EnterpriseBench: Simulating Enterprise Environments for Testing and Evaluating LLM-based Agents

Anonymous ACL submission

Abstract

Enterprise systems are crucial for enhancing productivity and decision-making among employees and customers. Integrating LLM based systems into enterprise systems enables intelligent automation, personalized experiences, and efficient information retrieval, driving operational efficiency and strategic growth. However, developing and evaluating such systems is challenging due to the inherent complexity of enterprise environments, where data is fragmented across multiple sources and governed by sophisticated access controls. We present EnterpriseBench, a comprehensive benchmark that simulates realistic enterprise settings, featuring 550 diverse tasks across software engineering, HR, finance, and administrative domains. Our benchmark uniquely captures key enterprise characteristics including data source fragmentation, access control hierarchies, and cross-functional workflows. Additionally, we provide a novel data generation pipeline that creates internally consistent enterprise datasets from organizational metadata. Experiments with state-of-the-art LLM agents demonstrate that even the most capable models achieve only 21.5% task completion, highlighting significant opportunities for improvement in enterprise-focused AI systems. `Anonymous version of our code / dataset:` `EnterpriseBench`

1 Background and Introduction

Large Language Models (LLMs) are fundamentally transforming how enterprises operate, driving improvements in productivity across departments (Plumb, 2025; Meta, 2024; Carlini, 2024). These models have demonstrated remarkable capabilities in automating knowledge-intensive tasks, from question answering and code generation to report writing and data analysis (Brachman et al., 2024; Jiang et al., 2024a; GitHub, 2024). Recent advancements have led to emergence of Compound AI Systems (CAI) (Zaharia et al., 2024; Lin et al., 2024) (also referred to as Agents (LangChain, 2024; Anthropic, 2024)) that can orchestrate complex workflows for solving various tasks. These systems, exemplified by tools like Devin (Labs, 2024) and Glean (Glean), can automatically search across information sources, analyze data, and even initiate actions when human intervention is needed.

However, developing effective CAI systems for enterprises faces a critical challenge: enterprise data is inherently complex and fragmented across multiple sources, including email systems, Customer Relationship Management (CRM) platforms, SharePoint sites, internal wikis, and ticketing systems. This fragmentation is further complicated by sophisticated access control mechanisms that govern who can access specific information resources. Even seemingly simple queries often require orchestrating data gathering from multiple sources, executing database calls, and performing complex reasoning across diverse information types. While current research has made progress in developing CAI systems for specific use-cases relevant to enterprises, the unique challenges of enterprise environments—particularly around data fragmentation and access control—remain largely unaddressed with current CAI systems.

The lack of suitable evaluation data for developing CAI systems specific to enterprises compounds this challenge. There are currently no public datasets that adequately capture the complexity of enterprise environments, primarily because real enterprise data is often proprietary and subject to strict privacy regulations. This data scarcity and lack of benchmarks significantly hampers the development and validation of enterprise-focused AI systems. Furthermore, enterprises seeking to prototype and evaluate AI agents for their specific needs face a chicken-and-egg problem: they need to test agents on realistic enterprise scenarios, but cannot use their actual data during the initial exploration and development phases.

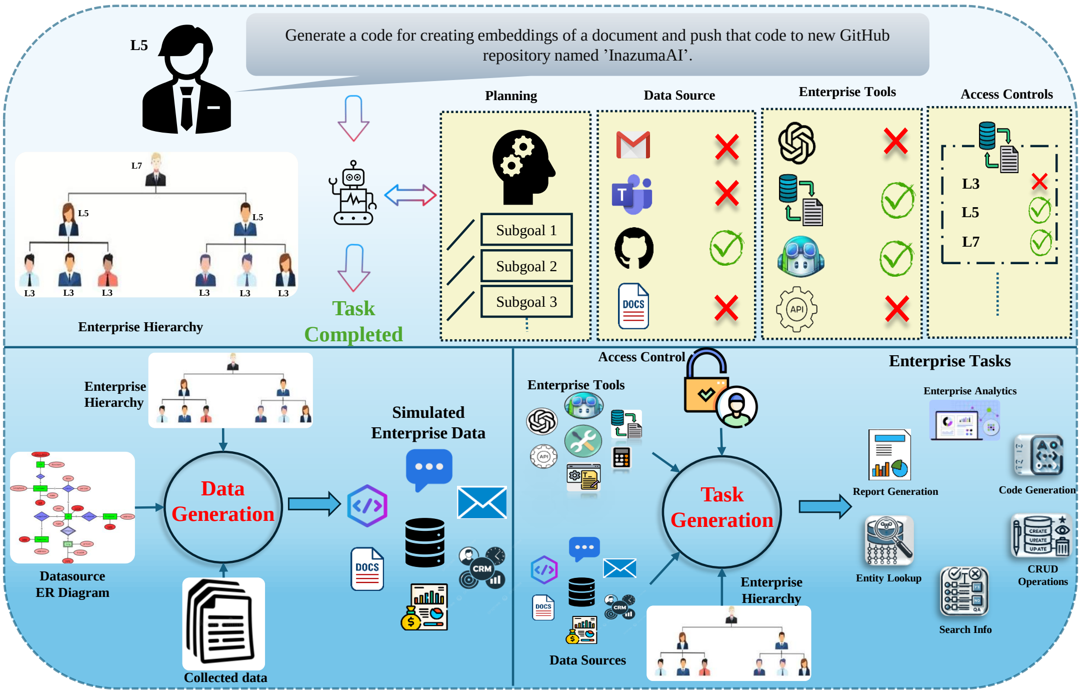


Figure 1: **Overview of EnterpriseBench framework showing the interplay between data generation and task evaluation components.** The framework consists of two main cycles: (1) Data Generation, which combines collected data, ER diagrams, and enterprise hierarchies to create simulated enterprise data, and (2) Task Generation, which leverages this data along with enterprise tools and access controls to create realistic evaluation scenarios. The top panel demonstrates an example task execution where an L5 employee attempts to create a GitHub repository, showing how access controls and tool availability influence task completion.

Most of the existing benchmarks for developing CAI systems address only partial aspects of enterprise environments. WorkArena (Drouin et al.) and WorkArena++ (Boisvert et al., 2024) evaluate the performance of web agents on knowledge work tasks. OSWorld (Xie et al., 2024) and Windows Agent Arena (Bonatti et al.) focus on open-ended computer-based tasks on popular Operating Systems. Agent Company (Xu et al., 2024a), simulates tasks commonly seen in small software companies but does not fully capture or focus on enterprise data fragmentation and access control hierarchies. SWE-Bench (Jimenez et al.) and DevBench (Li et al., 2024) focus solely on software engineering tasks. We present a detailed comparison of existing works in Section A.3 of Appendix

To illustrate challenges and complexities of the CAI, consider an enterprise specific scenario: an employee asks, *"Can I apply for a week's leave in December without overlapping project deadlines?"* This seemingly straightforward request requires a complex workflow that traditional approaches like Retrieval-Augmented Generation (RAG) (Bruckhaus, 2024) and existing LLM agents (Talebirad and Nadiri, 2023; Zhang et al.; Li et al., 2019) struggle to handle. A robust enterprise-specific CAI system must orchestrate multiple subtasks for this: querying HR systems for leave balances, checking project management tools for deadlines, and cross-referencing team calendars for conflicts—all while respecting access controls and organizational hierarchies. These requirements highlight the need for sophisticated CAI systems that can (1) integrate multiple enterprise data sources, (2) enforce access controls, (3) coordinate multiple tasks, and (4) maintain context across system interactions (as shown in Figure 1).

To enable development of such systems, we introduce EnterpriseBench, a comprehensive benchmark that simulates the data from real enterprise environments. By providing a rich, realistic dataset that mirrors complexities of real-world scenarios without using sensitive real data, EnterpriseBench enables rapid prototyping and evaluation of CAI systems for enterprise settings. This allows organizations to validate and refine their CAI systems before deploying them on actual enterprise data. Our dataset spans multiple domains, including Software Engineering (code repositories, documentation), Sales and CRM (customer interactions), Finance (budgets, expense reports), IT support (ticketing systems, incident reports), HR (policies, employee records), and Internal Communication platforms (simulated team and email conversations). EnterpriseBench emphasizes persona-based tasks that require adherence to access controls and organizational hierarchies. Additionally, we propose a novel synthetic data generation process that constructs realistic enterprise datasets using structured inputs such as employee directories, organizational hierarchies, data source descriptions, and access policies. This approach ensures internal consistency while reflecting real-world enterprise scenarios and relationships. Our key contributions are listed below.

- A comprehensive benchmark of 550 enterprise tasks across IT, HR, Sales and Finance, featuring multi-step reasoning, access controls, and cross-functional workflows.
- Our comprehensive evaluations shows a significant performance gap in current CAI systems, with even state-of-the-art models achieving only 21.5% task completion.
- A novel data generation pipeline that transforms organizational metadata into internally consistent datasets while preserving hierarchical relationships and access controls in an enterprise.
- A persona-based task framework that generates contextually appropriate challenges, testing both technical capabilities and organizational constraints.

2 EnterpriseBench: Crafting a Simulated Enterprise Benchmark

Developing a enterprise sandbox environment requires careful consideration of four key components: data sources, security layers, task frameworks, and dynamic operations. Building on the challenges outlined in Section 1, we present a systematic approach to creating a simulated enterprise environment that captures the complexity of real-world scenarios while enabling controlled experimentation.

2.1 Enterprise Data Foundation

2.1.1 Data Description

Our framework combines collected and synthetically generated data across multiple enterprise domains within our simulated organization, Inazuma.co. The data spans HR, IT, Sales, Finance, Management, and Software Development domains. This hybrid approach ensures both authenticity and comprehensive coverage of enterprise components, from Customer Relationship Management (CRM) systems to code repositories. To maintain real-world fidelity, we establish connections between disparate data sources—for example, Customer Support data incorporates both Customer profiles and Product Sentiment information. Table 7 provides detailed statistics of the simulated enterprise data across domains (refer Appendix A.4 for more details).

2.1.2 Data Development

Generating realistic inter-connected enterprise data using LLMs presents three key challenges: (1) Context adherence: LLMs may drift from provided specifications, affecting data fidelity (2) Terminology preservation: Critical domain-specific terms must be preserved to ensure alignment with the data source (3) Diversity: Generated data should respect semantic and contextual diversity, avoiding repetitive patterns.

To address these challenges, our data generation pipeline requires three key inputs: department-wise employee hierarchy, entity-relationship (ER) diagram, and collected reference data. The pipeline systematically constructs and validates these components to ensure data consistency and realism.

Employee Hierarchy Generation: We collect general organizational hierarchy information from the web and enrich it using LLMs to create level-wise distributions across departments, ensuring realistic descriptions aligned with organizational structures. Department-specific rules are defined through prompting the LLM and are manually verified. In the final output, employees are classified into levels (e.g., L8, L9), and rules are refined with LLMs to meet level-specific requirements. The employee hierarchy construction process is shown in Figure 2.

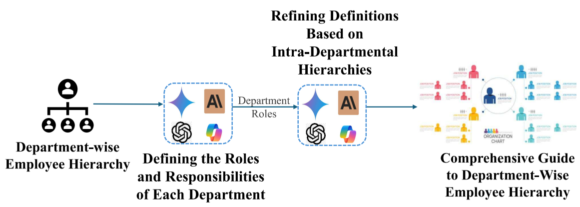


Figure 2: Employee Hierarchy Generation process demonstrating the transformation from basic organizational structure to detailed department-level roles.

ER Diagram Construction: Starting with human-annotated descriptions of data sources, we construct a comprehensive ER diagram mapping entities, attributes, and relationships. Expert knowledge and LLM assistance help define detailed attributes—for example, Employee entities include ID, name, email, position, department, and skills. The relationships are validated through human review to ensure proper primary and foreign key map-

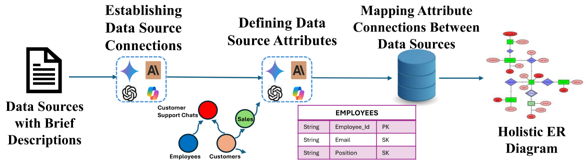


Figure 3: ER Diagram Construction pipeline illustrating the progression from raw data descriptions to a structured enterprise schema.

pings. This ER diagram serves as the blueprint for enterprise data structuring and database design (complete ER diagram in Appendix Figure 7a).

Data Source Generation Framework Building on (Xu et al., 2024b)'s approach to conversation dataset generation, we develop a four-stage pipeline for creating interconnected enterprise data sources (Figure 4):

(1) *Subject Generation*: The pipeline first generates context-appropriate subjects using employee roles and data source attributes. Employee roles determine subject categories—engineers discuss code deployment and system architecture, while HR personnel focus on policy and workplace culture. For interdepartmental communications, subjects bridge multiple domains, such as *"HR-Engineering: Joint Initiative on Data Security Training"*. (2) *Context QA Generation*: Using collected reference data and identified subjects, we generate domain-specific QA pairs that capture realistic enterprise interactions. For conversational data (emails, chats), subjects are mapped to appropriate departmental relationships (e.g., Customer-Support, HR-IT) based on the organizational hierarchy. Non-conversational sources like GitHub issues are generated independently. This helps the LLM preserve key terminology accurately when generating the final data. (3) *Semantic Clustering*: To ensure content diversity, we employ K-means clustering (Likas et al., 2003) on Sentence-BERT (Reimers, 2019) embeddings. This groups semantically similar questions, enabling us to filter redundant content while maintaining comprehensive domain coverage. Each cluster represents a distinct aspect of enterprise communication or documentation. (4) *Instance Generation*: The final stage transforms filtered questions into data instances using source-specific attributes. For example, email generation combines sender_id, recipient_id, and subject with the question context to create complete messages. Each instance undergoes LLM-based paraphrasing to introduce natural language variation while preserving essential information and context.

All prompts used for data generation are detailed in Appendix A.9.1.

2.2 Enterprise Security Layer

To mirror real enterprise environments, we implement a dynamic security layer that enforces role-based access controls based on organizational hierarchy. Access permissions are determined through a combination of roles classified by the level in the organization (L9-L14), tasks and data source sensitivity, and cross-departmental relationships, following the ER diagram (Figure 7a in the Appendix). For example, while enterprise social platforms are accessible to all employees (L9-L14), GitHub access is restricted to specific teams and their management chain. These rules are generated using LLM assistance and validated by humans to ensure realistic security constraints. Detailed access control specifications are provided in Appendix A.5.

