COPPE
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia
UFRJ

COMPARATIVE ANALYSIS OF SINGLE AND MULTI-AGENT LARGE LANGUAGE MODEL ARCHITECTURES FOR DOMAIN-SPECIFIC TASKS IN WELL CONSTRUCTION

Vitor Brandão Sabbagh

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Geraldo Bonorino Xexéo

Rio de Janeiro
Julho de 2025

COMPARATIVE ANALYSIS OF SINGLE AND MULTI-AGENT LARGE LANGUAGE MODEL ARCHITECTURES FOR DOMAIN-SPECIFIC TASKS IN WELL CONSTRUCTION

Vitor Brandão Sabbagh

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Geraldo Bonorino Xexéo

Aprovada por: Prof. Nome do Primeiro Examinador Sobrenome
Prof. Nome do Segundo Examinador Sobrenome
Prof. Nome do Terceiro Examinador Sobrenome
Prof. Nome do Quarto Examinador Sobrenome
Prof. Nome do Quinto Examinador Sobrenome

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2025

Brandão Sabbagh, Vitor

Comparative Analysis of Single and Multi-Agent Large Language Model Architectures for Domain-Specific Tasks in Well Construction/Vitor Brandão Sabbagh. – Rio de Janeiro: UFRJ/COPPE, 2025.

X, 18 p.: il.; 29, 7cm.

Orientador: Geraldo Bonorino Xexéo

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2025.

Referências Bibliográficas: p. 16 – 16.

1. Large Language Models. 2. Agentes. 3. Construção de Poços de Petroleo. I. Bonorino Xexéo, Geraldo. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Para Carolina, minha companheira de vida.

Agradecimentos

Gostaria de agradecer a todos. [família, amigos, etc]

Estendo minha gratidão aos especialistas em engenharia de poços, Marcelo Grimberg, Rafael Peralta e Lorenzo Simonassi, cuja expertise e dedicação contribuíram significativamente para esta pesquisa.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

COMPARATIVE ANALYSIS OF SINGLE AND MULTI-AGENT LARGE LANGUAGE MODEL ARCHITECTURES FOR DOMAIN-SPECIFIC TASKS IN WELL CONSTRUCTION

Vitor Brandão Sabbagh

Julho/2025

Orientador: Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

Apresenta-se, nesta tese, a aplicação de grandes modelos de linguagem (LLM) no setor de petróleo e gás, especificamente em tarefas de construção e manutenção de poços. O estudo avalia o desempenho de uma arquitetura baseada em LLM de agente único e de múltiplos agentes no processamento de diferentes tarefas, oferecendo uma perspectiva comparativa sobre sua precisão e as implicações de custo de sua implementação. Os resultados indicam que sistemas multiagentes oferecem desempenho melhorado em tarefas de perguntas e respostas, com uma medida de veracidade 28% maior do que os sistemas de agente único, mas a um custo financeiro mais alto. Especificamente, a arquitetura multiagente incorre em custos que são, em média, 3,7 vezes maiores do que os da configuração de agente único, devido ao aumento do número de tokens processados. Por outro lado, os sistemas de agente único se destacam em tarefas de texto para SQL (Linguagem de Consulta Estruturada), especialmente ao usar o Transformador Pré-Treinado Generativo 4 (GPT-4), alcançando uma pontuação 15% maior em comparação com as configurações multiagentes, sugerindo que arquiteturas mais simples podem, às vezes, superar a complexidade. A novidade deste trabalho reside em seu exame original dos desafios específicos apresentados pelos dados complexos, técnicos e não estruturados inerentes às operações de construção de poços, contribuindo para o planejamento estratégico da adoção de aplicações de IA generativa, fornecendo uma base para otimizar soluções contra parâmetros econômicos e tecnológicos.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

COMPARATIVE ANALYSIS OF SINGLE AND MULTI-AGENT LARGE LANGUAGE MODEL ARCHITECTURES FOR DOMAIN-SPECIFIC TASKS IN WELL CONSTRUCTION

Vitor Brandão Sabbagh

July/2025

Advisor: Geraldo Bonorino Xexéo

Department: Systems Engineering and Computer Science

This article explores the application of large language models (LLM) in the oil and gas sector, specifically within well construction and maintenance tasks. The study evaluates the performances of a single-agent and a multi-agent LLM-based architecture in processing different tasks, offering a comparative perspective on their accuracy and the cost implications of their implementation. The results indicate that multi-agent systems offer improved performance in question and answer tasks, with a truthfulness measure 28% higher than single-agent systems, but at a higher financial cost. Specifically, the multi-agent architecture incurs costs that are, on average, 3.7 times higher than those of the single-agent setup due to the increased number of tokens processed. Conversely, single-agent systems excel in text-to-SQL (Structured Query Language) tasks, particularly when using Generative Pre-Trained Transformer 4 (GPT-4), achieving a 15% higher score compared to multi-agent configurations, suggesting that simpler architectures can sometimes outpace complexity. The novelty of this work lies in its original examination of the specific challenges presented by the complex, technical, unstructured data inherent in well construction operations, contributing to strategic planning for adopting generative AI applications, providing a basis for optimizing solutions against economic and technological parameters.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Experimento 2

1.1 Methodology (generated)

1.1.1 Overview

This chapter describes the experimental methodology used to evaluate large language model (LLM) agents and workflows for answering questions in the oil well operations domain. The experiment integrates multiple LLM configurations, agent architectures, and retrieval-augmented generation (RAG) tools, leveraging Petrobras datasets.

[POR ÚLTIMO, INSERIR OBJETIVOS DO EXPERIMENTO E HIPOTESES] *<insert brief summary of research objectives and hypotheses here>*

1.1.2 Experimental Workflow

Dataset Preparation

The experimental workflow was designed to provide a thorough and reproducible evaluation of language model agents within oil well operations. The process begins with the careful preparation of the dataset, which is composed of questions and corresponding ground truth answers derived from a diverse range of operational records, incident reports, and lessons learned. To ensure the quality and relevance of the data, questions undergo a filtering and preprocessing phase where clarity, diversity, and alignment with real-world scenarios are prioritized. This includes removing duplicates, standardizing terminology, and confirming that each question is properly paired with an accurate answer. The dataset is further validated for completeness and consistency, ensuring it represents the full spectrum of operational challenges, such as safety, cementing, and intervention scenarios.

Model and Setup Selection

Following dataset preparation, the experimental design incorporates a variety of agent architectures. These include approaches where questions are routed to specialized agents, single-agent systems that centralize all reasoning and retrieval, and multi-agent frameworks that leverage collaboration among specialized agents under a supervisory structure. Each of these configurations is evaluated using different language models, allowing for a comprehensive assessment of how model choice and agent setup influence performance. The agents are also provided with access to advanced retrieval tools and domain-specific knowledge bases, enabling them to draw on a broad foundation of operational expertise.

Execution Loop

The core of the experimental workflow is an execution loop depicted in Algorithm 1. For each combination of question, agent setup, and language model, the system systematically loads the relevant data, configures the agent, and executes the workflow. Throughout this process, all responses and intermediate reasoning steps are meticulously logged. This approach not only ensures systematic coverage of all experimental conditions but also provides full traceability for subsequent analysis. The automation of these procedures guarantees consistency and reproducibility, while the comprehensive logging facilitates in-depth evaluation and comparison of agent performance across a range of operational scenarios.

