Journal of the Brazilian Computer Society, 2026, 32:1, doi: 10.5753/jbcs.2026.6044

STELLAR: A Structured, Trustworthy, and Explainable LLM-Led Architecture for Reliable Customer Support

Matheus Ferracciú Scatolin [**Universidade Estadual de Campinas (UNICAMP)** | ***m252099@dac.unicamp.br***]
Helio Pedrini [**Universidade Estadual de Campinas (UNICAMP)** | ***helio@ic.unicamp.br***]

Instituto de Computação, Universidade Estadual de Campinas (UNICAMP), Av. Albert Einstein, 1251 - Cidade Universitária, Campinas - SP, 13083-852, Brazil.

Received: 19 May 2025 • **Accepted:** 17 December 2025 • **Published:** 21 February 2026

Abstract. While Large Language Models (LLMs) offer transformative potential for automating customer support, significant hurdles remain concerning their reliability, explainability, and consistent performance in complex, sensitive interactions. This paper introduces **STELLAR (Structured, Trustworthy, and Explainable LLM-Led Architecture for Reliable Customer Support)**, a novel architectural blueprint designed to address these issues. STELLAR utilizes a **Directed Acyclic Graph (DAG) structure** composed of nine specialized modules and eleven predefined workflows to orchestrate support interactions in a structured and predictable manner. This design promotes enhanced traceability, reliability, and control compared to less constrained systems. The architecture integrates components for few-shot classification, Retrieval-Augmented Generation (RAG), urgency-aware human escalation, compliance verification, user interaction validation, and knowledge base refinement through a semi-automated loop. This modular design deliberately balances LLM-driven innovation with operational requirements such as human-in-the-loop integration and ethical safeguards through embedded checks. We evaluated the core modules of STELLAR in key tasks - classification, retrieval, and compliance - demonstrating strong performance and reliability. Together, these features position STELLAR as a robust and transparent foundation for the next generation of intelligent, reliable customer support systems.

Keywords: Large Language Models (LLMs), Intelligent Customer Support Systems, Structured LLM Architectures, Reliable and Trustworthy AI, Retrieval-Augmented Generation (RAG).

1 Introduction

Effective customer support remains a cornerstone of business success, directly impacting customer satisfaction, loyalty, and brand reputation. The recent rise of powerful Large Language Models (LLMs) presents unprecedented opportunities to automate, scale, and enhance the quality of customer interactions [Wulf and Meierhofer, 2024]. There is significant interest in leveraging these models for complex tasks in customer support scenarios, such as understanding nuanced queries, retrieving relevant information, and generating helpful responses.

However, deploying LLMs directly for critical customer support functions faces substantial obstacles. Despite their fluency, standalone LLMs often exhibit limitations in consistency, factual trustworthiness (including susceptibility to hallucination [Xu *et al*., 2024]), and explainability, making their direct application in sensitive, high-stakes interactions problematic. These models may struggle to maintain context over long interactions or reliably execute complex, multi-step reasoning processes required in many support situations. Consequently, relying solely on zero-shot LLM capabilities for robust customer support is often insufficient.

Attempts to mitigate these issues through agent frameworks (e.g., CrewAI, LangChain), while conceptually promising, frequently introduce their own challenges [Sumers *et al*., 2024]. Many such systems lack predictability, and lead to non-deterministic behaviors that can be difficult to manage, debug, and trust [Xi *et al*., 2023] – attributes fundamentally misaligned with the requirements for reliable and consistent customer support operations.

To address this, a trend towards more structured approaches, such as predefined pipelines and workflows, is emerging. These methods aim to harness the capabilities of LLMs within controlled, modular frameworks, thereby enhancing reliability and predictability. In this context, this paper introduces STELLAR: a Structured, Trustworthy, and Explainable LLM-Led Architecture for Reliable Customer Support.

STELLAR provides a modular framework that supports:

- Inquiry Classification (Module 1);
- RAG-based FAQ Retrieval (Module 2);
- Direct Contact Routing (Module 3);
- Human Escalation with Summarization (Module 4);
- Sentiment Analysis (Module 5);
- Feedback Collection (Module 6);
- Knowledge Base Improvement (Module 7);
- Query Resolution Verification (Module 8);
- Compliance Checking (Module 9).

The primary objective of STELLAR is to serve as a comprehensive blueprint for designing and implementing next-generation intelligent customer support systems. It provides a systematic approach designed to balance cutting-edge inno-

vation with operational practicality, integrating robust techniques such as Retrieval-Augmented Generation (RAG) and vital human-in-the-loop escalation points. Furthermore, ethical and reliability concerns are embedded directly into the pipeline through dedicated modules and enforced workflows.

The architecture is based on a Directed Acyclic Graph (DAG), which facilitates explainability, prevents uncontrolled loops, and ensures a predictable, reliable flow tailored specifically for the demands of modern customer support. This paper details the STELLAR architecture, its constituent modules, interaction flows, and evaluation considerations, positioning it as a viable foundation for building more effective customer support solutions with LLMs.

While STELLAR is broadly applicable across diverse industries, this paper grounds its implementation and illustrative examples in the insurance domain, leveraging publicly available data. This sector offers a representative high-stakes environment that demonstrates the capabilities of the architecture while preserving generalizability to other business contexts.

2 Related Work

The pursuit of automating and enhancing customer support systems has evolved significantly over decades. Initial approaches relied heavily on rule-based expert systems and keyword-matching techniques [Weizenbaum, 1966]. While offering basic automation, these systems often struggled with natural language nuances, and required extensive manual effort to create and maintain comprehensive rule sets. Subsequent integration of traditional machine learning models improved capabilities such as intent recognition and sentiment analysis, leading to more sophisticated chatbots and virtual assistants [Chen *et al*., 2018]. However, these often still lacked deep contextual understanding and the ability to handle complex, multi-turn dialogues fluidly [Bodonhelyi *et al*., 2024].

The advent of Large Language Models (LLMs) has marked a paradigm shift, offering unprecedented capabilities in natural language understanding, generation, and reasoning [Brown *et al*., 2020; Dong *et al*., 2025]. Their potential application in customer support is vast, promising more natural, empathetic, and knowledgeable interactions capable of addressing a wider range of queries [Thoppilan *et al*., 2022]. Yet, as discussed previously, deploying standalone LLMs for customer support introduces significant challenges concerning reliability and factual consistency [Xu *et al*., 2024; Sreekar *et al*., 2024], including susceptibility to hallucination, particularly within complex steps, such as Retrieval-Augmented Generation (RAG) [Zhang and Zhang, 2023; Jiang *et al*., 2023; Shao *et al*., 2023]. These limitations hinder their direct use in scenarios demanding high levels of trust and predictable outcomes [Hudeček and Dušek, 2023; Nehring *et al*., 2024].

To harness LLM capabilities while mitigating their weaknesses, various orchestration frameworks and agent-based systems have emerged (e.g., LangChain [Chase, 2022], CrewAI [Moura, 2023], AutoGen [Wu *et al*., 2023]). These frameworks facilitate the construction of complex applications by composing sequences of LLM calls, integrating external tools, and managing memory or state. They offer considerable flexibility, enabling rapid prototyping and the development of sophisticated multi-step reasoning agents. However, this inherent flexibility, often allowing for dynamic, less constrained interactions between components, can compromise predictability and traceability, which are crucial for robust customer support operations. The potential for non-deterministic behavior and the difficulty in debugging complex interaction chains in such open-ended systems [Xi *et al*., 2023] may not align well with the need for consistent service delivery and auditable processes in customer-facing applications.

In contrast to these highly flexible but potentially unpredictable frameworks, the proposed STELLAR architecture adopts a structured approach grounded in a Directed Acyclic Graph (DAG). This architectural choice is designed to enforce predictable execution flows and enhance the overall reliability and explainability of the customer support process. By defining explicit pathways and transitions between specialized modules, the DAG structure provides a necessary level of control absent in more free-form agentic systems, making it inherently better suited for the demands of consistent and trustworthy customer service.

Within this structured framework, STELLAR integrates several innovative techniques:

- **Efficient Classification**: Module 1 utilizes LLMs with in-context learning (few-shot prompting) for initial query classification, demonstrating an effective approach to building accurate routers without requiring large, labeled datasets.
- **Optimized Information Retrieval**: Beyond standard RAG (used in Module 2), STELLAR incorporates a dedicated path (Module 3) leveraging the strong in-context retrieval capabilities of modern LLMs for relatively static, bounded information (e.g., contact details). Recognizing recent findings on near-perfect recall in moderate and large contexts [Li *et al*., 2025] (e.g., "Needle in a Haystack" benchmarks) and the potential for techniques such as Prompt Caching to reduce latency and cost for repeated access to static prompts [Gim *et al*., 2024], this path offers a highly reliable and potentially more efficient alternative to RAG for specific data types.
- **Semi-Automatic Knowledge Improvement**: Module 7 implements a human-in-the-loop [Shavit *et al*., 2023] workflow for knowledge base augmentation. Triggered by user queries that could not be solved by the RAG module (Module 2), it uses an LLM to draft potential new FAQ entries which are then queued for human review and approval before integration. This creates a continuous, targeted improvement cycle for the knowledge base, driven by real user interactions while maintaining human oversight to ensure factual trustworthiness.
- **Contextual Escalation & Feedback**: Module 4 incorporates an urgency assessment based on multiple factors (sentiment, query category, insurance type) to prioritize human escalations effectively. Furthermore, Module 6 provides systematic feedback collection and automated classification/routing, closing the loop for system improvement based on direct user input.

By combining a reliability-focused DAG architecture with these specialized modules and techniques, STELLAR presents a blueprint that balances the power of LLMs with the practical requirements of predictable, and continuously improving customer support systems.

