Supporting human operators during customer service interactions with agentic-RAG

Juan Barrionuevo-Valenzuela[1], Daniel Calderón-González[1], Zoraida Callejas[1,2], David Griol[1,2]
[1]Dept. of Software Engineering, University of Granada, Granada, Spain
[2]Center for Information and Communications Technologies (CITIC-UGR), Granada, Spain
Correspondence: {zoraida, dgriol}@ugr.es

Abstract

This paper focuses on improving customer service in call centers, where finding accurate answers in the shortest possible time is crucial. The proposed solution is the development of a conversational AI system that acts as a "copilot" for human operators. The main goal of this copilot is to assist the operator in real time by providing conversation summaries, relevant domain information, and suggested responses that help guide the interaction toward a successful resolution. To achieve this, different approaches to Retrieval Augmented Generation (RAG) have been explored.

The proposed *agentic-RAG* architecture integrates multiple autonomous agents for routing, retrieval validation, and response generation, achieving consistent improvements in real-time performance, grounding, and overall user experience across diverse service scenarios. Empirical results with the Action-Based Conversations Dataset (ABCD) corpus show that the use of agents proved to be effective in handling unstructured conversational data. The proposed approach showed an improvement in the quality, relevance, and accuracy of the generated responses with respect to a *naïve* RAG baseline. It is important to emphasize that this system is not intended to replace the operator, but rather to act as a support tool to enhance efficiency and customer satisfaction.

1 Introduction

Customer-service call centers constitute a primary interface between companies and their users; operational efficiency in this setting directly affects perceived service quality and customer satisfaction (Pacella et al., 2024). Previous evidence highlights that first-call resolution is the most decisive driver of satisfaction: even longer wait times may be tolerated if the interaction with the agent is effective and empathetic (Chicu et al., 2019). At the same time, human operators are required to combine domain knowledge with socio-emotional skills in dynamic dialogs, a combination that can increase stress and degrade both well-being and service outcomes (Pacella et al., 2024).

Real-time AI integrations in call centers have reported measurable improvements, including reductions in average handle time and a higher probability of first-call resolution, suggesting benefits for productivity, customer outcomes, and the operator experience (Gudipati, 2025). Nevertheless, users often prefer human-supported service over fully automated bots and frequently report frustration with chatbot-only experiences (Zhang et al., 2024).

This work proposes a conversational *copilot* that assists the operator during the interaction. The system follows the conversation in real time, summarizes relevant information, retrieves domain knowledge, and suggests responses that help guide the dialog toward resolution. A central design choice is to keep the human–human interaction while using AI to increase effectiveness, aligning with evidence that discourages full automation in this context (Wirtz et al., 2018).

Technically, we build on RAG to reduce hallucinations (Lewis et al., 2021). We explore and compare RAG configurations, including graph-based and agentic variants, aimed at handling unstructured conversational data and multi-topic queries (Brühl, 2024; Wulf and Meierhofer, 2024). The resulting pipeline combines careful corpus preparation (anonymization and formatting) with specialized retrievers and agents that iteratively assess, refine, and route context before producing final suggestions for the operator.

For development and evaluation, we use ABCD, an open corpus with more than 10,000 human–human dialogs organized into 10 topics and 55 subtopics (Chen et al., 2021). We design an evaluation that contrasts the proposed system against a naïve RAG baseline, using both manual annotation and an LLM-based evaluator. The results indicate

Proceedings of the 16th International Workshop on Spoken Dialogue Systems Technology, pages 348–356
February 26–March 1, 2026.

consistent improvements in the *quality*, *relevance*, and *accuracy* of suggestions, with particularly pronounced gains under ambiguous and multi-topic scenarios.

The paper makes four main contributions: (i) a real-time *conversational copilot* for call centers that increases agent productivity rather than replacing human agents; (ii) an *agentic-RAG* pipeline tailored to unstructured dialog data; (iii) a *data preparation* process (anonymization and structuring) that improves retrieval and generation; and (iv) a *comparative evaluation* demonstrating benefits over a naïve RAG baseline on practical utility metrics.

