A VIABILIDADE DE UMA ARQUITETURA HÍBRIDA DE CHATBOT COM RASA E LLMS LOCAIS

https://doi.org/10.22533/at.ed.1281125170311

Data de aceite: 12/03/2025

Sergio Juniors Garcez
Instituto Federal de Ciência e Tecnologia do Amazonas (IFAM) Campus Manaus Zona Leste

Joethe Moraes de Carvalho
Instituto Federal de Ciência e Tecnologia do Amazonas (IFAM) Campus Manaus Centro
http://lattes.cnpq.br/8157292652509113

RESUMO:Em um cenário onde o uso de inteligência artificial cresce exponencialmente, especialmente em ambientes corporativos e industriais, torna-se essencial garantir controle, segurança e eficiência de custos. Este trabalho propõe e avalia uma arquitetura híbrida inovadora para chatbots, onde o framework Rasa é utilizado e aplica seu modelo de linguagem menor, personalizado e controlável para atuar como gestor e filtro de uma Large Language Model (LLM) mais complexa e robusta. A proposta é utilizar o Rasa como interface principal com o usuário, responsável por interpretar comandos e aplicar regras, delegando tarefas à LLM apenas quando a complexidade da requisição exige maior capacidade de compreensão de linguagem. A viabilidade técnica da abordagem foi validada por meio de testes de acurácia e latência, que, embora tenham revelado uma latência considerável em um ambiente de desenvolvimento, confirmam o potencial da arquitetura para um modelo de custo fixo e a sua aplicabilidade em diversos cenários práticos.

PALAVRAS-CHAVE: Rasa, Large Language Model (LLM), Chatbot, Arquitetura Híbrida.

THE FEASIBILITY OF A HYBRID CHATBOT ARCHITECTURE WITH RASA AND LOCAL LLMS

ABSTRACT: In a scenario where the use of artificial intelligence is growing exponentially, especially in corporate and industrial environments, ensuring control, security, and cost efficiency is essential. This paper proposes and evaluates an innovative hybrid architecture for chatbots, utilizing the Rasa framework and applying its smaller, personalized, and controllable language model to act as a manager and filter for a more complex and robust Large Language Model (LLM). The proposal is to use Rasa as the main user interface, responsible for interpreting commands and applying rules, delegating tasks to the LLM only when the

complexity of the request requires greater language understanding. The technical feasibility of the approach was validated through accuracy and latency tests, which, although revealing considerable latency in a development environment, confirm the architecture's potential for a fixed-cost model and its applicability in various practical scenarios.
KEYWORDS: Rasa, Large Language Model (LLM), Chatbot, Hybrid Architecture.

INTRODUÇÃO

O presente estudo surge com o objetivo de aprimorar os chatbots tradicionais, que são amplamente utilizados no cenário digital. Com a crescente adoção de agentes de inteligência artificial no mercado, o usuário tende a preferir soluções mais 'inteligentes' em detrimento das que oferecem apenas respostas padronizadas e pouco flexíveis (KARANIKOLAS et al., 2023). Neste trabalho, o termo 'Large Language Model (LLM)' será utilizado para se referir a modelos de linguagem generativos, independentemente de seu tamanho de parâmetros, incluindo modelos de menor escala (Small Language Models) ou SLMs, por serem capazes de raciocinar e responder perguntas comuns (BEATTY, 2024). Entretanto, adotar um agente de inteligência artificial ou um modelo LLM não é tão simples, barato e seguro, tendo em vista que os mais conhecidos são terceirizados (SHASHIDHAR et al., 2023).

A ascensão das Large Language Models (LLMs) de código aberto, como o Phi-3, Deepseek-r1, oferecem uma alternativa robusta para resolver essa limitação. No entanto, a adoção completa de LLMs em sistemas de produção pode ser desafiadora, apresentando questões como a imprevisibilidade das respostas, o gerenciamento de custos e os requisitos computacionais (GARCIA, 2025). Para mitigar esses desafios, as arquiteturas híbridas emergem como uma solução promissora. A aplicação do controle e robustez de uma ferramenta de IA conversacional altamente usada no mercado, Rasa e a complexidade de uma LLM para compreender as intenções do usuário através das respostas combinam em uma arquitetura híbrida de muito potencial (KARANIKOLAS et al., 2023).

A proposta deste trabalho é avaliar uma arquitetura híbrida que integra a flexibilidade, rapidez e controle de um framework de IA conversacional tradicional (Rasa) com o poder generativo de uma LLM de código aberto hospedada de forma local no Ollama. O objetivo principal é verificar a viabilidade técnica dessa integração, analisando o desempenho da solução em termos de acurácia e latência.