2.3 Task Framework

2.3.1 Task Design Principles

Our benchmark comprises 550 enterprise tasks, each designed to evaluate CAI systems capabilities in enterprise scenarios. Tasks are structured around three key dimensions: (1) Employee personas and associated access controls (2) Tool usage (3) Expected outcomes and evaluation criteria. As shown in Figure 5b, tasks span four main categories: Search, CRUD (Create. Read, Update, and Delete) operations, Access Denied scenarios, and Unanswerable queries. Figure 5a shows the classification based on the output. Each task requires systematic execution through: Decomposing primary tasks, data source identification and appropriate tool selection. This division enables step-by-step evaluation of CAI systems.

The resource distribution (Figure 5c) demonstrates the multi-step nature of these tasks, with most requiring 2-4 distinct data sources for completion. Table 16 presents an example from each task category.

2.3.2 Task Generation Pipeline

Our task generation process (Figure 6) involves creating tasks that require access to mutltiple data sources and tools while adhering to persona-specific access controls and can be divided into five stages:

1. Dependency Path Selection: We employ Depth-First Search (DFS) on the ER diagram, randomly selecting a single path starting from the `Employee`

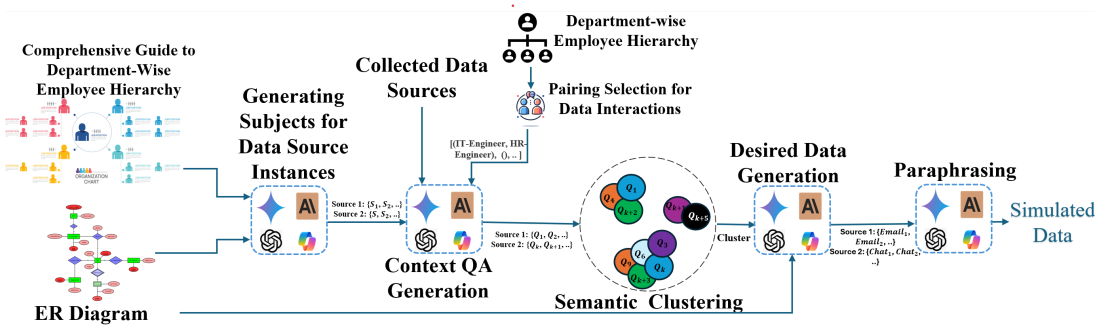


Figure 4: **Enterprise Data Source Generation Framework** demonstrating the end-to-end pipeline for creating realistic enterprise data.

node. Paths are constrained to length 1-5 to manage context complexity. Each data source along the path represents a distinct processing hop, following the traversal's chronological sequence.

2. Persona-Based Goal Generation: We select relevant personas based on the chosen data path and use LLMs to generate contextualized primary goals. Each goal decomposes into practical sub-goals—for example, "Improve our sales performance and customer relationships?" breaks down into "retrieve customer interaction history" and "analyze feedback trends."

3. Tool Integration: Following Zhuang et al. (2023), we map tools to sub-goals while enforcing access control policies (as outlined in Section 2.2). Each tool operation validates user permissions before execution, returning "Access Denied" when appropriate. A detailed description of the tool pool and its specifications are detailed in Appendix Table 8.

4. Template Generation: We extract entities from data sources and categorize them into head, torso, and tail groups by frequency. After sampling an entity type from this distribution and selecting a specific entity, we extract associated triples from the knowledge graph to build task templates, following Yang et al. (2024).

We construct the knowledge graph by extracting triples from data sources (Kertkeidkachorn and Ichise, 2017) and incorporate self-reflection (Ji et al., 2023) to enrich knowledge representation. This knowledge graph (Figure 13, Appendix A.6) provides triples that guide the LLM in generating persona-specific, tool-dependent task templates, ensuring greater generalizability and reusability.

5. Final Assembly: With our predefined triple-chunk mapping, we combine templates and entity-source mapping to construct tasks, assigning relevant data sources, tools, subgoals, and final answers for search queries.

Task generation prompts in Appendix A.9.2.

2.4 Dynamic Operations

To fully simulate enterprise environments, EnterpriseBench implements dynamic data management capabilities that reflect real-world organizational changes. These changes span employee turnover, project updates, customer interactions, and organizational restructuring, requiring continuous adaptation of the underlying data structures.

Central to this capability is an LLM-mediated system that manages CRUD operations across the enterprise data foundation. This system enables: (1) Real-time updates to employee roles and permissions. (2) Dynamic adjustment of access controls. (3) Maintenance of data relationships across sources

For instance, when an employee is promoted from Manager (L-12) to Director (L-14), the system automatically: Updates their role and responsibilities, Adjusts resource access permissions, Modifies related data dependencies.

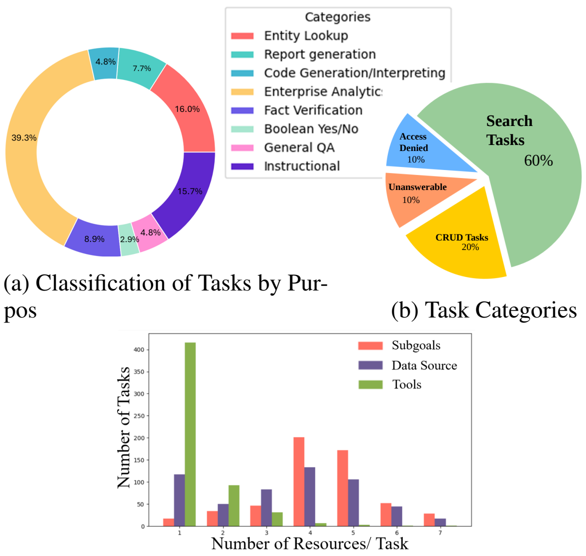
The LLM Agent processes these changes through natural language queries, while built-in control mechanisms ensure all operations adhere to established persona definitions and access control policies (examples in Figure 16 in Appendix).

Through this dynamic framework, EnterpriseBench provides a realistic testbed for evaluating LLM Agents' ability to handle evolving enterprise scenarios. Further implementation details can be found in Appendix A.5.

3 Experimental Setup

3.1 Enterprise LLM Agent Setup

To efficiently solve our enterprise search tasks, we design an LLM-based agent that follows a structured multi-step approach. Given a primary goal P,



(a) Classification of Tasks by Purpos

(b) Task Categories

(c) Resource-Based Task Distribution

Figure 5: Task Design Overview illustrating the multi-faceted nature of enterprise tasks through three key perspectives. (a) A detailed breakdown of tasks by their output / purpose (b) High-level categorization of tasks into four main types: Search, CRUD, Access Denied, and Unanswerable queries, (c) Distribution of resource requirements per task, comparing the number of subgoals, data sources, and tools needed for task completion.

the agent decomposes it into meaningful sub-goals $G = \{g_1, g_2, \ldots, g_n\}$ using a reasoning-based method. These sub-goals are then refined into well-defined, solvable steps $S = \{s_1, s_2, \ldots, s_n\}$. The agent, defined as $\mathcal{A} = f(\Theta, \mathcal{K})$, where Θ and $\mathcal{K}$ are model parameters and prior knowledge, selects the appropriate tools T and data sources D to optimize information retrieval and processing. It then iteratively solves each sub-goal, constructing the final answer A to the primary goal. The entire process is formulated as follows:

$$G = \text{decompose}(P; \Theta, \mathcal{K}); \tag{1}$$

$$S = \text{reason}(G; \Theta, \mathcal{K}); \tag{2}$$

$$(T, D) = \text{select}(S; \mathcal{T}, \mathcal{K}); \tag{3}$$

$$A = \text{execute}(T, D, S; \mathcal{A}). \tag{4}$$

This structured framework ensures reliable execution of enterprise search tasks by leveraging LLMs for multi-step reasoning, tool utilization, and precise information retrieval.

3.2 Experimental Settings

This section outlines our experimental setup, detailing the baseline methods used to evaluate our benchmark, the evaluation metrics employed, and the implementation specifics.

3.2.1 Baseline Methods

To evaluate state-of-the-art performance on the EnterpriseBench benchmark, we conducted experiments under two factors: *Resource Selection* and *Execution Accuracy*. These two factors are evaluated under two scenarios: *w/o Planning* and *w/ Planning*. In the *w/ Planning* scenario, we evaluate using following techniques: Chain-of-Thought (CoT) (Wei et al., 2022), ReAct (Yao et al., 2022b), and planning instructions(Sub-goals for that Primary goal) as input.

The system is built using the following LLMs: `GPT-4o`[1], `o1-mini`[2] (via Azure AI Foundry), and `Anthropic Claude 3.5-Sonnet`[3] (`anthropic.claude-3-5-sonnet-20240620-v1:0`) from Amazon Bedrock. The system role is to decompose primary goals into subgoals, selects relevant data sources and tools, verifies access controls, and executes tasks end-to-end. Our baseline methods are inspired by the SoTA approach in TPTU-v2 (Kong et al.) and the innovative solutions from Quantologic [4].

3.2.2 Implementation Details

Experiments were performed using two NVIDIA A30 GPUs (24GB each) and LLMs inference APIs.

- *Data Source Generation*: We utilized `GPT-4o`[1] to generate all components of EnterpriseBench, ensuring consistency and high-quality data synthesis, it took approximately 3 minutes and 30 seconds to generate a single data instance.
- *Task Generation*: The task generation process was conducted using `GPT-4o`[1], implementing an end-to-end pipeline. Additionally, `Anthropic Claude 3.5-Sonnet`[3] was employed for self-reflection, contextual reasoning, and final quality assessment of the generated tasks. It took approximately 2 minutes and 20 seconds to generate a single task.
- *Tool Dependency and Execution*: Tool dependencies were defined using a structured `JSON` file containing detailed descriptions of all tools within EnterpriseBench. For tool execution, API calls were made to invoke various external tools. Further details on tool specifications and implementations can be found in Table 8.
- *Data Source Retrieval*: We implemented hybrid retrievers (BM25 + Dense) (Chen et al., 2022; Ma et al., 2021) for text-based data, Colpali (Faysse et al., 2024) for PDF documents, and query-to-SQL retrievers inspired by (Zhang et al., 2025) for tabular content.

[1] https://platform.openai.com/docs/models#gpt-4o
[2] https://platform.openai.com/docs/models#o1
[3] https://aws.amazon.com/bedrock/claude/
[4] https://www.quantalogic.app/

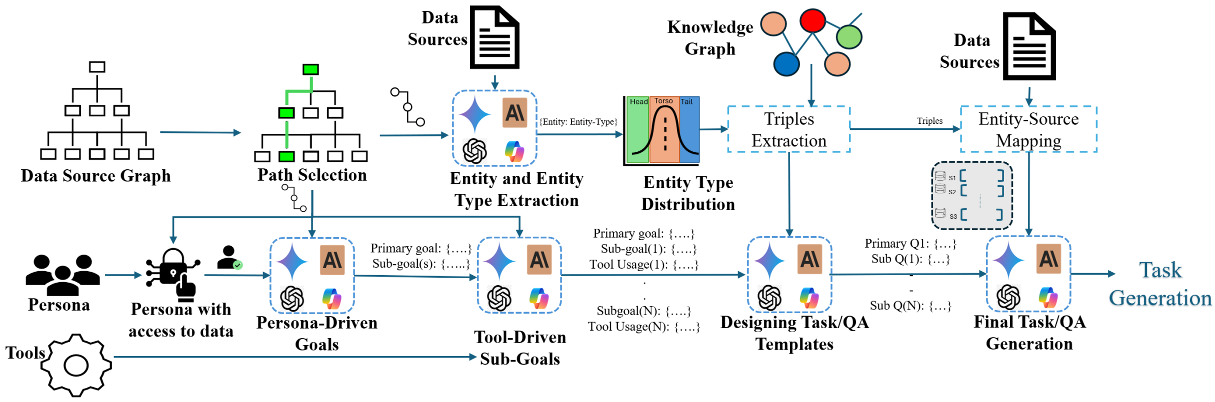


Figure 6: **Enterprise Task Generation Pipelines** demonstrating the end-to-end process of creating realistic enterprise tasks.

3.3 Evaluation Metric

To systematically evaluate Compound AI systems, we define a stepwise scoring metric inspired by Xu et al. (2024a) that assesses key execution stages: Resource Selection (data source and tool selection) and Subgoal Execution (decomposition and execution). Based on scenarios described in Section 3.2.1 (detailed evaluation process in Appendix A.7), our metric penalizes incomplete or incorrect executions and enforces systematic flow—if a penultimate execution fails, subsequent stages are not executed, ensuring robust performance assessment.

Full Execution Score:

$$\text{Score}_{\text{full}} = \begin{cases} 1, & \text{if execution is fully correct} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Partial Execution Score:

$$\text{Score}_{\text{partial}} = \sum_{i=1}^{d \cdot \mathbb{P}} I_i \cdot W_i \cdot O_i \tag{6}$$

where each component is defined as:

$$W_i = \frac{1}{2^i}, \text{ penalty score for step } i \tag{7}$$

$$O_i = \text{ LLM judge score for step } i \tag{8}$$

$$\mathbb{P} = \begin{cases} 1, & \text{if planning} \\ 0, & \text{if no planning} \end{cases} \tag{9}$$

The flow vector **I** ensures consistent execution checks:

$$I_d = \begin{cases} 1, & \text{if } O_{\text{resource}} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

$$I_i = \begin{cases} 1, & \text{if } I_{i+1} = 1 \text{ and } O_i = 1 \\ 0, & \text{otherwise} \end{cases} \quad \forall i \in [1, d-1] \tag{11}$$

Depth (d) switches between planning and no-planning modes as shown in Figure 14b and Figure 14a. For more detailed evaluation procedures, refer to Appendix A.8.

Furthermore, the Human Evaluation Scores are obtained by averaging the scores from three annotators working in an X enterprise. We use a smaller subset of tasks for this analysis due to the resource-intensive nature of human evaluation.

4 Results and Analysis

In this section, we present the evaluation of our benchmark, EnterpriseBench, using three state-of-the-art reasoning models: GPT-4o, Claude-3.5-Sonnet, and o1-mini. We also provide a detailed analysis of EnterpriseBench Generation and Evaluation. Section 4.2 presents the error analysis from the benchmark evaluation, while Section 4.3 examines EnterpriseBench generation using the LLM-as-a-Judge approach (Zheng et al., 2024) for data source and task creation in enterprise simulation. Additionally, we perform human evaluation to assess task realism and conduct grounding tests (Tang et al., 2024) to verify the contextual accuracy of the generated data.

4.1 Evaluation on Enterprise Search Tasks

Table 1 presents the evaluation of our benchmark using three LLM models, each individually used to set up the LLM Agent for task execution. We assess performance in two settings—Resource (Tools + Data Source) Selection and Final Task Execution—across four scenarios: a) no plan, b) CoT, c) ReAct, and d) gold plan, using the aggregated metric (Section 3.3).