The rest of the paper is organized as follows: Section 2 reviews related work and the technical background on RAG in call-center contexts. Section 3 details datasets, architecture, and prompts. Section 4 presents results and discussion. Finally, Section 5 concludes and outlines future work.

2 Related Work

Prior research in customer-service settings could be split into two avenues: (i) *replacing* the human operator with autonomous systems and (ii) *optimizing* the operator's work to improve productivity and response quality. Representative efforts on the replacement side include institutional help desks that combine fine-tuned LLMs with RAG to improve precision and availability (Ordóñez-Camacho et al., 2024), as well as field experiments where voice-based AI substitutes interactive voice response (IVR) for simple issues (Wang et al., 2023). In contrast, work focused on optimizing the operator's work targets assistive functions such as real-time call summarization and guidance (Sachdeva et al., 2023).

In customer-service copilots, responses must be grounded in task knowledge and the conversation history, that is the reason why RAG is used in many systems to ground responses in trusted evidence. The *naïve* RAG pipeline comprises indexing (chunking and embedding documents into a vector store), retrieval (semantic search and, increasingly, hybrid search that combines lexical and vector queries), and generation (conditioning the LLM on the retrieved context) (Gao et al., 2024). "Advanced" and "modular" RAG variants add intermediate stages (e.g., re-ranking and query transformation) to mitigate recall/precision gaps observed in the naïve setup (Bianchini, 2025).

This paper explores two types of RAG: Graph RAG and Agentic RAG. *Graph RAG* captures entity and relation structure to improve disambiguation and context selection in multi-topic dialogs; practical implementations extract a knowledge graph with an LLM, map query entities to graph nodes, expand neighborhoods, and feed the adapted subgraph to the generator. Despite its benefits, graph-based pipelines entail nontrivial costs (graph construction/maintenance, traversal, and coherence between structured facts and generated text), which can limit scalability in noisy domains (Brühl, 2024; Bianchini, 2025). *Agentic RAG* uses autonomous agents to plan, decide when and how to retrieve, invoke tools, and iteratively self-evaluate and refine outputs, making it possible to exploit short-/long-term memory for multi-step reasoning and reduce error propagation (Wulf and Meierhofer, 2024; Singh et al., 2025)

Within this landscape, our approach aligns with the *optimization* trajectory and builds on RAG variants suited to unstructured and multi-topic customer-service dialogs. Thus, we emphasize techniques that (i) preserve human control, (ii) improve retrieval fidelity under domain constraints, and (iii) support iterative refinement toward actionable, operator-facing suggestions.

3 Proposal

The methodology proposed to generate the copilot system to assist operators includes the following steps:

1. **Corpus selection:** Choosing a dataset of customer service conversations is critical to ensure system effectiveness. Human-to-human conversations are preferred, as they reflect the unpredictability of real scenarios.

2. **Data preparation:** Structuring the corpus is essential for a retrieval-augmented generation (RAG) system to efficiently utilize the information.

3. **RAG system development:** The LLM acts as a copilot, generating contextually relevant responses based on similar conversations from the corpus. The generation process is explained in the following sections.

4. **Evaluation:** Finally, assessment measures whether the proposed system enhances existing approaches.

3.1 Data and corpus preparation

Two customer-service human–human dialog datasets were analyzed for development. First, the *LUNA Corpus* (Italian, 60 dialogs annotated with Penn Discourse Treebank (PDTB)), which required transforming its JSON files (tokens and turn groups) into LLM-readable text. Although the script reconstructed turns and reduced tokens, the format introduced ambiguities among developers who were also not fluent in Italian, so it was discarded as the main basis.

The system was ultimately built on the ABCD corpus (over 10,000 acted dialogs, 10 topics and 55 subtopics). Starting from the original JSON, conversations were exported to a topic-based folder hierarchy (`account_access`, `manage_account`, ..., `troubleshoot_site`), creating ten vector databases, one per topic. Before indexing, key preprocessing was applied: removal of emoticons and systematic anonymization to prevent leakage of sensitive data *and* LLM confusions (e.g., "recovering" a user from another conversation). Twelve distinct types of sensitive entities were considered, each requiring anonymization through specific utilities. The anonymization pipeline used (i) *regex* for cards, passwords, ZIP/PIN codes, emails, phones, order/account IDs (with a strict application order to resolve overlaps), (ii) *Named Entity Recognition (NER)* (*bert-large-NER*) for names and addresses with a confidence threshold, and (iii) targeted manual replacements (e.g., security-question answers or certain usernames). This pipeline enabled indexing of anonymized and structured text by topic, improving retrieval and reducing the risk of inappropriate responses.