3 The STELLAR Architecture

The STELLAR architecture is systematically engineered to create a robust and adaptable framework for developing next-generation intelligent customer support systems. Its design is explicitly guided by a set of core principles aimed at overcoming the limitations of both standalone LLM deployments and overly complex agentic systems. These principles are:

- **Reliability and Predictability**: Ensuring consistent system behavior, predictable outcomes for similar inputs, and robust handling of errors or unexpected situations.
- **Explainability and Traceability**: Facilitating the understanding of why the system produced a particular response or took a specific action, allowing for easier debugging, auditing, and trust-building.
- **Balanced Integration**: Pragmatically combining cutting-edge AI innovations (usage of LLMs for a broad range of tasks) with practical necessities (e.g., predefined paths, human-in-the-loop escalation points) and embedded ethical safeguards (compliance checks).

3.1 Design Principles & Overview

Central to achieving these principles is the foundational choice of a Directed Acyclic Graph (DAG) structure to orchestrate the interactions between specialized functional modules. This architectural decision offers advantages over alternatives. While simple linear pipelines often lack the flexibility to handle the branching logic inherent in diverse customer support scenarios, and highly dynamic, free-form agentic frameworks can introduce unpredictability and hinder traceability [Xi *et al*., 2023], the DAG model provides a controlled, structured environment. Therefore, it inherently prevents infinite processing loops, and significantly enhances traceability by providing clear, auditable paths for each user interaction.

The overall STELLAR architecture, illustrating the constituent modules and their primary interconnections, appears in Figure 1(a) and Figure 1(b). As shown, the system is composed of 9 distinct modules, each performing a specialized function within the customer support lifecycle. Interactions start universally at Module 1 (Inquiry Classifier), which serves as the primary router. Based on its analysis of the user's initial query, Module 1 directs the workflow into one of three main conceptual pathways:

1. **FAQ Pathway**: Activated for general informational queries likely answerable by the existing knowledge base. This path primarily engages Module 2 (FAQ Retrieval), utilizing RAG techniques.
2. **Direct Information Pathway**: Employed for specific, often factual queries (e.g., contact numbers, plan details) potentially handled more effectively via targeted retrieval from a constrained knowledge source, engaging Module 3 (Direct Contact & Consultation Information).
3. **Human Escalation Pathway**: Triggered for urgent cases, complex problems exceeding automated capabilities, explicit user requests for human help, or as a fallback from other pathways. This path involves Module 5 (Sentiment Analysis) and culminates in Module 4 (Human Support Escalation).

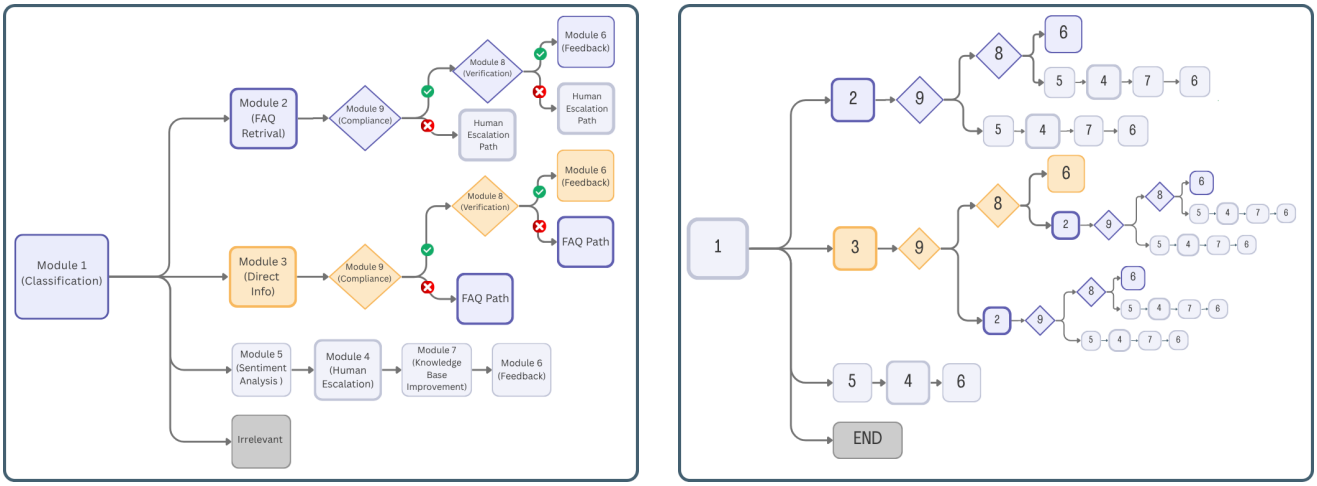
While these pathways define the initial routing, the progression through the DAG is not strictly linear and incorporates critical decision points that allow for dynamic, yet controlled, adjustments based on intermediate results or user feedback. Key points of flux divergence include:

- **Module 1** (Initial Routing): As mentioned, selects the initial pathway based on query classification.
- **Module 8** (User Verification): Following a potential resolution generated by Module 2 or 3 (after passing through Module 9), this module actively confirms with the user if their issue was resolved. A negative response ("No, my problem is not solved") triggers a rerouting mechanism: an unresolved query from the Direct Information Pathway (Module 3 origin) is typically rerouted to the FAQ Pathway (Module 2) for a broader search, while an unresolved query from the FAQ Pathway is escalated to the Human Escalation Pathway (Module 4).
- **Module 9** (Compliance Check): This crucial module validates the outputs of generative LLM modules (primarily Modules 2 and 3) against predefined compliance and quality criteria before presenting them to the user. If a response is flagged as non-compliant, Module 9 permits one retry of the preceding generative module. If the response fails compliance again, Module 9 acts as a safety net, possibly redirecting the workflow similarly to Module 8 (e.g., diverting from Module 3's path towards Module 2, or from Module 2's path towards Module 4), assuming the automated response generation is unreliable for the given query. This also ensures problematic queries that couldn't be answered adequately by Module 2 eventually reach Module 7 (via Module 4) to potentially improve the knowledge base.

The interplay between the specialized modules and these defined control flow mechanisms results in a finite set of 12 distinct, predefined end-to-end workflows, also illustrated in Figure 1(b). Each workflow represents a valid, traceable path through the DAG, starting from the initial query classification by Module 1 and concluding, upon successful resolution or managed escalation, with Module 6 (Feedback Collector). This structured yet adaptable design forms the core of the STELLAR architecture, aiming to provide a reliable and explainable foundation for intelligent customer support.

3.2 Adaptability to Domains and Modalities

The decomposition of the STELLAR architecture into nine specific modules is guided by the principle of *Separation of Concerns*. Each module is designed to encapsulate a dis-



(a) Simplified illustration of the architecture with all possible workflows.

(b) Complete illustration of all possible workflows.

Figure 1. Comparison between simplified and complete workflow illustrations.

tinct, logically coherent function within the customer support lifecycle.

This design choice makes STELLAR inherently domain-agnostic. To adapt it to other domains (e.g., e-commerce, healthcare), one would primarily need to update the knowledge sources (Modules 2 and 3) and reconfigure prompt patterns and classifiers (Modules 1 and 4). The core workflow logic, however, remains largely unchanged.

The architecture also supports multimodal inputs. To handle voice interactions, two lightweight wrappers can be added: a speech-to-text transcriber, and a text-to-speech synthesizer. Specifically, voice input would be transcribed in the operation of Modules 1, 3, 6, or 8, and generated text from Modules 2, 3, 6, or 8 could be converted back to speech. The internal DAG and modules continue to operate in text, preserving STELLAR's core logic while enabling flexible modality adaptation.

3.3 Module Descriptions

This subsection provides a detailed description of each of the 9 functional modules comprising the STELLAR architecture. For each module, we outline its specific role, required inputs, expected outputs, internal workflow and core technologies, and briefly summarize key evaluation results demonstrating its efficacy.

Our evaluation methodology was designed to ensure fairness and transparency. Each module was validated using a distinct, module-specific test set, and the knowledge base for Module 2 (FAQ Retrieval, RAG) was constructed by scraping publicly available FAQ data from a real-world insurance provider. Test cases were then synthetically generated using GPT-4o, followed by human verification and refinement to ensure plausibility and diversity of query types and complexities. All code implementations and test sets are publicly available in GitHub repository.

During development, prompt engineering was performed with a small number of few-shot examples that were strictly separated from the final test sets to avoid bias. Although these sets are suitable for demonstrating feasibility and functionality, we acknowledge their modest size. The results should therefore be interpreted as a proof-of-concept validation, with larger and more diverse testing left as important future work to strengthen statistical confidence and assess broader generalizability.

3.3.1 Module 1: Inquiry Classifier

Role. Module 1 serves as the initial entry point and primary router for all incoming user interactions within the STELLAR system. Its core function is to analyze the user's initial query and classify it into one of several predefined categories, thereby directing the workflow to the most appropriate downstream pathway (FAQ, Direct Information, Human Escalation, or filtering out irrelevant requests). The operation of this module is illustrated in Figure 2.

Inputs/Outputs.

- **Input**: The user's query text.
- **Output**: Upon successful classification, the module outputs a structured dictionary containing two key fields:
  - "Explanation": a textual justification generated by the LLM for its classification decision;
  - "Category": an integer representing the assigned pathway: 0 for FAQ, 1 for Direct Information, 2 for Human Escalation, or 3 for Irrelevant.

Internal Workflow & Techniques. Module 1 leverages a Large Language Model (LLM) configured with few-shot prompting to perform classification without requiring extensive labeled training data. The prompt combines predefined input-output examples (demonstrating the desired classification, explanation, and JSON output format for each category) with the new user query. Including exemplar explanations before the category labels in the prompt examples is designed to encourage more reasoned classification by the LLM, akin to chain-of-thought prompting [Wei *et al.*, 2023], as the LLM can think of why a given category is a great choice before actually consolidating its choice.

The module invokes the LLM, attempts to parse the expected JSON structure from the response, and validates the extracted category. A retry mechanism (up to two retries, total three attempts) handles potential LLM generation errors or parsing failures. If a valid response is not obtained, the module defaults to category 0 (FAQ Pathway) for robustness. It is important to mention that, with the new generation of LLMs, these parsing errors rarely occur. For instance, in the experiments mentioned as follows, this error did not occur once.