For resource selection, the LLM Agent must select the appropriate data sources and tools through reasoning. ReAct outperforms the no-plan approach,

Models	Resource Selection				Task Execution			
Methods	w/o Planning	CoT (Wei et al., 2022)	ReAct (Yao et al., 2022b)	w/ Gold Planning	w/o Planning	CoT (Wei et al., 2022)	ReAct (Yao et al., 2022b)	w/ Gold Planning
GPT-4o	28.43	38.28	41.03	65.61	8.82	11.15	14.28	36.81
Claude-3.5-Sonnet	45.20	45.70	46.13	67.54	10.56	9.88	15.73	42.26
o1-mini	45.03	40.10	45.34	66.01	11.34	10.42	21.53	40.37

Table 1: **EnterpriseBench Evaluation.** We evaluate our benchmark on Compound AI systems built using GPT-4o, Claude-3.5-Sonnet, and o1-mini for resource selection (data source and tool selection) and task execution. Evaluation is based on an aggregated metric (Equation 5, 6) for the following settings: *w/o Planning; CoT (Chain-of-Thought, step-by-step instruction to LLM); ReAct (Observation, Thought, Action); w/ Gold Planning (providing gold planning instructions to the LLM).*

	Task Execution (0/1)			
Methods	w/o Planning	CoT (Wei et al., 2022)	ReAct (Yao et al., 2022b)	w/ Gold Planning
GPT-4o	3.00	5.00	5.00	19.00
Claude-3.5-Sonnet	6.00	6.00	8.00	27.00
o1-mini	11.00	7.00	15.00	23.00

Table 2: **EnterpriseBench Human Evaluation.** We evaluate our benchmark using three human annotators. Each annotator checks whether the final task is executed correctly. If the task is completed correctly, the annotator assigns a score of 1; otherwise, the score is 0.

but the gold plan (decomposed tasks) achieves the highest accuracy, highlighting limitations in current reasoning models.

In Final Task Execution, performance drops significantly compared to resource selection. The best setup with ReAct achieves 21.53%, while providing all subgoals (gold plan) improves this to 42.26%, emphasizing the need for better decision-making and long-term reasoning in LLM Agents.

Table 2 further shows human-evaluated task execution results(aggregate of 3 annotators), with o1-mini achieving the accuracy: 23% with the gold plan and 15% with ReAct. Additionally, Figure 15a illustrates o1-mini's performance across different task categories.

We evaluated human CAI (humans acting as an LLM agent) in task execution, as shown in Figure 15b. While they achieve high accuracy, it comes at the cost of increased time, revealing a trade-off between precision and efficiency. The results also highlight the performance gap between an LLM Agent and human agents in task execution.

4.2 EnterpriseBench Evaluation Analysis

We analyzed 100 random samples for task execution using o1-mini ReAct and found errors in 85% of the tasks. Among them, 67% were due to LLM invocation issues, including subgoal decomposition, resource selection, and response generation, while 18% resulted from retrieval and tool execution failures. For a detailed analysis, refer to Section A.1 in the Appendix.

4.3 EnterpriseBench Analysis

Our benchmark creation involves two parts: Enterprise Simulated Data Creation and Enterprise Tasks Creation. For data creation, we conducted grounding and realism tests to assess contextual consistency and compare generated data with human-curated data (detailed in Section A.2.1 in the Appendix). For task creation, we evaluated realism, performed detailed error analysis to identify inaccuracies, and conducted human evaluation to verify task authenticity (detailed in Section A.2.2 in the Appendix).

Findings: a) Our results show that grounding tests scored 60-80% across Roberta and MiniCheck models. b) LLM-as-a-judge rated 80-90% human-likeness for Claude-3.5-Sonnet and o1-mini, with a 75% agreement rate. c) For tasks, we achieved 80% human-likeness using LLM-as-a-judge and 67% realism in human evaluation. d) Task Creation error analysis of 100 samples revealed 23 incorrect tasks, categorized into entity-persona alignment, KG faults, and LLM generation faults.

5 Conclusion

In this paper, we highlight the importance of Compound AI Systems in enterprise settings and the need for a benchmark to evaluate their performance. To address this, we introduce EnterpriseBench, a novel benchmark designed to assess CAI systems on complex enterprise tasks. Our experiments show that even state-of-the-art models face significant challenges with these tasks. To create a realistic evaluation environment, we also propose a simulated enterprise data generation pipeline and an enterprise task framework, enabling the construction of comprehensive benchmarks with minimal input requirements.

Limitations

The limitations of our work are as follows: 1) Our enterprise data generation process requires an initial set of real enterprise data, which can be costly to obtain. Relying solely on synthetic data may affect the realism of generated tasks. 2) Human experts are needed to verify intermediate steps during task generation, adding to the complexity and cost. 3) While we achieve high accuracy in enterprise task generation, some errors remain, suggesting areas for future improvement. 4) The evaluation of our benchmark relies on the current capabilities of reasoning models, which are likely to improve over time. 5) Integration with real enterprise tools like MS Teams and interface-based frameworks was not achieved due to permission constraints. 6) Our experiments did not involve large-scale data generation with terabytes of data, which would better represent real-world enterprise-scale scenarios.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2024. Building effective agents. https://www.anthropic.com/research/building-effective-agents.

Navid Ayoobi, Sadat Shahriar, and Arjun Mukherjee. 2023. The looming threat of fake and llm-generated linkedin profiles: Challenges and opportunities for detection and prevention. In *Proceedings of the 34th ACM Conference on Hypertext and Social Media*, pages 1–10.

Daniil A Boiko, Robert MacKnight, and Gabe Gomes. 2023. Emergent autonomous scientific research capabilities of large language models. *arXiv preprint arXiv:2304.05332*.

Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault Le Sellier De Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, and Alexandre Drouin. 2024. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Rogerio Bonatti, Dan Zhao, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Keunho Jang, et al. Windows agent arena: Evaluating multi-modal os agents at scale. In *NeurIPS 2024 Workshop on Open-World Agents*.

Michelle Brachman, Amina El-Ashry, Casey Dugan, and Werner Geyer. 2024. How knowledge workers use and want to use llms in an enterprise context. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–8.

Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. 2023. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, pages 287–318. PMLR.

Tilmann Bruckhaus. 2024. Rag does not work for enterprises. *arXiv preprint arXiv:2406.04369*.

Nicholas Carlini. 2024. How i use "ai"? https://nicholas.carlini.com/writing/2024/how-i-use-ai.html.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Xilun Chen, Kushal Lakhotia, Barlas Oguz, Anchit Gupta, Patrick Lewis, Stan Peshterliev, Yashar Mehdad, Sonal Gupta, and Wen-tau Yih. 2022. Salient phrase aware dense retrieval: Can a dense retriever imitate a sparse one? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 250–262.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, et al. Workarena: How capable are web agents at solving common knowledge work tasks? In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2024. Colpali: Efficient document retrieval with vision language models. *arXiv preprint arXiv:2407.01449*.

Tiantian Feng, Digbalay Bose, Tuo Zhang, Rajat Hebbar, Anil Ramakrishna, Rahul Gupta, Mi Zhang, Salman Avestimehr, and Shrikanth Narayanan. 2023. Fedmultimodal: A benchmark for multimodal federated learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4035–4045.

Chen Gao, Xiaochong Lan, Zhihong Lu, Jinzhu Mao, Jinghua Piao, Huandong Wang, Depeng Jin, and Yong Li. 2023. S3: Social-network simulation system with large language model-empowered agents. *arXiv preprint arXiv:2307.14984*.

Alireza Ghafarollahi and Markus J Buehler. 2024. Protagents: protein discovery via large language model multi-agent collaborations combining physics and machine learning. *Digital Discovery*.

GitHub. 2024. Github copilot: Your ai pair programmer. Accessed: Feb. 11, 2025.

Glean. Glean: Work ai for all. Accessed: February 11, 2025.

Karthik Gopalakrishnan, Behnam Hedayatnia, Qinlang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tür. 2019. Topical-Chat: Towards Knowledge-Grounded Open-Domain Conversations. In *Proc. Interspeech 2019*, pages 1891–1895.

Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. 2023. Towards mitigating llm hallucination via self reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1827–1843.

Feihu Jiang, Chuan Qin, Kaichun Yao, Chuyu Fang, Fuzhen Zhuang, Hengshu Zhu, and Hui Xiong. 2024a. Enhancing question answering for enterprise knowledge bases using large language models. In *International Conference on Database Systems for Advanced Applications*, pages 273–290. Springer.

Yucheng Jiang, Yijia Shao, Dekun Ma, Sina Semnani, and Monica Lam. 2024b. Into the unknown unknowns: Engaged human learning through participation in language model agent conversations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9917–9955, Miami, Florida, USA. Association for Computational Linguistics.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*.

Natthawut Kertkeidkachorn and Ryutaro Ichise. 2017. T2kg: An end-to-end system for creating knowledge graph from unstructured text. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*.

Vivek Khetan, Roshni Ramnani, Mayuresh Anand, Shubhashis Sengupta, and Andrew E Fano. 2020. Causal bert: Language models for causality detection between events expressed in text. *arXiv preprint arXiv:2012.05453*.

Yilun Kong, Jingqing Ruan, YiHong Chen, Bin Zhang, Tianpeng Bao, Hangyu Mao, Ziyue Li, Xingyu Zeng, Rui Zhao, Xueqian Wang, et al. Tptu-v2: Boosting task planning and tool usage of large language model-based agents in real-world systems. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Cognition Labs. 2024. Introducing devin, the first ai software engineer. Accessed: February 11, 2025.

LangChain. 2024. What is an ai agent? `https://blog.langchain.dev/what-is-an-agent/`.

Bowen Li, Wenhan Wu, Ziwei Tang, Lin Shi, John Yang, Jinyang Li, Shunyu Yao, Chen Qian, Binyuan Hui, Qicheng Zhang, et al. 2024. Devbench: A comprehensive benchmark for software development. *arXiv preprint arXiv:2403.08604*.

Nian Li, Chen Gao, Yong Li, and Qingmin Liao. 2023. Large language model-empowered agents for simulating macroeconomic activities. *Available at SSRN 4606937*.

Xu Li, Mingming Sun, and Ping Li. 2019. Multi-agent discussion mechanism for natural language generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6096–6103.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *Transactions on Machine Learning Research*.

Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. 2003. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461.

Matthieu Lin, Jenny Sheng, Andrew Zhao, Shenzhi Wang, Yang Yue, Yiran Wu, Huan Liu, Jun Liu, Gao Huang, and Yong-Jin Liu. 2024. Llm-based optimization of compound ai systems: A survey. *arXiv preprint arXiv:2410.16392*.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations*.

Xing Han Lù, Zdenek Kasner, and Siva Reddy. Weblinx: real-world website navigation with multi-turn dialogue (2024). *URL https://arxiv. org/abs/2402.05930*, 3(8).

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2024. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, pages 1–11.

Xueguang Ma, Kai Sun, Ronak Pradeep, and Jimmy Lin. 2021. A replication study of dense passage retriever. *arXiv preprint arXiv:2104.05740*.

Meta. 2024. Large language models: Transforming the future of work. https://forwork.meta.com/blog/how-large-language-models-are-changing-the-future-of-work/.

Taryn Plumb. 2025. Here's how google is using llms for complex internal code migrations. https://www.infoworld.com/article/3804552/heres-how-google-is-using-llms-for-complex-internal-code-migrations.html.

N Reimers. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Milad Shokouhi and Luo Si. 2011. Federated search". foundations and trends in information retrieval (ftir). *Foundations and Trends in Information Retrieval*.

Yashar Talebirad and Amirhossein Nadiri. 2023. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *arXiv preprint arXiv:2306.03314*.

Liyan Tang, Philippe Laban, and Greg Durrett. 2024. Minicheck: Efficient fact-checking of llms on grounding documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024a. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Shuai Wang, Ekaterina Khramtsova, Shengyao Zhuang, and Guido Zuccon. 2024b. Feb4rag: Evaluating federated search in the context of retrieval augmented generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 763–773.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Frank F Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, et al. 2024a. Theagentcompany: benchmarking llm agents on consequential real world tasks. *arXiv preprint arXiv:2412.14161*.

Weijie Xu, Zicheng Huang, Wenxiang Hu, Xi Fang, Rajesh Cherukuri, Naumaan Nayyar, Lorenzo Malandri, and Srinivasan Sengamedu. 2024b. Hr-multiwoz: A task oriented dialogue (tod) dataset for hr llm agent. In *Proceedings of the First Workshop on Natural Language Processing for Human Resources (NLP4HR 2024)*, pages 59–72.

Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, et al. 2024. Crag–comprehensive rag benchmark. *arXiv preprint arXiv:2406.04744*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Rui Ye, Rui Ge, Xinyu Zhu, Jingyi Chai, Yaxin Du, Yang Liu, Yanfeng Wang, and Siheng Chen. 2024. Fedllm-bench: Realistic benchmarks for federated learning of large language models. *arXiv preprint arXiv:2406.04845*.

Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. 2024. Assistantbench: Can web agents solve realistic and time-consuming tasks? *arXiv preprint arXiv:2407.15711*.

Matei Zaharia, Omar Khattab, Lingjiao Chen, Jared Quincy Davis, Heather Miller, Chris Potts, James Zou, Michael Carbin, Jonathan Frankle, Naveen Rao, and Ali Ghodsi. 2024. The shift from models to compound ai systems. `https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/`.

Bing Zhang, Mikio Takeuchi, Ryo Kawahara, Shubhi Asthana, Md Maruf Hossain, Guang-Jie Ren, Kate

Soule, and Yada Zhu. 2024a. Enterprise benchmarks for large language model evaluation. *arXiv preprint arXiv:2410.12857*.

Jintian Zhang, Xin Xu, Ningyu Zhang, Ruibo Liu, Bryan Hooi, and Shumin Deng. Exploring collaboration mechanisms for llm agents: A social psychology view. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. 2025. Murre: Multi-hop table retrieval with removal for open-domain text-to-sql. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5789–5806.

Yuxiang Zhang, Jing Chen, Junjie Wang, Yaxin Liu, Cheng Yang, Chufan Shi, Xinyu Zhu, Zihao Lin, Hanwen Wan, Yujiu Yang, Tetsuya Sakai, Tian Feng, and Hayato Yamana. 2024b. Toolbehonest: A multi-level hallucination diagnostic benchmark for tool-augmented large language models. *arXiv preprint arXiv:2406.20015*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems*, 36:50117–50143.