3.2 Selection of the RAG approach

Two strategies were evaluated. Graph RAG was tested first, using Neo4j and Memgraph as property-graph engines. For *LUNA*, entities and relations were defined (e.g., *Device, Problem, Troubleshooting Action, Verification/Diagnosis, Solution*), but with unstructured dialogs the graph proved inconsistent (orphan nodes, collisions among anonymized entities, low repeatability, and high construction/traversal cost). In the ABCD corpus, despite conducting multiple experiments, various problems were encountered in the generation of knowledge graphs when the information was presented in the form of conversation, which is a very unstructured form of knowledge with relevant information dispersed across turns, thus not guaranteeing good graph generation.

Consequently, *Agentic RAG* was adopted for its flexibility with unstructured data: agents that *plan*, *retrieve* dynamically, *assess* relevance, and *refine* queries and outputs iteratively. This architecture was implemented with LangChain and LangGraph and tailored to the ABCD domain with 11 agents (10 topics + `off_topic`).

3.3 System architecture

As a proof-of-concept, we developed a real-time *copilot* integrated into a client–operator chat web application in the ABCD domain.

An overview of the main components and their interactions is illustrated in Figure 1, which outlines the overall system architecture, including the backend, frontend, and supporting services.

3.3.1 Backend

Responsibilities were separated into two servers: Flask-SocketIO for real-time messaging (WebSocket) and a Worker for the Agentic RAG pipeline. The communication between them is asynchronous via Redis (pub/sub pattern in Docker), preventing blocking and enabling scalability, making the system fault tolerant. Each server operates independently, maintaining a loosely coupled architecture.

3.3.2 LLM

Gemini 1.5 Flash was used in the initial experiments for its latency and availability; when the provider announced its deprecation, the system migrated to Gemini 2.5 Flash while keeping the architecture and prompts unchanged.

Following this transition, the overall responsiveness and quality of the system's outputs improved, with faster inference times and more coherent responses observed across tasks.

3.3.3 Agentic RAG pipeline

The proposed pipeline comprises the following components:

1. **Topic router** (11 domains: the 10 from ABCD + `off_topic`) classifies the latest turn and selects the appropriate topic's *retriever* and *vector DB*.

2. **Retrieval** of k fragments from the selected topic (with separate per-topic stores to avoid *cross-topic mixing*).