Evaluation & Results. The selection of the optimal LLM and prompt configuration was guided by systematic experimentation. Initial experiments focused on prompt engineering, using a smaller model (Llama 3.2 3B [Meta, 2024a]) for clearer differentiation and a 100-item test set representing typical query distributions (initially, with only categories 0, 1 and 2). An iterative approach, analyzing the confusion matrix to identify common misclassifications (e.g., between category 0 and 1) and strategically adding targeted examples to the few-shot prompt, proved effective.

This process demonstrated that careful, targeted additions of examples could significantly improve accuracy (raising it from 87% to 95% in one instance), while also confirming that simply increasing example quantity does not guarantee better performance. Subsequently, the optimized prompt was tested across various LLMs, considering both performance and cost. The Llama 3.1 70B [Meta, 2024c] model was ultimately selected, achieving 100% classification accuracy on the initial 100 test cases.

The later introduction of category 3 ('Irrelevant') with 25 additional test cases maintained this 100% accuracy on the expanded 125-item set, validating the module's effectiveness in accurately routing diverse queries.

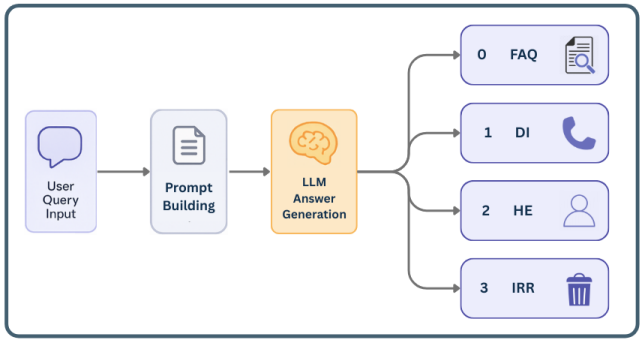


Figure 2. Module 1 (Inquiry Classifier) operation.

3.3.2 Module 2: FAQ Retrieval (RAG)

Role. Module 2 serves as the primary engine for addressing user queries that fall into the general information category (Category 0 from Module 1). It implements a Retrieval-Augmented Generation (RAG) workflow to retrieve relevant information from a curated FAQ knowledge base and synthesize a coherent, contextually appropriate answer for the user.

Inputs/Outputs.

- **Input**: The natural language customer query text, as passed from Module 1.
- **Output**: The synthesized answer (text generated by the LLM based on retrieved FAQs) and sources (a list of FAQ IDs used to generate the answer).

Note: This module also includes auxiliary functionalities for managing the FAQ knowledge base itself, such as adding new FAQs (utilized by Module 7) and initially building the underlying vector database (using ChromaDB in our implementation) from a structured data source (e.g., JSON).

Internal Workflow & Techniques. Module 2 executes a multi-stage RAG pipeline (as shown in Figure 3) designed to maximize retrieval relevance and answer quality:

1. **Hybrid Search**: To leverage the strengths of both lexical and semantic matching [Rayo *et al.*, 2025], a hybrid retrieval strategy is employed. It first performs semantic search using vector embeddings over a ChromaDB database to retrieve documents comprising 70% of the target retrieval count (k). Subsequently, it uses BM25, a keyword-based algorithm, to retrieve the remaining 30% of documents, ensuring that any documents already retrieved via semantic search are not considered to maximize diversity.
2. **Re-ranking**: The initial set of retrieved FAQs (from both search methods) is then re-ranked for relevance using an LLM. A prompt is constructed containing the original user query and the text of the retrieved FAQs. The LLM is tasked with evaluating and ordering these FAQs based on their direct relevance to the query. This step refines the initial retrieval set, prioritizing the most pertinent documents. Error handling and retries are included in the LLM call for robustness.
3. **Context Construction**: The top-ranked FAQs, as determined by the re-ranking step, are selected to form the context that will be provided to the final generation LLM.
4. **Response Generation**: A final prompt is constructed using a predefined template, incorporating the original user query and the curated context from the re-ranked FAQs. This prompt is sent to an LLM, which generates a synthesized, natural language answer addressing the user's query based only on the provided context.
5. **Formatting**: The generated answer text and the list of source FAQ IDs used in the context construction phase are formatted into the final answer.

Evaluation & Results. The performance of Module 2's RAG pipeline was evaluated using a dataset of 100 FAQs and 100 corresponding test queries, where each query had 1-3 ground truth relevant FAQ IDs associated with it. Key retrieval metrics were used:

- **Hit-Rate@n**: The proportion of queries for which at least one correct FAQ is retrieved within the top n results;

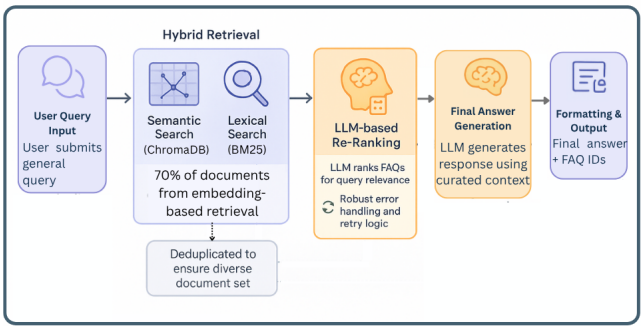


Figure 3. Module 2 (FAQ Retrieval) operation.

- **Recall@n**: The average proportion of a query's relevant FAQs retrieved within the top n results;
- **Mean Reciprocal Rank (MRR)**: The average reciprocal rank of the first relevant FAQ retrieved. The formula is as follows:

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\text{rank}_i}, \tag{1}$$

where:

  - N: Number of queries or test cases (for this experiment: $N = 100$ test cases).
  - rank_i: Position (rank) of the first relevant item in the results for the i-th query.

Two main experiments were conducted (they are both available at GitHub):

1. **Embedding Model Selection**:
   Four common embedding models (all-MiniLM-L6-v2 [sentence-transformers, 2020b], all-mpnet-base-v2 [sentence-transformers, 2021], multi-qa-MiniLM-L6-cos-v1 [sentence-transformers, 2020c], all-MiniLM-L12-v2 [sentence-transformers, 2020a]) were compared using only semantic search. Based on a balance of MRR, Hit-Rate@5, Recall@5, and average latency (See Table 1 and Figure 4). All-MiniLM-L6-v2 was selected as the optimal embedding model for subsequent experiments.
2. **RAG Strategy Comparison**:
   Using the selected all-MiniLM-L6-v2 model, four different RAG strategies were evaluated: Semantic Search (SS) alone, SS with LLM re-ranking (SS+R), Hybrid Search (HS) alone, and HS with LLM re-ranking (HS+R). Results indicated that Hybrid Search with Re-ranking (HS+R) yielded the best overall retrieval performance across MRR, Hit-Rate@5, and Recall@5 (See Table 2 and Figure 5), achieving scores of 0.939, 0.99, and 0.98 respectively. As expected, this strategy led to the highest average latency due to the added computational steps. The fact that HS+R achieves the highest performance metrics aligns with previous research, which suggested that combining hybrid retrieval with re-ranking is often a best practice for RAG [Wang *et al.*, 2024].
   However, the superior performance of the HS+R strategy comes at the cost of the highest average latency (5.18 seconds), a point that warrants discussion regarding its practical viability. In real-time customer support chat applications, such a delay could negatively impact the user experience. This highlights a critical trade-off between retrieval accuracy and response time. For instance, an organization prioritizing minimal latency could opt for the "Hybrid Search without reranking" strategy. This approach offers a compelling balance, reducing latency by over 40% to 2.96 seconds while maintaining a high average performance score (0.95). The ability to select the most appropriate strategy based on specific business needs is a core feature of the STELLAR blueprint. Furthermore, the latency of the HS+R strategy could be further mitigated in a production environment through optimizations such as using a smaller, faster model specifically for the re-ranking task or implementing caching for common queries.

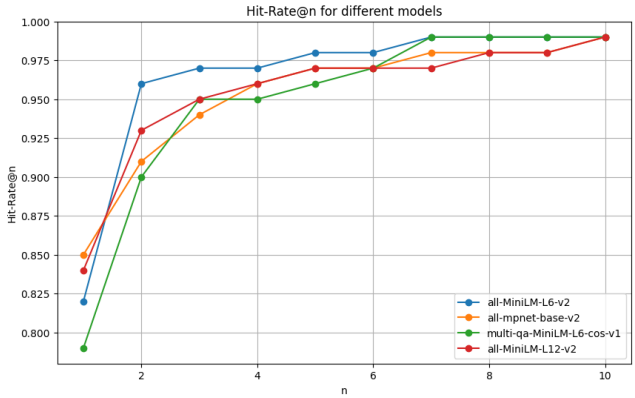


Figure 4. Hit-Rate@n performance comparison across different embedding models for n ranging from 1 to 10.

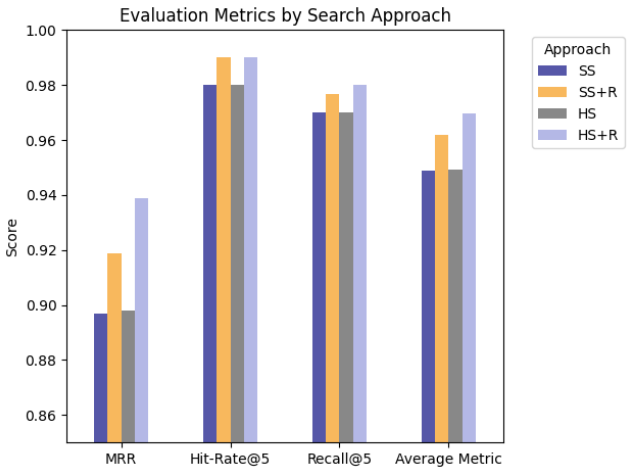


Figure 5. Bar chart comparing evaluation metrics (MRR, Hit-Rate@5, Recall@5) for different RAG approaches: Semantic Search (SS), Semantic Search + Reranking (SS+R), Hybrid Search (HS), Hybrid Search + Reranking (HS+R)).

3.3.3 Module 3: Direct Information Retrieval

Role. Module 3 is specialized for handling queries classified by Module 1 as seeking specific, often factual, information that is relatively static and contained within a predefined

Table 1. Comparison of embedding models based on MRR, Hit-Rate@5, Recall@5, an averaged metric (average of the 3 previous metrics), and average latency (rounded to 2 decimal places).