Algorithm 1 Experiment Execution Loop

Require: questions, setups, models
Ensure: results

```
function RunExperiment
    results ← {}
    for all question ∈ questions do
        ground_truth ← question.ground_truth
        for all setup ∈ setups do
            for all model ∈ models do
                agent ← InitializeAgent(setup, model)
                response ← agent.ProcessQuestion(question)
                metrics ← EvaluateResponse(response, ground_truth)
                results[question, setup, model] ← {
                        "response" : response,
                        "metrics" : metrics,
                        "execution_trace" : agent.trace
                }
            end for
        end for
    end for
    return AggregateResults(results)
end function
```

Evaluation and Metrics

Following the execution of all experimental combinations, a comprehensive evaluation framework is applied to assess agent performance. The system calculates a suite of quantitative metrics for each question, setup, and model combination by comparing the generated answers against the established ground truth. These metrics include standard performance indicators such as accuracy, precision, recall, and F1 score, which provide a multifaceted view of response quality. For open-ended questions where binary correctness measures are insufficient, a confusion matrix approach is implemented to capture nuances in answer quality and content coverage. Additionally, the system measures answer size ratio relative to ground truth, offering insights into model verbosity and conciseness. These metrics are then aggregated across different dimensions to enable meaningful comparisons between agent architectures and language models, revealing patterns in performance across various operational scenarios and question types.

Reproducibility and Quality Control

To ensure scientific rigor and reproducibility, the experimental methodology incorporates robust tracking of all environmental variables and configuration parameters. The system maintains detailed logs of the computational

environment, including software versions, dependency specifications, and hardware characteristics that might influence results. All experimental parameters, from model identifiers to dataset specifications, are systematically recorded alongside the results they generate. Throughout the experimental process, periodic validation checks are performed to maintain data integrity and result consistency, with anomalies flagged for investigation. This comprehensive approach to reproducibility not only facilitates verification of findings but also enables future extensions of the research with comparable baselines. The quality control measures embedded in the workflow ensure that conclusions drawn from the experiments rest on a foundation of methodological soundness and data reliability.

<insert workflow diagram or pseudocode here to illustrate the above stages>

1.1.3 Data Sources

The experimental evaluation relies on a carefully curated collection of data sources that represent the diverse knowledge domains relevant to oil well operations. At the core of the experiment is a comprehensive questions dataset containing structured entries that simulate real-world queries an operator might encounter. This dataset was developed through extensive collaboration with domain experts and analysis of historical operational records. Each entry in the dataset contains a question formulated in natural language, a unique identifier, categorical metadata to facilitate analysis, and a corresponding ground truth answer validated by subject matter experts. The questions span various complexity levels, from factual inquiries to complex reasoning scenarios that require integration of multiple knowledge sources.

To provide the language models with the necessary domain knowledge, the experiment incorporates several specialized knowledge bases that reflect different aspects of oil well operations:

- **Knowledge Bases and Tools**:
  - **Lessons**: A repository of knowledge items capturing insights, best practices, and technical know-how from past oil well operations. These lessons represent institutional memory and expertise accumulated over years of operational experience.
  - **Alertas SMS**: A collection of safety alerts and incident reports documenting past events, near-misses, and accidents, providing critical safety information and preventative measures.
  - **Cronoweb**: A comprehensive database of scheduling information and intervention records, detailing maintenance activities, equipment deployments, and operational timelines.
  - **SITOP**: Detailed daily operational logs from drilling rigs, containing technical parameters, operational decisions, and situational reports from active drilling operations.

These knowledge sources were preprocessed to ensure consistency, remove sensitive information, and optimize retrieval performance. The integration of these diverse data sources enables a holistic evaluation of how language model agents navigate the complex informational landscape of oil well operations, from technical specifications to safety protocols and historical precedents.

<insert table or image summarizing datasets and tools here>

1.1.4 System Architecture

The experimental system was implemented using modern Python frameworks specialized for language model orchestration and agent workflows. The architecture leverages the LangChain and LangGraph ecosystems,

which provide robust foundations for building complex language model applications with multiple components and state management. This subsection details the modular design of the system, highlighting how different components interact to enable systematic evaluation of language model agents in oil well operations.

Experiment Orchestration

At the core of the system architecture is an experiment orchestration layer responsible for coordinating the entire evaluation process. This component manages the loading of questions from the dataset, systematically iterates through different model and setup combinations, and ensures proper logging of results. The orchestrator maintains experiment state across multiple runs, handles error recovery, and implements checkpointing to allow for resumption of long-running experiments. By centralizing control flow, this component ensures that all experimental conditions are tested consistently and that results are captured in a standardized format for subsequent analysis.

Agent Workflow Frameworks

The system implements multiple agent workflow frameworks to evaluate different approaches to question answering in the oil well domain. These frameworks define the flow of information and decision-making processes within and between language model agents. The implemented workflows include a Linear-Flow with Router (CORTEX) that directs questions to specialized processing paths, a Single-Agent approach that centralizes all reasoning and tool use, and a Multi-Agent Supervisor framework that coordinates multiple specialized agents. Each workflow is defined declaratively, specifying the sequence of operations, decision points, and information exchange patterns that govern agent behavior during question processing.

Nodes and Tool Integration

The system architecture includes specialized nodes that implement specific reasoning steps and tool-calling logic. These nodes serve as the building blocks of agent workflows, encapsulating discrete functionality such as question analysis, knowledge retrieval, and answer synthesis. The tool integration layer provides agents with access to external knowledge sources through a standardized interface, enabling semantic search over domain-specific corpora, structured data queries, and other specialized operations. This modular approach to tool integration allows for consistent evaluation of how different agent architectures leverage available tools and knowledge sources.

Prompt Engineering and System Messages

A critical component of the architecture is the prompt engineering layer, which defines the instructions and context provided to language models. This includes carefully crafted system messages that establish the role and capabilities of each agent, prompt templates that structure inputs consistently across experimental conditions, and few-shot examples that guide model behavior. The system maintains a library of prompt variants optimized for different tasks within the question-answering workflow, ensuring that each agent receives appropriate guidance while maintaining experimental control.

State Management and Metrics

The architecture incorporates a comprehensive state management system that tracks the progress of experiments, maintains contextual information across agent interactions, and captures intermediate reasoning steps.

This component is tightly integrated with the metrics calculation subsystem, which computes performance indicators in real-time as experiments progress. The metrics framework implements various evaluation approaches, from simple accuracy measures to sophisticated semantic similarity calculations, providing multi-dimensional assessment of agent performance. All experimental data, including intermediate states and final results, is persisted in structured formats to enable both immediate feedback and in-depth post-experiment analysis.

<insert system architecture diagram here>

1.1.5 Experimental Setups

To comprehensively evaluate language model performance in well construction operations, the experiment employed multiple agent architectures and model configurations. This subsection details the different experimental setups, highlighting their design principles, operational characteristics, and the rationale behind their selection. The experimental design deliberately incorporates contrasting approaches to agent architecture, enabling comparative analysis of different strategies for complex question answering in specialized domains.

Linear-Flow

The Linear-Flow architecture represents the simplest RAG design, where user input is processed in a strictly sequential manner.