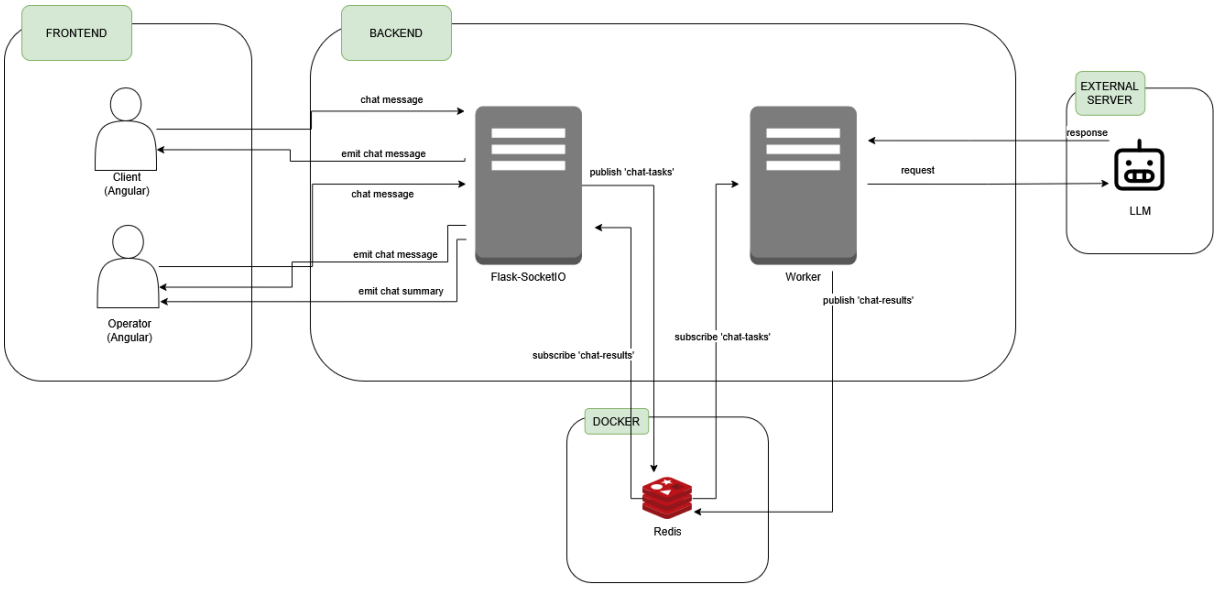

Figure 1: System architecture diagram

3. **Relevance evaluator** checks whether the retrieved documents are useful for the current intent. *If the evaluation is affirmative*, continues to step 5. *If negative*, triggers step 4.

4. **Query rewriting** (clarifying entities/objective utilizing an LLM) and re-routing; then repeats retrieval and validation.

5. **Suggestion generator** produces several candidate replies for the operator, strictly related to the retrieved context. If the case is `off_topic`, the system skips RAG and guides the operator to query for useful information.

The overall flow of the agentic RAG process is illustrated in Figure 2, which visualizes the interactions among the components described above.

3.3.4 Frontend

The *customer* sees a simple chat. The *operator* has: (i) the same chat, (ii) a panel with a summary, key entities (e.g., item/problem in ABCD scenarios) and detected business rules, and (iii) actionable suggestions that can be edited before sending.

Figure 3 illustrates the operator's user interface of the application. The left panel displays the ongoing conversation with the customer, while the right panel is divided into two sections.

The lower section displays a list of suggested responses generated for the current customer's message, which the operator can modify if necessary. This component presents the results of the the Agentic RAG system, operating independently from the real-time WebSocket chat.

The upper section provides supplementary information to assist the operator in managing the interaction, including related items, the identified issue topic, and a concise summary of the entire conversation. This system incorporates conversation memory, progressively refining and completing the displayed fields as the interaction develops and more information becomes available.

3.4 Prompt design

The *copilot*'s performance largely depends on sound *prompt engineering*. For the suggestion generator, the final prompt was obtained by iteratively refining an initial draft, and including sections that enforce the desired behavior: (i) role (contextualizes how it should act), (ii) task (what to produce and from which context), (iii) hard constraints (prevent hallucinations and require grounding in the retrieved documents), (iv) output format (ready-to-send lines), (v) internal process (number of candidates, checklist, and diverse selection), and (vi) final output.

Below are the specific aspects considered for the prompts defined for each agent and the rationale behind their design:

- **Suggestion generator (operator).** Assigns a "call-center assistant" role and restricts the output to ready-to-send phrases, strictly grounded in the retrieved documents. The *internal process* specifies: drafting 5–10 candidates; run-