A Appendix

In this section, we provide detailed related work and additional results and analysis that we could not include in the main paper due to space constraints. In particular, this appendix contains the following:

- Error Analysis on Evaluation of EnterpriseBench
- EnterpriseBench Analysis
- Extended Related Work
- EnterpriseBench Creation Additional Details
- EnterpriseBenchSecurity Layer Details
- Knowledge Graph Formation for Task Creation in EnterpriseBench
- Evaluation Process of EnterpriseBench
- Extended Evaluation Metric
- LLM Prompts

A.1 Error Analysis on Evaluation of EnterpriseBench

Our error analysis on 100 sampled examples for the o1-mini ReAct during Human Evaluation—10% unanswerable queries, 10% access-denied cases, 20% CRUD operations, and 60% search tasks—revealed an 85% failure rate on Task Execution. The identified error categories are as follows:

- **Errors in LLM Invocation (67):** The following LLM errors have arisen due to multiple factors, including hallucination, context misalignment, intent recognition.
    1. *Subgoal Decomposition*: For complex tasks requiring subgoal decomposition, LLMs often generate oversimplified subgoals, deviating from the primary objective. For instance, when extracting email IDs of ≥ 1 recipients, the model may hallucinate fake addresses instead of retrieving them from the given data.
    2. *Data Source Selection*: The LLM sometimes misselects data sources when its pre-trained knowledge conflicts with provided descriptions, occasionally referencing non-existent sources, leading to context misalignment.
    3. *Tool Selection*: The LLM exhibits semantic parsing failures and weak generalization in tool selection. It correctly invokes the "Report Generation" tool for explicit commands like "Generate a Report" but fails to recognize equivalent requests such as "Provide an analysis document." Similar inconsistencies occur with tools like "Data Analysis Tool" and "LLM Call Tool," highlighting issues in intent recognition and instruction mapping.
    4. *Access Control*: The LLM exhibits access control failures, often misjudging permissions when they depend on specific identifiers, such as *emp_id* in *Collaboration Tools* or *email_id* in *Enterprise Mail System*.
    5. *Response Generation*: The LLM sometimes fails to answer search-related queries despite having full context due to reasoning limitations. For unanswerable questions, it hallucinates responses based on prior knowledge instead of recognizing them as unanswerable.
- **Retrieval Errors (7):** The retriever component occasionally fetches irrelevant or incomplete data, resulting in inaccurate responses or erroneous "Context not sufficient" outputs.
- **Tool Execution (11):** Tool execution failures hinder task completion. Errors include misstructured nested SQL queries, incorrect parameter parsing in CRUD functions, which leads inconsistency in the information.

A.2 EnterpriseBench Analysis

Our benchmark creation involves two parts: Enterprise Simulated Data Creation and Enterprise Tasks Creation. 1) For Enterprise Simulated Data, we conduct a grounding test to evaluate the LLM's ability to generate contextually consistent outputs and a realism test to compare the generated data with human-curated data. 2) For Enterprise Tasks, We conduct a quality check, a detailed error analysis to identify inaccuracies, and human evaluation to verify task realism.

A.2.1 Analysis of Generated Data in EnterpriseBench

1. **Grounding on the EnterpriseBench data**: LLMs often hallucinate, generating text that

deviates from the given context. To evaluate our data generation approach, we conduct a grounding test using the methodology from Tang et al. (2024) [Minicheck].

Table 3 shows the grounding task results across various models referenced in Minicheck, with 70%-80% of the generated data being grounded in the actual content. This performance is attributed to the Context QA generation step before data generation, which enhances grounding, reduces hallucinations, and improves contextual alignment.

2. **Quality check for Generated Data**: To evaluate the quality of our dataset, we conducted a comparative relevance analysis against human-annotated email and chat corpora. Our email dataset was benchmarked against the Enron Email Corpus (Enron Emails), while our chat dataset was compared to the Topical-Chat dataset (Gopalakrishnan et al., 2019).

   We employed an LLM-based assessment framework, utilizing o1-mini and Claude 3.5-Sonnet as evaluators to determine alignment with human-authored content. The evaluation assessed key linguistic and contextual factors such as coherence, conversational flow, topic adherence, and stylistic similarity.

   Our results demonstrate a high degree of relevance to human-curated data, with LLM-based evaluations scoring 93% and 94% for emails and 86% and 82% for chats, respectively. Additionally, the inter-rater reliability, measured using Cohen's Kappa (Cohen, 1960), showed strong agreement between the two LLM evaluators, yielding scores of 0.8521 for emails and 0.7623 for chats (Table 4).

A.2.2 Analysis of Tasks in EnterpriseBench

1. **Error Analysis** For human evaluation, we sampled data from each task classification by proportionally scaling to 100 samples and selecting instances randomly. From our generation pipeline, 23 tasks were rejected, with the following breakdown.

   - **LLM Fault - 13**:
     - **Data Source Dependency and Persona Selection (6)**: The LLM occasionally struggles to integrate dependencies across multiple ($\geq$ 2) data sources when generating persona-based goals and subgoals. It also at times fails to consider employee hierarchy, leading to inaccurate task generation.
     - **Subgoal Generation(5)**: While decomposing subgoals, LLM(s) don't stay consistent with the Primary Goal and generates subgoals that are diverged from the provided enterprise environment.
     - **Tool Alignment (1)**: Tool selections for particular subgoal can sometimes diverge from the primary goal.
     - **Invalid Unanswerable Questions (1)**: The LLM incorrectly generates answers for unanswerable queries using its world knowledge instead of identifying them as unanswerable, despite the availability of a Web Search API.
   - **Entity-Persona Alignment (3)**: Extracted entities sometimes misalign with the persona's primary goal, leading to the retrieval of irrelevant context for task completion.
   - **KG Fault and Entity-Source Mapping (7)**: The self-reflection KG sometimes fails to preserve entities with the same keywords as in the data source, making source mapping difficult.

2. **Quality check for Tasks**:
   LLM as a Judge: We conducted a realism evaluation to compare our task dataset against ToolBeHonest (Zhang et al., 2024b), which consists of 700 manually annotated evaluation samples across seven distinct tasks. For this analysis, we randomly sampled 100 instances and employed a large language model (LLM) as a judge to assess whether the text exhibits characteristics expected in human annotation, including coherence, logical consistency, factual correctness, reasoning depth, linguistic diversity, adherence to task-specific constraints, and contextual appropriateness. Our evaluation aimed to assess the degree to which the Tasks from EnterpriseBench aligns with real-world tasks.

   The experiment was conducted using two state-of-the-art models, GPT-4o and Claude

Models	Data Sources			
	Collaboration Tools	Enterprise Mail System	Github Issues	Customer Support Chats
Roberta Large	81.23	74.81	65.88	74.64
Flan T5 Large	85.01	77.34	68.50	71.35
MiniCheck 7B	77.60	69.30	59.78	72.04

Table 3: Context Grounding on Data Generated by Various Models

	Human-Likeness Score (%)		Cohen's Kappa
Category	**Claude 3.5-Sonnet**	**o1-mini**	
	Emails		
Enron Emails	96.0	100.0	
Enterprise Mail System(EnterpriseBench)	93.0	94.0	0.8521
	Chats		
Topical Chat (Gopalakrishnan et al., 2019)	91.0	97.0	
Collaboration Tools(EnterpriseBench)	86.0	82.0	0.7623

Table 4: LLM as a Judge for Realism: Comparison of Human-Likeness Score (%) and agreement scores across different datasets.

3.5 Sonnet, yielding Human Likeness Scores of 74.32% and 77.00%, respectively. These scores indicate the proportion of tasks that closely resemble real-world scenarios. Furthermore, we computed the inter-model agreement using Cohen's Kappa coefficient(Cohen, 1960), obtaining a score of 0.6865, which signifies moderate-to-substantial agreement between the two models. Table 5 presents our results, illustrating the extent to which individual instances exhibit human-like characteristics.

Human as a Judge: We conducted a survey involving 20 human annotators on a randomly sampled set of 50 task instances. The results indicate that 67% of the tasks looks like they were curated by human annotators.

A.3 Extended Related Work

Compound AI Systems LLMs have emerged as powerful tools, demonstrating excellence in tasks such as processing and generating human-like text (Team et al., 2023; Achiam et al., 2023), writing code (Chen et al., 2021), and performing complex reasoning (Khetan et al., 2020). Beyond these fundamental capabilities, LLMs show immense potential as agents within Compound AI Systems, enabling collaborative problem-solving, dynamic interactions, and advanced decision-making (Yao et al., 2022b; Xi et al., 2023; Wang et al., 2024a). As tasks grow in complexity and scope, leveraging multiple LLMs in a cooperative framework becomes a natural strategy to enhance their effectiveness. A Compound AI System comprises of multiple LLMs working together to achieve a shared objective, with each LLM assigned a specific role tailored to particular tasks. These agents can access distinct tools, make independent decisions, and communicate seamlessly with one another, creating a synergistic system capable of tackling sophisticated challenges.

Compound AI System Benchmarks Compound AI Systems have been developed to address a wide range of tasks, including scientific experimentation (Ghafarollahi and Buehler, 2024; Boiko et al., 2023; M. Bran et al., 2024), embodied intelligence (Brohan et al., 2023), and societal simulations (Gao et al., 2023; Li et al., 2023). In scenarios requiring diverse resources or distributed systems, such as federated search (Shokouhi and Si, 2011), the integration of multiple LLMs becomes crucial to enhance efficiency and performance. To support the evaluation of such models serving as LLM agents, various benchmarks have emerged. For instance, FedLLM (Ye et al., 2024), FedMultimodal (Feng et al., 2023), and FEB4RAG (Wang et al., 2024b) address challenges like heterogeneous data distributions and privacy constraints. Similarly, environment-based benchmarks such as Mind2Web (Deng et al.), WebArena (Zhou et al.), and WebShop (Yao et al., 2022a) offer testing grounds for task-specific LLM agents in controlled settings. Despite these advancements, a significant gap persists in the development of enterprise simulated environments that accurately represent real-world

	Human-Likeness Score (%)		Cohen's Kappa
Category	Claude 3.5-Sonnet	o1-mini	
ToolBeHonest	87.53	95.71	
EnterpriseBench	77.00	74.32	0.6875

Table 5: LLM as a Judge for Realism: Comparison of human percentage estimates across models and agreement score.

conditions. A comparison of our proposed EnterpriseBenchwith existing benchmarks is presented in Table 7.

Enterprise Search: An Underexplored Area Enterprise Search systems provide team members with a unified platform to access the diverse and dispersed knowledge within an organization. Liang et al. highlights the importance of enterprise-specific benchmarks, particularly in domains like finance. However, there is a notable gap in the availability of comprehensive benchmarks tailored to real-world enterprise scenarios. While efforts such as Zhang et al. (2024a) have aimed to address this issue, their evaluations are often limited in scope and fail to reflect the complexities of practical enterprise settings. As Bruckhaus (2024) highlights, RAG in enterprise contexts is far from straightforward and introduces unique challenges. Enterprises manage vast volumes of data distributed across multiple domains, formats, and systems. Additionally, enterprise systems must meet stringent requirements, including compliance, accuracy, seamless integration, and scalability. However, the lack of suitable benchmarks tailored to these complex settings significantly impedes the development of such advanced systems. To address this gap, we propose a novel benchmark, EnterpriseBench, specifically designed for enterprise scenarios. This benchmark provides a robust framework to evaluate LLM-based agents under realistic and domain-relevant conditions, facilitating the development of effective and reliable enterprise systems.

A.4 EnterpriseBench Creation Additional Details

EnterpriseBench represents a real-world organizational structure, providing both a high-level overview and a detailed breakdown of its components and their operations. Figure 7a illustrates the organizational architecture of our dataset, where every component is linked to either Employee data or Customer data, as these serve as the primary reference entities for other components. Figure 7b depicts the departmental structure within EnterpriseBench, showcasing hierarchical relationships within each department to simulate a realistic organizational environment.

A.4.1 Data Collection

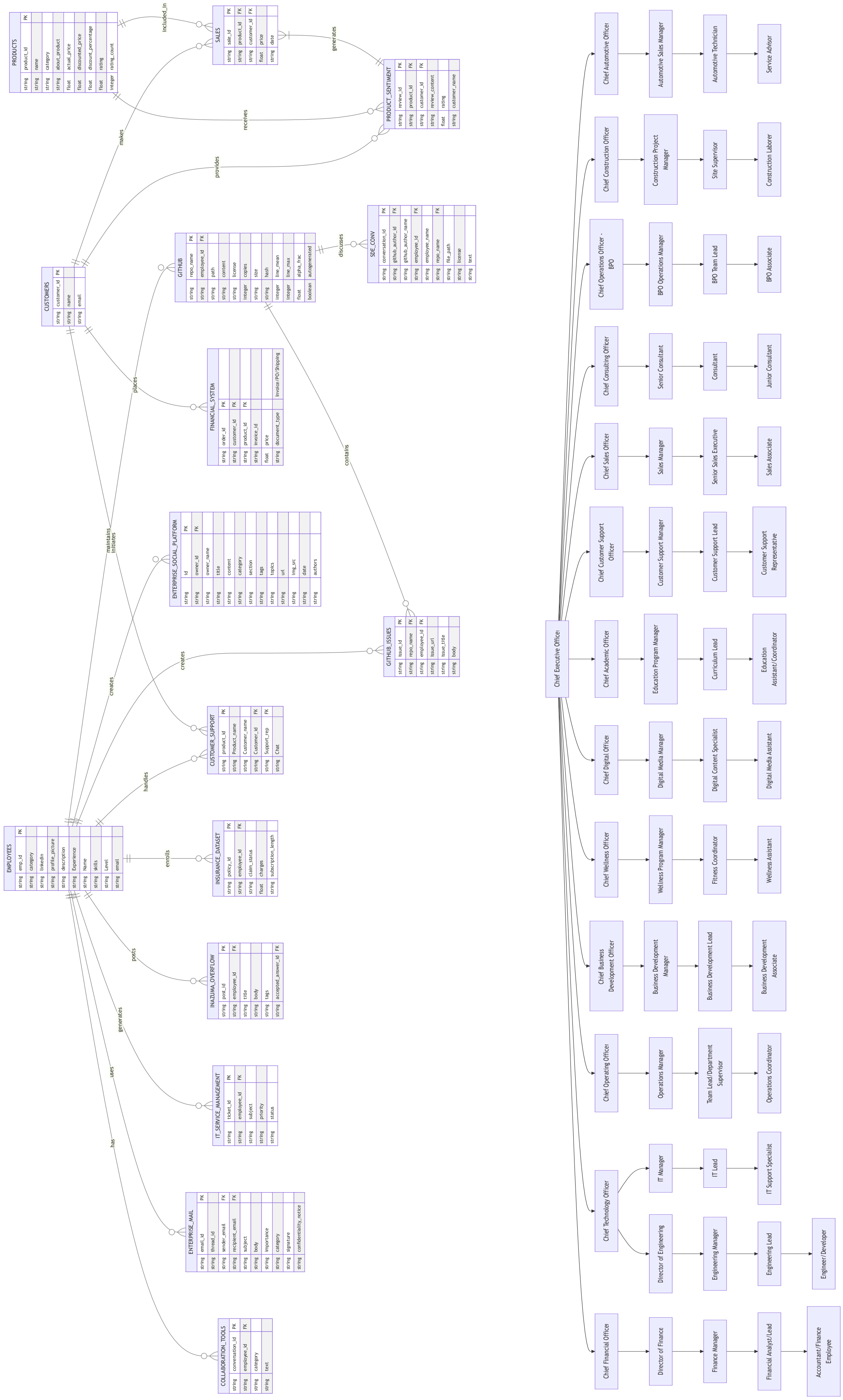
The data collection process is designed to align with the enterprise structure. To ensure data authenticity, we sourced information from reliable and verified sources. After collection, the data was parsed to extract relevant attributes. For example, from the collected product sentiment data, we extracted customer and product information and synchronized it with the sales dataset. Table 7 & Figure 8 provides a detailed overview of the data sources, the number of instances in EnterpriseBench, and their respective collection origins.