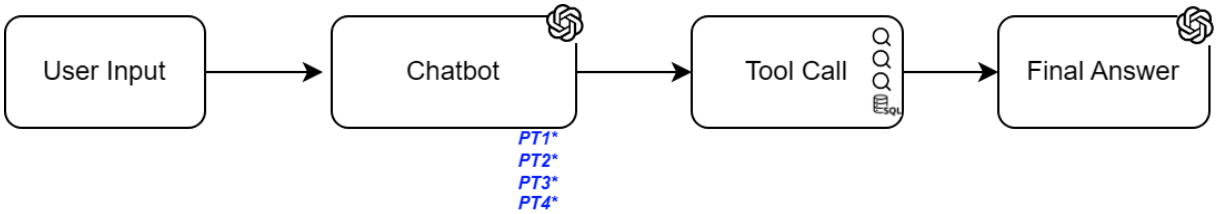

Figura 1.1: Linear-Flow architecture. PT1 indicates Prompt for Tool 1 and so on.

Linear-Flow with Router

[VERSÃO 1] The Linear-Flow with Router architecture implements a sequential processing pipeline with routing, in order to divide and reduce tool instruction prompts. In this setup, questions first pass through a router that analyzes the query content and determines the most appropriate specialized node for the user query at hand.

[VERSÃO 2] The Linear-Flow with Router paradigm extends the basic linear flow by introducing a routing mechanism that enables the distribution of tool instruction prompts into specialized nodes with smaller prompts. User questions first pass through a router that determines the most appropriate specialized node. As illustrated in Figure 1.2, instead of a single node generating different types of retrieval queries, we have several nodes (or sub-queries), each with its own specialized tool. Each sub-query is then processed independently by separate tool invocations.

This approach offers several advantages:

- **Increased Throughput:** By distributing sub-tasks across multiple tools, the system can handle more complex or multi-faceted user requests efficiently.
- **Specialization:** Each tool can be tailored to address a specific aspect of the user's query, allowing for more accurate and relevant results.

- **Scalability:** The architecture naturally supports scaling, as additional tools can be added to handle more sub-queries or specialized tasks.

In practice, the router acts as an orchestrator, analyzing the user input and generating multiple targeted queries (PT1*, PT2*, PT3*, PT4* in the figure). These queries are dispatched to their respective tools, and the results are aggregated to form the final answer. This method is particularly effective for tasks that can be decomposed into independent components, such as multi-part questions or workflows requiring different types of expertise.

Compared to the standard linear flow, the use of a router introduces additional complexity in query generation and result aggregation but enables a significant boost in system flexibility and performance.

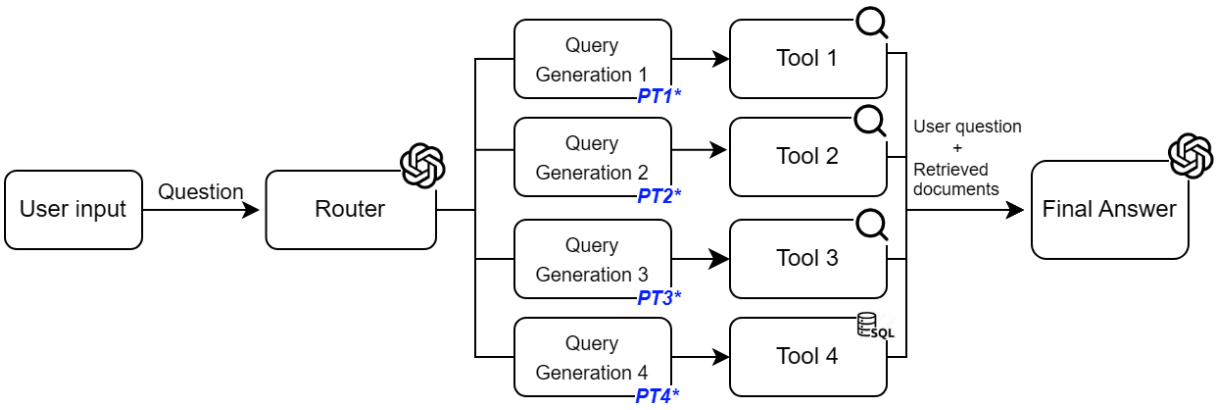

Figura 1.2: Linear-Flow with Router architecture.

Single-Agent

The Single-Agent approach represents a centralized architecture where a single language model agent handles the entire question-answering process. This agent has access to the full suite of retrieval tools and knowledge sources, making independent decisions about which tools to invoke and how to synthesize information into coherent answers. The design emphasizes end-to-end reasoning within a unified context, allowing the model to maintain a consistent understanding throughout the process. This approach tests the capability of language models to manage complex workflows autonomously, balancing between exploration of different knowledge sources and focused answer generation without the overhead of inter-agent communication.

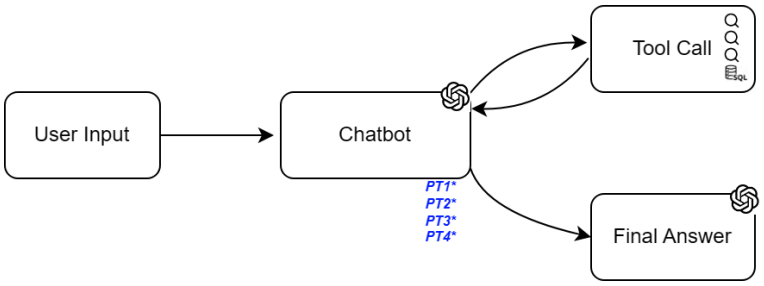

Figura 1.3: Single-Agent architecture

Multi-Agent Supervisor

The Multi-Agent Supervisor setup implements a collaborative approach where multiple specialized agents work together under the coordination of a supervisor agent. Each specialized agent focuses on a specific domain of

knowledge or reasoning skill, such as retrieval, analysis, or explanation generation. The supervisor agent orchestrates the collaboration, delegating subtasks to appropriate specialized agents, integrating their contributions, and ensuring coherence in the final answer. This architecture explores the potential benefits of distributed cognition, where complex reasoning is decomposed into manageable components handled by purpose-built agents. The framework includes mechanisms for resolving conflicts between agents and synthesizing potentially divergent perspectives into unified responses.

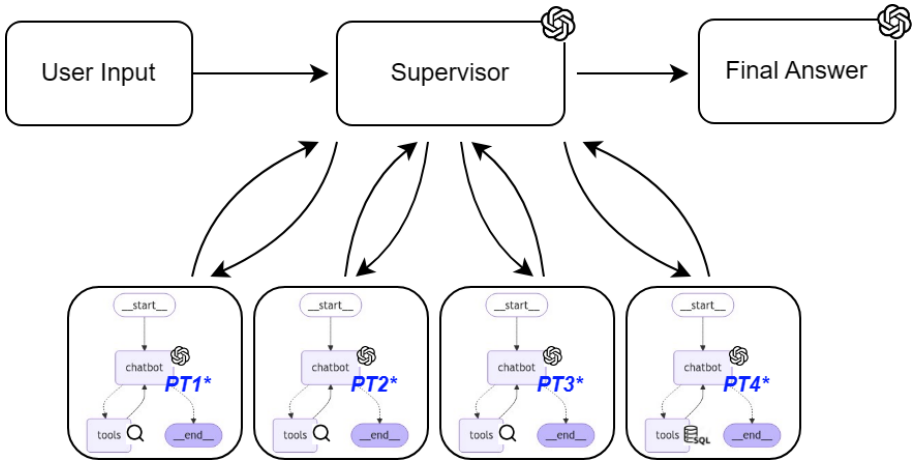

Figura 1.4: Multi-Agent setup with one supervisor and 4 specialist agents.