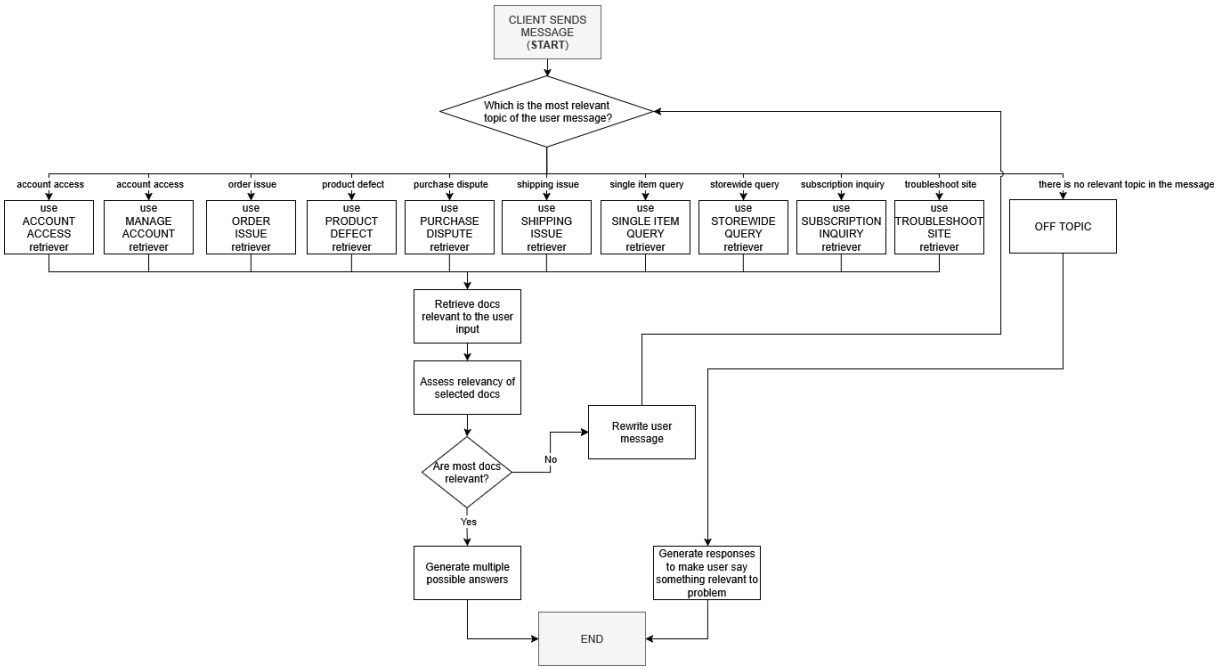

Figure 2: Flowchart of the Agentic RAG structure

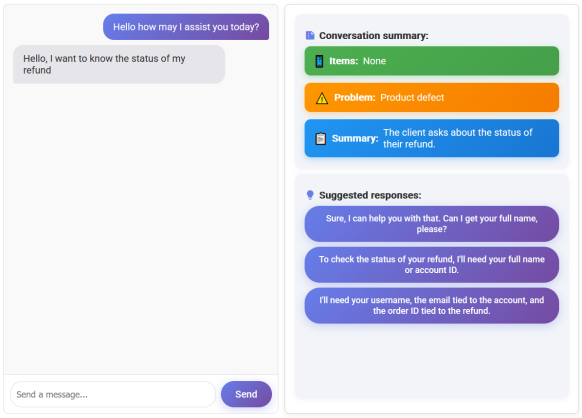

Figure 3: Operator user interface

ning a checklist (grounding to context, usefulness, clarity, and non-duplication); and returning 3–8 suggestions that cover different angles (clarifying question, step-by-step guidance, confirmation, next action). If the incident is resolved, the output must be exactly `Issue resolved`. Additionally, it employs few-shot prompting to ensure that the generated responses are fully aligned with the desired output.

- **Topic router.** Provides clear definitions of the 10 ABCD domains with operational descriptions to classify the query (`off_topic` when it does not fit). It decides which vectorial database will be retrieved based on the user input.

- **Relevance evaluator.** Reuses those descriptions and applies binary rules (yes/no) to decide whether the retrieved snippets help with the current intent. If the answer is no, it triggers *query rewriting* to clarify entities/objective and repeat retrieval and validation.

- **`off_topic` assistant.** When the query does not fit the ten topics, it guides the user with brief questions to steer toward a useful domain, maintaining the same output format and without adding text outside the available context.

This agent-based design separated responsibilities (routing, context validation, generation) and improved consistency: the router prevents cross-topic mixing; the evaluator filters unhelpful retrievals; and the generator emits actionable suggestions consistent with the retrieved material and the conversation state.

3.5 Practical considerations

A backend server was used for real-time message management utilizing WebSocket, and another for the RAG logic, which eliminated possible workload bottlenecks and improved latency, stability, and scalability. The use of Redis with Docker

simplified development and communication between both servers. The ten per-topic vector stores combined with the router ensured contextual consistency (e.g., avoiding retrieval from “order issue” when the intent was “account access”). Prior anonymization was critical to prevent inappropriate responses (e.g., “remembering” a user from another conversation) and to protect privacy.