A.4.2 Simulated Conversations

The conversations generated in EnterpriseBench span various departmental teams, covering a wide range of topics—from *simple inquiries* to *comprehensive discussions about a specific GitHub repository*. These conversations are context-dependent and are designed to closely simulate real-world interactions, following the generation process of the proposed holistic pipeline. Figure 9 presents an example of a chat between two employees, `Steve` and `John`, from the engineering department, based on the GitHub repository maintained by `Steve`.

A.4.3 Simulated Customer Support Chat

The customer support conversations are generated based on product sentiment data. Persona-based interactions subjects are created by incorporating details of both the customer and a sales representative(employee from sales department). These interactions simulate a conversation where the representative responds to the customer's sentiment by proposing a potential solution to resolve the issue. Figure 10 illustrates an example of such a conversation between a customer and a sales representative.



(a) ER Diagram

(b) Employee Hierarchy

Figure 7: Enterprise Structure

Benchmarks	Diverse Real-World Tasks	Task Domains	# Data Sources Interaction	Step-by-Step Evaluation	Automated Task Generation	Access Controls	Persona-based Tasks
MiniWob++ (Liu et al., 2018)	✗	Browsing*	🗎	✗	✓	✗	✗
Mind2Web (Deng et al., 2024)	✗	Browsing*	🗎	✗	✗	✗	✗
WebLINX (Lù et al.)	✗	Browsing*	🗎	✗	✗	✗	✗
AssistantBench (Yoran et al., 2024)	✗	Browsing*	🗎	✗	✗	✗	✓
WebArena (Zhou et al.)	✗	Browsing*	🗎	✗	✗	✗	✗
SWE-bench (Jimenez et al.)	✗	SWE	🗎	✗	✓	✗	✗
DevBench (Li et al., 2024)	✗	SWE	🗎	✗	✓	✗	✗
WorkArena (Drouin et al.)	✓	Enterprise Software	🗎🗎	✗	✓	✗	✗
OSWorld (Xie et al., 2024)	✓	Office, Coding	🗎🗎	✗	✓	✗	✗
Windows Agent Arena (Bonatti et al.)	✓	Browsing*, Office, Coding	🗎🗎🗎	✗	✓	✗	✗
TheAgentCompany (Xu et al., 2024a)	✓	SWE, HR, Admin, PM, Research, Finance	🗎🗎🗎	✓	✓	✗	✗
EnterpriseBench	✓	SWE, HR, Admin, IT tickets, Sales, Finance, CRM, etc.	🗎🗎🗎🗎	✓	✓	✓	✓

Table 6: Comparison of benchmarks based on diverse real-world work, task categories, interaction requirements, and interface support.

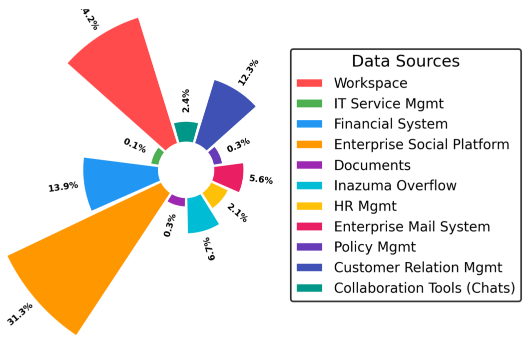


Figure 8: Distribution of Data-source in EnterpriseBench

A.4.4 Simulated Enterprise Mail System

The email simulations are generated based on threaded conversations, where each email exchange belongs to a specific thread. Within a thread, multiple messages are exchanged between the sender and recipient, maintaining continuity and context. Figure 11 presents an example of an email thread between two employees from the HR department.

A.5 EnterpriseBenchSecurity Layer Details

In enterprise environments, ensuring secure and regulated data access is critical. The **Access Control Layer** plays a fundamental role in enforcing access policies and preventing unauthorized data access. Our work, EnterpriseBench, implements a structured approach by integrating access control rules in a JSON format for each data source. A **Large Language Model (LLM) agent** is responsible for verifying access permissions based on an employee's credentials and the requested data.

A.5.1 Access Verification Mechanism

The Access Control Layer operates in conjunction with the retrieval process. When a query is processed, the Retriever first gathers relevant contextual data. Before the information is presented to the user, it is passed through the Access Control Layer, where all inaccessible content is filtered out based on predefined rules.

For instance, as illustrated in **Figure 12**, the access control rules dictate that a **GitHub repository** is accessible only to its owner and senior employees within the organizational hierarchy. If an employee from a different department, or even from the same department but with an `emp_id` different from the `repo_owner_id`, attempts to access the repository, the agent will respond with **"Access Denied."** Furthermore, if an employee at the same level attempts to perform a task requiring edit access to the repository, the agent will revoke the request, ensuring

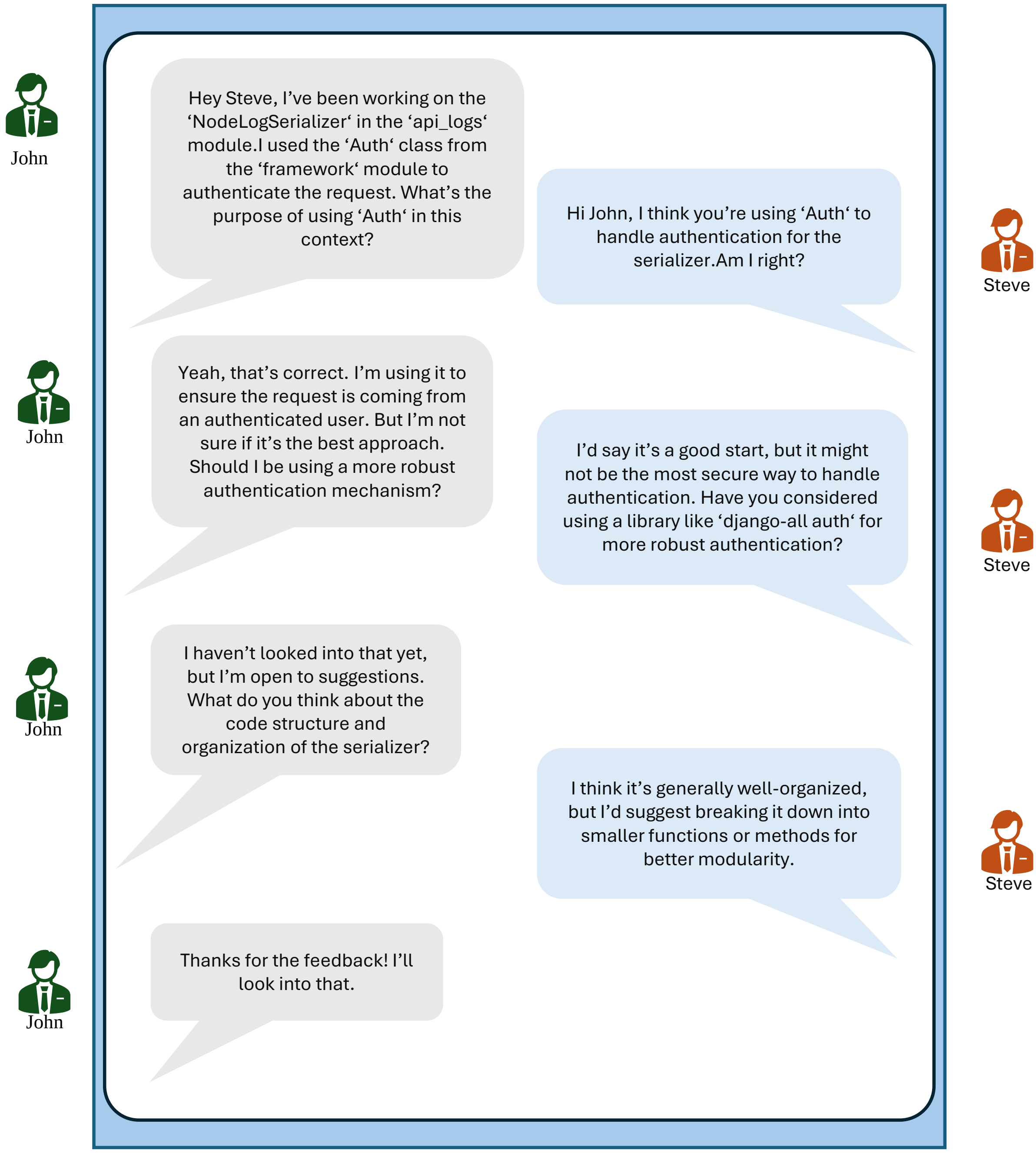


Figure 9: Collaboration Tools chat

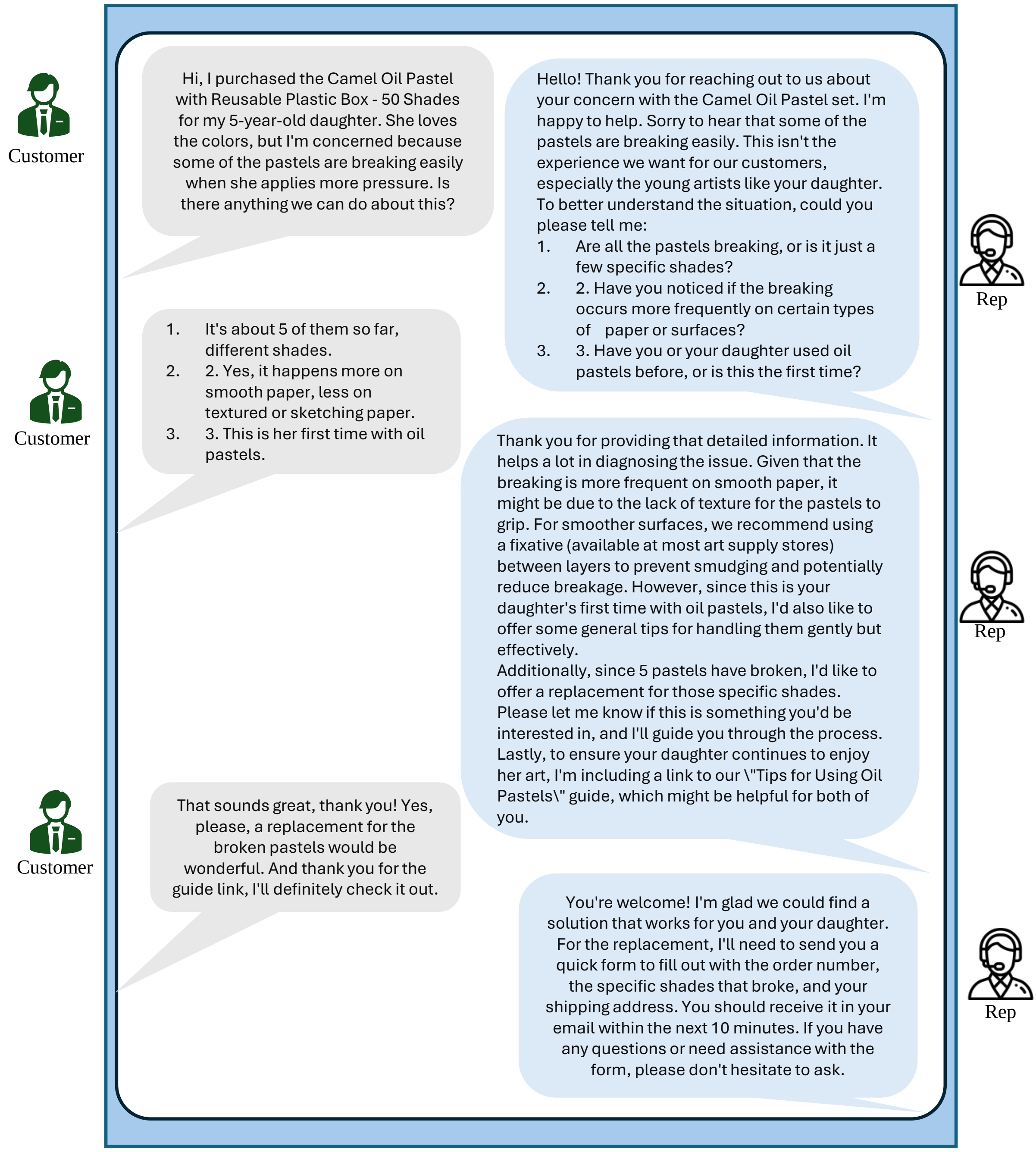


Figure 10: Customer Support Chat

Date: 2012-03-18 06:58:29 IST
Sender: aji.joseph@inazuma.com,
Recipient: deepa.gangadharan@inazuma.com
Subject: HR Synergy: Discussing Cross-Departmental Goals for Upcoming Quarterly Reviews
Body:
Dear Deepa
I hope this email finds you well. As we approach the upcoming quarterly reviews, I wanted to touch base with you regarding the HR synergy project. As you know, our department's goals are closely aligned with the company's overall objectives. I'd like to schedule a meeting to discuss how we can work together to achieve these goals. Please find attached the draft goals document for your review. I've highlighted the key areas where we need your input. Looking forward to hearing from you soon.
Importance: High
Category: INTERNAL
Signature: Aji Joseph, HR Generalist, Inazuma Corporation, Bangalore
Confidentiality Notice: This email contains confidential information intended only for the use of the individual or entity named above. Any unauthorized use, disclosure, or distribution is strictly prohibited.

