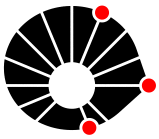

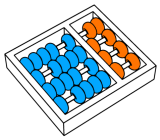

Moderando Conteúdo Textual via Sistema Multiagentes com Modelos de Linguagem

Tiago Feliciano Gomes *Julio Cesar dos Reis*

Relatório Técnico - IC-PFG-25-03
Projeto Final de Graduação
2025 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Moderando Conteúdo Textual via Sistema Multiagentes com Modelos de Linguagem

Tiago Feliciano Gomes Julio Cesar dos Reis *

Resumo

Respostas automáticas geradas artificialmente para consumidores em sistemas de comércio eletrônico demandam diversos cuidados para preservar aspectos éticos e de privacidade. Esse contexto exige moderação e análise específica sobre o conteúdo das respostas geradas. Contudo, devido ao alto volume de questões, soluções computacionais são altamente necessárias apesar dos desafios em se lidar com diversidade léxica, semântica e de idiomas. Este estudo propõe uma solução de arquitetura multiagentes baseados em LLMs projetada para realizar a moderação automática de respostas em sistemas de atendimento ao cliente. Em nossa solução, cada agente é responsável por uma sub-tarefa em relação a avaliação e geração automatizada de respostas (revisão semântica, verificação de aderência contextual, recomendação de melhorias, reescrita e decisão final) dentro de um fluxo iterativo e colaborativo entre agentes. A arquitetura proposta visa maximizar a confiabilidade, otimizar o custo computacional ao interromper revisões desnecessárias e assegurar transparência e auditabilidade em cada etapa, demonstrando-se efetividade na melhoria contínua da qualidade das respostas frente aos desafios de gerações de respostas imprecisas, incompletas ou inadequadas que comprometem a experiência do usuário.

1 Introdução

A crescente adoção de modelos de Inteligência Artificial **(IA)** por empresas para responder às perguntas de consumidores destaca a importância da qualidade dessas respostas para a experiência do usuário. Modelos de linguagem, como o *ChatGPT*[1] da OpenAI, demonstraram capacidades notáveis em compreender e gerar texto coerente em linguagem natural, destacando-se em tarefas de Perguntas e Respostas **(QA)**. No entanto, muitos desses modelos são de código fechado, limitando a transparência, personalização e adaptabilidade às necessidades específicas de diferentes domínios.

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.

[1] `https://chatgpt.com/`

Em contexto de sistemas de comercio eletrônico, a diversidade de **idiomas**, **tipos de perguntas** e **categorias de produtos** nas interações com os consumidores exige sistemas capazes de lidar com uma ampla variedade de variações linguísticas e contextuais. Isso ressalta a necessidade de abordagens que possam adaptar-se efetivamente a diferentes cenários sem depender de grandes volumes de dados de treinamento específicos para essas tarefas.

Apesar dos avanços significativos na área de Processamento de Linguagem Natural **(PLN)**, os modelos de linguagem atuais enfrentam limitações que impactam diretamente a qualidade das respostas geradas. Uma das principais questões é a tendência desses modelos de produzir respostas que, embora gramaticalmente corretas, podem carecer de fundamentação no contexto fornecido para o produto, fenômeno conhecido como **"alucinação"** [1]. Essa característica compromete a confiabilidade das respostas geradas aos usuários finais, especialmente em contextos que exigem precisão e aderência estrita às informações disponíveis.

Muitos modelos de linguagem são desenvolvidos com base em dados **predominantemente em inglês**, o que limita sua efetividade em outras línguas, como o português e o espanhol [2, 3, 4]. Essa limitação linguística impede a ampla aplicação desses modelos em contextos multilíngues, restringindo seu uso em mercados diversos e globalizados.

A incapacidade dos modelos de reconhecer quando **não possuem informações suficientes** para responder adequadamente a uma pergunta é uma preocupação crescente. Estudos indicam que, em vez de admitir a falta de conhecimento, os modelos tendem a fornecer respostas imprecisas ou incorretas [5], o que pode comprometer a confiança dos usuários nas soluções baseadas em IA. Diante dessas limitações, torna-se evidente a necessidade de desenvolver **soluções mais robustas e adaptáveis**, capazes de mitigar os problemas identificados e aprimorar a qualidade das respostas geradas por sistemas de Perguntas e Respostas baseados em inteligência artificial.

Este trabalho propõe uma arquitetura composta por **múltiplos agentes de IA especializados**, incluindo revisores semânticos e contextuais, um recomendador de melhorias, um reescritor e um decisor final. Esses agentes interagem entre si por meio de um **contexto compartilhado**, permitindo a troca de informações e a construção de um **fluxo de trabalho iterativo**. O processo inicia-se com a avaliação da resposta original pelos revisores, que atribuem pontuações e justificativas. Com base nessas avaliações, o recomendador propõe melhorias na resposta, que são implementadas pelo reescritor. A nova resposta proposta é então reavaliada, e o decisor determina se ela atende aos critérios estabelecidos ou se novas revisões são necessárias.

Esse encadeamento permite mitigar de forma precisa possíveis falhas comuns para nesse tipo de sistema, como alucinações ou descontextualizações, gerar sugestões direcionadas para o problema, reformular respostas originalmente incorretas e deixá-las

sem resposta caso não sejam capazes de atender os critérios de qualidade preestabelecidos, graças a um ciclo de melhoria iterativa que pode ser reiniciado até um máximo de três vezes, se necessário.

Para implementar essa arquitetura, utilizamos o *AG2*[2], uma ferramenta que facilita a integração entre agentes, utilizando recursos como ferramentas `(tools)`, funções `(functions)`, variáveis de contexto `(context variables)` e transições `(transitions)` no padrão de conversação em grupo `(group chat)`[3]. Essa abordagem modular e colaborativa permite que o sistema **identifique e corrija deficiências nas respostas**, além de facilitar a escalabilidade e a adaptabilidade do sistema a diferentes domínios ou aplicações.

Este relatório está organizado da seguinte forma: A Seção 2 apresenta a fundamentação teórica para os conceitos implementados. A Seção 3 descreve a proposta de solução. A Seção 4 apresenta os experimentos conduzidos e a avaliação experimental. A Seção 5 discute os resultados obtidos. Por fim, a Seção 6 apresenta as conclusões.

2 Fundamentos

2.1 Agentes de IA e LLMs

Historicamente, os primeiros *conversational agents* [6] surgiram na década de 1960 com o **ELIZA**, um sistema de regras simples baseado em padrões de texto. Tais agentes clássicos possuíam **capacidades limitadas de entendimento** e frequentemente geravam respostas padronizadas sem qualquer profundidade semântica, apenas a partir de condições pré-estabelecidas.

Agentes de IA são por definição sistemas de software projetados para executar tarefas de **forma autônoma**, interagindo com usuários ou outros sistemas por meio de linguagem natural, percepção do ambiente e tomada de decisão baseada em objetivos claros e instruções pré-definidas [7]. Com o advento dos Large Language Models **(LLMs)** [8], tornou-se possível adotar esses agentes de habilidades avançadas de compreensão e geração de texto, graças ao treinamento em enorme diversidade de dados e contextos.

A arquitetura de *Transformer*, introduzida por Vaswani *et al.* (2017) [9], utiliza **mecanismos de atenção** para modelar relações de longo alcance em sequências de entrada. Essa abordagem permite processar eficientemente contextos extensos e gerar respostas coerentes, superando em desempenho arquiteturas anteriores como **RNNs** e **LSTMs**. A chegada de arquiteturas LLMs como **GPT-3** [10] e **PaLM**

[2] `http://docs.ag2.ai/`

[3] `https://docs.ag2.ai/latest/docs/user-guide/advanced-concepts/orchestration/group-chat/introduction/`

[11] expandiu enormemente essas fronteiras, possibilitando diálogos mais naturais e adaptativos, abrindo caminho para aplicações em atendimento ao cliente, criação de conteúdo e análise de informações complexas [12].

Segundo Zhao *et al.* [13], **agentes baseados em LLMs** apresentam vantagens significativas em termos de armazenamento de conhecimento, capacidade de raciocínio e flexibilidade para tarefas diversas. Os autores destacam que componentes como planejamento, memória e uso de ferramentas (*tool use*) são essenciais para estruturar **fluxos de trabalho complexos** e garantir tomadas de decisão fundamentadas em dados concretos.

2.2 Orquestração de Agentes

Orquestração de agentes é o processo de **coordenar múltiplos agentes de IA** especializados em diferentes tarefas para alcançar objetivos complexos de forma colaborativa e eficiente [14]. Ao contrário de sistemas monolíticos ou sequenciais, onde **um único agente** trata as solicitações, a orquestração permite decompor esses fluxos de trabalho em **sub-tarefas** distribuídas segundo as competências atribuídas para cada agente [15]. Esse modelo se assemelha ao papel de um maestro em uma orquestra, garantindo que cada agente atue no momento adequado e baseado nas **instruções corretas**, mantendo a sincronia entre suas ações e os resultados obtidos para cada etapa [16].

As soluções de orquestração frequentemente fornecem componentes como roteadores de solicitações, gerenciadores de estado e mecanismos de retry, assegurando **robustez e resiliência do sistema** nas mais diversas etapas [17]. Em arquiteturas modernas baseadas em LLMs, a orquestração pode incluir a integração de **ferramentas e APIs externas**, ampliando as capacidades dos agentes e automatizando chamadas de função conforme demandado pelo fluxo conversacional [18]. Padrões de orquestração(como fluxos centralizados, descentralizados e hierárquicos) permitem customizar o **grau de autonomia e controle** em cada etapa, selecionados conforme requisitos de desempenho, segurança e escalabilidade [19].

2.3 Comunicação, Uso de Ferramentas e Contexto Compartilhado entre Agentes

Com a evolução dos sistemas de IA, surgiram os *chats em grupo*, nos quais múltiplos **agentes colaboram entre si** dentro de um contexto compartilhado. Um gerente de chat seleciona qual agente deve ser acionado, e isso pode ser feito por uma ordem pré-definida, aleatoriamente, manualmente ou via LLM, permitindo *handoffs* explícitos, especialização dinâmica e escalabilidade sem a necessidade grandes mudanças estruturais, habilitando uma **maior escalabilidade** das soluções envolvendo esse tipo de

arquitetura [4].