4 Results

This section presents the results obtained from applying the developed agentic RAG copilot, utilizing the anonymized conversational data, and discusses the quality and relevance of the generated responses. The methodology involved comparing the proposed Agentic RAG system against a baseline normal RAG architecture.

4.1 Evaluation Design

The evaluation employed a test battery consisting of 58 messages derived from the ABCD corpus. This battery included one conversation per subtheme (55 total) and three additional queries categorized as "off-topic". A specific challenge addressed during testing involved ambiguous queries designed to confuse the retriever, such as a user asking about a forgotten password while simultaneously mentioning an order status, which could misleadingly direct the retrieval toward "order issue" instead of the correct "account access" domain.

4.2 RAG System Results

The implementation of the RAG system developed, as described in detail in the design phase, was rigorously compared against a simpler naïve RAG baseline (referred to as "normal RAG"). Both systems were standardized using the same final prompt, embedding models (multilingual-e5-base), chunking strategy, top-k (k=5), and the Gemini-2.5-flash LLM. The evaluation utilized a test battery comprising 58 total queries, covering 55 distinct sub-topics and three off-topic scenarios, resulting in the analysis of 521 generated responses.

The quality of the generated responses was categorized manually into three classes: inappropriate, too general, or appropriate and useful. Responses were deemed appropriate if they successfully solicited the necessary data from the customer to resolve the issue or provided a coherent, complete answer (e.g., specific product details); and too general if it moves the dialog forward but entails a longer, less direct interaction. Conversely, responses were labeled as inappropriate if they were repetitive or failed to contribute meaningfully to the customer’s problem resolution.

The manual assessment demonstrated that the proposed RAG system achieved a higher average percentage of appropriate responses and a corresponding reduction in inappropriate responses across the majority of conversational themes compared to the normal RAG. This superior performance confirms the value of incorporating steps to select the appropriate domain topic and evaluate the relevance of the retrieved documents before generation.

Category	Inappropriate	Too General	Appropriate
account_access	0.35	0.12	0.53
manage_account	0.41	0.05	0.54
order_issue	0.50	0.09	0.41
product_defect	0.38	0.07	0.55
purchase_dispute	0.28	0.10	0.62
shipping_issue	0.47	0.00	0.53
single_item_query	0.29	0.07	0.64
storewide_query	0.19	0.25	0.56
subscription_inquiry	0.41	0.11	0.48
troubleshoot_site	0.24	0.00	0.76
off_topic	0.82	0.00	0.18

Table 1: Manual evaluation of responses in each topic for a standard RAG system.

Category	Inappropriate	Too General	Appropriate
account_access	0.22	0.11	0.67
manage_account	0.28	0.03	0.69
order_issue	0.25	0.17	0.58
product_defect	0.31	0.10	0.59
purchase_dispute	0.26	0.11	0.63
shipping_issue	0.30	0.00	0.70
single_item_query	0.21	0.14	0.64
storewide_query	0.21	0.47	0.32
subscription_inquiry	0.48	0.04	0.48
troubleshoot_site	0.06	0.00	0.94
off_topic	0.00	0.00	1.00

Table 2: Manual evaluation of responses in each topic for the proposed system.

In addition to qualitative evaluation, a secondary assessment was conducted using a separate LLM designated as an evaluator, which applied the same quality criteria. This LLM-based evaluation generally aligned with the human assessment. While the normal RAG exhibited highly variable performance, showing occasional high scores in specific domains (e.g., "troubleshoot site"), the proposed system achieved significantly greater consistency and stability, maintaining useful response rates typically above 60% across categories.

Customer service centers are under pressure to achieve efficient resolution of customer inquiries, ideally during the first interaction. To address this challenge, we propose an approach to develop copilots based on agentic RAG based to support human operators, fostering performance and productivity,

Category	Inappropriate	Too General	Appropriate
account_access	0.00	0.29	0.71
manage_account	0.28	0.33	0.38
order_issue	0.03	0.50	0.47
product_defect	0.41	0.21	0.38
purchase_dispute	0.13	0.18	0.69
shipping_issue	0.16	0.32	0.53
single_item_query	0.00	0.14	0.86
storewide_query	0.06	0.25	0.69
subscription_inquiry	0.26	0.30	0.44
troubleshoot_site	0.00	0.18	0.82
off_topic	0.36	0.27	0.36

Table 3: Evaluation with an LLM evaluator of responses in each topic for a standard RAG system.