Model	MRR	Hit-Rate@5	Recall@5	Avg Metrics	Avg Delay
all-MiniLM-L6-v2	0.90	0.98	0.97	0.95	3.36
all-mpnet-base-v2	0.90	0.97	0.96	0.94	3.17
multi-qa-MiniLM-L6-cos-v1	0.87	0.96	0.94	0.92	2.89
all-MiniLM-L12-v2	0.90	0.97	0.96	0.94	2.98

Table 2. Comparison of RAG strategies based on MRR, Hit-Rate@5, Recall@5, an averaged metric (the average of the previous 3 metrics), and average latency (rounded to 2 decimal places).

Method	MRR	Hit-Rate@5	Recall@5	Avg Metrics	Avg Delay
Semantic Search without reranking	0.90	0.98	0.97	0.95	2.83
Semantic Search with reranking	0.92	0.99	0.98	0.96	4.08
Hybrid Search without reranking	0.90	0.98	0.97	0.95	2.96
Hybrid Search with reranking	0.94	0.99	0.98	0.97	5.18

knowledge corpus (e.g., contact phone numbers, email addresses, specific plan details). Instead of employing RAG, it utilizes direct in-context learning with an LLM for potentially higher precision retrieval within a bounded information set.

Inputs/Outputs.

- **Input**: The user's query text.
- **Output**: Depending on the analysis, one of the following:
  - A direct textual answer containing the requested information.
  - A follow-up question prefixed with "-1", requesting clarification from the user.
  - An "information unavailable" message prefixed with "-2".

(Note: The prefixes "-1" and "-2" serve as internal signals for the workflow orchestration logic).

Internal Workflow & Techniques. This module leverages the strong in-context learning capabilities of modern LLMs. The process (shown in Figure 6) is as follows

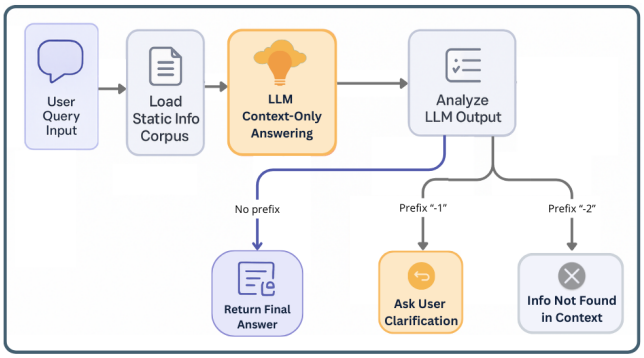


Figure 6. Module 3 (Direct Information Retrieval) operation.

1. **Context Loading**:
   A predefined corpus of relevant information (in our implementation, approximately 2,000 tokens of contact details for Bradesco Seguros, structured similarly to a markdown file) is loaded.
2. **Prompt Construction**:
   The user's query is combined with the entire preloaded information corpus within a single prompt provided to the LLM. The prompt instructs the LLM to answer the query based only on the provided context. Crucially, the prompt also specifies the use of prefixes "-1" (for necessary follow-up questions) and "-2" (if the answer is definitively not in the context).
3. **LLM Query & Response Analysis**:
   The LLM processes the prompt. The module then analyzes the beginning of the response:
   - If it starts with "-1", the subsequent text is treated as a follow-up question to be posed back to the user. The system allows for one round of follow-up; if the user responds, their clarification is combined with the original query for a second LLM call.
   - If it starts with "-2", the subsequent text is treated as a message indicating the information is unavailable within this module's scope.
   - Otherwise, the entire response is considered the direct answer to the user's query.

This approach is chosen over RAG for this specific type of data due to its relatively static nature and moderate size. Recent benchmarks demonstrate near-perfect recall for LLMs on "Needle-in-a-Haystack" tasks within moderate context windows [Li *et al.*, 2025], suggesting high reliability for this method. Furthermore, the static nature of the information corpus makes it amenable to techniques such as Prompt Caching [Gim *et al.*, 2024; Fu and Feng, 2023], which can significantly reduce latency and computational cost by caching the embedding of the large, static context portion of the prompt.

Evaluation & Results. Module 3's performance was assessed in two stages using the Llama 3.1 70B model:

1. **Direct Retrieval Accuracy**:
   A test set of 50 queries requiring direct lookup within the 2k-token context was used. The module achieved 100% accuracy (50/50 correct retrievals), confirming the LLM's ability to reliably extract information explicitly

present in the provided context, aligning with expectations for capable models on such tasks.

2. **Handling Ambiguity & Unavailability**: The module's functionality was extended to allow follow-up questions ("-1") and indicate unavailability ("-2"), tested with a new set of 50 diverse queries (including direct, ambiguous, and unanswerable ones). In this more complex scenario, the module achieved 90% task success rate. The primary source of errors was the LLM generating a follow-up question ("-1") when the original query, although potentially slightly underspecified, could have been answered directly based on common assumptions (e.g., asking for clarification on pension type when a general pension number exists). While this reflects a conservative approach by the LLM, aiming for precision, it highlights a potential area for refinement through further prompt tuning (e.g., adding more examples to guide when not to ask follow-up questions). However, the ability to correctly utilize the "-1" and "-2" mechanisms was successfully demonstrated.

3.3.4 Module 4: Human Support Escalation

Role. Module 4 serves as the critical interface between the automated STELLAR system and human customer support agents. It is activated when an issue requires human intervention, either through direct classification by Module 1 (Category 2) or escalation from Modules 2 or 3 via Modules 8 or 9. Its primary purposes are to:

- Efficiently prepare the context;
- Assess urgency;
- Assign an appropriate human agent;
- Provide the agent with a tailored starting message for the interaction.

Inputs/Outputs.

- **Input**:
  - `sentiment_analysis`: output from Module 5, containing probability scores for positive, neutral, and negative sentiment;
  - `chat_history`: the conversation transcript up to the escalation point.
- **Output**:
  - `customer_name`;
  - `insurance_type`;
  - `issue_summary`;
  - `query_category`;
  - `query_subcategory` (extracted from history);
  - `urgency_score` (calculated);
  - `human_attendant_id`, `human_attendant_name` (if assigned);
  - `recommended_message`: a draft message for the human agent to use or adapt.

Note: This module also manages an internal set of human agents, including their availability status and areas of specialization (e.g., by insurance type, query category), and includes functions for adding new agents and updating their status (e.g., setting to 'Busy' on assignment, freeing them post-interaction).

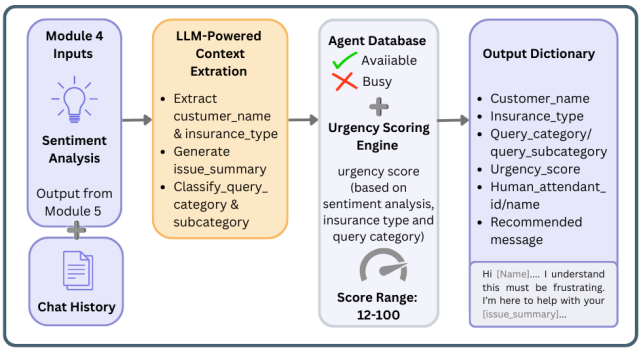


Figure 7. Module 4 (Human Support Escalation) operation.

Internal Workflow & Techniques. Module 4 orchestrates several steps to facilitate a smooth handover to a human agent (as shown in Figure 7):

1. **Context Extraction**: Leverages LLM-powered functions to parse the `chat_history`. One function extracts `customer_name` and `insurance_type`; another generates a concise `issue_summary` and identifies the relevant `query_category` and `query_subcategory`.
2. **Urgency Calculation**: Computes a numerical `urgency_score` to prioritize incoming escalations. This score is a weighted sum of three components:
   - Sentiment Score (Range: 12–50): Derived from Module 5's output, heavily weighting negative sentiment: score $= 25 \times (\text{negative} \times 2.0 + \text{neutral} \times 1.0 + \text{positive} \times 0.5)$
   - Category/Subcategory Score (Range: 0–30): Based on predefined weights assigned to different issue types, reflecting inherent business priority.
   - Insurance Type Score (Range: 0–20): Based on predefined weights associated with different product lines.
   - The combined score provides a nuanced measure of priority, ranging from 12 to 100.
3. **Human Agent Assignment**: Attempts to match the escalation with an available human agent, prioritizing agents specialized in the extracted `insurance_type` and `query_category`. If a suitable agent is found, their status is updated to `'Busy'`. If no agent is immediately available, the customer is added to a waiting list, ordered by the calculated `urgency_score`.
4. **Recommended Message Generation**: Utilizes an LLM to generate a recommended message for the assigned (or future) human agent. This message is designed to be empathetic, personalized (using extracted customer name and issue summary), contextually appropriate (reflecting sentiment and formality), and aims to facilitate a smooth transition by ending with an engaging next step or question.

Evaluation & Results. Key LLM-driven components of Module 4 were evaluated individually using the Llama 3.1

70B model and a test set of 20 diverse chat history examples (including varying formality and some ambiguity):

- **Customer Detail Extraction**: The function responsible for extracting `customer_name` and `insurance_type` achieved 100% accuracy (20/20 correct extractions for both fields), demonstrating robustness even with varied inputs.
- **Issue Summarization & Categorization**: The function extracting `issue_summary`, `query_category`, and `query_subcategory` achieved 97.5% accuracy (39/40 correct category/subcategory pairs), highlighting high overall performance despite the inherent subjectivity in this task.

3.3.5 Module 5: Sentiment Analysis

Role. Module 5 is responsible for evaluating the emotional tone conveyed in the customer's communications. Its primary function is to analyze the chat history and quantify the sentiment expressed (positive, neutral, or negative). This sentiment score serves as a key input for Module 4 (Human Support Escalation).

Inputs/Outputs.

- **Input**: A single string containing the `chat_history` of the customer interaction up to the point where sentiment analysis is required.
- **Output**: A dictionary containing the normalized probabilities for each sentiment class, e.g., `{positive: 0.1, neutral: 0.8, negative: 0.1}`.