Para além da mera troca de mensagens, agentes são capazes de invocar **ferramentas** através de chamadas de função (*tool use*), podendo interagir com APIs, realizar cálculos ou realizar o registro de dados de acordo com suas instruções. Isso permite a realização de tarefas muito mais **especializadas e orientadas** pelo fluxo de execução dos agentes, de acordo com as necessidades específicas da solução [5].

Outro conceito importante para a comunicação entre agentes são as **variáveis de contexto**, elas funcionam como uma **memória compartilhada** entre os agentes, armazenando pares chave–valor acessíveis a todos os componentes do sistema. Esse repositório comum preserva o histórico de decisões, dados coletados e estado da conversa, evitando reprocessamento de dados e aumentando a **eficiência e segurança** nas trocas de informações relevantes [6].

2.4 Framework AG2

A biblioteca *AG2*[7] foi escolhida neste estudo por oferecer um framework completo e consolidado para **orquestração de múltiplos agentes LLM**, unificando o conceito de chats em grupo (substituindo o conceito "swarm" [20] presente nas versões anteriores), uso de ferramentas e funções, contexto compartilhado entre agentes e padrões de transição controlados. Tudo em sua versão 0.9, lançada em abril de 2025.

No AG2, um `GroupChatManager` é responsável por selecionar iterativamente o próximo agente com base no **padrão** escolhido: automático, sequencial, aleatório, manual ou explícito (`DefaultPattern`). Esse mecanismo assegura interações em **contexto compartilhado**, sustentado por variáveis de estado acessíveis a qualquer agente ou ferramenta, assim como acesso irrestrito a invocação dos agentes previamente inicializados, caso necessário.

O suporte ao `DefaultPattern` é essencial quando é necessário **controle rigoroso**: cada *handoff* deve ser explicitamente definido, ou a conversa é encerrada. Evitando execuções implícitas e garantindo **precisão e controle** durante todo o workflow. Combinado ao gerenciamento centralizado e à memória compartilhada, o AG2 oferece uma base sólida para sistemas multiagente **previsíveis e escaláveis** para uma grande variedade de cenários.

[4] `https://docs.ag2.ai/latest/docs/user-guide/advanced-concepts/orchestration/group-chat/introduction/`

[5] `https://docs.ag2.ai/latest/docs/user-guide/advanced-concepts/orchestration/group-chat/agent-tools-functions/`

[6] `https://docs.ag2.ai/latest/docs/user-guide/advanced-concepts/orchestration/group-chat/context-variables/`

[7] `https://docs.ag2.ai/`

3 Arquitetura Multiagentes para Revisão Iterativa de Conteúdo

A Figura 1 apresenta o fluxo de trabalho **multiagente** especificado e implementado em que diferentes agentes como revisores semântico e conxtextual, recomendador de melhorias, reescritor e um decisor interagem de **maneira iterativa** para realizar a revisão e aprimoramento de respostas geradas por IA a partir de perguntas de usuários sobre produtos em sistemas de comércio eletrônico combinadas com informações relevantes para a geração dessas respostas, como o contexto que contém informações detalhadas dos produtos e os metadados que contém regras específicas para lojas ou do sistema em geral.

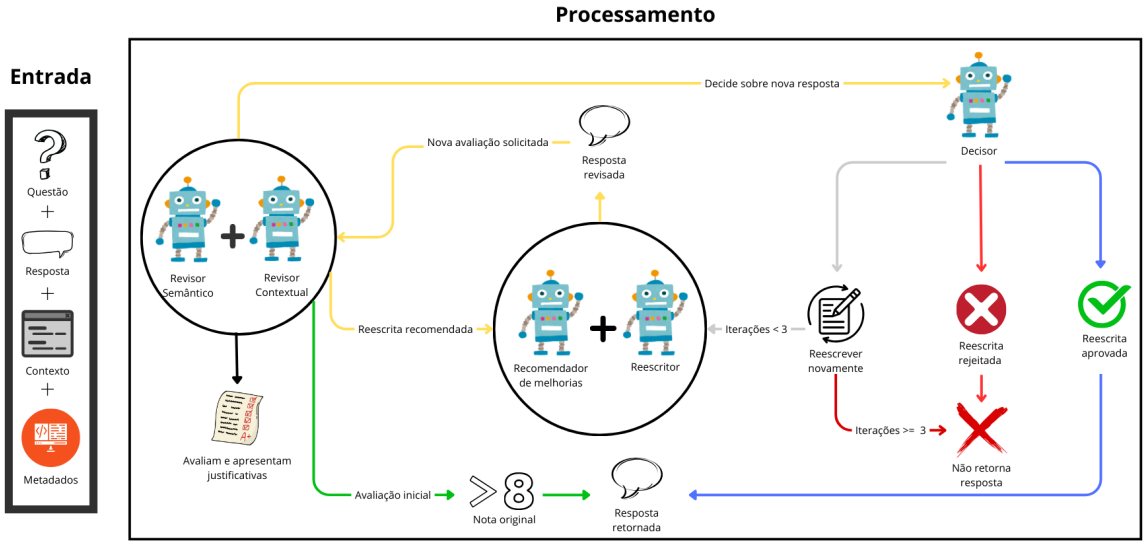

Figura 1: Design do sistema multiagente desenvolvido.

Cada agente assume uma responsabilidade específica: avaliação de qualidade das respostas originais, sugestão de melhorias para respostas que não atenderam os critérios de qualidade estabelecidos, reformulação textual para novas respostas e decisão final em relação as novas alternativas de respostas. Nossa proposta de design modular e colaborativo entre os agentes visa mitigar falhas comuns de sistemas LLMs isolados (por exemplo, alucinações e inconsistências), garantindo uma maior robustez e confiabilidade operacional para o processo.

A abordagem proposta segue padrões consolidados na literatura de sistemas multiagente com LLMs **(LLM-MAS)** [21, 22], que destacam estruturas como orquestração hierárquica, refinação iterativa e especialização de papéis. Por exemplo, frameworks como HALO [23] empregam arquiteturas hierárquicas com divisão de tarefas entre planejamento, design dos papéis e execução de tarefas, essas estratégias resultaram em ganhos de até **14,4%** em benchmarks de geração de código e raciocínio.

Em cenários de raciocínio matemático, o MAgICoRe [24] evidenciou melhorias nos resultados ao integrar os agentes "Solver", "Reviewer" e "Refiner" em **laços de retroalimentação**, superando métodos como Self-Consistency [25] e Self-Refine [26]. Levantamentos recentes enfatizam a eficácia de **refinamento iterativo e avaliação cruzada** comuns em arquiteturas "Evaluator–Optimizer" e "Orchestrator–Workers" para elevar a qualidade das respostas [27].

Esses estudos inspiram diretamente a solução adotada nesse trabalho: agentes compartimentalizados (especializados) que trabalham colaborativamente e realizam avaliações através de notas de 0 a 5, fornecem **sugestões** e pontos de melhorias e realizam **reescritas iterativas** até que uma nova resposta atinja o patamar de qualidade definido ou seja descartada.

Na abordagem proposta, cinco agentes trabalham de forma coordenada, cada qual incumbido de uma função especializada. Os agentes **Revisor Semântico** e **Revisor Contextual** atuam como **críticos experientes**, analisando coerência semântica e a aderência da resposta fornecida inicialmente ao contexto e metadados disponíveis. Atribuindo notas de 0 a 5 para cada um desses critérios e apresentado justificativas de possíveis falhas, inconsistências ou lacunas. Seguindo a proposta do MAgICoRe [24], esse agente traduz pontuações de etapas (via *reward models*) em **feedback acionável e localizado**.

Com base nas observações dos Revisores, o **Recomendador de Melhorias** propõe **ajustes concretos**, seja no conteúdo, tom ou organização da resposta. A ideia é que o *feedback* seja traduzido em **sugestões de edição**, aproximando-se do padrão "*Evaluator–Optimizer*" observado em sistemas como *Self-Refine* [26].

O **Agente Reescritor** recebe a versão original da resposta e as sugestões apresentadas para o **processo de reescrita**, buscando clareza, completude e precisão. Essa etapa consolida o refinamento sugerido e precede a classificação final, exercendo papel similar ao agente **"*Refiner*"** em arquiteturas hierárquicas para LLMs [23].

O **Agente Decisor** verifica se a resposta aprimorada cumpre os **critérios de aceitação**. Por exemplo, adequação ao produto, ausência de informações imprecisas ou irrelevantes e alinhamento com o tom desejado. Caso esses critérios não sejam atendidos, esse ciclo pode ser reiniciado até o limite de 3 iterações estabelecido. Esse **controle iterativo basal** é comparável ao que ocorre em modelos multifásicos como *HALO* [23] e *Optima* [28].

Ao **separar** as responsabilidades, a arquitetura multiagente assegura que cada agente atue dentro de sua **especialidade**, promovendo especialização, revisão localizada e auditabilidade com mensagens e registros claros para cada etapa, permitindo **rastreabilidade e avaliação** de processos complexos em sub-etapas menores.

Padrões generalizáveis. Essa arquitetura incorpora padrões amplamente reconhecidos na literatura de sistemas multiagente LLM:

- *Decomposição modular de tarefas*: dividir o processo em **fases claras** (geração, crítica, sugestão, reformulação e validação) que permite **especialização e paralelismo**, ao passo que ajuda a reduzir os efeitos adversos de falhas em sistemas monolíticos [28].

- *Feedback iterativo com refinamento adaptativo*: ciclos de feedback "revisor → refiner → verificação" são capazes de promover **aprimoramento contínuo** da solução, com controle dinâmico de iterações, como demonstrado no *MAgICoRe* [24], em que o refinamento só prossegue caso necessário e baseado em *scoring* interno.

- *Orquestração hierárquica inspirada em estruturas humanas*: ao adotar uma pipeline organizada e com um agente decisor ao final, o sistema assemelha-se a processos de **engenharia de software ou curadoria editorial**, em que cada subtarefa é refinada em etapas sequenciais bem definidas [29].