Data: 2021-03-21 22:52:11 IST
Sender: deepa.gangadharan@inazuma.com
Recipient: aji.joseph@inazuma.com
Subject: Re: HR Synergy: Discussing Cross-Departmental Goals for Upcoming Quarterly Reviews
Body:
Hi Aji,
Thanks for reaching out and sharing the draft goals document. I've reviewed it and have some suggestions for the 'Employee Engagement' section. I've attached my feedback document for your reference. Would it be possible to schedule the meeting or next Wednesday at 2 PM? I've also invited our team lead, Rohan, to join the discussion. Looking forward to your confirmation.
Importance: Normal,
Category: FOLLOW-UP
Signature: Deepa Gangadharan, HR Manager, Inazuma Corporation, Bangalore
Confidentiality notice: This email contains confidential information intended only for the use of the individual or entity named above. Any unauthorized use, disclosure, or distribution is strictly prohibited.

Data: 2021-03-22 14:52:11 IST
Sender: aji.joseph@inazuma.com
Recipient: deepa.gangadharan@inazuma.com
Subject: Re: HR Synergy: Discussing Cross-Departmental Goals for Upcoming Quarterly Reviews
Body:
Hi Deepa,
Thanks for your prompt response and valuable feedback. I've reviewed your suggestions and incorporated them into the revised goals document (attached). I've also confirmed the meeting with Rohan for next Wednesday at 2 PM. Please note that I've marked the 'Employee Retention' section as 'High Priority' in the revised document. I'd like to discuss this further during the meeting. See you both on Wednesday..
Importance: High,
Category: FOLLOW-UP
Signature: : Aji Joseph, HR Generalist, Inazuma Corporation, Bangalore
Confidentiality notice: This email contains confidential information intended only for the use of the individual or entity named above. Any unauthorized use, disclosure, or distribution is strictly prohibited.

Figure 11: Enterprise Mail System

Data Source	Data Source Elements	Data Formats	Collected/Generated	# Instances	Data Origin	Collected Source link
Collaboration Tools (Chats)	HR Finance Sales Mgmt IT SDE	JSON	Generated	500 500 500 500 500 500	Employees.csv + GitHub Code + Policy Documents + Sales + etc.	-
Customer Relation Management	Customer Support Chats Product Sentiments Customers.csv	CSV JSON JSON	Generated Collected Collected	1000 13500 832	Product Sentiments + Customer.csv + Employees.csv + Sales.csv	Product Sentiments Customers (Extracted from Product sentiments)
Documents and Policy Management	Policy Documents Employee Insurance Details	PDF CSV	Collected	23 * 15 1265	-	Documents Collected from Google Datasets Insurance Policies
Enterprise Mail System	HR Finance Sales Management IT SDE Other Dept Emails	JSON	Generated	7000	Employees.csv + GitHub Code + Policy Documents + Sales + etc.	-
Enterprise Social Platform	Tech Crunch Posts (Social Platform)	JSON	Collected	39115	-	Tech Crunch Posts
Financial System	Customer Orders Products Product Sales Stocks	PDF CSV CSV CSV	Generated Collected Collected Collected	832 1352 13511 1700	Product Sales	Extracted from Product Sentiment dataset
HR Management	Employees.csv Resumes Roles	CSV PDF PDF	Collected Generated Generated	1265 1265 1 * 32	Employees.csv	LinkedIn Profiles (Ayoobi et al., 2023)
Inuzuma Overflow	Technical Posts (like StackOverflow)	JSON	Collected	8398 Posts	-	Stack Overflow Posts
IT Service Management	Help Desk Tickets	JSON	Collected	163Tickets	-	Helpdesk Customer Tickets
Workspace	GitHub Repository GitHub Repository Issues	JSON	Collected Generated	29241 957	GitHub + Employees.csv	GitHub Code

Table 7: Enterprise Data Source Statistics (explain all column names)

strict compliance with access policies.

A.5.2 Dynamic and Customizable Access Control

The **Access Control Layer** is designed to be flexible, allowing dynamic modification of access rules. This adaptability enables organizations to customize security policies according to evolving requirements while ensuring robust data protection. By maintaining granular control over data accessibility, this framework enhances security and compliance within enterprise systems.

Data Dynamism Pipeline

```
from llmCrudOps import EnggConvCRUD
from llmCrudOps import GitHubCRUD
from llmCrudOps import GitIssuesCRUD
...

class DataDynamismPipeline:
    def __init__(self, llm):
        self.llm = AzureChatOpenAI(llm)

    def fetch_crud_control(...):
        # Returns CRUD controller for selected data source
        return control
```

```
def run_CAI_pipeline(user_persona, user_query):

    # 1. Break down Primary Tasks into Subtasks
    prompt_CoT=ChatPromptTemplate.from_messages
    task_breakdown = prompt_CoT | self.llm
    generated_subtasks = chain_task_breakdown.invoke(...)

    for subtask in generated_subtasks:

        # 2. Determine Data Source
        prompt_ds=ChatPromptTemplate(...)
        chain_data_source = prompt_ds | self.llm
        selected_data_sources_str = chain_data_source.
            invoke(...)

        # 3. Determine Function and Parameters
        prompt_fn=ChatPromptTemplate(...)
        chain_function = prompt_fn | self.llm
        selected_function_name_str = chain_function.
            invoke(...)

        # 4. Check Access Permissions
        for function_name in selected_function_name_list:
            prompt_acc=ChatPromptTemplate(...)
            chain_access = prompt_acc | self.llm
            access_status = chain_access.invoke(...)

            # If Allowed, Execute CRUD Operation and
                Return Response

            if access_status == "Allowed":
                control = self.fetch_crud_control()

                if function_name -> read:
                    result = control.read(*params)
                elif function_name -> create:
                    result = control.create(*params)
                elif function_name -> update:
```

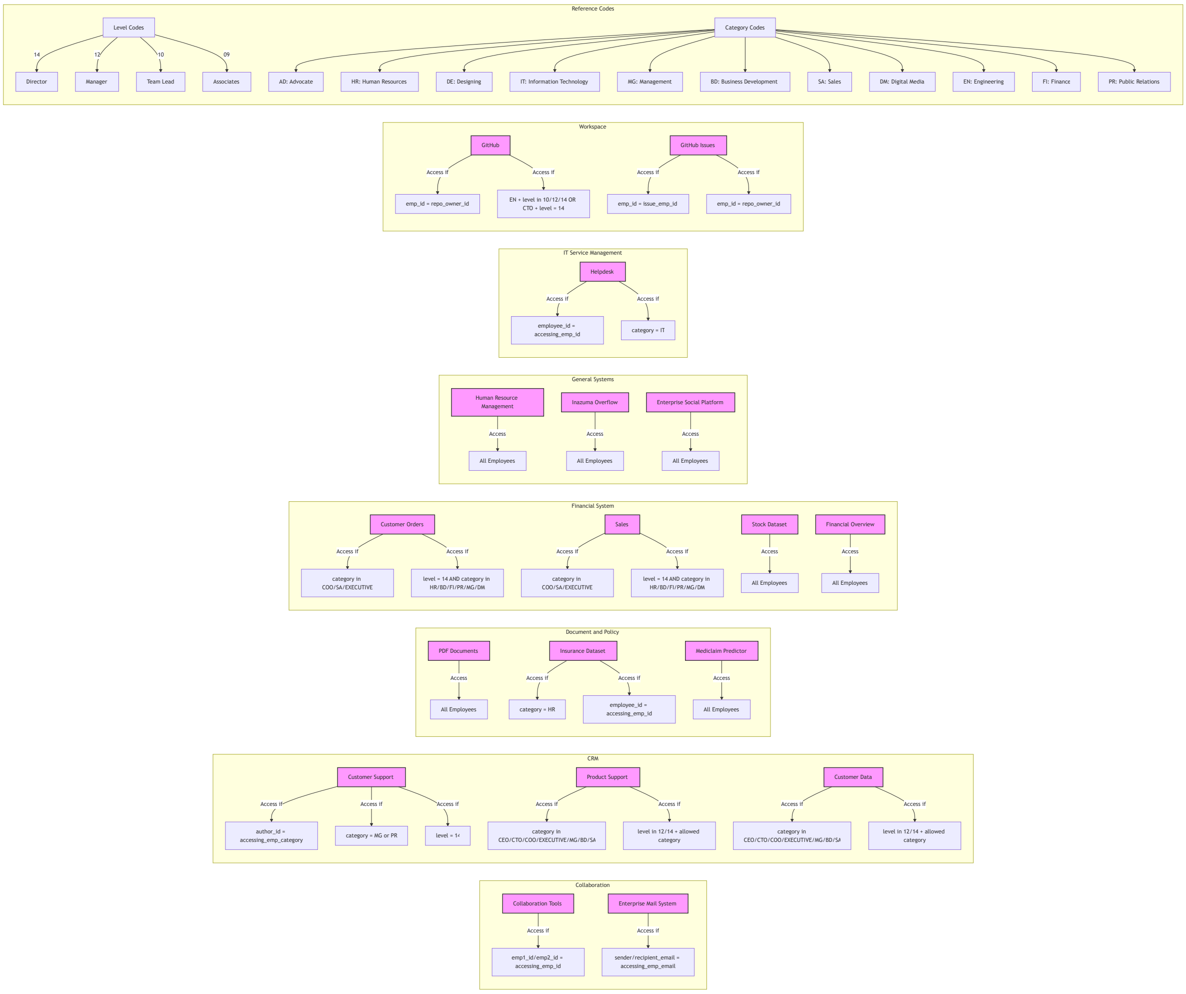


Figure 12: Access Control

```
                result = control.update(*params)
            elif function_name -> delete:
                result = control.delete(*params)

        return responses
```

GitHub CRUD Script

```
from accesscontrol import GitHubAccess

class GitHubCRUD:
    def __init__(self, employees_csv_path, code_json_path):
        self.access = access_control
        self.employees_df = ...
        self.code_data = ...
        self.code_json_path = ...

    def read(self, emp_id, path):
        """Reads GitHub code."""
        check -> access.is_valid_employee(emp_id):

        if (access.path_exists(...) and
            (access.is_owner(...) or
             access.is_engg_lvl_10_or_above(...) or
             access.is_cto_or_lvl_14(...))):
            for entry in self.code_data:
                if entry["path"] == path:
                    return entry
            print("Error: Code not found.")
        else:
            print("Error: Access denied.")


    def create(repo_name, emp_id, path, ...):
        """Creates a new GitHub code entry."""
        ....


    def update(self, emp_id, path, content, ...):
        """Updates an existing GitHub code entry."""
        check -> access.path_exists(...)
        check- > access.is_valid_employee(...)

        if (access.is_owner(...) or
            access.is_engg_lvl_10_or_above(...) or
            access.is_cto_or_lvl_14(...)):
            for entry in self.code_data:
                if entry["path"] == path:
                    # update entry
            print("Error: Code not found.")
        else:
            print("Error: Access denied for update.")


    def delete(self, emp_id, path):
        """Deletes a GitHub code entry."""
        ....
```

GitHub Access Check

```
class GitHubAccess:
    def __init__(self, employees_csv_path, code_json_path):
        self.employees_df = ...
        self.code_data = ...
        self.code_json_path = ...

    def path_exists(self, path, code_json_path) -> bool:
        """Checks if the GitHub code path exists."""
        ....

    def is_valid_employee(self, emp_id) -> bool:
        """Checks if the employee ID exists and is valid."""
        ....

    def is_owner(self, path, emp_id) -> bool:
        """Checks if the employee is the owner of the code
            path."""
        ...

    def is_engineer_lvl_10_or_above(self, emp_id) -> bool:
        """Checks if the employee is an Engineer with level
            >= 10."""
        ....

    def is_cto_or_lvl_14(self, emp_id) -> bool:
        """Checks if the employee is a CTO with level 14."""
        ....
```

A.6 Knowledge Graph Formation for Task Creation in EnterpriseBench

The **Knowledge Graph (KG)** plays a crucial role in the formation of task templates. The quality of KG construction directly impacts the accuracy and relevance of the generated templates. A well-structured KG ensures comprehensive task representation, minimizing inconsistencies and missing information. Our self-reflection framework (Figure 13) is inspired from methodology proposed by (Kertkeidkachorn and Ichise, 2017) which provides an approach to improving KG formation by incorporating a self-reflection mechanism.

A.6.1 Self-Reflection Mechanism for KG Construction

Self-reflection serves as a feedback loop wherein the **Large Language Model (LLM)** acts as its own evaluator, verifying whether the generated triples are consistent with the original data source. This consistency check is essential in reducing errors that may lead to missing critical information during KG construction. By ensuring that the extracted triples accurately represent the underlying data, self-reflection enhances the overall quality of the KG.

A.6.2 Handling Redundancy in KG Formation

Apart from ensuring consistency, it is equally important to **identify and amend redundant facts** in the KG. The presence of redundant or duplicate triples can lead to the generation of repetitive task templates, negatively impacting their efficiency and usability. By systematically refining the KG and eliminating redundancy, the framework ensures that extracted triples contribute meaningfully to task template formation, leading to a more structured and coherent representation.

Thus, by integrating self-reflection and redundancy correction, the proposed framework enhances the robustness of KG-based task template formation, ultimately improving the effectiveness of task execution in various applications.

A.7 Evaluation Process of EnterpriseBench

To systematically assess the performance of our **Compound AI System**, we define a structured evaluation framework tailored to different types of tasks (refer Figure 16). Our evaluation approach leverages **LLM-as-a-Judge** to assign scores, ensuring objective assessment across various categories.

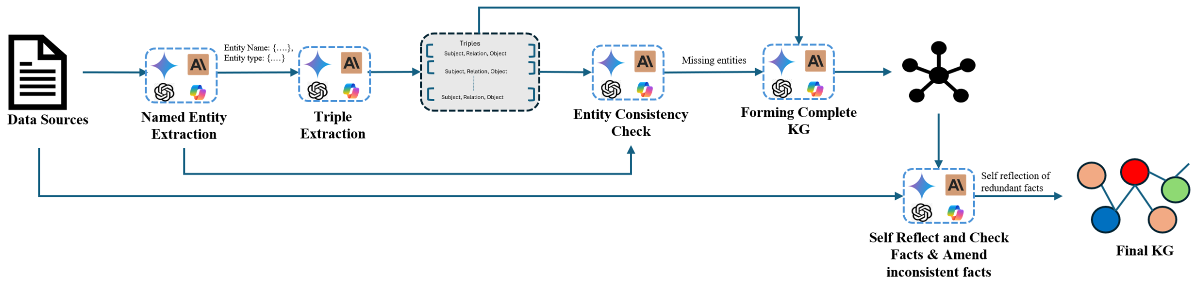


Figure 13: Self Reflecting KG

Below, we detail the evaluation methodology for different task types.