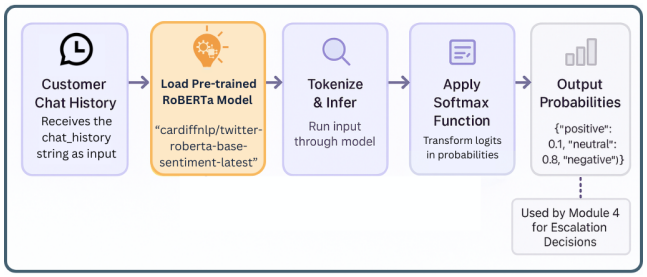


Figure 8. Module 5 (Sentiment Analysis) operation.

Internal Workflow & Techniques. The module employs the pre-trained "cardiffnlp/twitter-roberta-base-sentiment-latest" [cardiffnlp, 2021] model to analyze the input `chat_history`. The model processes the text to produce a normalized probability distribution over the sentiment classes (positive, neutral, negative), which serves as the final sentiment score.

Evaluation & Results. The performance of the selected sentiment analysis model (cardiffnlp/twitter-roberta-base-sentiment-latest) was validated on a test set of 50 examples manually labeled as predominantly positive (10 cases), neutral (30 cases), or negative (10 cases). The model correctly classified the primary sentiment in 48 out of 50 cases (96% accuracy), demonstrating its suitability for reliably gauging customer sentiment within the STELLAR workflow.

3.3.6 Module 6: Feedback Collector

Role. Module 6 serves as the concluding step in most STELLAR interaction workflows, responsible for systematically gathering customer feedback on their experience. It collects both quantitative ratings and qualitative comments, categorizes the textual feedback using an LLM, and routes this information to relevant internal teams for analysis and system improvement.

Inputs/Outputs.

- **Input**: Contextual information about the completed interaction, including `chat_history`, `human_attendant_name` (if applicable), sentiment analysis results (from Module 5), `insurance_type`, `issue_summary`, `query_category`, and `query_subcategory` (from Module 4).
- **Output**: A structured dictionary containing the collected feedback, organized by feedback key (corresponding to specific questions asked). Each key maps to a sub-dictionary containing the `rating` (integer 1–5), `follow_up_response` (text comment, if provided), and `categories` (a list of LLM-assigned category labels for the comment).

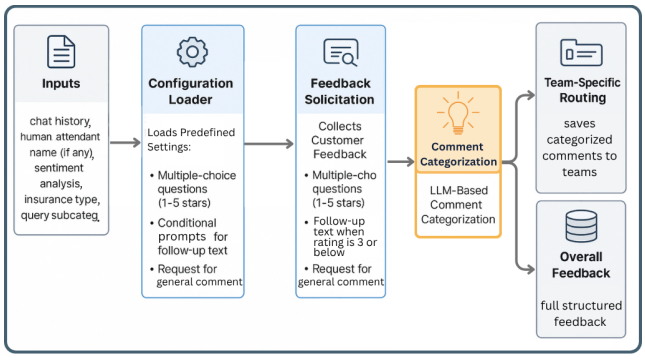


Figure 9. Module 6 (Feedback Collector) operation.

Internal Workflow & Techniques. This module's workflow (as shown in Figure 9) is:

1. **Configuration Loading**:
   Retrieves predefined configurations, including the set of feedback questions to ask (e.g., satisfaction with resolution, ease of interaction), the list of possible feedback categories (e.g., "Agent Performance", "Usability Issue", "System Error"), and the mapping rules for routing categorized feedback to designated teams.
2. **Feedback Solicitation**:
   Interacts with the customer by presenting a series of multiple-choice questions, requesting ratings on a 1-5 scale corresponding to the predefined feedback keys. It conditionally prompts for textual follow-up comments if a rating is low (e.g., ≤ 3) and always requests general comments.

3. **Comment Categorization**:
   Any provided textual feedback comments are processed using an LLM-powered function (`categorize_comment`). This function analyzes the comment's content and classifies it into one or more of the predefined feedback categories using few-shot prompting.
4. **Saving and Routing**:
   The structured feedback (ratings, comments, assigned categories) is compiled into the output dictionary. Simultaneously, the module logs the categorized comments to team-specific files based on the loaded routing rules and saves the complete interaction context and feedback summary to the overall log file.

Evaluation & Results. The critical LLM component of this module – automatic categorization of free-text feedback comments – was evaluated. A test set of 50 representative feedback comments was created, with each comment manually assigned one or two ground truth categories from the predefined list (e.g., "Agent Performance", "Claims Processing Issue", "Improvement Suggestion"). The LLM categorization function was tested against this dataset. It achieved 92.5% accuracy in correctly identifying the primary intended category for the comments, demonstrating its effectiveness in reliably interpreting and structuring qualitative user feedback for targeted internal review and action.

3.3.7 Module 7: Knowledge Base Builder

Role. Module 7 functions as a semi-automatic mechanism for improving the underlying FAQ knowledge base used by Module 2 (RAG). It is triggered downstream from Module 4 (Human Support Escalation) specifically for interactions where the previous automated retrieval modules (Module 2 or Module 3) failed to provide a satisfactory resolution, suggesting a potential gap in the existing knowledge base. Module 7 aims to streamline the process of identifying and filling these gaps by generating draft FAQ entries based on unresolved user queries, facilitating subsequent human review and approval. The workflow is illustrated in Figure 10.

Inputs/Outputs.

- **Input**:The original `user_question` text that ultimately required human intervention and its associated `insurance_type` (providing context for the draft generation).
- **Output**:
  - Writes draft FAQ suggestions (including the original user query, generated question, draft answer, and metadata such as timestamp) to a dedicated review queue file.
  - Upon human approval (via an external review process), triggers updates to Module 2's knowledge base (both the source JSON file and the corresponding vector database).

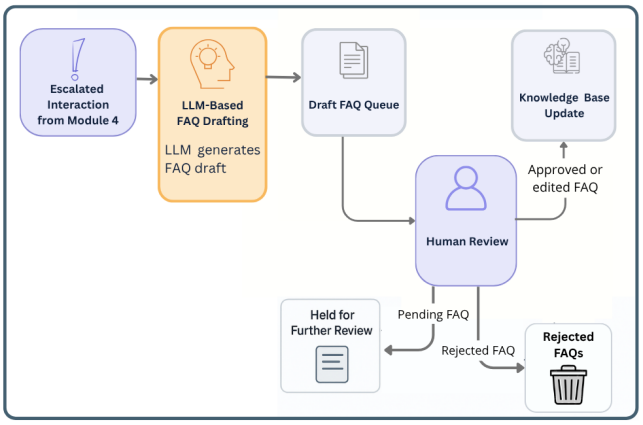


Figure 10. Module 7 (Knowledge Base Builder) operation.

Internal Workflow & Techniques.

1. **Draft FAQ Generation**: An LLM is prompted with the unresolved user question and insurance type. Its task is to generate a candidate FAQ entry, consisting of a well-formed question derived from the user's query and a plausible draft answer. The module includes basic validation to ensure the LLM response adheres to the expected format.
2. **Review Queue Management**: The generated draft FAQ, along with relevant metadata (original query, timestamp), is added to a persistent "pending review" queue (e.g., a JSON file or database table).
3. **Human Review Interface (External)**: This module relies on an external process or interface where human subject matter experts or support agents can access the pending queue. They can review each draft, choosing to:
   - Approve: Accept the draft FAQ as is.
   - Rewrite: Edit the generated question and/or answer for accuracy and clarity.
   - Reject: Discard the draft if it's irrelevant or incorrect.
   - Keep Pending: Leave it for later review.
4. **Knowledge Base Integration**: When a draft is approved (or approved after rewriting) via the external review process, Module 7 facilitates its integration. The approved FAQ is added to the FAQ dataset, and Module 2's database update function is invoked to add the corresponding embedding to the vector database, making it available for future retrievals.

Evaluation & Results. Direct performance evaluation of Module 7 using metrics such as accuracy is less applicable, as its primary goal is not autonomous, perfect FAQ generation but rather process facilitation for knowledge base improvement. Its value lies in automatically identifying potential knowledge gaps (based on failed automated resolutions) and reducing the human effort required to formulate relevant questions and initial draft answers. The critical human-in-the-loop review step ensures the accuracy and quality of the information added to the knowledge base, preventing the propagation of potentially incorrect LLM-generated content.

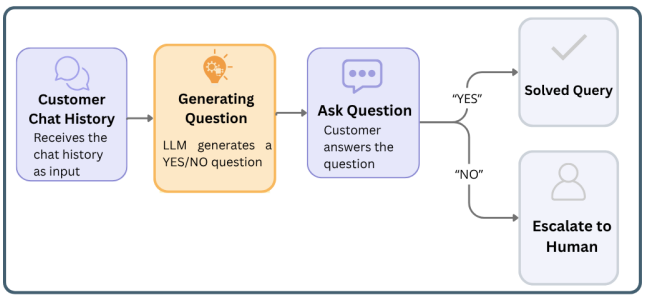


Figure 11. Module 8 (Resolution Verifier) operation.

3.3.8 Module 8: Resolution Verifier

Role. Module 8 acts as a crucial checkpoint within the workflow, specifically after an automated response has been generated and considered compliant (by Module 9). Its purpose is to explicitly verify with the customer whether the provided information (by Module 2 or 3) has successfully resolved their query before concluding the interaction or initiating an escalation. The module's operation is illustrated in Figure 11.

Inputs/Outputs.

- **Input**: The `chat_history` encompassing the user's query, and the system's generated response (from Module 2 or 3, post-Module 9 check).
- **Output**: A Boolean value: True if the customer confirms their issue is resolved, False otherwise, signaling the need for further action (escalation via Module 4).

Internal Workflow & Techniques.