- *Auditabilidade e interpretabilidade*: cada agente registra suas entradas e saídas, facilitando **análise retroativa, rastreamento de falhas e justificativas**. Características valorizadas em aplicações críticas como suporte técnico e revisão sistemática [30, 31].

- *Parada condicional e economia de recursos*: semelhante ao *MAgICoRe* [24] e frameworks como *Optima* [28], o sistema interrompe o ciclo quando os **critérios de qualidade** estabelecidos já foram atendidos, evitando refino redundante e otimizando o uso dos LLMs e os custos envolvidos.

Esses princípios conferem à arquitetura propsta **adaptabilidade** a diversos domínios, como sistemas de revisão acadêmica, criação assistida de código ou apoio automatizado ao cliente. Onde **confiança, clareza e escalabilidade** são essenciais.

Detalhes de Implementação. A Seção A apresenta o código que contém o conjunto de **instruções** específicas definidas para cada agente, assim como as **funções de registro e decisão** definidas para eles pode ser encontrado. Esse código está escrito na linguagem Python [8] e faz o uso de ferramentas presentes na versão 0.9 da biblioteca de código aberto *AG2* [9] para realizar a **comunicação** entre os agentes.

4 Avaliação Experimental

A Subseção 4.1 apresenta os dados utilizados para a realização da avaliação de desenvolvimento e validação da solução; a Subseção 4.2 apresenta os testes exploratórios

[8] `https://www.python.org/`
[9] `https://docs.ag2.ai/0.9`

que foram realizados para chegar na solução final; a Subseção 4.3 apresenta os procedimentos da avaliação experimental realizada; e a Subseção 4.4 apresenta os resultados dos experimentos realizados.

4.1 Conjunto de Dados

Para a realização dos experimentos com as diferentes **arquiteturas de agentes**, foram utilizados dois conjuntos de dados (`datasets`) contendo perguntas e respostas, extraídas diretamentes dos dados de produção da GoBots. A GoBots é uma empresa que atua no setor de e-commerce, oferecendo soluções automatizadas de atendimento ao cliente para lojas cadastradas em plataformas virtuais, como Shopee e Mercado Livre. O primeiro `dataset`, utilizado durante o desenvolvimento da aplicação, continha **1072** respostas para dúvidas de usuários.

O segundo dataset possuí originalmente **9640** respostas, das quais foram selecionadas **6000** após um **processo de filtragem**, sendo utilizado para validar a **eficácia e aderência** da solução para diferentes cenários e extração de **métricas de desempenho**.

4.1.1 Dados de Desenvolvimento

O conjunto de dados de desenvolvimento foi formado por **perguntas realizadas pelos clientes**, as respostas fornecidas originalmente e as informações utilizadas pelos modelos da GoBots para gerar essas respostas. Esse conjunto também inclui **feedbacks dos lojistas** sobre a corretude das respostas fornecidas pelos modelos da GoBots, juntamente com **justificativas** para os casos em que as respostas foram consideradas inadequadas. Outras informações adicionais presentes incluem o **idioma da pergunta** (considerando que a GoBots atende diversos países da América Latina, as perguntas podem estar em português ou espanhol), uma inferência classificada entre 0 e 1 sobre a **intenção do consumidor** ao relizar a pergunta, e a **categoria do produto** referente à pergunta.

Esse conjunto incluiu **712** perguntas em português e **360** em espanhol. Quanto às **intenções das perguntas**, foram identificadas **39** categorias distintas, sendo as mais frequentes: Especificação de Produto (**322 perguntas**), Compatibilidade (**268 perguntas**), Disponibilidade (**173 perguntas**), Itens Inclusos (**42 perguntas**) e Problema (**35 perguntas**). Já em relação às **categorias de produtos**, o conjunto abrangia perguntas sobre **32** categorias diferentes, destacando-se: Acessórios para Veículos (**486 perguntas**), Casa, Móveis e Decoração (**147 perguntas**), Ferramentas (**62 perguntas**), Construção (**53 perguntas**) e Eletrodomésticos (**29 perguntas**).

Fornecemos aos agentes os atributos *contexto* e *metadados*, **essenciais para a geração das respostas**. Esses atributos incluem descrições completas dos produ-

tos, regras específicas definidas pelos lojistas em relação à possíveis respostas, informações de envio, disponilidade, garantia, políticas da loja, condições do produto, marca, padrões de saudações que devem estar presentes nas respostas e qualquer outra informação considerada relevante para atender as mais diversas necessidades dos clientes. Quando disponível, também é fornecido o processo originalmente utilizado pelos modelos para gerar a resposta, incluindo o **custo**, **total de tokens** utilizados e o **modelo responsável** pela geração de texto.

4.1.2 Dados de Teste

Cada dado do conjunto de teste possui **exatamente** os mesmos atributos presentes no conjunto de desenvolvimento, também tendo sido extraídos dos **dados de produção da GoBots**. Entretanto, o conjunto de teste contém uma quantidade significativamente **maior de exemplos** e cobre uma diversidade mais ampla de **categorias e intenções**.

O conjunto original passou por uma filtragem rigorosa para garantir maior relevância e precisão das respostas. Essa seleção baseou-se no **grau de confiança** atribuído pela inferência sobre as intenções das perguntas. Especificamente, foram escolhidas as 3000 perguntas classificadas como corretas e as 3000 perguntas classificadas como incorretas que apresentaram os maiores índices de confiança no processo de inferência da intenção, todos acima de 90%. Esse procedimento assegura que as perguntas sejam altamente relevantes, elevando a qualidade da análise das métricas de desempenho.

Esses dados foram organizados em **60** arquivos JSON para facilitar a realização **de testes em lote** em uma infraestrutura adequada. Cada arquivo continha **100** perguntas, divididas igualmente entre **50** perguntas classificadas como corretas e **50** classificadas como incorretas, ordenadas pelo **nível de confiabilidade** do processo de inferência da intenção.

O conjunto original continha inicialmente **6747** perguntas em português e **2893** em espanhol. Após o processo de filtragem mencionado anteriormente, esses números foram reduzidos para **4411** perguntas em português e **1589** perguntas em espanhol, garantindo assim **maior relevância e precisão** no conjunto final.

Quanto às **intenções das perguntas**, o conjunto original apresentava **40** categorias diferentes. As intenções mais frequentes inicialmente eram: Compatibilidade (**2623 perguntas**), Especificação de Produto (**2583 perguntas**), Disponibilidade (**1556 perguntas**), Itens Inclusos (**427 perguntas**) e Problema (**355 perguntas**). Após a filtragem, essas intenções permaneceram as mais comuns, porém com uma distribuição ajustada, conforme apresentado na Tabela 1.

Figura 2: Exemplo de dado do conjunto de desenvolvimento


```
{
   "id":13292648659,
   "question":"HOLA, LAS MEDIDAS QUE APARECEN EN SU DESCRIPCION: Medidas:
   20 cm x 13 cm x 2 cm SON LAS MEDIDAS DEL EMBALAJE, \u00d3 SON LAS",
   "answer":"Hola, gracias por contactarnos Las medidas de 20 cm de alto
   por 13 cm de ancho corresponden al tama\u00f1o del Term\u00f3metro
   Digital con Sensor de Humedad (higr\u00f3metro) Steren modelo TER-120,
   no al embalaje. Este dispositivo es compacto y perfecto para
   monitorear las condiciones ambientales en diversos espacios.  Saludos
   de Electr\u00f3nica Steren.",
   "correct":false,
   "feedback":"Revisi\u00f3n v\u00e1lida Solo para este producto\n
   \u00a1Hola! Las medidas del producto son 68mm x 80mm x 16mm (base,
   altura y profundidad). Saludos. ",
   "locale":"es",
   "intent":{
      "confidence":0.52419597,
      "name":"Disponibilidad"
   },
   "context":{
      "title":"Term\u00f3metro Digital Con Sensor De Humedad
   (higr\u00f3metro)",
      "BRAND":"Steren",
      "COLOR":"Blanco",
      ...
   },
   "metadata":[
      {
         "total_tokens":670,
         "cost":0.008180000000000002,
         "language":"spanish",
         "inference":"<answer>Las medidas de 20 cm de alto por 13 cm de
   ancho corresponden al tama\u00f1o del Term\u00f3metro Digital con
   Sensor de Humedad (higr\u00f3metro) Steren modelo TER-120, no al
   embalaje. Este dispositivo es compacto y perfecto para monitorear las
   condiciones ambientales en diversos espacios.</answer>",
         "model":"gpt4",
         "prompt":"no-filter-prompt"
      }
   ],
   "category":"Herramientas"
}
```

Tabela 1: Intenções de perguntas mais frequentes no *dataset* gerado pós-filtragem

Intenção	**Quantidade de ocorrências**
Compatibilidade	1706
Especificação de Produto	927
Disponibilidade	870
Itens Inclusos	315
Problema	107

Com relação às **categorias**, o conjunto original abrangia **48** categorias distintas. As mais frequentes inicialmente eram: Acessórios para Veículos (**4645 perguntas**), Casa, Móveis e Decoração (**1276 perguntas**), Construção (**619 perguntas**), Ferramentas (**463 perguntas**) e Informática (**337 perguntas**). Após o processo de filtragem, as categorias com maior ocorrência sofreram alterações, sendo as cinco mais comuns descritas na Tabela 2.

Tabela 2: Categorias de produtos mais frequentes no *dataset* gerado pós-filtragem

Categoria	**Quantidade de ocorrências**
Acessórios para Veículos	3127
Casa, Móveis e Decoração	523
Ferramentas	331
Construção	313
Casa, Móveis e Jardim	166

4.2 Experimentos Exploratórios

Nesta fase exploratória, avaliamos **duas propostas de arquiteturas distintas** para a comunicação entre os agentes que deveriam trabalhar no processo de melhoria das respostas:

Dupla de agentes conversacionais [10] O primeiro experimento utilizou uma *dupla de agentes conversacionais* voltados à compreensão básica de perguntas, avaliação de qualidade das respostas fornecidas originalmente e geração de novas respostas automatizadas quando necessário. Foram realizados testes controlados com **subconjuntos extraídos do conjunto de desenvolvimento** mencionado anteriormente, focando na precisão e qualidade das respostas, na identificação de falhas iniciais de entendimento e na aderência entre os resultados obtidos e os feedbacks das respostas fornecidos pelos lojistas.