A.7.1 Search-Based Tasks

We adopt the methodology proposed by (Zheng et al., 2023), which demonstrates how to assess LLM-generated responses across different scenarios given a question and its corresponding answer. Search-based tasks are evaluated by comparing the system's generated response with the **gold answer** provided in the dataset for the **Primary Question**. The correctness of the response is determined based on semantic similarity and factual accuracy, as assessed by an **LLM-based evaluation metric** (refer to Section 3.3). This methodology ensures that the system retrieves and presents information accurately.

A.7.2 Tool Execution Evaluation

For tasks involving tool execution, we employ the following evaluation criteria:
External Tool Dependencies: For tasks requiring external tools, correctness is primarily assessed based on appropriate tool selection by the resource selection agent, given the assumption of reliable tool performance.
CRUD Operations: For Create, Update, and Delete operations, verification is performed through subsequent read operations:

- For Create and Update: The read output must match the tool inputs exactly
- For Delete: The read operation should return "Entry not found"

A.8 Extended Evaluation Metrics

To perform step-by-step evaluation of the Compound AI system under the defined scenarios, we designed a metric that penalizes the system for failing to complete a step or executing it incorrectly. The Final end to end execution of LLM is scored by equations (5) & (6)

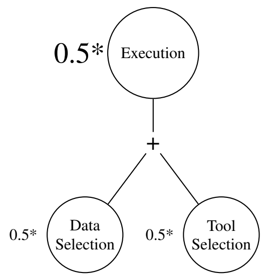
Here, $W[i]$ is the Penalty Factor, which is calculated as $1/2^i$ where i in the i_{th} intermediary step while execution of a task. The goal of the Penalty Factor is to dynamically allocate penalties based on the complexity of the intermediate steps of LLM agent for any particular task, essentially assigning lower penalties to more difficult steps(end-end execution and reasoning) and higher penalties to easier steps(Resource selection). This complexity hierarchy is represented through depth of graph in Figures 14b and 14a.
The flow vector I functions as a control mechanism that regulates the propagation of execution correctness from deeper levels to their parent nodes within the execution graph. It behaves similarly to a cascading AND gate, where execution validity depends on the correctness of previous stages. However, unlike a conventional AND gate that invalidates the entire execution if any condition is false, I only invalidates the portion of the execution path that follows the first incorrect decision.

For instance, in data source selection, if an incorrect data source is chosen, evaluating subgoal decomposition and execution beyond that point is redundant, as it may lead to misleading assessments by the LLM. Similarly, if the decomposed subgoals are not relevant to the primary task, evaluating their individual executions is unnecessary. However, any execution path that remains unaffected by the first incorrect decision continues to be evaluated independently.

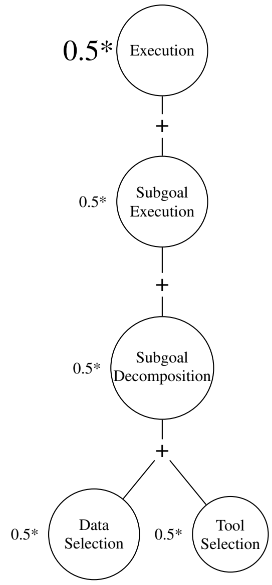
Putting all these together, we compute the final accuracy as follows:
Full Execution Score =

$$\begin{cases} 1, & \text{if full execution is correct} \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

Partial Execution Score (No Planning) =



(a) Graph depicting execution without planning.



(b) Graph depicting execution with planning.

Figure 14: Comparison of Execution Strategies: (a) Without Planning, (b) With Planning.

$$
\begin{aligned}
&0.5 \cdot \text{Full Execution} \\
&+ 0.5 \cdot \Big(0.5 \cdot I_1 \cdot \text{O[Data Selection]} \\
&+ 0.5 \cdot I_1 \cdot \text{O[Tool Selection]} \Big)
\end{aligned}
\tag{13}
$$

Partial Execution Score (With Planning) =

$$
\begin{aligned}
&0.5 \cdot \text{Full Execution} \\
&+0.5 \cdot \Big(0.125 \cdot I_1 \cdot \big(\text{O[Data Selection]} \\
&+I_1 \cdot \text{O[Tool Selection]}\big) \\
&+0.25 \cdot I_2 \cdot \text{O[Subgoal Decomposition]} \\
&+0.5 \cdot I_3 \cdot \text{O[Subgoal Execution]} \Big)
\end{aligned}
\tag{14}
$$

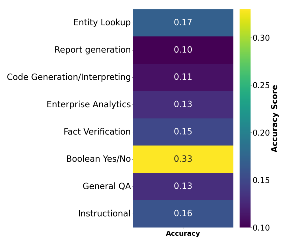
where,

$$
W[i] = \frac{1}{2^i}, \quad \text{with } i \text{ being the depth of execution} \tag{15}
$$

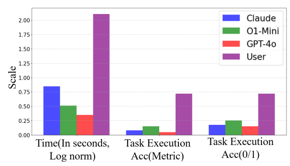
$$
O_i = \text{ LLM judge score for step } i \tag{16}
$$

Thus, the above unified metric automatically adjusts the weights of the Data Sources and Tools components in the planning and no-planning scenarios while including extra planning-related terms when appropriate.

All the above scores are assigned based on evaluations conducted by an LLM acting as a judge.



(a) Accuracies of different classes of tasks on o1-mini



(b) Time vs Accuracy plot between Humans (bounded by 10 min/task) and AI System

Figure 15: (a) Accuracies of different task classes on o1-mini, (b) Time vs Accuracy comparison between Humans and AI.

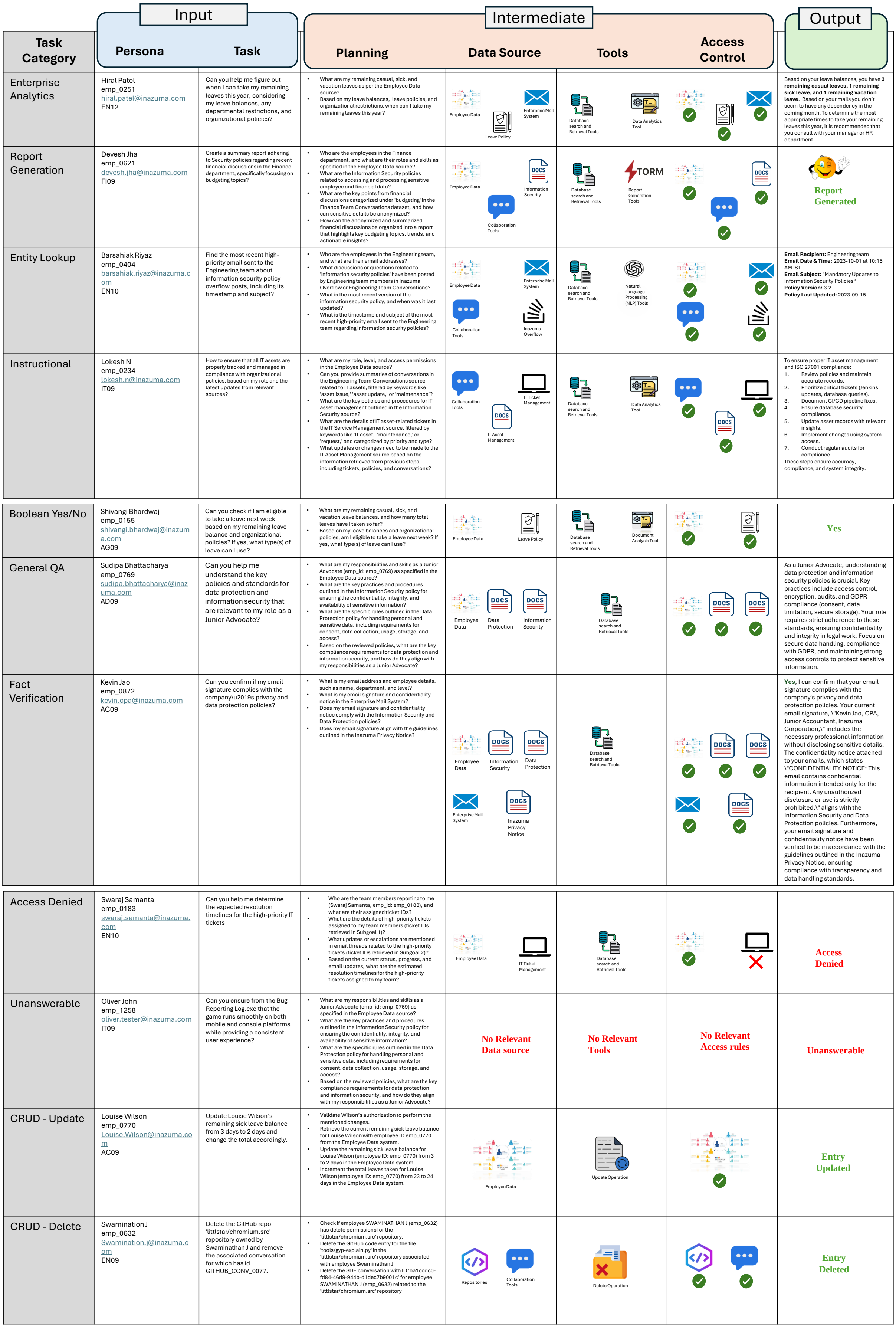


Figure 16: Task Execution Flow

Tool Name	Description	Usage	Source
Calculator	A tool for performing accurate numerical computations, ensuring precision in enterprise operations.	Used in financial analysis, operational planning, and engineering calculations.	Python function
Web Search API	A real-time tool for retrieving up-to-date information from the internet, aiding in enterprise decision-making.	Used for market research, competitive analysis, and staying updated with industry regulations.	Rapid-api
Code Interpreter/-Completion	A utility for generating, debugging, and completing code in various programming languages, optimizing enterprise applications.	Used in software development, automation of internal processes, and quick prototyping.	Claude 3.5-Sonnet
Code Compiler	A tool for compiling and executing code in multiple languages, validating and testing enterprise applications.	Used in testing and deploying applications that support business processes.	Rapid-api
Data Analysis Tools	A suite of tools for processing, analyzing, and visualizing structured data for enterprise decision-making.	Used in financial forecasting, operational optimization, and customer behavior analysis.	Code generation to generate plots based on query using Claude 3.5-Sonnet
Document Analysis	Tools for extracting, processing, and summarizing enterprise documents such as contracts and invoices.	Used in legal, finance, and compliance departments to streamline document-heavy workflows.	Colpali (Faysse et al., 2024)
Natural Language Processing (NLP) Tools	APIs and models for advanced text processing, enabling analysis of unstructured data and automation of workflows.	Used in customer service, market analysis, and sentiment tracking.	Claude 3.5-Sonnet
Report Generation Tool	A tool for automatically generating structured and visually appealing reports, ensuring accuracy and efficiency.	Used in IT operations, project management, and business analysis for periodic updates.	Co-STORM (Jiang et al., 2024b)
Database Search and Retrieval Tools	Tools for efficiently searching internal enterprise data sources for relevant information.	Used for retrieving compliance documents, customer insights, and historical team conversations.	Seperate Hybrid Retrievers for each data-source
CRUD Functions	Python functions for performing Create, Read, Update and Delete functionalities, providing a dynamic angle to the Datasource	Used for making dynamic changes in the Dataset	Python Functions

Table 8: Enterprise Tools Overview

A.9 LLM Prompts

Below are the mentioned prompts used for LLM based generation. The prompts are generated using the System prompt generated and then human intervention to refine them.

A.9.1 Prompts for Data Generation

Roles and Responsibilities Generation

Task: You are an expert Roles and Responsibilities Generating Agent. Your task is to generate precise and structured job roles and responsibilities based on an employee hierarchy. You ensure that each role aligns with industry standards and organizational needs.Analyze the given employee hierarchy, including department, level, and position details, and generate clear, structured roles and responsibilities. Your response must be tailored to the employee's seniority and function within the organization.

Input:

- **Employee Hierarchy:** {hierarchy_description}

Instructions:

- **Understand** the employee hierarchy, identifying role levels (Entry, Mid, Senior, Executive).
- **Identify** department-specific functions and responsibilities.
- **Break Down** responsibilities based on role level:
  - **Entry-Level(09):** Task-based execution.
  - **Mid-Level(10):** Process ownership, reporting.
  - **Manager-Level(12):** Strategy, leadership, cross-functional coordination.
  - **Director-Level(14):** Visionary leadership, policy development.
- **Analyze** industry benchmarks for role expectations.
- **Formulate** structured role definitions with specific, measurable responsibilities.
- **Validate** role alignment with organizational hierarchy.
- **What Not To Do:**
  - **DO NOT** generate vague or generic responsibilities.
  - **DO NOT** misalign responsibilities with the employee's seniority.
  - **DO NOT** create redundant or overlapping responsibilities.
  - **DO NOT** ignore the department context.
  - **DO NOT** exclude leadership responsibilities for managerial roles.

Output Format:

```
Role:[Job Title]
Department:[Department Name]
Level:[Entry/Mid/Senior/Executive]

Role Overview:[Brief role description]

Core Responsibilities:
1. [Specific responsibility]
2. [Another relevant responsibility]
3. [Aligned with seniority level]
4. [Distinct and measurable contribution]
5. [Ensure clarity, no redundancy]

Leadership Expectations (if applicable):
- [Leadership, mentoring, or strategic responsibility]
- [Cross-functional collaboration expectations]

Key Performance Indicators (if applicable):
- [KPI related to role function]
- [Measurable performance target]
```

Example Input:

{......}

Example Output:

{......}

Subject Generation Expert

Task: You are a subject generation expert, responsible for creating highly relevant and engaging subject lines tailored to different platforms. Your objective is to analyze the provided employee details and platform context to generate effective subject lines that align with the employee's role, responsibilities, and communication style.

Input:

- **persona of employee:** {persona}
- **platform type:** {platform}
- **platform description:** {platform_description}
- **primary communication objective:** {objective}

Instructions:

- **understand the input information**
- analyze the employee's role, department, and seniority level to align subject generation with their communication style.
- assess the platform type (e.g., email, chat, crm notifications, ticketing system, social media) and its intended function in the workflow.