<insert table summarizing experimental setups and models here>

1.1.6 Execution Details

The experiment was driven by a script without manual intervention during the evaluation process. A main execution loop systematically iterated through all combinations of questions, agent setups, and language models defined in the experimental design.

Tool Integration and Knowledge Access

During execution, the agent systems accessed domain-specific knowledge through a standardized tool interface layer. This layer provided consistent access patterns across all experimental configurations, ensuring that differences in performance could be attributed to agent architecture rather than variations in knowledge availability. The tool integration framework supported a diverse range of knowledge access methods, including semantic search over unstructured text corpora, structured queries against relational databases, and specialized information extraction routines tailored to the oil well operations domain. Each tool invocation was executed within a controlled environment that captured performance metrics such as latency and resource utilization, providing additional dimensions for analysis beyond answer correctness. The standardization of tool interfaces across agent architectures was a critical design decision that enabled fair comparison while still allowing each architecture to implement its own strategy for tool selection and result interpretation.

Comprehensive Logging and Observability

A cornerstone of the experimental methodology was the implementation of comprehensive logging throughout the execution process. The system captured detailed records of each step in the question-answering workflow, from initial question parsing to final answer generation. These logs included intermediate reasoning steps, tool invocations with their inputs and outputs, and internal state transitions within the agent systems. All

experimental artifacts were persisted in structured formats that facilitated both automated analysis and manual inspection. The logging system implemented a hierarchical organization that linked high-level metrics to the detailed execution traces that produced them, enabling root cause analysis of performance patterns. This observability infrastructure was essential for understanding not just what results were produced, but how and why different agent architectures arrived at their answers, providing insights into their reasoning processes and failure modes.

<insert code snippet or pseudocode of main execution loop here>

1.1.7 Evaluation Metrics

The evaluation of each experimental run is grounded in a comprehensive set of metrics designed to capture both the correctness and the quality of the system's responses. Standard quantitative measures such as accuracy, precision, recall, and F1 score are calculated by comparing the answers generated by the agent systems to the established ground truth for each question. These metrics provide a multifaceted view of performance, indicating not only how often the system produces correct answers but also how well it balances false positives and false negatives.

For questions that are open-ended or less amenable to binary correctness, the evaluation framework employs a confusion matrix approach. This allows for a more nuanced assessment, capturing partial correctness and the degree to which the system's response overlaps with the expected content. Additionally, the methodology includes the calculation of the answer size ratio, which measures the verbosity of the generated answer relative to the ground truth. This metric helps to identify tendencies toward overly concise or excessively verbose responses, offering further insight into the models' behavior and suitability for practical deployment.

1.1.8 Reproducibility

Ensuring reproducibility is a cornerstone of the experimental methodology. To this end, every aspect of the computational environment is meticulously documented. This includes recording the exact Python version used, as well as all package dependencies and their respective versions. Hardware specifications, such as processor type and available memory, are also logged to account for any potential influence on experimental outcomes.

Beyond the environment, the system systematically records all configuration parameters relevant to each experimental run. This encompasses model names, hyperparameters, and dataset paths, as well as any other settings that might affect the results. By maintaining this comprehensive record, the methodology enables other researchers to replicate the experiments precisely or to build upon them with confidence that baseline conditions are well understood and controlled.

1.1.9 Limitations

While the experimental methodology strives for rigor and comprehensiveness, several limitations must be acknowledged. One key limitation concerns the coverage of the dataset: although the question set is carefully curated to represent a broad range of operational scenarios, it may not capture the full diversity of real-world challenges encountered in oil well operations. Similarly, the models and agent architectures evaluated are constrained by the available computational resources and the current state of language modeling technology, which may limit their ability to generalize beyond the scenarios tested.

Another limitation arises from the reliance on ground truth answers, which, despite expert validation, may still reflect subjective judgments or incomplete information in certain cases. Furthermore, the evaluation metrics,

while robust, may not fully capture qualitative aspects of answer usefulness or clarity, especially in highly technical or ambiguous situations. Recognizing these limitations is essential for interpreting the results and for guiding future research aimed at addressing these gaps.

1.1.10 Summary

This methodology provides a systematic framework for comparing different agent architectures and large language models in the context of complex question answering for oil well operations. By integrating rigorous evaluation metrics, robust reproducibility practices, and a clear acknowledgment of limitations, the approach enables meaningful insights into the strengths and weaknesses of various system designs. The findings derived from this methodology can inform both the deployment of language model agents in operational settings and the ongoing development of more capable and reliable AI systems for specialized industrial domains.

1.2 Resultados e Discussão

1.2.1 Performance

[GERAR TEXTO AQUI]

... ANTONIAK *et al.* (2016)

...

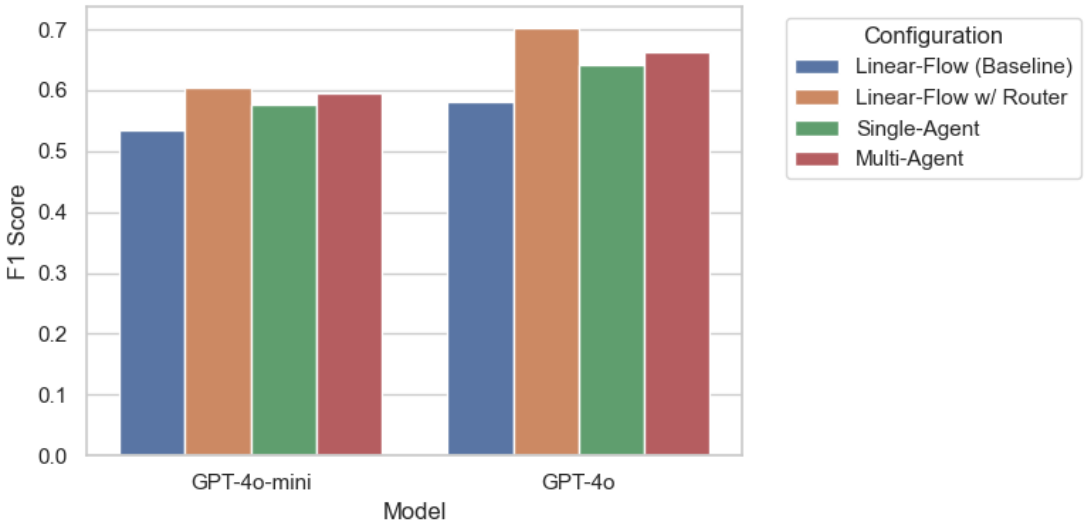

Figura 1.5: Best F1 score by model and configuration.

...

...

...

...

F1 Score

[GERAR TEXTO AQUI]

...

...