[10] `https://docs.ag2.ai/latest/docs/user-guide/advanced-concepts/orchestration/two-agent-chat/`

Group chat [11] Em seguida, adotou-se a arquitetura de *group chat*, na qual **múltiplos agentes** atuavam colaraborativamente na elaboração das respostas. Essa versão introduziu três agentes especialistas: Reescritor, Revisor e Avaliador, além de um `GroupChatManager`, responsável por **coordenar as iterações**, e um `UserProxy` para **iniciar a conversa**. O objetivo era verificar se o trabalho colaborativo gerava respostas mais completas, contextualizadas e coerentes, sem um aumento muito grande de custo computacional e financeiro, devido à distribuição de **papéis bem definidos e especializados** entre os agentes.

Para ambos os experimentos, avaliamos diversos **modelos proprietários consolidados**: ChatGPT 4o, GPT 4.1, GPT 4.1 mini, Sabiá 3, GPT 4o mini e o3 mini. Observou-se variação significativa de efetividade entre essas arquiteturas e modelos, bem como em parâmetros como temperatura, evidenciando **vantagens e limitações específicas** de cada combinação.

Na versão 0.9, o AG2 unificou a arquitetura *swarm*(comprovadamente a mais precisa) com a estratégia de *group chat*. Essa integração se revelou acertada: testes subsequentes demostraram que a combinação do paradigma `swarm` [20], com sua colaboração distribuída, e o controle de fluxo entre agentes, elevou significativamente a **qualidade das respostas** em relação às duas abordagens separadas. Além de evoluir o processo de transições entre agentes, devido às novas estratégias de troca de controle implementadas.

As arquiteturas iniciais (dupla de agentes e group chat) desempenharam papel fundamental como **etapas exploratórias**, mas foram descartadas após a validação da **superioridade** da abordagem combinada adotada em nossa solução final e avaliação experimental final.

Em nossos experimentos (cf. Subseção 4.3), decidimos utilizar **modelos** `open-source` para melhor avaliar a **reprodutibilidade** da solução em diferentes contextos e cenários devido ao nível de **transparência** dos modelos e a possibilidade de rodá-los localmente em um **ambiente controlado**.

4.3 Procedimentos da Avaliação Experimental

A Figura 3 apresenta o processo de tomada de decisão em relação às respostas no nosso experimento, todas elas passam por uma **avaliação inicial** e as que já atendem os critérios de qualidade estabelecidos são **retornadas sem alterações**, enquanto as que **necessitam de melhorias** passam por um processo de **revisão iterativa** limitado a **3** etapas, caso após essas etapas ainda não seja possível obter uma resposta melhor, opta-se por **não fornecer uma resposta** para essa pergunta.

[11] `https://docs.ag2.ai/latest/docs/user-guide/advanced-concepts/orchestration/group-chat/introduction/`

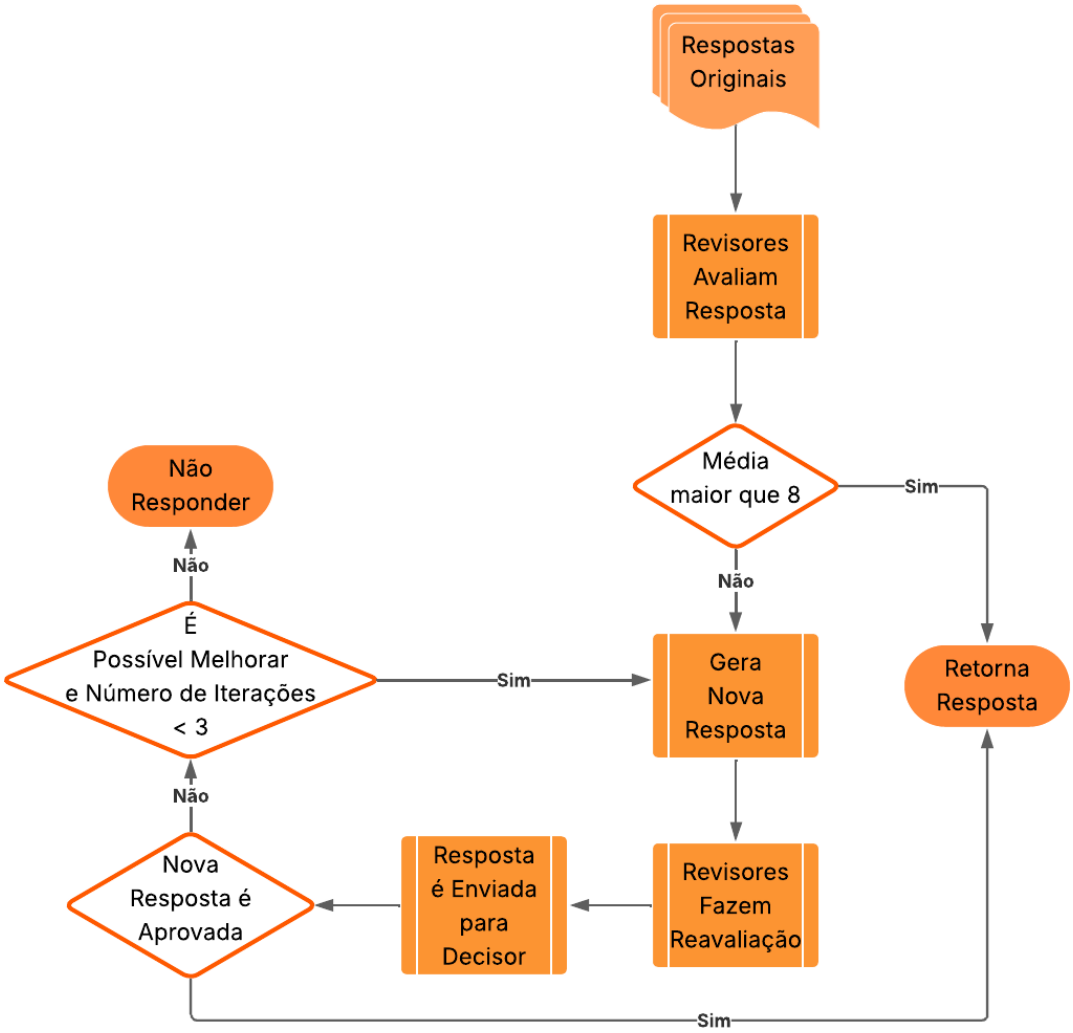

Figura 3: Fluxograma da interação entre agentes no experimento

Para avaliar o impacto de diferentes modelos, selecionamos dois modelos de código aberto da família Qwen3 [12]: o denso **Qwen3 32B** e o modelo esparso MoE **Qwen3 30B-A3B**. O Qwen3 32B consiste em um *transformer* tradicional com **32 bilhões** de parâmetros ativos a cada geração, demandando aproximadamente **20GB** de memória e mantendo **latência previsível** em tarefas de inferência. Já o Qwen3 30B-A3B emprega a arquitetura *Mixture of Experts*, na qual **30 bilhões** de parâmetros são distribuídos entre diversos submodelos ("experts"), mas apenas **3 bilhões são ativados** dinamicamente por token através de uma rede de roteamento treinável, reduzindo o **custo computacional médio** por passo de inferência e demandando aproximadamente **18GB** de memória no total. [13]

Adicionalmente, o Qwen3 implementa um mecanismo de modos de operação (*"thinking mode"* para raciocínio multi-etapa e *"non-thinking mode"* para respostas diretas), permitindo que o orçamento de parâmetros ativados seja **ajustado dinamicamente** conforme a complexidade do prompt. Esse controle presente em ambos os modelos combina-se de forma sinérgica com o paradigma **MoE** [32], em que apenas **uma fração dos especialistas** é invocada por vez, otimizando ainda mais a relação entre **capacidade e custo computacional**.

A vantagem prática de fazer o uso dessas **duas arquiteturas** de estruturas diferentes reside em analisar trade-offs de latência, consumo de memória e desempenho. Embora o Qwen3 32B ofereça inferência **estável e previsível**, o Qwen3 30B-A3B apresentou tempos de treinamento e inferência cerca de duas vezes mais rápidos (aprox. **45–60 tokens/s** em CPU de ponta) e reduziu custos de implantação em até **30%**. Em benchmarks padrão de raciocínio e codificação, o modelo MoE igualou ou superou o Qwen3 32B, evidenciando que a capacidade ociosa pode ser aproveitada seletivamente, sem necessariamente penalizar a qualidade das respostas. Então nós podemos analisar se esses resultados obtidos nos `benchmarkes` são refletidos de forma equivalente na nossa estrutura de agentes, ou se é possível notar de forma evidente a **diferença de desempenho** entre eles.

Em ambos os experimentos, foram utilizados o mesmo **conjunto de dados** filtrados mencionado na Subseção 4.1.2, mantendo constantes variáveis como *temperatura* e características de desempenho do modelo para garantir a **comparabilidade** e ser possível quantificar rigorosamente a influência do número de **parâmetros ativos** e da arquitetura na qualidade das novas respostas, na adequação ao cumprimento de instruções e na avaliação da qualidade das respostas originais.

Mantendo constantes a temperatura (0.0), o fluxo de execução de agentes e scripts rodando em um notebook Python, avaliamos separadamente as seguintes configurações:

[12] `https://arxiv.org/abs/2505.09388`
[13] `https://qwenlm.github.io/blog/qwen3/`

Experimento #1 Utilizou-se o modelo `qwen3:32b`, carregado **integralmente na GPU** via um túnel SSH nos servidores da ENQII, com execução local pelo **Ollama** por meio de uma API. O conjunto de **60** arquivos de teste (`test_output_1.json` a `test_output_60.json`) foi processado sequencialmente, em lotes de **100** perguntas cada, com média de **1h e 30 min** de tempo total de execução por arquivo. Todas as chamadas seguiram o mesmo **pipeline de agentes**, utilizando as mesmas **instruções e funções** previamente apresentadas nos Apêndices A.1 e A.2.