Category	Inappropriate	Too General	Appropriate
account_access	0.11	0.39	0.50
manage_account	0.28	0.21	0.52
order_issue	0.11	0.28	0.61
product_defect	0.03	0.31	0.66
purchase_dispute	0.03	0.29	0.68
shipping_issue	0.15	0.40	0.45
single_item_query	0.07	0.29	0.64
storewide_query	0.00	0.42	0.58
subscription_inquiry	0.26	0.37	0.37
troubleshoot_site	0.11	0.11	0.78
off_topic	0.00	0.15	0.85

Table 4: Evaluation with an LLM evaluator of responses in each topic for the proposed system.

rather than replacing them.

Globally, the proposed system demonstrated better overall results. For instance, a critical advantage of the proposed architecture, which uses an agent router to classify the query topic, is the assurance of retrieval consistency. When the proposed system retrieved documents, they belonged exclusively to the classified topic, whereas the normal RAG often retrieved irrelevant documents, such as mixing "order issue" documents with "account access" queries concerning password retrieval. Furthermore, in the "off topic" category, the proposed system achieved a 100% appropriate rate by correctly determining that RAG was unnecessary, instead focusing on effective user redirection, resulting in a much higher quality outcome than the baseline. In summary, the evaluation validates that the Agentic RAG architecture provides enhanced consistency, stability, and higher overall quality compared to a naïve RAG approach.

5 Conclusions and Future Work

Throughout the development process, a complete workflow was designed for collecting, preprocessing, and exploiting call center dialog data. The preprocessing phase proved crucial, as it ensured that LLMs could correctly interpret conversational context. The applied anonymization and formatting strategies significantly improved the relevance and coherence of the model‘s responses.

Two RAG approaches were explored to identify the most suitable framework for this domain. Although Graph RAG initially appeared promising, its performance was limited by the unstructured and dynamic nature of conversational data. In contrast, Agentic RAG delivered superior adaptability and quality, effectively handling the contextual and semantic complexity of dialog-based inputs.

Finally, a functional web application was developed to demonstrate the proposed system. This prototype enables real-time interactions between two simulated users, a customer and an operator, where the operator benefits from an integrated copilot assistant. The system identifies and summarizes key information, applies business rules, and generates multiple response options to assist the human agent in decision-making.

The system was used to demonstrate the feasibility of our proposal and to evaluate it in comparison with a naïve RAG approach, showing positive results in terms of response appropriateness in a practical use case with the ABCD corpus, which contains conversations across 11 topics under the umbrella of customer service.

Future research will focus on optimizing the Agentic RAG workflow through improved agent coordination, advanced prompt engineering, and slot-filling mechanisms to strengthen contextual consistency. Further evaluation with a larger number of human operators will enable a more comprehensive assessment of the system’s scalability and adaptability. Moreover, we plan to incorporate additional call center data obtained from real-world settings within the CRYSTAL research project.

6 Acknowledgments

The research described in this paper was supported by the TrustBoot Research Project (ref. PID2023-150584OB-C22 and PID2023-150584OB-C21), financed by MICIU/AEI/10.13039/501100011033 and FEDER; and the European Union’s Horizon Europe MSCA Staff Exchanges CRYSTAL project (grant agreement No. 101182965).

References

Filippo Bianchini. 2025. Retrieval-Augmented Generation. In Francesca De Luzi, Flavia Monti, and Massimo Mecella, editors, *Engineering Information Systems with Large Language Models*, pages 139–172. Springer Nature Switzerland.

Volker Brühl. 2024. Generative Artificial Intelligence

– Foundations, Use Cases and Economic Potential. *Intereconomics*, 59(1):5–9.

Derek Chen, Howard Chen, Yi Yang, Alexander Lin, and Zhou Yu. 2021. Action-Based Conversations Dataset: A Corpus for Building More In-Depth Task-Oriented Dialogue Systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3002–3017, Online.