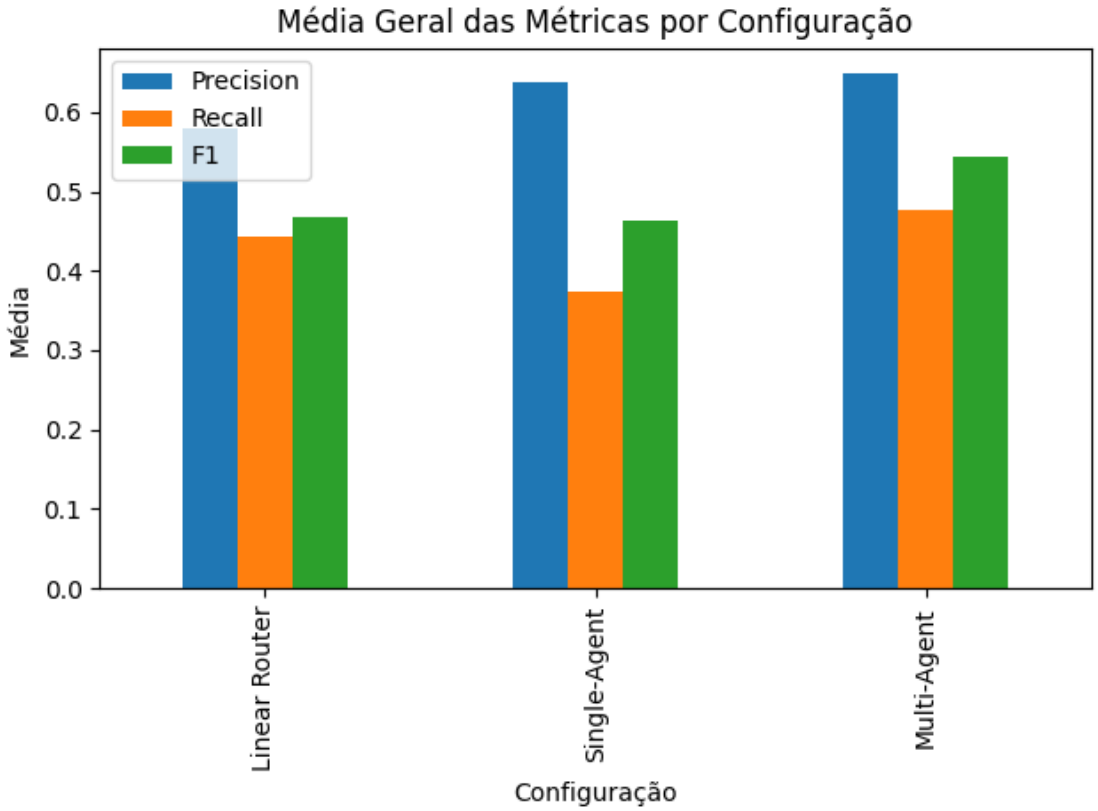

Figura 1.6: Precisão, recall e f1 por configuração. [GERAR GRÁFICO NOVO AQUI]

Figura 1.7: Image 1 caption

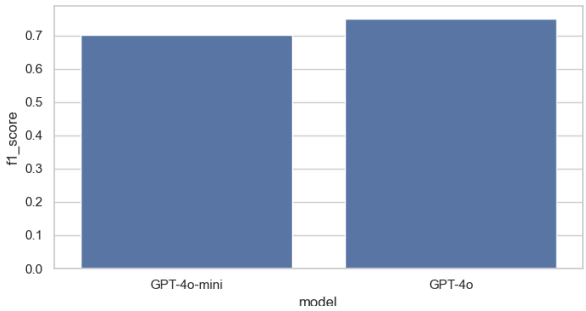

Figura 1.8: Image 2 caption

...
[GERAR TEXTO AQUI]
...
...
...
[GERAR TEXTO AQUI]
...
...
...
[GERAR TEXTO AQUI]
...
...
...