Experimento #2 Alterou-se apenas o **modelo** em que os agentes foram executados, que passou a ser o `qwen3:30b`, com quantização e offload de 27B de parâmetros para a CPU, mantendo inalterados a estratégia de execução por meio de **túnel SSH**, ambiente local **Ollama** e notebook Python. O mesmo conjunto de **60** arquivos também foi processado em lotes de **100**, tendo uma redução no tempo médio de execução por arquivo para aproximadamente **45 min**, sem ajustes extras em hiperparâmetros, no **fluxo de agentes** ou nas **instruções e funções** associadas, presentes nos Apêndices A.1 e A.2.

4.4 Resultados

Esta seção apresenta os resultados obtidos dos **dois modelos** utilizados nos experimentos (**Qwen3 32B** e **Qwen3 30B-A3B**) ao realizar **avaliações das respostas** dadas para o mesmo conjunto de **6000** perguntas mencionadas na Subseção 4.1.2 em relação às **3 principais análises** feitas pelos agentes em relação as respostas.

A Subseção 4.4.1 apresenta a **classificação inicial de corretude** comparada com os feedbacks fornecidos pelos lojistas, a Subseção 4.4.2 reporta as **decisões tomadas** para as respostas que foram inicialmente classificadas como incorretas e a Subseção 4.4.3 apresenta uma **análise da acurácia segmentada** por idiomas, principais intenções de perguntas(extraídas da Tabela 1) e categorias de produtos mais frequentes(extraídas da Tabela 2). Enquanto a Subseção 4.4.4 exibe uma comparação entre as pontuações originais atribuídas para as respostas e as **notas atribuídas após o processo de reescrita iterativa**. Para cada uma dessas métricas, destacamos em **negrito** o modelo que apresentou melhor desempenho.

4.4.1 Classificação das Respostas

A primeira avaliação que nós podemos fazer em relação aos resultados é sobre a classificação inicial realizada para as **respostas originais**, comparando-as com os **feedbacks** realizados pelos lojistas em relação a elas (salientando que essas informações não foram fornecidas aos agentes durante as avaliações, apenas foram usadas para extração de métricas). Como mostra a Tabela 3, o sistema de agentes realizou avaliações das respostas para as mesmas **6000** perguntas utilizando os modelos Qwen3

32B e Qwen3 30B-A3B. Com o Qwen3 32B, apenas **757 (12,6%)** das interações apresentaram falhas no seguimento do conjunto de instruções pelo modelo entrar em "thinking mode" e não realizar as chamadas de funções as quais ele foi instruído a fazer, resultando em **5243** perguntas processadas integralmente, com concordância nas análises em **3121 (59,6%)** dos casos e divergência em **2123 (40,5%)**, o que resultou em uma acurácia geral de **59,6%** e F1-score **59,0%**. Já com o Qwen3 30B-A3B, em **1242 (20,7%)** das análises houveram falhas no seguimento de instruções pelo modelo entrar em "thinking mode", com **4758** das respostas sendo avaliadas no total, com concordância em **2818 (59,2%)** dos casos e divergência em **1942 (40,8%)**, resultando em uma acurácia geral de **59,2%** e F1-score **61,0%**.

Tabela 3: Comparativo de classificação das respostas entre modelos

Métrica	**Qwen3 32B**	**Qwen3 30B-A3B**
Total de perguntas	6000	6000
Respostas sem avaliação adequada	**757**	1242
Corretas avaliadas adequadamente	**1694**	1535
Incorretas avaliadas adequadamente	**1427**	1283
Corretas avaliadas incorretamente	965	**899**
Incorretas avaliadas incorretamente	1158	**1043**
Acurácia geral	**59,6%**	59,2%
F1-score	59,0%	**61,0%**

4.4.2 Decisões para Respostas Classificadas Inicialmente como Incorretas

Como mostra a Tabela 4, o **fluxo de revisão iterativa** sobre as respostas inicialmente classificadas como incorretas revelou **diferenças marcantes** entre os dois modelos. No caso do Qwen3 32B, de um total de **2522** respostas, **1492** foram de fato falsos positivos (classificação correta como incorreta), das quais **897 (60,1%)** passaram por reescritas e tiveram novas respostas aprovadas e **595 (39,9%)** delas deixaram de ser respondidas por não atender os critérios de qualidade estabelecidos; das **1030** respostas classificadas inadequadamente (falsos negativos), **698 (67,8%)** tiveram novas respostas aprovadas e **332 (32,2%)** ficaram sem resposta após o fluxo de revisão iterativo. Isso resultou em **1595 (63,2%)** respostas aprimoradas e **927 (36,8%)** perguntas sem resposta, com uma média de reescritas de **1,23** por pergunta (e **1,05** para as que tiveram novas respostas aprovadas).

Já para o Qwen3 30B-A3B, das **1273** respostas marcadas como incorretas, **524 (41,2%)** foram reescritas e tiveram novas respostas aprovadas, enquanto **749 (58,8%)** foram rejeitadas e ficaram sem resposta; das **888** que eram falsos negativos, **479 (54,0%)** receberam novas respostas reescritas e aprovadas e **409 (46,0%)** foram

rejeitadas. No total, **1003 (46,4%)** respostas foram aprimoradas e **1158 (53,6%)** perguntas ficaram sem resposta, com média de reescritas de **1,26** por pergunta (e **1,03** para as com respostas aprovadas).

Tabela 4: Comparativo de decisões para perguntas inicialmente incorretas

Métrica	**Qwen3 32B**	**Qwen3 30B-A3B**
Reescritas e aprovadas (class. correta)	**897**	524
Rejeitadas (class. correta)	**595**	749
Reescritas e aprovadas (class. incorreta)	**698**	479
Rejeitadas (class. incorreta)	**332**	409
Média total de reescritas	**1,23**	1,26
Média de reescritas aprovadas	1,05	**1,03**

4.4.3 Acurácia por Segmento

Para avaliar a consistência dos modelos em diferentes cenários, comparamos a **acurácia média segmentada** por idioma, principais intenções de perguntas (extraídas da Tabela 1) e principais categorias de produto (extraídas da Tabela 2) presentes no conjunto de dados (Tabela 5). Observa-se que, em perguntas em português, o Qwen3 30B-A3B obteve uma leve vantagem (**58,99%** vs. **58,81%**), enquanto em espanhol a diferença foi mais expressiva (**63,30%** vs. **60,89%**), reforçando melhor tratamento de estruturas no segundo modelo, porém esse número pode ter sido um pouco "inflado"pelo conjunto ter um número menor de perguntas em espanhol e o modelo MOE ter tido mais problemas no seguimento de instruções. Quanto às intenções, ambos se saíram de forma semelhante em "Compatibilidade" (**63,62%** vs. **63,52%**), mas o Qwen3 30B-A3B levou uma pequena vantagem em "Disponibilidade" (**52,52%** vs. **52,14%**) e em "Especificação de Produto" (**61,99%** vs. **60,56%**). Nas categorias de produto, o Qwen3 32B obteve melhores resultados em "Casa, Móveis e Decoração" (**55,90%** vs. **53,33%**) e em "Ferramentas" (**67,54%** vs. **63,78%**), enquanto o Qwen3 30B-A3B teve melhor desempenho em "Acessórios para Veículos" (**61,38%** vs. **60,44%**).

Tabela 5: Comparativo de acurácia média segmentada

Métrica	**Qwen3 32B**	**Qwen3 30B-A3B**
Idioma (Português)	58,81%	**58,99%**
Idioma (Espanhol)	60,89%	**63,30%**
Intenção (Compatibilidade)	63,52%	**63,62%**
Intenção (Disponibilidade)	52,14%	**52,52%**
Intenção (Especificação de Produto)	60,56%	**61,99%**
Categoria (Acessórios para Veículos)	60,44%	**61,38%**
Categoria (Casa, Móveis e Decoração)	**55,90%**	53,33%
Categoria (Ferramentas)	**67,54%**	63,78%

4.4.4 Diferença de Notas para Respostas Revisadas

A Figura 4 apresenta a **distribuição das diferenças de pontuação** (final – original) para as respostas reescritas e aprovadas nos modelos Qwen3 32B e Qwen3 30B-A3B. No caso do Qwen3 32B, o pico de frequência de evolução das notas ocorreu em **+2 pontos (30,5%)**, seguido de +3 (**18,8%**) e +4 (**13,0%**), com ganho médio de **+2,9** pontos (mediana = **+3**), sendo **89,9%** das revisões consideradas melhores que as respostas originais, **6,0%** estáveis e **4,0%** piores. Para o Qwen3 30B-A3B, o pico de evolução também foi em **+2 pontos (24,5%)**, seguido de +1 (**15,3%**) e +3 (**14,4%**), com evolução média de **+2,3** pontos (mediana = **+2**), **80,0%** de melhorias, **10,7%** sem alteração e **9,3%** de pioras.