1. **Tailored Question Generation**:
   Instead of using a generic prompt such as "Is your issue resolved?", this module leverages an LLM to generate a context-aware, personalized verification question. The LLM is prompted with the chat history and tasked with formulating a specific binary (yes/no) question that references the user's original issue or the solution provided (e.g., "Did the contact number provided for Auto Assistance help you?", "Were you able to find the details needed in the FAQ about claims processing?"). This degree of personalization aims to provide a more natural and engaging user experience [Joko *et al.*, 2024].
2. **User Interaction & Response Processing**:
   The generated question is presented to the customer. The module then awaits the user's response to determine whether the resolution was successful from the customer's perspective. A positive confirmation maps to True, while a negative response or indication of continued issues maps to False.

Evaluation & Results. Similar to Module 7, Module 8's effectiveness is not typically measured by standard accuracy metrics in isolation. Its primary value is judged qualitatively by its contribution to the user experience and its functional role in the workflow logic. The use of an LLM to generate tailored verification questions is a design choice aimed at improving perceived interaction quality compared to static prompts. The module's success lies in reliably capturing the user's confirmation, which directly determines whether the interaction proceeds to feedback collection (Module 6) or requires rerouting/escalation.

3.3.9 Module 9: Compliance Checker

Role. Module 9 acts as an essential quality assurance and safety layer within the STELLAR architecture. It intercepts responses generated by preceding LLM-based modules (specifically Module 2 and Module 3) before they are presented to the customer. Its purpose is to evaluate these responses against predefined compliance, ethical, and quality standards, ensuring they are appropriate, safe, and meet required criteria.

Inputs/Outputs.

- **Input**: The user question that prompted the response, and the candidate LLM response generated by the preceding module (e.g., Module 2 or 3).
- **Output**: A dictionary containing:
  - "compliance": A Boolean value (True if the response passes all checks, False otherwise).
  - "violation": A string indicating the specific type of violation if compliance is False (e.g., "Inadequate Tone", "Disclosure of Confidential Information"), or a null/specific string such as "No violation" if compliance is True.

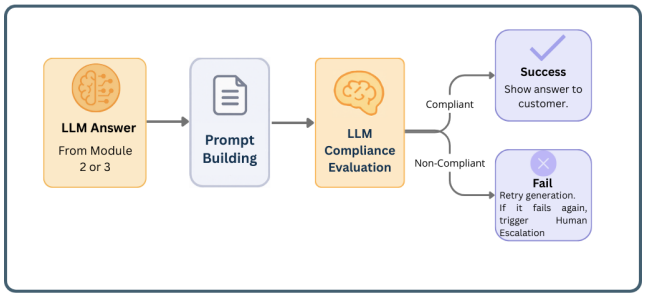


Figure 12. Module 9 (Compliance Checker) operation.

Internal Workflow & Techniques.

1. **Evaluation Prompting**: The module utilizes an LLM, prompted with the user question and the candidate LLM response. The prompt instructs the LLM to evaluate the response based on a defined set of compliance criteria. In our implementation, these criteria include:
   - Disclosure of Confidential Information: Checking if sensitive customer or internal data is inappropriately shared.
   - Discrimination or Prejudice: Identifying biased, unfair, or prejudiced language.
   - Incomplete or Vague Response: Assessing if the response fails to adequately address the user's query.
   - Inadequate Tone: Evaluating professionalism, respectfulness, and appropriateness of the language tone.

2. **Classification**: Based on the evaluation against these criteria, the LLM is prompted to classify the response as either "Compliant" or "Non-Compliant" and, if non-compliant, to specify which criterion (or criteria) was violated.
3. **Output Formatting**: The LLM's classification result is parsed and formatted into the output dictionary (compliance Boolean, violation string). Non-compliant responses trigger the retry/rerouting logic described in the architecture overview (Section 3.1), preventing potentially harmful or low-quality responses from reaching the user. Violations are also logged for monitoring and potential system refinement.

The workflow is illustrated in Figure 12.

Evaluation & Results. Experiments were conducted to validate the ability of LLMs to perform this compliance checking task accurately. The primary objectives were to confirm consistent binary compliance classification (Compliant/Non-Compliant) and accurate identification of the specific violation type. A test set of 50 responses was created, including 10 compliant examples and 10 examples specifically designed to violate each of the four non-compliance criteria.

Using the Llama 3.1 70B model, the module achieved 100% accuracy both in the binary classification task (correctly identifying all 50 responses as either compliant or non-compliant) and in the multi-class classification task (correctly identifying the specific type of violation for all 40 non-compliant examples).

To evaluate the feasibility of using smaller, more resource-efficient models for this critical safety task, we also tested the Gemma2 9B [Google, 2024] and Llama 3 8B [Meta, 2024b] models. The Gemma2 9B model also achieved 100% accuracy, making it, alongside the Llama 3.1 70B, a strong candidate for scenarios where absolute reliability is non-negotiable. The Llama 3 8B model achieved 94% accuracy, representing a viable, lower-cost alternative only when a minor performance trade-off is acceptable. Given the high stakes of this application, models achieving perfect accuracy remain the preferred choice, though these results underscore the module's flexibility in balancing reliability and cost.

These results highlight the feasibility and effectiveness of using capable LLMs as automated compliance monitors within the customer support workflow.

3.4 Interaction Workflows

The operational flow of interactions within the STELLAR architecture is governed by a Directed Acyclic Graph structure, which defines a finite set of possible pathways between modules. These predefined sequences ensure controlled execution and predictable behavior. Based on the defined module connections and branching logic, there are 12 distinct end-to-end workflows possible within the current STELLAR implementation, as illustrated in Figure 2. These are enumerated as follows:

1. $1 \rightarrow 2 \rightarrow 9 \rightarrow 8 \rightarrow 6$
2. $1 \rightarrow 2 \rightarrow 9 \rightarrow 8 \rightarrow 5 \rightarrow 4 \rightarrow 7 \rightarrow 6$
3. $1 \rightarrow 2 \rightarrow 9 \rightarrow 5 \rightarrow 4 \rightarrow 7 \rightarrow 6$
4. $1 \rightarrow 3 \rightarrow 9 \rightarrow 8 \rightarrow 6$
5. $1 \rightarrow 3 \rightarrow 9 \rightarrow 8 \rightarrow 2 \rightarrow 9 \rightarrow 8 \rightarrow 6$
6. $1 \rightarrow 3 \rightarrow 9 \rightarrow 8 \rightarrow 2 \rightarrow 9 \rightarrow 8 \rightarrow 5 \rightarrow 4 \rightarrow 7 \rightarrow 6$
7. $1 \rightarrow 3 \rightarrow 9 \rightarrow 8 \rightarrow 2 \rightarrow 9 \rightarrow 5 \rightarrow 4 \rightarrow 7 \rightarrow 6$
8. $1 \rightarrow 3 \rightarrow 9 \rightarrow 2 \rightarrow 9 \rightarrow 8 \rightarrow 6$
9. $1 \rightarrow 3 \rightarrow 9 \rightarrow 2 \rightarrow 9 \rightarrow 8 \rightarrow 5 \rightarrow 4 \rightarrow 7 \rightarrow 6$
10. $1 \rightarrow 3 \rightarrow 9 \rightarrow 2 \rightarrow 9 \rightarrow 5 \rightarrow 4 \rightarrow 7 \rightarrow 6$
11. $1 \rightarrow 5 \rightarrow 4 \rightarrow 6$
12. $1 \rightarrow$ end

The logic driving the navigation through these workflows is determined by the outputs of key decision-making modules, primarily Modules 1, 9, and 8:

- **Module 1 (Initial Routing)**:
  Every interaction begins here. Based on its classification of the initial user query, it selects the primary pathway. Category 0 directs the flow towards Module 2 (FAQ Pathway), Category 1 towards Module 3 (Direct Information Pathway), and Category 2 towards Module 5 and then Module 4 (Direct Human Escalation Pathway, workflow 11). Category 3 (Irrelevant) terminates the interaction early (workflow 12).
- **Module 9 (Compliance Check)**:
  Positioned after the generative modules (2 and 3), Module 9 acts as a quality and safety gate. If a generated response is deemed 'Compliant' (compliance: True), the workflow proceeds (to Module 8). If 'Non-Compliant' (compliance: False), the system attempts one retry of the preceding generative module (this retry logic is embedded within the transition but not explicitly shown as separate nodes in the high-level workflows). If the response remains non-compliant after the retry, Module 9 triggers a fallback:
  - Following Module 3 failure: Reroutes to Module 2 (e.g., initiating workflow 8: $1 \rightarrow 3 \rightarrow 9 \rightarrow 2 \rightarrow$ ...);
  - Following Module 2 failure: Escalates to human support (e.g., initiating workflow 3: $1 \rightarrow 2 \rightarrow 9 \rightarrow 5 \rightarrow$ ...). This path ensures that queries failing automated resolution via RAG are captured for potential knowledge base improvement via Module 7.
- **Module 8 (User Verification)**: This module introduces user feedback directly into the control flow after a compliant response has been generated by Module 2 or 3 (and passed Module 9). If the user confirms resolution (output: True), the interaction proceeds to Module 6 (Feedback Collection), completing workflows such as 1 and 4. If the user indicates the issue persists (output: False), a fallback logic similar to Module 9's failure path is invoked:
  - Following Module 3 path: Reroutes to Module 2 (e.g., initiating workflow 5: $1 \rightarrow 3 \rightarrow 9 \rightarrow 8 \rightarrow 2 \rightarrow$ ...), attempting resolution via RAG;
  - Following Module 2 path: Escalates to human support (e.g., initiating workflow 2: $1 \rightarrow 2 \rightarrow 9 \rightarrow 8 \rightarrow 5 \rightarrow$ ...), again ensuring Module 7 is eventually triggered.

These branching points explain the variations in the workflow list. For example, workflow 5 represents a scenario where Direct Information (Module 3) initially seemed appropriate but failed user validation (Module 8), leading to a second attempt via FAQ Retrieval (Module 2). Workflows involving Module 4 and subsequently Module 7 represent paths where automated resolution failed either initially, after compliance checks, or after user validation, necessitating human intervention and simultaneously flagging a potential knowledge gap. All defined workflows ultimately conclude with Module 6 (Feedback Collector) after resolution confirmation or after the human escalation process is logged.