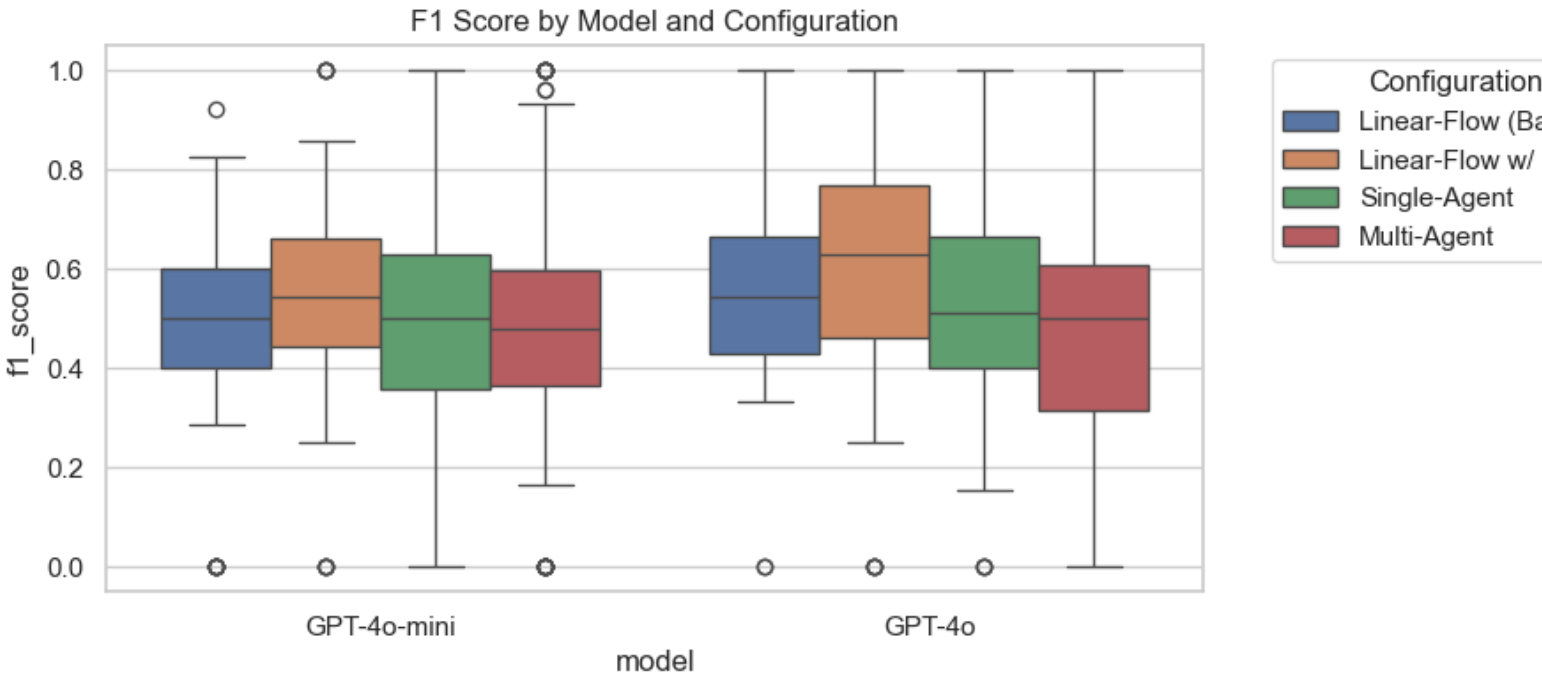

Figura 1.9: F1 Score distribution by model and configuration of agents

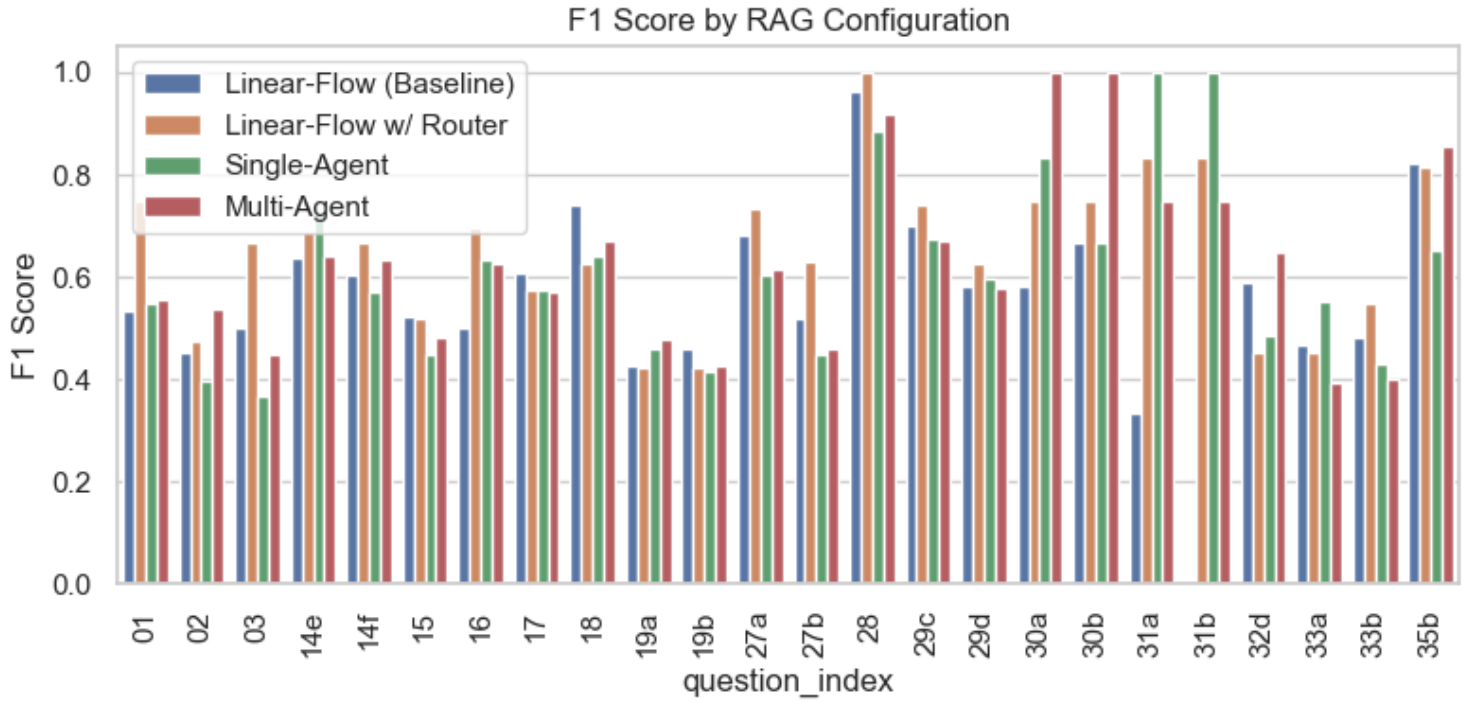

Figura 1.10: Enter Caption

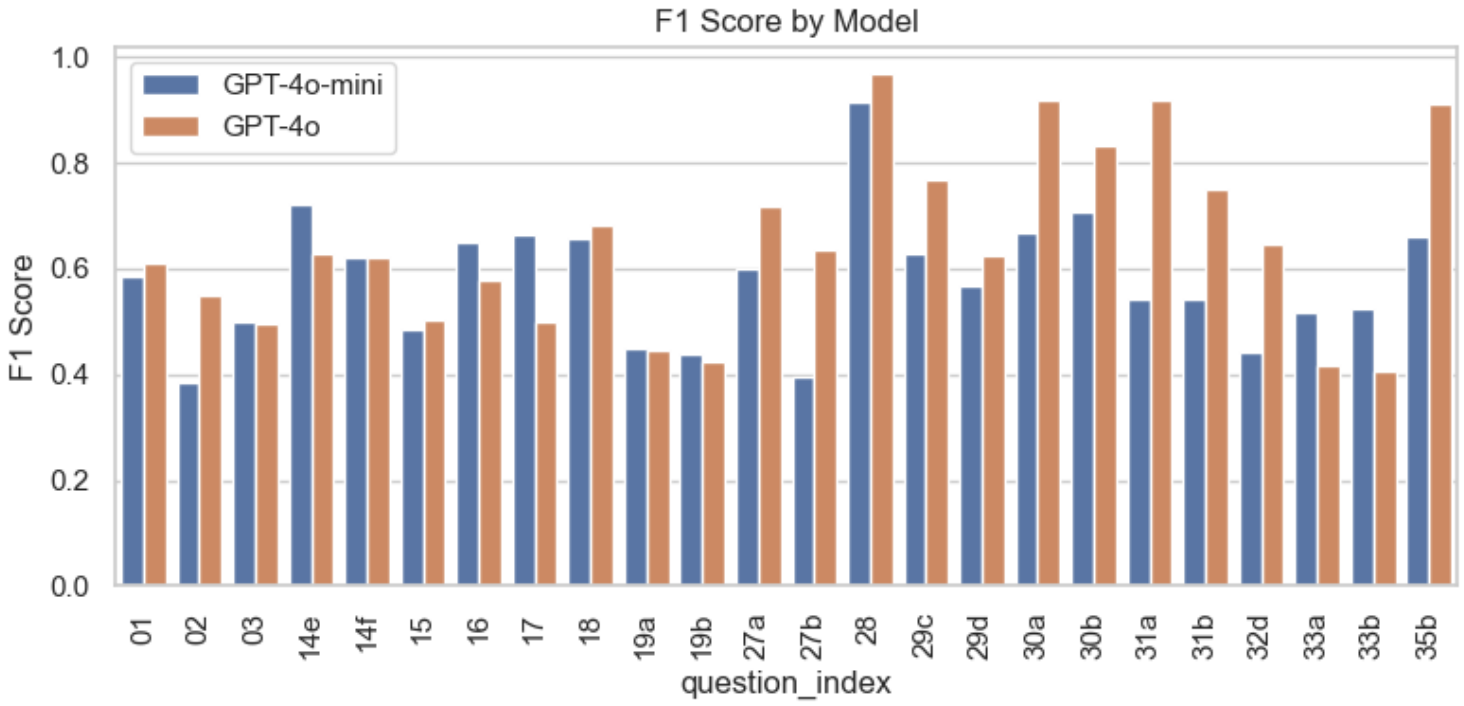

Figura 1.11: Enter Caption

[GERAR TEXTO AQUI]

...

...

...

Precisão

[GERAR TEXTO AQUI]

...

...

...

[GERAR TEXTO AQUI]

...

...

...

[GERAR TEXTO AQUI]

...

...

Recall

[GERAR TEXTO AQUI]

...

...

[GERAR TEXTO AQUI]

...