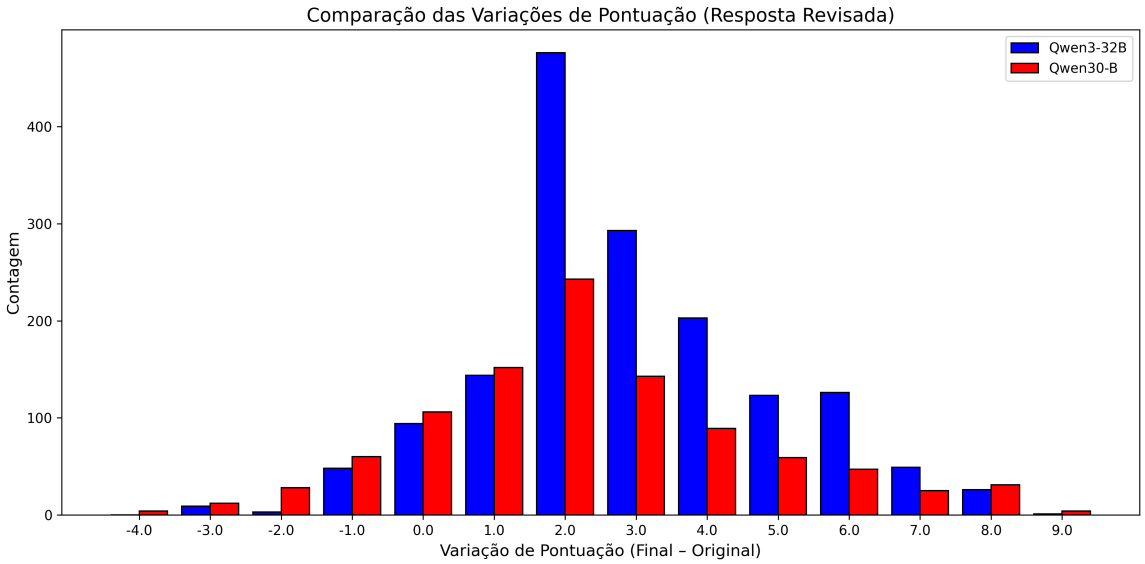

Figura 4: Diferença de notas revisadas em relação às originais

5 Discussão

Discutimos em detalhes os resultados obtidos nos Experimentos 1 e 2, avaliando a efetividade da **arquitetura multiagente** proposta e comparando os desempenhos dos modelos **Qwen3 32B** e **Qwen3 30B-A3B**. Destacamos os pontos fortes e fracos deles e apresentamos sugestões de melhorias de trabalhos futuros.

Análise Geral dos resultados Nos Experimentos 1 e 2 foram realizados considerando o mesmo conjunto de **6000** perguntas, porém utilizando **configurações de modelo distintas**. No Exp. 1 (Qwen3-32B), observou-se uma acurácia geral de aproximadamente **59,6%** e F1-score em torno de **59%**, enquanto no Exp. 2 (Qwen3-30B-A3B) a acurácia ficou em **59,2%** com F1-score de cerca de **61%** (Tabela 3). Observa-se, portanto, que o Qwen3 32B apresentou uma acurácia ligeiramente maior, enquanto o Qwen3 30B-A3B atingiu um equilíbrio um pouco melhor entre precisão e recall, refletido em um F1-score ligeiramente superior. Observamos que o modelo MoE (30B-A3B) manteve desempenho comparável ao modelo denso, o que corrobora a hipótese de que arquiteturas *Mixture of Experts* podem oferecer boa **relação custo-desempenho** em cenários de orquestração de agentes.

A proporção de falhas de avaliação (agentes entrando em *"thinking mode"*) foi maior no Exp. 2 (**20,7%**) do que no Exp. 1 (**12,6%**), sugerindo que o modelo esparso apresenta maior propensão a ignorar instruções de uso de ferramentas e registro de dados, possivelmente devido ao roteamento dinâmico de parâmetros. Esse comportamento ressalta a necessidade de afinar as `system messages` e os prompts para **reduzir desvios de fluxo** em arquiteturas **MoE**.

Efetividade da solução multiagente A abordagem com a combinação dos agentes revisores semântico e contextual, recomendador de melhorias, reescritor e decisor final mostrou-se capaz de aprimorar **63,2%** das respostas inicialmente marcadas como incorretas no Exp. 1 e **46,4%** no Exp. 2 (Tabela 4).

A distribuição dos ganhos de pontuação (Figura 4) evidenciou melhorias médias de **+2,9** pontos no Exp. 1 e **+2,3** pontos no Exp. 2, comprovando a eficácia do ciclo crítico–refinador. O maior ganho alcançado com o Qwen3 32B sugere que modelos com maior número de parâmetros ativos por token tendem a produzir revisões mais consistentes. Isso demonstra que em cenários como esse pode ser mais adequado a utilização de um modelo mais robusto, enquanto em outros é possível abrir mão de poder computacional em troca de eficiência e agilidade. Como visto nas análises por acurácia segmentada (Tabela 5), o Qwen3 30B-A3B obteve resultados ligeiramente superiores ao modelo denso para a maioria dos casos, mesmo com a quantidade muito menor de parâmetros ativos.

O limiar de aceitação ($N_{\mathbf{sem}} + N_{\mathbf{ctx}} > 8$) e o máximo de **três iterações de reescrita** também revelaram-se adequados para obter um equilíbrio entre qualidade e custo computacional. Em apenas **4,0%** (Exp. 1) e **9,3%** (Exp. 2) das revisões houve piora na nota, indicando robustez do processo, mas também apontando para casos em que sugestões do recomendador divergiram dos julgamentos realizados pelos revisores semânticos e contextuais, especialmente em perguntas com alta complexidade, ambiguidade ou quando possivelmente existe uma ausência de informações para gerar uma resposta adequada e os agentes não são capazes de fazer essa avaliação.

Ressaltamos ainda que embora o experimento apresentado se concentre em atendimento para comércio eletrônico, nossa solução é **extensível a outros domínios** que exijam revisão contínua de conteúdos, como suporte técnico, educação automatizada e curadoria de informação, reforçando a versatilidade e a relevância da arquitetura apresentada.

Prós e contras dos modelos Qwen3

Qwen3-32B (denso)

- **Prós:** inferência estável, menor taxa de falha no seguimento de instruções e revisões com maior ganho médio de pontuação.
- **Contras:** maior consumo de memória (≈ 20 GB), latência de processamento mais alta (≈ 1 h 30 min por lote de 100 perguntas).

Qwen3-30B-A3B (MoE)

- **Prós:** redução de custos computacionais (≈ 18 GB de memória; tempo médio de 45 min por lote), desempenho próximo ao modelo denso em acurácia e F1.
- **Contras:** maior propensão a ignorar instruções de orquestração (*por entrar em "thinking mode"de maneira inadequado*) e ganho médio nas revisões ligeiramente inferior.

Limitações e sugestões de trabalhos futuros

1. *Refinamento de prompts e mensagens do sistema:* pode ser necessário revisar as **instruções dos agentes** para evitar que eles entrem em *"thinking mode"* e garantam **consistência** no uso de ferramentas, especialmente em arquiteturas MoE, devido a menor quantidade de parâmetros ativos.

2. *Enriquecimento do contexto:* é possível incorporar mecanismos de recuperação de documentos (RAG) ou conhecimento externo para reduzir alucinações e **melhorar a fundamentação das respostas**, garantindo uma maior autonomia aos agentes.

3. *Aprendizado online:* algo que poderia ser proveitoso para obter resultados mais precisos e consistente seria implementar chamadas a **APIs de busca** para perguntas que podem ser respondidas online, já que perguntas gerais em relação aos produtos vindo de **outras fontes** poderiam ser usadas para essa finalidade.

4. *Combinar diferentes modelos no pipeline de agentes:* como demonstrado nos experimentos realizados, modelos **MOE** podem ter desempenho comparável a modelos densos para tarefas de menor complexidade. Para otimizar o custo computacional envolvido e o desempenho geral da arquitetura, um **sistema de modelos misto** pode ser implementado.

6 Conclusão

Este trabalho apresentou uma proposta de **arquitetura multiagentes LLMs** para refinamento de um sistema de **Perguntas e Respostas** utilizado para responder clientes de comércio eletrônico. A arquitetura multiagentes foi composta por distintos agentes: revisores semântico e contextual, recomendador de melhorias no conteúdo, reescritor e decisor final em um fluxo iterativo de execução. Os experimentos realizados com os modelos **Qwen3 32B (denso)** e **Qwen3 30B-A3B (MoE)** demonstraram que ambos são capazes de produzir resultados comparáveis em relação acurácia e F1, embora o modelo denso apresente revisões com ganho ligeiramente superior e menor taxa de falhas execução de fluxo. O modelo MoE reduziu custos de memória e tempo de inferência, o que reforça seu potencial em cenários de produção em que **recursos computacionais são limitados**. Observamos que a abordagem desenvolvida "crítico-refinador" se demostrou uma maneira efetiva de lidar com problemas que exigem uma melhora iterativa dos resultados, podendo ser aplicado em diversos cenários em que é necessário um maior nível de **confiabilidade nas avaliações** sem um grande aumento de custo, tornando esse tipo sistema e arquitetura apto a atender a requisitos de domínios diversos em ambientes reais. Trabalhos futuros envolvem tanto avaliar a solução com conjunto de dados maiores quanto aplicar o mesmo para outros domínios de aplicação.

Agradecimentos

Este trabalho foi conduzido em colaboração com o projeto "Construção e uso de bases de conhecimento em sistemas automatizados de questão e respostas" – convênio

93454 – entre Unicamp e a GoBots Soluções Inteligentes LTDA. Agradecemos toda a colaboração dos envolvidos na empresa no contexto desse projeto e fornecimento dos créditos para o treinamento dos modelos utilizados. Este trabalho teve igualmente apoio do aluno de Mestrado do IC/UNICAMP, Rodrigo Oliveira Caus.