REFERENCIAL TEÓRICO

Chatbots e Rasa

Os chatbots têm sido uma tecnologia central para a automação de interações, evoluindo de sistemas baseados em regras pré-definidas para agentes conversacionais mais sofisticados. No entanto, a arquitetura tradicional desses sistemas é frequentemente limitada. Chatbots "tradicionais", também conhecidos como sistemas baseados em

intenções, operam através de um fluxo de trabalho estruturado, dependendo de uma arquitetura que processa a entrada do usuário em estágios distintos (GHOLAMI; OMAR, 2023).

O primeiro componente crucial dessa arquitetura é a Compreensão de Linguagem Natural (NLU). Neste momento, a entrada de texto do usuário é analisada para identificar duas informações principais: a intenção (o objetivo do usuário) e as entidades (as informações relevantes na frase) (VASLOV; BOCKLISCH, 2017). Essa abordagem de classificação de intenções, apesar de eficaz para tarefas bem definidas, apresenta dificuldades. A principal limitação reside na dependência de conjuntos de dados de treinamento bem rotulados. Se uma frase não se encaixa perfeitamente em uma intenção pré-definida, o classificador pode falhar em compreendê-la, levando a respostas incorretas ou ao "fallback" (GHOLAMI; OMAR, 2023).

O *Rasa Open Source* é um dos frameworks de código aberto mais utilizados para a construção de chatbots que seguem essa arquitetura baseada em intenções (SHARMA; JOSHI, 2020), seu design une a NLU e o gerenciamento de diálogo. O componente agora definido como *Rasa Open Source* (Figura 1) é responsável pela classificação de intenções, extração de entidades e utiliza um modelo de "política" para prever a próxima ação com base no histórico da conversa (GMBH, 2025). Embora essa abordagem ofereça controle, clareza e previsibilidade, a sua falta de flexibilidade em cenários com falha na classificação da intenção demonstra a necessidade de uma estratégia mais robusta para lidar com a ambiguidade da linguagem natural (Yuan et al., 2025).

Como dito por (SOARES, 2023, p. 7) "Um chatbot desenvolvido em Rasa é um misto de baseado em regras e baseado em Inteligência Artificial e o seu desempenho é fundamentalmente dependente do conjunto de dados de pares pergunta-resposta utilizado para treinar".

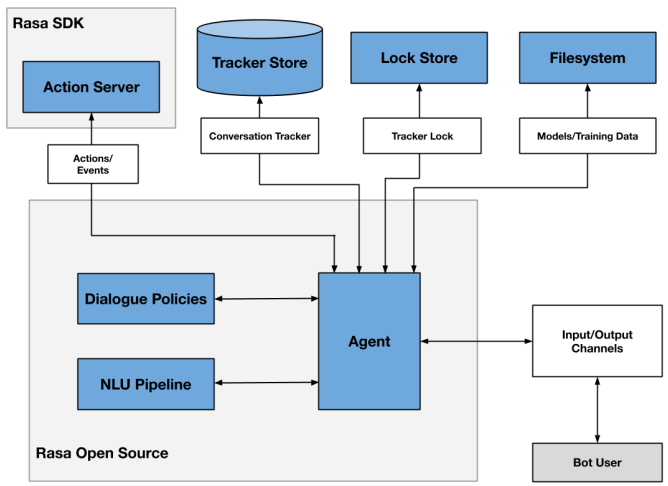

Figura 1 - Diagrama da Arquitetura Rasa

Fonte: (GMBH,2025)

Viabilidade dos LLMs

Com o surgimento das Large Language Models (LLMs) uma nova era no desenvolvimento de agentes conversacionais começou, superando as limitações dos chatbots tradicionais baseados em Compreensão de Linguagem Natural (NLU) e regras (VASLOV; BOCKLISCH, 2017). Enquanto um modelo NLU comum é treinado para classificar uma intenção e extrair entidades de um conjunto de dados predefinidos, os LLMs são modelos generativos, capazes de compreender e produzir texto de forma fluida, abrangendo um vasto conhecimento e inferindo respostas para cenários não previstos nos dados de treinamento (KARANIKOLAS et al., 2023).

A viabilidade de integrar LLMs em aplicações privadas traz consigo dois problemas: o custo elevado das APIs de modelos proprietários e as preocupações com a privacidade dos dados. Para cada interação, um custo é cobrado, e os dados do usuário precisam ser enviados para servidores de terceiros, o que pode violar políticas de segurança e soberania de dados.

No entanto, um avanço significativo que mudou esse panorama é a democratização de LLMs de código aberto e a capacidade de realizar a "inferência local" (SHASHIDHAR et al., 2023). Rodar modelos localmente oferece uma série de vantagens críticas como aumento na privacidade e segurança. Os dados sensíveis dos usuários não saem da infraestrutura local da empresa, assegurando a permanência das possíveis informações sigilosas. (GARCIA, 2025, s.p.)

Outro fator é a redução de custos, ao eliminar a dependência de APIs de terceiros e de seus modelos de cobrança por token, resultando em um custo operacional fixo, ligado apenas ao hardware. O equipamento fixo e dedicado, de forma local, impacta na ausência da latência de rede entre o cliente e o servidor da API, permitindo respostas quase instantâneas. (GARCIA, 2025, s.p.)