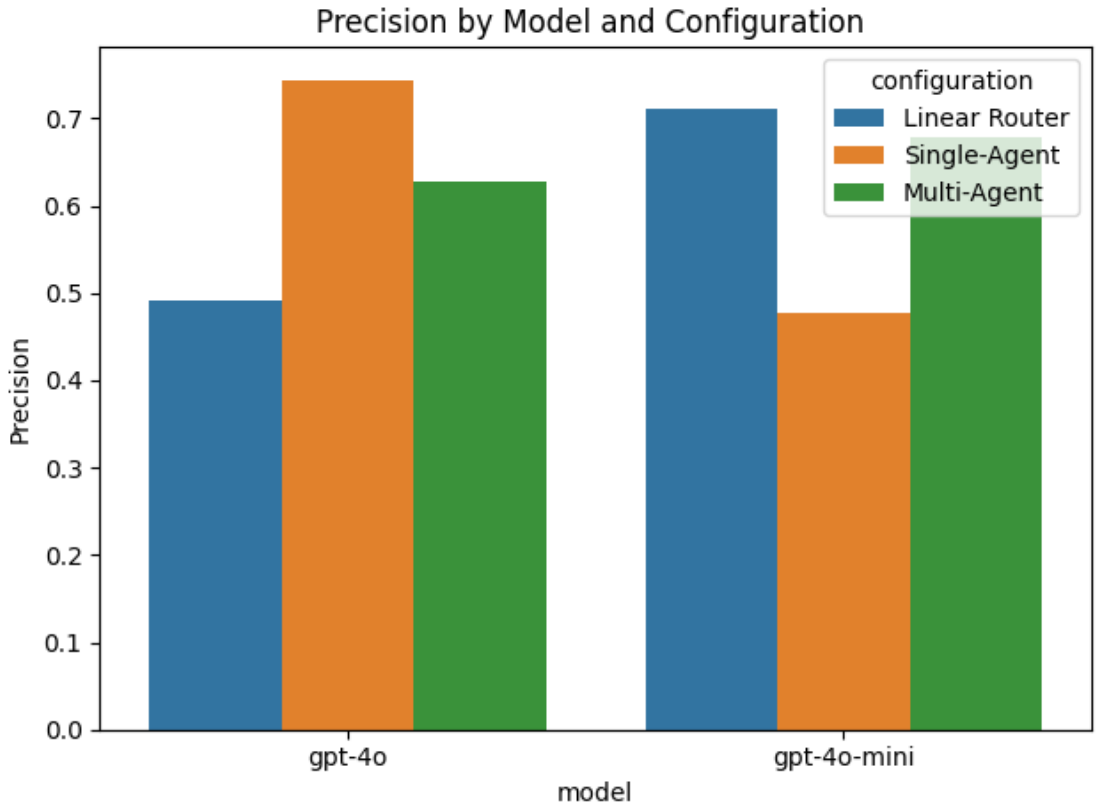

Figura 1.12: Precisão por modelo e configuração.

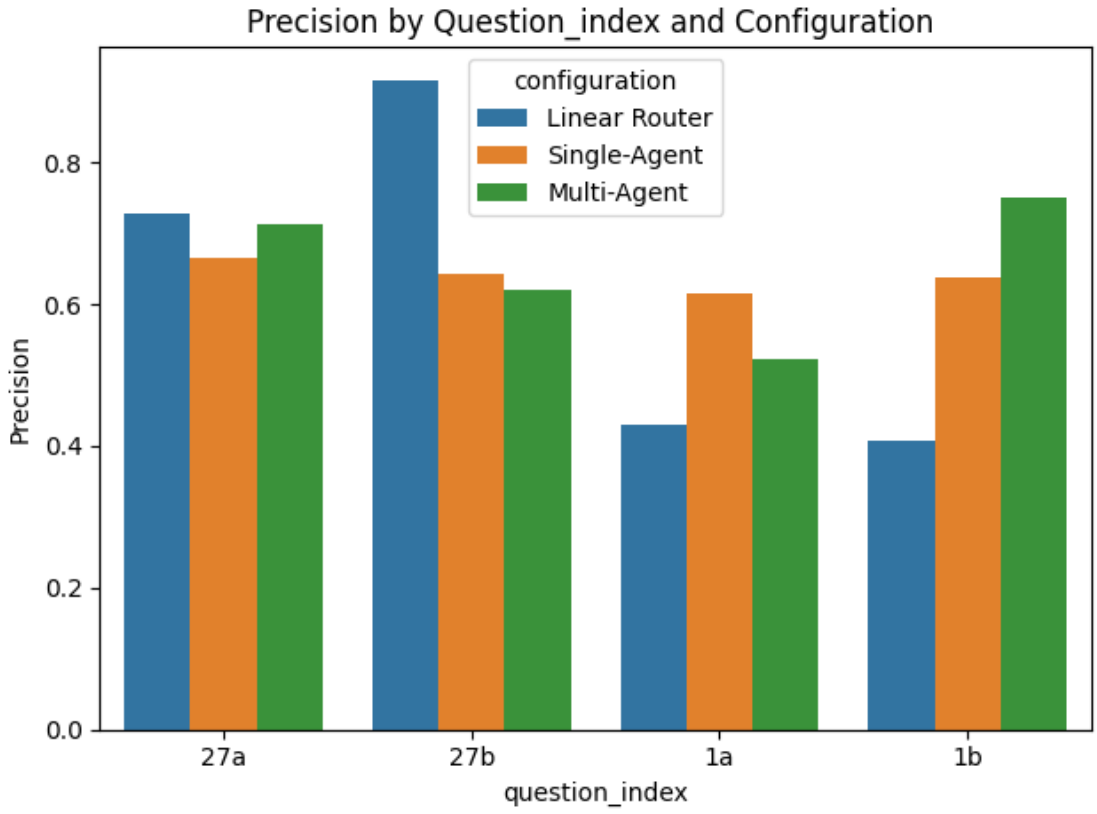

Figura 1.13: Precisão por pergunta e configuração.

...

...

[GERAR TEXTO AQUI]