Referências

[1] Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., & Liu, T. (2023). *A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions*. https://arxiv.org/abs/2311.05232

[2] SimulTrans Team. (2024). Limitations of Language Models in Other Languages. SimulTrans Blog. https://www.simultrans.com/blog/limitations-of-language-models-in-other-languages

[3] Schut, L., Gal, Y., & Farquhar, S. (2025). Do Multilingual LLMs Think In English? https://arxiv.org/abs/2502.15603

[4] Center for Democracy & Technology. (2023). Large Language Models in Non-English Content Analysis Primer. https://cdt.org/wp-content/uploads/2023/05/non-en-content-analysis-primer-051223-1203.pdf

[5] Yin, Z., Sun, Q., Guo, Q., Wu, J., Qiu, X., & Huang, X. (2023). Do Large Language Models Know What They Don't Know? https://arxiv.org/abs/2305.18153

[6] Følstad, A., & Skjuve, M. (2023). Charting the Evolution and Future of Conversational Agents. Information Systems Frontiers, 26(2), 729–754. https://doi.org/10.1007/s10796-023-10375-9

[7] Oliveira, F., & Pereira, H. (2025). AI-Based Conversational Agents: A Scoping Review. ResearchGate. https://www.researchgate.net/publication/374265061_AI-Based_Conversational_Agents_A_Scoping_Review

[8] Zhang, X., Wang, Y., & Li, Z. (2023). A Comprehensive Overview of Large Language Models. https://arxiv.org/abs/2307.06435

[9] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. https://arxiv.org/abs/1706.03762

[10] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). *Language Models are Few-Shot Learners*. https://arxiv.org/abs/2005.14165

[11] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, E., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., & Zoph, B. (2022). *PaLM: Scaling Language Modeling with Pathways*. https://arxiv.org/abs/2204.02311

[12] Gomez, C., Yin, J., Huang, C.-M., & Unberath, M. (2024). How LLM-Powered Conversational Agents Influence Non-Experts in Making Triage Decisions. Frontiers in Computer Science, 6, Article 1427463. https://www.frontiersin.org/articles/10.3389/fcomp.2024.1427463/full

[13] Zhao, P., Chen, L., & Martínez, S. (2023). An In-depth Survey of Large Language Model-based Artificial Intelligence Agents. https://arxiv.org/abs/2309.14365

[14] Sun, L., Yang, Y., Duan, Q., Shi, Y., Chang, Y.-C., Lin, C.-T., & Shen, Y. (2025). Multi-Agent Collaboration Mechanisms: A Survey of LLMs. https://arxiv.org/abs/2501.06322

[15] Darin, C., & SSRN Editors. (2025). A Comprehensive Survey of AI Agent Frameworks and Their Transformative Impact on the Financial Services Industry. SSRN. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5252182

[16] IBM. (2025, June 5). How do AI agents collaborate? Inside the world of multi-agent orchestration. IBM Think Insights. https://www.ibm.com/think/insights/boost-productivity-efficiency-multi-agent-orchestration

[17] Guo, T., et al. (2024). A Survey on Multi-Generative Agent Systems: Recent Advances and Challenges. https://arxiv.org/abs/2412.17481

[18] Darin, C., & SSRN Editors. (2025). A Comprehensive Survey of AI Agent Frameworks and Their Transformative Impact on the Financial Services Industry. SSRN. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5252182

[19] Nguyen, T., Patel, R., & Gomez, L. (2024). Large Language Models Empowered Agent-Based Modeling and Simulation. Human Behavior and Emerging Technologies, 6(1), 45–58. https://doi.org/10.1057/s41599-024-03611-3

[20] Beni, G., & Wang, J. (1989). Swarm Intelligence in Cellular Robotic Systems. In P. Dario, G. Sandini, & P. Aebischer (Eds.), *Robots and Biological Systems: Towards a New Bionics?* (NATO ASI Series, Vol. 102, pp. 703–712). Springer. https://doi.org/10.1007/978-3-642-58069-7_38

[21] Wooldridge, M. (2009). *An Introduction to MultiAgent Systems* (2nd ed.). John Wiley & Sons. https://www.wiley.com/en-us/An%2BIntroduction%2Bto%2BMultiAgent%2BSystems%2C%2B2nd%2BEdition-p-9780470519462

[22] Shoham, Y., & Leyton-Brown, K. (2009). *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press. https://books.google.com/books/about/Multiagent_Systems.html?id=bMR_qScakukC

[23] Hou, Z., Tang, J., & Wang, Y. (2025). HALO: Hierarchical Autonomous Logic-Oriented Orchestration for Multi-Agent LLM Systems. https://arxiv.org/abs/2505.13516

[24] Chen, J.-C. Y., Prasad, A., Saha, S., Stengel-Eskin, E., & Bansal, M. (2024). MAgICoRe: Multi-Agent, Iterative, Coarse-to-Fine Refinement for Reasoning. https://arxiv.org/abs/2409.12147

[25] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., & Zhou, D. (2023). Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *International Conference on Learning Representations (ICLR) 2023*. Disponível em: https://arxiv.org/abs/2203.11171

[26] Madaan, A., et al. (2023). Self-Refine: Iterative Refinement with Self-Feedback. https://arxiv.org/abs/2303.17651

[27] Lin, Y.-C., Chen, K.-C., Li, Z.-Y., et al. (2025). Creativity in LLM-based Multi-Agent Systems: A Survey. https://arxiv.org/abs/2505.21116

[28] Chen, W., Yuan, J., Qian, C., Yang, C., Liu, Z., & Sun, M. (2024). Optima: Optimizing Effectiveness and Efficiency for LLM-Based Multi-Agent System. https://arxiv.org/abs/2410.08115

[29] Sami, H., ul Islam, M., Charas, S., Gandhi, A., Gaillardon, P.-E., & Tenace, V. (2025). Nexus: A Lightweight and Scalable Multi-Agent Framework for Complex Tasks Automation. https://arxiv.org/abs/2502.19091

[30] Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., Zaharia, M., Gonzalez, J. E., & Stoica, I. (2025). MAST: Multi-Agent System Failure Taxonomy. https://arxiv.org/abs/2503.13657

[31] Rouzrokh, P., & Shariatnia, M. (2025). LatteReview: A Multi-Agent Framework for Systematic Review Automation Using Large Language Models. https://arxiv.org/abs/2501.05468

[32] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive Mixtures of Local Experts. Neural Computation, 3(1), 79–87. https://doi.org/10.1162/neco.1991.3.1.79

A Apêndice

A.1 Mensagens de Sistema dos Agentes

Para cada um dos agentes foi definida uma `system message` em inglês, contendo instruções claras sobre o papel de cada um deles no todo e quais eram suas tarefas principais. A decisão de escrever as mensagens em inglês se deve ao fato da maioria dos LLMs terem sido treinados majoritariamente no idioma, garantindo uma maior adaptabilidade para diferentes modelos, além de garantir uma maior eficiência no seguimento de instruções.

```
semantic_reviewer = AssistantAgent(
    name="Semantic_Reviewer",
    llm_config=llm_config,
    system_message=(
        "You are the Semantic Reviewer. Your task is to critically
    evaluate the semantic accuracy of an answer provided to a user's
    question about a product.\n\n"
        "You will be provided with the following information:\n"
        "- **Question**: The user's inquiry regarding the product.\n"
        "- **Original Answer**: The initial response given to the user's
    question.\n"
        "- **Revised Answer**: The improved response provided by the
    Rewriter, if available.\n"
        "- **Category**: The category to which the product belongs.\n"
        "- **Intent**: The identified intent behind the user's
    question.\n\n"
        "Evaluation Instructions:\n"
```

```
13          "- If the Revised Answer and the Original Answer are not none,
    evaluate the Revised Answer and register the score and the
    justification for it.\n"
14          "- If the Revised Answer is none, evaluate the Original Answer and
    register the score and the justification for it.\n\n"
15          "Evaluation Criteria:\n"
16          "- The answer must directly and explicitly address all aspects of
    the user's question.\n"
17          "- It must be grammatically correct, free of spelling errors, and
    use appropriate language without mixing languages.\n"
18          "- The answer should be concise and avoid unnecessary
    information.\n"
19          "- Greetings and signatures shouldn't be taken into account in the
    evaluation, unless they are duplicated.\n"
20          "- Be particularly critical of answers that are vague, incomplete,
    or contain linguistic errors.\n\n"
21          "Provide a semantic score from 0 to 5, where 5 indicates a perfect
    semantic match.\n"
22          "You must always call the function register_semantic_score with
    your semantic_score and a brief justification in English, do nothing
    else.\n\n"
23      ),
24      functions=[register_semantic_score],
25  )
26
27  contextual_reviewer = AssistantAgent(
28      name="Contextual_Reviewer",
29      llm_config=llm_config,
30      system_message=(
31          "You are the Contextual Reviewer. Your task is to critically
    assess whether an answer provided to a user's question about a product
    aligns with the given context and metadata.\n\n"
32          "You will be provided with the following information:\n"
33          "- **Question**: The user's inquiry regarding the product.\n"
34          "- **Original Answer**: The initial response given to the user's
    question.\n"
35          "- **Revised Answer**: The improved response provided by the
    Rewriter, if available.\n"
36          "- **Category**: The category to which the product belongs.\n"
37          "- **Intent**: The identified intent behind the user's question.\n"
38          "- **Metadata**: Additional information and rules pertinent to the
    product or store policies.\n"
39          "- **Context**: Crucial details about the product, store, or other
```

```
    relevant information.\n\n"
40          "Evaluation Instructions:\n"
41          "- If the Revised Answer and the Original Answer are not none,
    evaluate the Revised Answer and register the score and the
    justification for it.\n"
42          "- If the Revised Answer is none, evaluate the Original Answer and
    register the score and the justification for it.\n\n"
43          "Evaluation Criteria:\n"
44          "- The answer must be consistent with the information provided in
    the context and metadata.\n"
45          "- It should not include information that cannot be inferred from
    the provided context.\n"
46          "- The answer should focus on information relevant to the user's
    question.\n"
47          "- Be particularly critical of answers that include assumptions,
    omit critical context, or misrepresent the provided information.\n\n"
48          "Provide a contextual score from 0 to 5, where 5 indicates perfect
    contextual alignment.\n"
49          "You must always call the function register_contextual_score with
    your contextual_score and a brief justification in English, do nothing
    else.\n\n"
50      ),
51      functions=[register_contextual_score],
52  )
53
54  suggester = AssistantAgent(
55      name="Suggester",
56      llm_config=llm_config,
57      system_message=(
58          "You are the Suggester. Your purpose is to suggest improvements
    for an answer provided to a user's question about a product.\n\n"
59          "You will be provided with:\n"
60          "- **Question**: The user's inquiry regarding the product.\n"
61          "- **Original Answer**: The response given to the user's
    question.\n"
62          "- **Semantic Score**: A score from 0 to 5 indicating the semantic
    accuracy of the answer.\n"
63          "- **Contextual Score**: A score from 0 to 5 indicating the
    contextual accuracy of the answer.\n"
64          "- **Justifications**: Brief explanations for the semantic and
    contextual scores.\n\n"
65          "Based on the scores and justifications provided by the reviewers,
    you must provide suggestions for improvement.\n"
```

```
66          "- Focus on addressing specific issues highlighted in the
    justifications.\n"
67          "- Ensure that your suggestions are actionable and aimed at
    enhancing the answer's quality.\n\n"
68          "Do not provide a revised answer, only suggestions for
    improvement.\n"
69          "The suggestions must be in English, while the question and answer
    may be in Portuguese or Spanish.\n"
70          "You must always call the function register_suggestions with your
    suggestions as a parameter, do nothing else.\n"
71      ),
72      functions=[register_suggestions],
73  )
74
75  rewriter = AssistantAgent(
76      name="Rewriter",
77      llm_config=llm_config,
78      system_message=(
79          "You are the Rewriter. Your task is to rewrite answers that have
    not been evaluated positively by the reviewers, ensuring they meet
    both semantic and contextual standards.\n\n"
80          "You will be provided with the following information:\n"
81          "- **Question**: The user's inquiry regarding the product.\n"
82          "- **Original Answer**: The initial response given to the user's
    question.\n"
83          "- **Suggestions**: Recommendations for improvement provided by
    the Suggester.\n"
84          "- **Context**: Crucial details about the product, store, or other
    relevant information.\n"
85          "- **Category**: The category to which the product belongs.\n"
86          "- **Intent**: The identified intent behind the user's question.\n"
87          "- **Metadata**: Additional information and rules pertinent to the
    product or store policies.\n\n"
88          "Evaluation Criteria:\n"
89          "- The revised answer must directly and explicitly address all
    aspects of the user's question.\n"
90          "- It must be consistent with the information provided in the
    context and metadata.\n"
91          "- The answer should be grammatically correct, free of spelling
    errors, and use appropriate language without mixing languages.\n"
92          "- Retain any greetings or signatures present in the original
    answer, only removing duplicates, if any.\n"
93          "- If there isn't enough information to provide a revised answer,
```

```
    return 'CANNOT REWRITE'.\n\n"
94          "Provide the revised answer in the original language of the
    question.\n"
95          "You must always call the function register_revised_answer with
    your revised answer as a parameter, do nothing else.\n\n"
96      ),
97      functions=[register_revised_answer],
98  )
99
100 decider = AssistantAgent(
101     name="Decider",
102     llm_config=llm_config,
103     system_message=(
104         "You are the Decider. Your task is to determine whether the
    revised answer provided to a user's question about a product is
    acceptable, requires further improvement, or if the question should
    not be answered at all.\n\n"
105         "You will be provided with the following information:\n"
106         "- **Question**: The user's inquiry regarding the product.\n"
107         "- **Original Answer**: The initial response given to the user's
    question.\n"
108         "- **Revised Answer**: The improved response provided by the
    Rewriter.\n"
109         "- **Context**: Crucial details about the product, store, or other
    relevant information.\n"
110         "- **Category**: The category to which the product belongs.\n"
111         "- **Intent**: The identified intent behind the user's question.\n"
112         "- **Metadata**: Additional information and rules pertinent to the
    product or store policies.\n"
113         "- **Semantic and Contextual Scores**: Scores and justifications
    provided by the reviewers.\n"
114         "- **Suggestions**: Recommendations for improvement provided by
    the Suggester.\n\n"
115         "Evaluation Criteria:\n"
116         "- Determine if the revised answer fully addresses the user's
    question with semantic and contextual accuracy.\n"
117         "- Do not accept answers that mention another product unless it is
    mentioned in the context or metadata, containing a link to it.\n"
118         "- Do not accept answers that state any part of the question
    cannot be answered due to insufficient information.\n"
119         "- Be particularly critical of answers that are vague, incomplete,
    or contain incorrect information.\n"
120         "- If the number of revisions is 2 or more and the revised answer
```

```
    is still not good enough, the decision must be 'DO_NOT_ANSWER'.\n\n"
121         "Possible Decisions:\n"
122         "- **ANSWER_REVISED**: The revised answer is acceptable and fully
    addresses the question.\n"
123         "- **REWRITE**: The revised answer is not good enough, but can be
    improved based on the given information.\n"
124         "- **DO_NOT_ANSWER**: The revised answer is not good enough and
    cannot be improved based on the given information.\n\n"
125         "You must always call the function register_decision with your
    decision and a brief justification in English, do nothing else.\n\n"
126     ),
127     functions=[register_decision],
128 )
```