This explicit definition of a finite set of interaction workflows, enabled by the DAG structure and the controlled decision logic at Modules 1, 8, and 9, is a core principle of the STELLAR architecture. It imbues the system with a high degree of predictability and manageability. Unlike systems with more dynamic or unrestricted inter-component communication, STELLAR's behavior can be precisely mapped and understood. This structure significantly simplifies traceability for debugging, auditing purposes, and performance analysis, as the exact sequence of module executions for any given interaction is constrained to one of these 12 paths.

4 Monitoring, Evaluation, and Improvement Strategies for STELLAR

While comprehensive, comparative benchmarks against alternative architectures represent important future work, this section outlines practical methodologies for the ongoing monitoring, evaluation, and iterative improvement of a deployed STELLAR system. A key advantage of STELLAR's modular DAG architecture is its inherent suitability for granular monitoring and targeted optimization, leveraging the detailed logs generated during operation.

A typical interaction log captures crucial data points, including the full chat history, the sequence of agents executed (mapping directly to one of the 12 defined workflows), the final state summarizing key outcomes, and detailed execution logs with per-module output, timestamp, and execution time. This rich data enables both real-time monitoring and in-depth offline analysis.

4.1 Defining Operational Success

In an operational context, evaluating the architecture's success involves monitoring key indicators derivable from interaction logs, reflecting effectiveness, efficiency, and safety:

- **Resolution Effectiveness**: A high rate of positive confirmations (response: True) from Module 8 indicates the system is frequently meeting user needs before potential escalation. Consistently positive feedback ratings and sentiment captured by Module 6 further corroborate user satisfaction.
- **Compliance Adherence**: A very low rate of non-compliant responses flagged by Module 9 (response.compliance: False) demonstrates the effectiveness of the safety layer.
- **Appropriate Automation**: A high percentage of interactions successfully resolved via automated paths (e.g., workflows 1, 4, 5, 8) without unexpected escalations suggests efficient handling of common queries.

4.2 Leveraging Built-in Monitoring Points & KPIs

The structured nature of STELLAR allows for granular performance monitoring by tracking the outputs and behavior of individual modules, treating them as Key Performance Indicators (KPIs):

- **Module 1 (Classifier)**: Monitor the distribution of classifications. A significant shift might indicate changing user query patterns or issues with the classifier prompt/model. Periodic spot-checking against chat history can assess classification accuracy.
- **Modules 2 & 3 (Generative/Retrieval)**: Track the failure rates leading to Module 9 flags or Module 8 user rejection. High rates pinpoint issues in RAG relevance, Direct Information retrieval accuracy, or generation quality.
- **Module 4 (Escalation)**: Monitor the overall escalation rate (frequency of Module 4 in sequence of agents). Analyze the distribution of urgency score to understand workload drivers. Track agent assignment success and wait time.
- **Module 5 (Sentiment)**: Analyze overall sentiment trends over time, potentially segmented by query category or insurance type, to measure user emotional response.
- **Module 6 (Feedback)**: Track average feedback ratings and the distribution of categorized comments. Segmenting by query category or workflow path can reveal specific pain points.
- **Module 7 (Knowledge Builder)**: Monitor the acceptance rate of draft FAQs by human reviewers. Track the number of new FAQs successfully added to Module 2's knowledge base per time period (e.g., weekly).
- **Module 9 (Compliance)**: Track the overall rate of compliance flags and the distribution of violation types to identify recurring safety or quality issues in generated content.
- **Execution Time**: Analyze the latency for each module and overall workflows to identify performance bottlenecks.

4.3 A/B Testing Potential

The modular design of STELLAR is well-suited for controlled experimentation in a live environment. Different versions of specific modules (e.g., testing a new LLM for Module 1, evaluating a different embedding model in Module 2, trying alternative prompt strategies for Module 9) can be deployed to a subset of traffic, allowing for data-driven decisions based on comparing relevant KPIs between the control and variant groups.

By employing these ongoing monitoring, evaluation, and improvement strategies, organizations utilizing the STEL-

LAR blueprint can ensure its continued effectiveness, reliability, and alignment with evolving user needs and business requirements.

4.4 Operational Considerations: Cost, Latency, and Model Dependency

A critical aspect of STELLAR's viability as a practical blueprint is its operational feasibility, including computational cost and model dependency. While our primary evaluations centered on the high-performance Llama 3.1 70B model to establish a quality baseline, the STELLAR architecture is fundamentally model-agnostic, allowing for the substitution of different LLMs to balance cost, latency, and performance.

Our experiments highlighted this flexibility. As noted in Subsection 3.3.1, initial prompt engineering for Module 1 was effectively performed using a smaller Llama 3.2 3B model. For critical Module 9 (Compliance Checker), further tests revealed that the Gemma2 9B model achieved identical 100% accuracy to the 70B model. This demonstrates that good results in the experiments are attainable without relying solely on large-scale models.

To provide a concrete cost analysis, we estimated the operational cost per 1,000 requests based on the average token usage across the 12 defined workflows (approximately 4,886 input and 882 output tokens per interaction) and pricing from a representative API provider (Groq). The estimated costs are presented in Table 3. Although our experiments used Llama 3.1 70B, Groq currently offers only Llama 3.3 70B [Meta, 2024d], which achieved slightly better results in follow-up tests, maintaining the validity of our analysis. The results indicate that the use of smaller models can reduce operational costs by more than an order of magnitude compared to the 70B model, allowing organizations to select models that better align with their operational budget and risk tolerance.

Beyond cost, latency is a decisive factor for user experience. To assess end-to-end user response time, we measured the API latency across 50 random, complete workflow executions using the Llama 3.1 70B model. The median response time per API call was 0.81 seconds, a result that is within the acceptable range for real-time conversational applications. Although overall latency is low, per module latency varies; for example, the RAG process in Module 2 can take between 2.8 and 5.2s (Table 2) and represents the most computationally intensive step. However, since the DAG structure typically executes only one such intensive module per user turn, the overall interaction remains responsive.

5 Conclusion

This section presents some final considerations and suggestions for future work.

5.1 Final Considerations

Deploying Large Language Models (LLMs) effectively and responsibly for automated customer support requires architectures that address inherent challenges in consistency, trustworthiness, and explainability. While standalone LLM applications struggle with complex, multi-step tasks and reliability, highly flexible agentic frameworks can introduce unpredictability. In this context, there is a clear need for structured, dependable solutions.

This paper introduced STELLAR (Structured, Trustworthy, and Explainable LLM-Led Architecture for Reliable Customer Support), a novel architectural blueprint designed specifically for the demands of intelligent customer support. Fundamentally grounded in a Directed Acyclic Graph (DAG) structure comprising nine specialized modules and eleven predefined workflows, STELLAR offers a systematic approach to building robust support systems. By enforcing controlled execution pathways, comprising many specialized modules, the architecture significantly enhances predictability, reliability, and traceability compared to less constrained approaches.

Key contributions include the multi-pathway design that optimizes resource allocation and human agent focus, the integration of innovative techniques such as semi-automatic knowledge base improvement (Module 7) and context-aware user verification (Module 8), and the embedding of crucial safety checks (Module 9). STELLAR strikes a deliberate balance, harnessing the power of LLMs for sophisticated tasks while ensuring operational robustness, practicality through human-in-the-loop integration, and adherence to ethical considerations. As demonstrated through module-specific evaluations, the components effectively perform their designated roles, contributing to the overall viability of the architecture. We acknowledge that the validation in this study was scoped to the insurance sector, primarily utilizing publicly available data and synthetically generated test cases. This approach, while effective for demonstrating the architecture's design and functional integrity, means that further validation is required to confirm its generalizability. Testing STELLAR's performance with real-world, unstructured user data and across different commercial sectors, such as e-commerce or healthcare, remains a crucial next step. Overall, STELLAR provides a robust and practical blueprint for developing dependable next-generation intelligent customer support systems.

5.2 Future Work

Building upon the foundation laid by STELLAR, there are several avenues for future research and development:

- **System Enhancements and Optimization**:

Further work can focus on optimizing the performance (latency, computational cost) of individual modules, potentially exploring knowledge distillation to create smaller, specialized models for tasks such as compliance checking [Ghosh *et al*., 2024; ShieldGemma Team, 2024] (Module 9) or leveraging advanced caching strategies for Module 3. Enhancing module capabilities, such as incorporating more sophisticated retrieval and generation techniques in Module 2 or refining ambiguity handling and follow-up logic in Module 3, would also be beneficial. Investigating minor, controlled dynamic adjustments to workflow routing within the DAG structure, perhaps based on fine-grained context, could add flexibility without sacrificing core predictability, ultimately enabling STELLAR to better adapt to diverse operational demands.

Table 3. Estimated cost per 1 million tokens and per 1,000 requests for different LLMs.

Model	Input Cost per 1M Tokens (USD)	Output Cost per 1M Tokens (USD)	Cost per 1,000 Requests (USD)
Llama 3.1 8B	$0.05	$0.08	$0.31
Gemma2 9B	$0.20	$0.20	$1.15
Llama 3.3 70B	$0.59	$0.79	$3.58

- **Comprehensive Evaluation and Benchmarking**:

While this paper establishes the architecture and evaluates individual components, rigorous comparative studies are essential. Future work should involve evaluating individual modules on larger test sets as well as benchmarking the end-to-end STELLAR system [Lin and Zhang, 2023; Jia *et al*., 2024] against both standalone LLM approaches and leading agent orchestration frameworks (e.g., LangChain, CrewAI) using standardized customer support datasets. Crucially, this comparative analysis would focus on key system-level metrics such as resolution rate, average response time, and human escalation rate, providing quantitative evidence of STELLAR's effectiveness. Furthermore, deploying STELLAR in real-world pilot programs is crucial to gather data on actual user satisfaction, operational efficiency, long-term maintenance costs, and the effectiveness of the monitoring strategies outlined in Section 4.

- **Broadening Applicability and Scope**:

The STELLAR blueprint holds potential beyond the initial domain. Future research should explore adapting and evaluating the architecture for diverse customer support contexts (e.g., technical support, e-commerce, financial services) by customizing modules and knowledge bases. Implementing and testing the architecture for multi-modal interactions, particularly voice support through integration with speech-to-text and text-to-speech technologies, represents a significant next step.