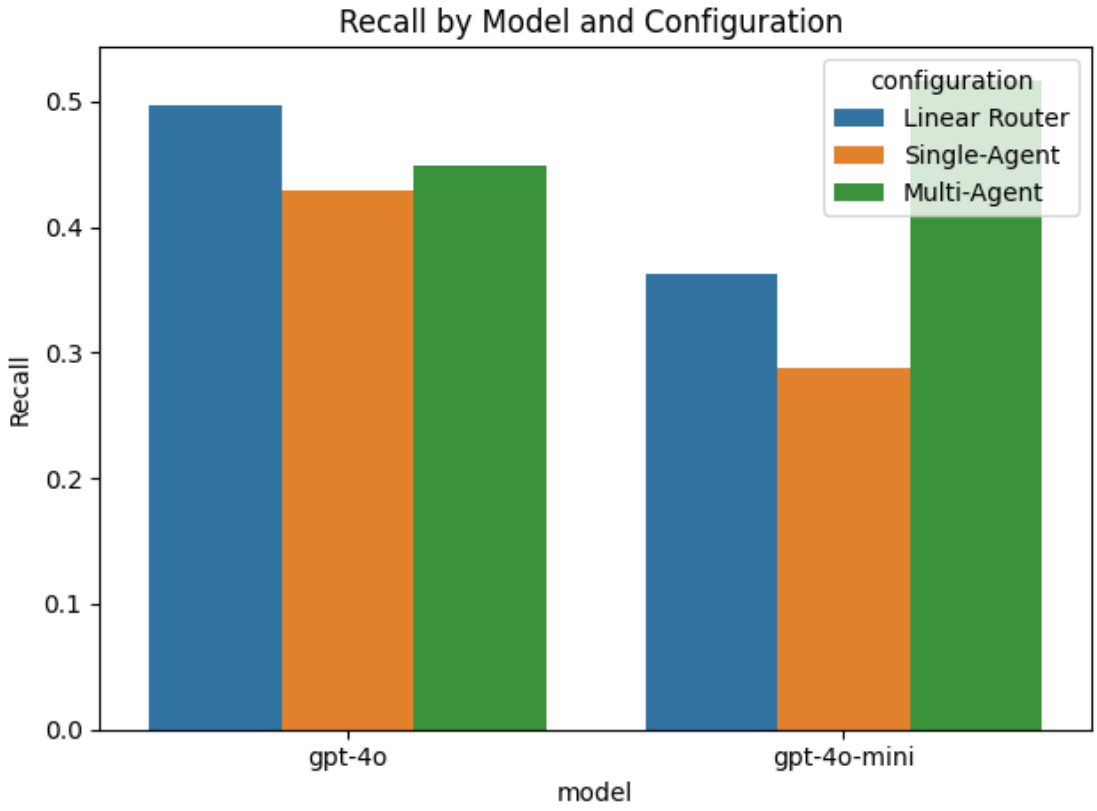

Figura 1.14: Recall por modelo e configuração.

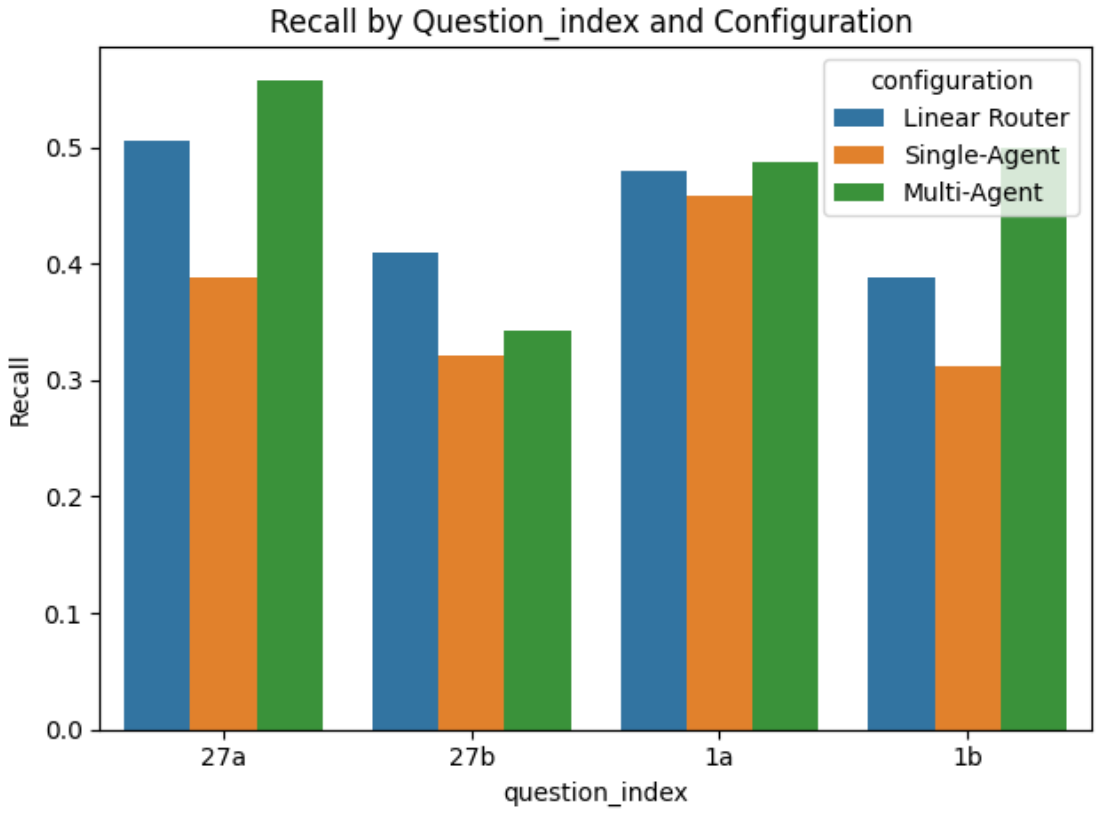

Figura 1.15: Recall por pergunta e configuração.

...

...

1.2.2 Linear-Flow

In this setup, the user's query is handled by a single LLM step, which carries all the instructions (PT1, PT2, PT3 and PT4, as depicted in 1.1) required for the generation of various types of search queries. These instruction prompts are often quite long, as they are carefully crafted to produce high-quality queries for the vector store. Due to the aggregation of all instruction prompts within a single LLM invocation, the resulting context becomes notably extensive. This can lead to performance degradation as the context length increases [O QUE PODE SER VISTO NO GRAFICO TAL EM CONTRASTE COM O SETUP TAL QUE DIVIDE OS PROMPTS EM PARTES].

...

...

...

1.2.3 Linear-Flow with Router

[GERAR TEXTO AQUI]

...

...

...

1.2.4 Single-Agent

[GERAR TEXTO AQUI]

...

...

...

1.2.5 Multi-Agent

[GERAR TEXTO AQUI]

...

...

...

Referências Bibliográficas

ANTONIAK, M., DALGLIESH, J., VERKRUYSE, M., et al. "Natural language processing techniques on oil and gas drilling data", *Society of Petroleum Engineers - SPE Intelligent Energy International Conference and Exhibition*, 2016. doi: 10.2118/181015-MS.

Apêndice A

Um apêndice

Segundo a norma da ABNT (Associação Brasileira de Normas Técnicas), a definição e utilização de apêndices e anexos seguem critérios específicos para a organização de documentos acadêmicos e técnicos.

Apêndice: O apêndice é um texto ou documento elaborado pelo autor do trabalho com o objetivo de complementar sua argumentação, sem que seja essencial para a compreensão do conteúdo principal do documento. O uso de apêndices é indicado para incluir dados detalhados como questionários, modelos de formulários utilizados na pesquisa, descrições extensas de métodos ou técnicas, entre outros. Os apêndices são identificados por letras maiúsculas consecutivas, travessão e pelos respectivos títulos. A inclusão de apêndices visa a fornecer informações adicionais que possam ajudar na compreensão do estudo, mas cuja presença no texto principal poderia distrair ou desviar a atenção do leitor dos argumentos principais.

Anexo A

Um Anexo

Segundo a norma da ABNT (Associação Brasileira de Normas Técnicas), a definição e utilização de apêndices e anexos seguem critérios específicos para a organização de documentos acadêmicos e técnicos.

Anexo: O anexo, por sua vez, consiste em um texto ou documento não elaborado pelo autor, que serve de fundamentação, comprovação e ilustração. O uso de anexos é apropriado para materiais como cópias de artigos, legislação, documentos históricos, fotografias, mapas, entre outros, que tenham relevância para o entendimento do trabalho do autor. Assim como os apêndices, os anexos são identificados por letras maiúsculas consecutivas, travessão e pelos respectivos títulos. Eles são utilizados para enriquecer o trabalho com informações de suporte, garantindo que o leitor tenha acesso a documentos complementares importantes para a validação dos argumentos apresentados no texto principal.