Dorina Chicu, Maria del Mar Pàmies, Gerard Ryan, and Christine Cross. 2019. Exploring the influence of the human factor on customer satisfaction in call centres. *BRQ Business Research Quarterly*, 22(2):83–95.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv preprint*. 2312.10997.

Satya Karteek Gudipati. 2025. Transforming the Working Style of Call Center Agents Through Generative AI. *International Journal of Advanced Computer Science and Applications*, 16(6):9–15.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *arXiv preprint*. 2005.11401.

Diego Ordóñez-Camacho, Rafael Melgarejo-Heredia, Mohsen Abbasi, and Lucía González-Solis. 2024. Aurel_ai: Automating an Institutional Help Desk Using an LLM Chatbot. In *Proceedings of the 28th World Multi-Conference on Systemics, Cybernetics and Informatics*, pages 81–84, Orlando, Florida, USA.

Massimo Pacella, Paride Vasco, Gabriele Papadia, and Vincenzo Giliberti. 2024. An Assessment of Digitalization Techniques in Contact Centers and Their Impact on Agent Performance and Well-Being. *Sustainability*, 16(2):1–19.

Aashraya Sachdeva, Sai Nishanth Padala, Anup Pattnaik, Varun Nathan, Cijo George, and Jithendra Vepa. 2023. Tailored Real-Time Call Summarization System for Contact Centers. In *Proceedings of Interspeech*, pages 5261–5262, Dublin, Ireland.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG. *arXiv preprint*. 2501.09136.

Lingli Wang, Ni Huang, Yili Hong, Luning Liu, Xunhua Guo, and Guoqing Chen. 2023. Voice-based AI in call center customer service: A natural field experiment. *Production and Operations Management*, 32(4):1002–1018.

Jochen Wirtz, Paul G. Patterson, Werner H. Kunz, Thorsten Gruber, Vinh Nhat Lu, Stefanie Paluch, and Antje Martins. 2018. Brave new world: service robots in the frontline. *Journal of Service Management*, 29(5):907–931.

Jochen Wulf and Jürg Meierhofer. 2024. Utilizing Large Language Models for Automating Technical Customer Support. *arXiv preprint*. 2406.01407.

Ruby Wenjiao Zhang, Xiaoning Liang, and Szu-Hsin Wu. 2024. When chatbots fail: exploring user coping following a chatbots-induced service failure. *Information Technology & People*, 37(8):175–195.

A Appendix

The final prompt used is:

```
You are a friendly assistant that acts as an
assistant in a call center, helping the
operator by providing relevant information
from past conversations that may be useful
in the current conversation.

TASK
- Provide several different phrases the
operator could say to help the customer resolve
their issue, grounded strictly in the context
of previous conversations (documents provided).

HARD CONSTRAINTS
- You cannot invent responses not related to
the documents.
- Do not add any explanation, notes, or extra
text beyond the responses.
- The responses must be written exactly as the
operator could say them.
- Before each response, add the symbol ">".

OUTPUT FORMAT
- Only output lines in the form:
  > Response 1...
  > Response 2...
  > Response 3...
- If the issue has been resolved, output exactly:
  > Issue resolved

INTERNAL PROCESS
(SILENT — DO NOT OUTPUT ANY OF THIS)
- Deliberate privately about user intent and which
snippets from past conversations are relevant.
- Draft 5 to 10 candidate operator phrases tailored
to the current context.
- Run this checklist and remove any candidate that
fails:
  [ ] Grounded in the provided documents
  (no external facts).
 [ ] Actionable/helpful next step for the customer.
 [ ] Clear, concise, and friendly tone suitable for
  an operator.
  [ ] Non-duplicative phrasing across suggestions.
- Perform a private verification pass:
  · Confirm each remaining phrase aligns with the
  customer's stated problem or with elicitation
  of missing info when out-of-domain.
  · If fewer than 3 valid suggestions remain,
  generate more candidates and re-check.
```

- Select the best diverse set (3 to 8) that covers different helpful angles (clarifying question, step-by-step instruction, confirmation, next-step offer).

FINAL OUTPUT
- Print only the selected suggestions, each on its own line starting with ">" as specified above.
- Do not reveal your internal reasoning or this checklist.