By pursuing these avenues, the principles embodied in the STELLAR architecture can be further refined, validated, and extended, contributing to the advancement of reliable, effective, and trustworthy AI-driven customer support solutions.

Declarations

Authors' Contributions

Matheus F. Scatolin contributed to the conceptualization, methodology, software, data curation, and original draft preparation of this study. Scatolin and Helio Pedrini contributed to validation, formal analysis, investigation, project administration, and writing – review and editing. Pedrini supervised the research. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Funding

This research was funded by Semantix.

Availability of data and materials

The datasets and software generated and/or analyzed during the current study are available in the following GitHub repository: https://github.com/Matheus-F-Scatolin/STELLAR.

References

Bodonhelyi, Z., Recski, G., and Iantovics, L. B. (2024). User Intent Recognition and Satisfaction with Large Language Models: A User Study with ChatGPT. *arXiv preprint arXiv:2402.02136*. DOI: 10.48550/arxiv.2402.02136.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901. DOI: 10.48550/arxiv.2005.14165.

cardiffnlp (2021). twitter-roberta-base-sentiment-latest. Available at:`https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest`.

Chase, H. (2022). LangChain. Available at:`https://github.com/langchain-ai/langchain`.

Chen, H., Liu, X., Yin, D., and Tang, J. (2018). A Survey on Dialogue Systems: Recent Advances and New Frontiers. *arXiv preprint arXiv:1711.01731v3*. DOI: 10.48550/arxiv.1711.01731.

Dong, H., Huang, Z., Xu, Y., Zhang, Y., and Yu, Y. (2025). ProTOD: Proactive Task-Oriented Dialogue System Based on Large Language Model. In *30th International Conference on Computational Linguistics (COLING)*, pages 9147–9164. Available at:`https://aclanthology.org/2025.coling-main.614`.

Fu, Y. and Feng, C. (2023). GPTCache: An Open-Source Semantic Cache for LLM Applications. In *3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS)*, pages 212–218. Available at:`https://aclanthology.org/2023.nlposs-1.24`.

Ghosh, D., Xiang, Z., Singh, M., and Choi, Y. (2024). Aegis: Online Adaptive AI Content Safety Moderation with Ensemble of LLM Experts. *arXiv preprint arXiv:2404.05993*. DOI: 10.48550/arxiv.2404.05993.

Gim, I., Chen, G., Lee, S.-S., Sarda, N., Khandelwal, A., and Zhong, L. (2024). Prompt Cache: Modular Attention Reuse for Low-Latency Inference. *arXiv preprint arXiv:2311.04934v2*. DOI: 10.48550/arxiv.2311.04934.

Google (2024). Gemma-2-9b-it. Available at:`https://huggingface.co/google/gemma-2-9b-it`.

Hudeček, O. and Dušek, O. (2023). Are Large Language Models All You Need for Task-Oriented Dialogue? In *24th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 216–228. DOI: 10.18653/v1/2023.sigdial-1.21.

Jia, R., Arora, S., Lee, H., Khaliq, B., Kwak, H., and Yang, D. (2024). Leveraging LLMs for Dialogue Quality Measurement. https://arxiv.org/abs/2406.17304. arXiv preprint arXiv:2406.17304.

Jiang, Z., Xu, F. F., Gao, L., Sun, Z., Dou, Q., Bing, L., Lin, R. W., and Han, S. (2023). Active Retrieval Augmented Generation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7969–7992. DOI: 10.18653/v1/2023.emnlp-main.495.

Joko, S., Sakai, T., Wu, D., and Joho, H. (2024). Doing Personal LAPS: LLM-Augmented Dialogue Construction for Personalized Multi-Session Conversational Search. In *47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 796–806. ACM. DOI: 10.1145/3626772.3657815.

Li, A., Gong, B., Yang, B., Shan, B., Liu, C., *et al*. (2025). MiniMax-01: Scaling Foundation Models with Lightning Attention. *arXiv preprint arXiv:2501.08313v1*. DOI: 10.48550/arxiv.2501.08313.

Lin, Z. and Zhang, Y. (2023). LLM-EVAL: Unified Multi-Dimensional Automatic Evaluation for Open-Domain Conversations with LLMs. In *5th Workshop on NLP for Conversational AI (NLP4ConvAI)*, pages 47–58. Available at:https://aclanthology.org/2023.nlp4convai-1.5.

Meta (2024a). Llama-3.2-3B. Available at:https://huggingface.co/meta-llama/Llama-3.2-3B.

Meta (2024b). Meta-Llama-3-8B-Instruct. Available at:https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct.

Meta (2024c). Meta-Llama-3.1-70B-Instruct. Available at:https://huggingface.co/meta-llama/Meta-Llama-3.1-70B-Instruct.

Meta (2024d). Meta-Llama-3.3-70B-Instruct. Available at:https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct.

Moura, J. (2023). CrewAI. Available at:https://github.com/joaomdmoura/crewAI.

Nehring, J., Augenstein, L., Gregor, B., Heinzerling, B., and Kern, R. (2024). Dynamic Prompting: Large Language Models for Task-Oriented Dialog. In *9th Italian Conference on Computational Linguistics (CLiC-it)*, pages 1–10. Available at:https://aclanthology.org/2024.clicit-1.72.

Rayo, J., Vila, J., Klinaku, S., and Schmidt, A. (2025). A Hybrid Approach to Information Retrieval and Answer Generation for Regulatory Texts. *arXiv preprint arXiv:2502.16767*. DOI: 10.48550/arxiv.2502.16767.

sentence-transformers (2020a). all-MiniLM-L12-v2. Available at:https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2.

sentence-transformers (2020b). all-MiniLM-L6-v2. Available at:https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.

sentence-transformers (2020c). multi-qa-MiniLM-L6-cos-v1. Available at:https://huggingface.co/sentence-transformers/multi-qa-MiniLM-L6-cos-v1.

sentence-transformers (2021). all-mpnet-base-v2. Available at:https://huggingface.co/sentence-transformers/all-mpnet-base-v2.

Shao, Z., Ren, Y., Sun, H., Wu, H., and Li, X. (2023). Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy. In *Findings of the Association for Computational Linguistics (EMNLP)*, pages 9248–9274. DOI: 10.18653/v1/2023.findings-emnlp.620.

Shavit, Y., Agarwal, S., Brundage, M., Adler, S., O'Keefe, C., Campbell, R., Kjellsson, H., Button, T., Sastry, G., Kokotajlo, D., Saunders, W., Knight, R., Schulman, J., and Solaiman, I. (2023). Practices for Governing Agentic AI Systems. Available at:https://openai.com/index/practices-for-governing-agentic-ai-systems/.

ShieldGemma Team (2024). ShieldGemma: Generative AI Content Moderation Based on Gemma. *arXiv preprint arXiv:2407.21772*. Available at:https://arxiv.org/abs/2407.21772.

Sreekar, K., Ashok, A., Lalwani, J., Rajanala, S., Joty, S., Lyzinski, V., Elazar, Y., and Potti, N. (2024). AXCEL: Automated eXplainable Consistency Evaluation using LLMs. In *Findings of the Association for Computational Linguistics (EMNLP)*, pages 14943–14957. DOI: 10.18653/v1/2024.findings-emnlp.878.

Sumers, T. R., Yao, S., Narasimhan, K., and Griffiths, T. L. (2024). Cognitive Architectures for Language Agents. *arXiv preprint arXiv:2309.02427v3*. DOI: 10.48550/arxiv.2309.02427.

Thoppilan, R., De Freitas, D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H. T., Chowdhery, A., Ichter, B., Le, Q. V., Salkin, R., Gao, L., Narang, S., Fiedel, N., Dean, J., and Roberts, A. (2022). LaMDA: Language Models for Dialog Applications. *arXiv preprint arXiv:2201.08239*. DOI: 10.48550/arxiv.2201.08239.

Wang, X., Wang, Z., Gao, X., Zhang, F., Wu, Y., Xu, Z., Li, Z., Zhang, W., Yuan, Z., Li, Z., Zhang, H., Li, H., Liu, Z., and Sun, M. (2024). Searching for Best Practices in Retrieval-Augmented Generation. *arXiv preprint arXiv:2407.01219v1*. DOI: 10.18653/v1/2024.emnlp-main.981.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. (2023). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv preprint arXiv:2201.11903v6*. Available at:https://arxiv.org/abs/2201.11903.

Weizenbaum, J. (1966). ELIZA – A Computer Program for the Study of Natural Language Communication Between Man and Machine. *Communications of the ACM*, 9(1):36–45. DOI: 10.1145/357980.357991.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Zhang, S., Zhu, E., Li, B., Jiang, L., Zhang, X., and Wang, C. (2023). AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework. *arXiv preprint arXiv:2308.08155*. Available at:https://arxiv.org/

abs/2308.08155.

Wulf, J. and Meierhofer, J. (2024). Utilizing Large Language Models for Automating Technical Customer Support. *arXiv preprint arXiv:2406.01407*. DOI: 10.48550/arxiv.2406.01407.

Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., Zheng, R., Fan, A., Wang, H., Gui, T., Zhang, Q., Wang, F., Zhang, B., Wang, Z., Zhao, H., Liu, S., Li, Z., Yuan, J., Wu, L., Liu, Z., Sun, M., and Zhang, Y. (2023). The Rise and Potential of Large Language Model Based Agents: A Survey. *arXiv preprint arXiv:2309.07864v3*. DOI: 10.1007/s11432-024-4222-0.

Xu, Z., Jain, S., and Kankanhalli, M. (2024). Hallucination is Inevitable: An Innate Limitation of Large Language Models. *arXiv preprint arXiv:2401.11817v2*. DOI: 10.48550/arxiv.2401.11817.

Zhang, W. and Zhang, J. (2023). Hallucination Mitigation for Retrieval-Augmented Large Language Models: A Review. *Mathematics*, 13(5):856. DOI: 10.3390/math13050856.