- evaluate the data source providing the content to ensure subject lines reflect key insights or critical information.
- determine the primary communication objective (e.g., request, report, alert, engagement) to craft a purpose-driven subject line.
- **generate effective subject lines**
- ensure clarity, conciseness, and engagement based on the platform's nature.
- incorporate relevant keywords from the data source to enhance specificity.
- adapt tone based on the employee's role and platform requirements.
- provide multiple subject variations to account for different contexts.
- **adapt subjects based on platform-specific requirements**
- for **emails**: ensure clarity, urgency (if needed), and professionalism.
- for **chat systems**: keep it short, direct, and actionable.
- for **crm**: highlight key insights or action items.
- for **ticketing systems**: clearly define the issue or request.
- for **social media posts**: optimize for engagement and visibility.

- **What Not To Do:**
  - never generate generic or irrelevant subject lines that do not align with the platform or employee role.
  - never ignore platform-specific requirements when formulating subject lines.
  - never use unnecessary jargon or overly complex language unless required by the platform.
  - never repeat the same subject structure without variation.

Output Format:

1. **platform type:** [state the platform here.]
2. **generated subject lines:**
   - **formal variation:** [subject line]
   - **concise variation:** [subject line]
   - **engagement-driven variation:** [subject line]
   - **urgent variation (if applicable):** [subject line]

Example Input:
{......}
Example Output:
{......}

1514

Context QA Generation

Task: You are an advanced QA Generating Agent. Your task is to generate question-answer (QA) pairs that are specifically grounded in the given subject and context while simulating an employee's perspective. The generated QA pairs must be highly relevant, realistic, and aligned with the employee's role. Analyze the given employee details, subject, and context, then simulate the employee's thought process to generate natural, role-specific questions along with precise and well-structured answers.

Input:

- **Persona**: {persona_description}
- **Subject**: {subject}
- **Context**: {context}

Instructions:

- **Understand** the employee's role, seniority level, and domain expertise.
- **Identify** key aspects of the subject relevant to the employee's function.
- **Analyze** the provided context to ensure realistic and context-aware QA pairs.
- **Simulate** real-world workplace scenarios where the employee might ask these questions.
- **Generate** insightful, natural-sounding questions that are aligned with the subject and context.
- **Formulate** clear, direct, and well-structured answers that accurately address the questions.
- **Validate** the QA pairs to ensure coherence, relevance, and correctness.
- **What Not To Do:**
  - **DO NOT** generate generic or unrelated questions.
  - **DO NOT** create QA pairs that are misaligned with the employee's role or context.
  - **DO NOT** provide vague or overly broad answers.
  - **DO NOT** introduce fictional or misleading information.
  - **DO NOT** ignore the subject—each question must be strongly tied to the given topic.

Output Format:

Employee:[Employee Name / Job Title]
Subject:[Topic]
Context:[Background Details]

QA Pairs:
Q1:[Question from the employee's perspective]
A1:[Accurate, concise, and contextually relevant answer]

Q2:[Another realistic question]
A2:[Well-structured and informative response]

Q3:[Ensure contextual alignment]
A3:[Direct and precise response]

Example Input:
{......}
Example Output:
{......}

Conversation-based data generation

Task: You are a conversation-based data generation agent, expert in creating realistic, contextually accurate conversations for different platforms such as email, MS Teams, Git issues, customer support chats, and more from a group of Question-Answer Pairs.

Input:

- **Persona:** {persona_description}
- **Clustered QA Pairs:** {clustered_qa_pair}
- **Platform description:** {data_source}

Instructions:

- **Understand the platform context:**
  - You will be given a type of conversation to generate (e.g., emails, chat logs, Git discussions).
  - You will receive semantically similar clustered question-answer pairs to inform your generation.
  - You will be provided with employee personas to ensure authenticity in style and tone.
- **Generate a realistic conversation:**
  - Incorporate the provided question-answer pairs organically into a fluid conversation.
  - Ensure the flow of the conversation feels natural, with a balance of formality and informality depending on the context.
  - Maintain contextual consistency, including references to projects, tasks, and previous messages if required.

- **Ensure authenticity in persona & tone:**
  - Adapt the language, response style, and tone to match the given persona (e.g., a senior engineer vs. a junior support rep).
  - Reflect realistic workplace behaviors such as greetings, acknowledgments, clarifications, and follow-ups.
- **Follow conversation structure based on the platform:**
  - **Emails:** Include greetings, formal sign-offs, and a professional structure.
  - **Chats:** Maintain a casual, concise tone with shorter sentences and possible emojis.
  - **Git Issues:** Structure discussions around problem-solution formats, including code snippets if relevant.
  - **Customer Support Chats:** Follow a helpful, professional, and empathetic tone.
- **Emulate organic human interactions:**
  - Include varied sentence structures, occasional typos, or edited messages (if informal chat).
  - Incorporate elements like response time gaps, follow-up questions, and clarifications to mimic real conversations.
- **Ensure variability & diversity in responses:**
  - Generate multiple variations of conversations using the same question-answer clusters to avoid repetitive patterns.
  - Introduce different levels of formality, detail, and word choice depending on context.
- **Chain of Thought (CoT) Process:**
  1. **Understand:** Read and analyze the provided question-answer clusters & employee personas.
  2. **Identify Basics:** Determine the type of conversation required (email, chat, Git issue, etc.).
  3. **Structure:** Organize the question-answer pairs into a natural dialogue flow.
  4. **Adapt:** Modify language, tone, and style based on the persona & context.
  5. **Refine:** Ensure smoothness, add transitions, and remove artificialness.
  6. **Review Edge Cases:** Check for consistency, coherence, and possible redundancies.
  7. **Finalize:** Output the conversation in the requested format.
- **What Not to Do:**
  - **DO NOT** generate generic or artificial responses that feel robotic.
  - **DO NOT** ignore the provided question-answer clusters or employee personas.
  - **DO NOT** create conversations that lack contextual consistency.

Example Input:
`{......}`
Example Output:
`{......}`

A.9.2 Prompts for Task Generation

Persona Specific Goal Generation

Task: You are a goal-generating agent that transforms task requests into actionable, step-by-step goals tailored to an employee persona's needs, the provided data dependency chain, and the specified goal category. Assume the persona is directly interacting with the system by framing their tasks as questions.

Input:

- **Persona Description**: {Persona_Description}
- **Data Source Dependency Chain**: {chain}
- **Each Data Source Description**: {Data_description}
- **Category of Goal**: {category}

Instructions:

- Understand the Persona's Question and Context
- Analyze the persona's **category**, **description**, **skills**, and **level** to interpret the question.
- Align the goal with their responsibilities and ensure the **goal category** influences sub-goals appropriately.
- Incorporate the Data Source Dependency Chain
- Interpret the **Data Source Dependency Chain** to structure the sequence and flow of data.
- Utilize the **Data Source Descriptions** to determine relevant inputs and outputs.
- Generate Goals Based on the Persona's Question
- Define a **Primary Goal** by rephrasing or expanding the persona's question into a clear, specific, and actionable task.
- Break down the Primary Goal into **Sub-Goals** that align sequentially with the **Data Source Dependency Chain**.

- Tailor Sub-Goals to the Goal Category
- Ensure Actionable Outputs
- All sub-goals except the last should involve retrieval, validation, or preparation.
- The final sub-goal should deliver insights, analysis, or decision-making support.

Output Format:

1. **Primary Goal:** [Clear and actionable objective reflecting the category.]
2. **Sub-Goals:**
   Retrieval or validation aligned with the chain of data source. Additional preparation or validation task if needed. Final actionable insight or output.

Example Input: {.......}

Example Output: {.......}

Tool Dependency Generation

Task: You are a Tool Dependency Generation Expert responsible for designing a detailed tool usage plan tailored to the persona's role, the provided goals and subgoals, and the enterprise environment's toolset. Your objective is to create an actionable plan that ensures efficient tool utilization across all steps of the workflow.

Input:

- **Persona of Employee**: {persona}
- **Tool Descriptions**: {tool_description}
- **Chain of Connected Data Sources**: {chain}
- **Description of Data Sources**: {data_description}
- **Primary Goal**: {primary_goal}
- **Subgoals**: {subgoals}

Instructions:

- **Understand the Input Information**
- Analyze the employee's role, skills, and level to recommend tools suited to their workflow and capabilities.
- Assess the features, functionality, and limitations of each tool to match them effectively with the goals and subgoals.
- Evaluate how data flows between sources to identify dependencies critical for tool selection.

- Understand the roles, inputs, and outputs of data sources to ensure tools align with data integration needs.
- Define the overarching objective the employee is tasked to achieve.
- Break down the primary goal into clear, actionable steps, considering data dependencies and tool functionalities.
- **Generate a Tool Dependency Plan**
- For **all subgoals except the last one**, focus on tools that facilitate data retrieval or preparation.
- For the **final subgoal**, recommend tools designed for analysis, actionable insights, or specific outcomes.
- Provide clear instructions for tool usage, ensuring alignment with the persona's skills and the data dependency chain.
- **Analyze the Persona and Goals**
- Use the persona's role, skills, and level to tailor tool recommendations to their proficiency and enterprise responsibilities.
- Ensure each tool aligns with the persona's workflow and enhances their productivity.
- **Evaluate Data Dependencies**
- Leverage the **Chain of Connected Data Sources** to map the logical flow of data retrieval and processing.
- Use the **Description of Data Sources** to align tool functionality with data inputs and outputs.
- **Design a Sequential Tool Usage Plan**
- For retrieval tasks, select tools that efficiently extract and organize data in alignment with the subgoal and data dependencies.
- For the final actionable task, recommend tools that synthesize data or provide insights, ensuring the output meets the enterprise's objectives.

Output Format:

1. **Primary Goal:** [State the overarching objective here.]
2. **Subgoals and Tool Usage Plan:**
   - **Subgoal(s):** [Describe the subgoal clearly.]
     Tool Usage: [Specify the retrieval tool(s) to be used.]
     How to Use the Tool(s): [Provide step-by-step instructions for using the tool(s).]
   - **Last Subgoal:** [Describe the final subgoal clearly, focusing on actionable insights or analysis.]
     Tool Usage: [Specify the analysis or processing tool(s) to be used.]
     How to Use the Tool(s): [Provide detailed instructions for using the tool(s).]

Notes:

- **Prioritize retrieval tools** for all subgoals except the final one, which requires an **analysis or actionable tool.**
- Ensure that tool recommendations align with the persona's skills and are practical for their level of expertise.
- Provide concise, enterprise-relevant instructions that can be directly implemented without ambiguity.
- The tool usage plan must follow the logical flow of data dependencies to ensure seamless integration.

Example Input:
{......}
Example Output:
{......}

Task Template Generation

Task: You are a Question Template Generating Agent responsible for creating a set of logically connected multi-hop question templates. These templates should systematically address subgoals while contributing to the primary goal. Each question must align with the provided entity names, entity types, and triples, ensuring answers are directly retrievable from the data. Tool dependencies should be evident from the triples, and for retrieval subgoals, the required information must be explicitly accessible within the triples.

Input:

- **Persona Description:** {persona_description}
- **Primary Goal:** {primary_goal}
- **Subgoals:** {subgoals}
- **Tools for Each Subgoal:** {tools_for_each_subgoal}
- **Dependent Data Source Chain:** {chain}
- **Data Source Descriptions and Triples:** {data}

Instructions:

- **Generate Multi-Hop Questions**
- Formulate one question per subgoal, ensuring the answer to each is found within the relevant triples.
- Structure questions to be dependent on answers from previous subgoals, forming a logical flow aligned with the dependency chain.
- Enable actionable insights through questions that systematically build toward achieving the primary goal.
- **For Each Retrieval Subgoal, Specify**

- **Data Resource:** Identify the specific data resource required, based on the dependency chain.
- **What to Access:** Clearly specify the exact attributes or entities to retrieve, using details from the data source description.
- **Tool to Access:** Identify the tool required to retrieve the data, if applicable.
- **Chain of Thought:** Explain how the retrieved data contributes to addressing the subgoal and advancing the primary goal.
- Ensure questions are logically connected, where the answer to one question provides information needed for the next.
- Follow the dependency flow of the data source chain.
- Frame questions such that the required information is directly retrievable from the triples.
- Use specific attributes, entities, or predicates from the triples in each question.
- Highlight the necessity of tools where applicable, ensuring the connection to the triples is clear.
- For retrieval subgoals, emphasize tools designed to access the relevant data.
- Write questions from the persona's perspective, making them clear, actionable, and aligned with their role.

Output Format:

1. **Primary Goal:** [State the overarching objective clearly.]
2. **Subgoals:**
    - **Subgoal:** [Describe the subgoal clearly.]
        - **Task Template:** [Write a task template based on the related triples for the first dependent data source in the chain.]
        - **Purpose of the Task:** [Explain how this Task contributes to achieving Primary Goal.]
        - **Data Resource:** [Specify which data resource to access.]
        - **What to Access:** [Describe what to access in that resource.]
        - **Tool to Access:** [Specify the tool required to access the data, if applicable.]
        - **Chain of Thought:** [Provide reasoning for how the data will address the subgoal.]

Example Input: {......}
Example Output: {......}

1527

Final Task Generation

Task: You are an agent tasked with generating a series of multi-hop, conversation-based questions tailored for an employee interacting with a chatbot. The questions must reflect the employee's persona, follow a logical data dependency chain, and be based on a provided question template.

Input:

- **Persona:** {persona_description}
- **Data Dependency Chain:** {data_dependency_chain}
- **Question Template:** {question_template}
- **Data:** {data}

Instructions:

- Analyze the employee's role, objectives, and context to craft questions that align with their conversational style and goals.
- Recognize the logical sequence in which data must be accessed to achieve the primary goal, ensuring questions follow this flow.
- Utilize and adapt the provided question templates to create specific, natural, and persona-focused queries.
- Extract precise information from the triples to replace placeholders in question templates and generate contextually accurate questions and answers.
  - Identify the employee's role and objectives based on the template.
  - Outline the sequence in which data must be accessed.
  - Determine the data source implied in the template.
  - Specify required data points (e.g., sales metrics, performance data).
  - Identify the method or system used to access the data.
- **Generate Tasks with labels:**
  - **For the Primary Goal:** Frame a single, first-person, conversational question summarizing the primary objective.
  - **For Subgoals:**
    - Break the main task into logical subgoals.
    - Write first-person query for each subgoal.
    - Provide exact answers derived from the data.
    - Specify data resources, required access, and tools used.

Output Format:

Persona: [Extracted persona from the question template],
Data Chain: [Logical sequence of data access],
Primary Goal: [Clearly defined objective],
Primary Goal Question: [Framed conversational question],
Subgoals: [List of subgoal questions and answers with supporting details]

Example Input:
{......}
Example Output:
{......}