Código 1: Systems messages definidas para os agentes

A.2 Funções de Registro de Variáveis de Contexto e Decisão do Próximo Agente

Cada um dos agentes também possui uma `function` associada, nas mensagens do sistema eles são instruídos a chamar essa função a cada iteração, passando os parâmetros adequados para cada finalidade. Através do parâmetro `target` do `ReplyResult`, essas funções também são responsáveis por invocar o agente que deve ser executado na sequência de acordo com os resultados registrados. Enquanto o atributo `message` faz um breve registro do que foi feito nessa função, armazenando-a no histórico do chat junto com as demais trocas de informação entre os agentes.

```
1  def register_semantic_score(semantic_score: int, justification: str,
       context_variables: ContextVariables) -> ReplyResult:
2      """
3      Register the semantic score and justification in the context variables.
4      """
5      if (context_variables.get("revised_answer") is None):
6          context_variables["semantic_score"] = semantic_score
7          context_variables["justification_semantic"] = justification
8      else:
9          context_variables["revised_answer_semantic_score"] = semantic_score
10         context_variables["revised_answer_justification_semantic"] =
       justification
11
12     return ReplyResult(
13         context_variables=context_variables,
14         target=AgentTarget(contextual_reviewer),
```

```
15          message="The semantic score and justification have been registered, handing over to the Contextual Reviewer for his review.",
16      )
17
18
19  def register_contextual_score(contextual_score: int, justification: str, context_variables: ContextVariables) -> ReplyResult:
20      """
21      Register the contextual score and justification in the context variables.
22      """
23      original_score = None
24      new_score = None
25
26      if (context_variables.get("revised_answer") is None):
27          context_variables["contextual_score"] = contextual_score
28          context_variables["justification_contextual"] = justification
29
30          semantic_score = context_variables["semantic_score"]
31          original_score = (contextual_score + semantic_score)
32
33          context_variables["original_score"] = original_score
34      else:
35          context_variables["revised_answer_contextual_score"] = contextual_score
36          context_variables["revised_answer_justification_contextual"] = justification
37
38          semantic_score = context_variables["revised_answer_semantic_score"]
39          new_score = (contextual_score + semantic_score)
40
41          context_variables["new_score"] = new_score
42
43      if original_score is not None and original_score > 8:
44          return ReplyResult(
45              context_variables=context_variables,
46              target=TerminateTarget(),
47              message="The original score is greater than 8, terminating the process.",
48          )
49      elif new_score is not None:
50          return ReplyResult(
51              context_variables=context_variables,
```

```
52              target=AgentTarget(decider),
53              message="The new score has been registered, handing over to
    the Decider to make a decision about the revised answer.",
54          )
55      else:
56          return ReplyResult(
57              context_variables=context_variables,
58              target=AgentTarget(suggester),
59              message="The contextual score and justification have been
    registered, handing over to the Suggester to suggest improvements.",
60          )
61
62
63  def register_suggestions(suggestions: str, context_variables:
    ContextVariables) -> ReplyResult:
64      """
65      Register the suggestions in the context variables.
66      """
67      context_variables["suggestions"] = suggestions
68
69      return ReplyResult(
70          context_variables=context_variables,
71          target=AgentTarget(rewriter),
72          message="The suggestions have been registered, handing over to the
    Rewriter to write a new answer.",
73      )
74
75
76  def register_revised_answer(revised_answer: str, context_variables:
    ContextVariables) -> ReplyResult:
77      """
78      Register the revised answer in the context variables.
79      """
80      context_variables["revised_answer"] = revised_answer
81      context_variables["number_of_revisions"] += 1
82
83      if re.search(r"^CANNOT REWRITE$", revised_answer):
84          context_variables["final_answer"] = "DO_NOT_ANSWER"
85
86          return ReplyResult(
87              context_variables=context_variables,
88              target=TerminateTarget(),
89              message="It's not possible to write a new answer, terminating
```

```
    the process.",
90          )
91
92      return ReplyResult(
93          context_variables=context_variables,
94          target=AgentTarget(semantic_reviewer),
95          message="The revised answer has been registered, handing over to
    the Semantic Reviewer to review it.",
96      )
97
98
99  def register_decision(decision: str, justification: str,
    context_variables: ContextVariables) -> ReplyResult:
100     """
101     Register the decision in the context variables.
102     """
103     context_variables["decision"] = decision
104     context_variables["decision_justification"] = justification
105
106     if decision == "ANSWER_REVISED":
107         context_variables["final_answer"] =
    context_variables["revised_answer"]
108
109         return ReplyResult(
110             context_variables=context_variables,
111             target=TerminateTarget(),
112             message="The decision is 'ANSWER_REVISED', terminating the
    process.",
113         )
114     elif decision == "REWRITE":
115         context_variables["original_answer"] =
    context_variables["revised_answer"]
116         context_variables["original_answer_semantic_score"] =
    context_variables["revised_answer_semantic_score"]
117         context_variables["original_answer_justification_semantic"] =
    context_variables["revised_answer_justification_semantic"]
118         context_variables["original_answer_contextual_score"] =
    context_variables["revised_answer_contextual_score"]
119         context_variables["original_answer_justification_contextual"] =
    context_variables["revised_answer_justification_contextual"]
120         context_variables["original_score"] =
    context_variables["new_score"]
121         context_variables["revised_answer"] = None
```

```
122         context_variables["revised_answer_semantic_score"] = None
123         context_variables["revised_answer_justification_semantic"] = None
124         context_variables["revised_answer_contextual_score"] = None
125         context_variables["revised_answer_justification_contextual"] = None
126         context_variables["new_score"] = None
127
128         return ReplyResult(
129             context_variables=context_variables,
130             target=AgentTarget(rewriter),
131             message="The decision is 'REWRITE', handing over to the
    Rewriter to write a new answer.",
132         )
133
134     context_variables["final_answer"] = "DO_NOT_ANSWER"
135
136     return ReplyResult(
137         context_variables=context_variables,
138         target=TerminateTarget(),
139         message="The decision is 'DO_NOT_ANSWER', terminating the
    process.",
140     )
```

Código 2: Funções de registro associadas aos agentes