A comunidade de código aberto tem desenvolvido LLMs menores, mas com alta capacidade, como a família de modelos Phi-3 da Microsoft, deepseek-r1 da Deepseek e qwen3 da Alibaba. Conforme (BEATTY, 2024) o artigo da Microsoft sobre o Phi-3, o modelo Phi-3-Mini, com apenas 3,8 bilhões de parâmetros, consegue competir em desempenho com modelos significativamente maiores em tarefas de raciocínio e compreensão de linguagem.

Para facilitar o processo de build desses modelos, surgiu o Ollama. Ele simplifica radicalmente o processo de download, gerenciamento e execução de LLMs. O Ollama atua como um servidor local que expõe uma API REST, facilitando a integração de

modelos a outros sistemas, como o Rasa Action Server, sem a necessidade de gerenciar complexidades de hardware (ROTIROTI, 2024, s.p.).

Arquiteturas Híbridas

Com o crescimento das Large Language Models (LLMs), surgiu um avanço inegável na capacidade dos agentes conversacionais de lidar com a complexidade da linguagem humana. Entretanto, a implementação desses modelos em cenários de produção, especialmente para tarefas rotineiras e de alta frequência, pode apresentar desafios relacionados à previsibilidade e ao custo (GARCIA, 2025, s.p.).

Nesse contexto, as arquiteturas híbridas surgem como uma solução pragmática e eficiente, combinando o melhor de dois mundos: a robustez e o controle dos sistemas de chatbot conversacionais baseados em regras e modelos pré-treinados, com a complexidade e raciocínio das LLMs.

O princípio central desta arquitetura híbrida é a lógica de fallback. Em vez de substituir completamente o sistema conversacional tradicional, a LLM é posicionada como um "plano B" ou uma "válvula de escape". O fluxo da conversa é planejado para que o *Rasa Open Source* (Figura 1) tente, prioritariamente, processar a solicitação do usuário usando suas regras e modelos de NLU internos. Isso garante que interações bem definidas e previsíveis (como "agendar uma consulta" ou "consultar saldo") sejam tratadas de forma rápida e eficiente. Somente quando o assistente falha em identificar uma intenção clara (classificando a mensagem como nlu_fallback) ou classificando a intenção como "baixa confiança", a conversa é redirecionada para a LLM.

Este redirecionamento é orquestrado por uma ação personalizada (custom action) no *Action Server* (Figura 1). Essa ação captura a mensagem do usuário e a encaminha para o servidor do back-end (Django) onde é adicionado todos os parâmetros necessários para o LLM. Após isso é redirecionado para o servidor de inferência local, que executa a LLM (por exemplo, via Ollama). O modelo então gera uma resposta com a verdadeira intenção e nível de confiança, que é retornada ao *Action Server*. Este servidor retorna uma mensagem personalizada para o usuário preenchendo, assim, a lacuna deixada pelo modelo tradicional. Essa abordagem tira proveito da previsibilidade e do baixo custo dos modelos tradicionais para a maioria das interações, enquanto reserva o poder computacional e a flexibilidade da LLM para os casos mais complexos, incertos ou que exigem aleatoriedade.

METODOLOGIA

Este estudo emprega uma metodologia de pesquisa aplicada, com o objetivo de desenvolver e validar uma arquitetura híbrida de chatbot capaz de superar as limitações dos modelos tradicionais de NLU. A abordagem proposta integra um LLM de código aberto, executado localmente, como um mecanismo de fallback para aprimorar a precisão da

classificação de intenções. A seguir, detalha-se a arquitetura do sistema, os modelos e as ferramentas utilizadas, e os procedimentos experimentais que foram conduzidos para avaliar o desempenho e a viabilidade da solução.

A escolha de uma arquitetura híbrida foi motivada, como dito anteriormente, pela necessidade de mitigar as limitações do processamento de linguagem natural (NLU) tradicional, especialmente a sua fragilidade na classificação de intenções em cenários com baixa confiança. Em situações onde um fallback genérico seria acionado, um toque de inteligência mudaria completamente a experiência do usuário. Esta solução foi esboçada para combinar a completude do framework Rasa no quesito, diálogos previsíveis e capacidade de raciocínio. Complementando isso, a generalização de um LLM de código aberto. A utilização de uma API Django como orquestradora foi uma decisão estratégica para retirar a lógica de classificação da LLM do núcleo do chatbot, visando maior praticidade, escalabilidade e a possibilidade de integrar diferentes modelos de IA no futuro.

Arquitetura do Sistema

A proposta é pautada em uma arquitetura de microsserviços, que integra um framework de chatbot tradicional (Rasa) com um Large Language Model (LLM) de código aberto. A comunicação é intermediada por uma API RESTful desenvolvida em Django, que atua como mediador do fluxo de dados. A (Figura 2) apresenta um diagrama de interação com o sistema que ilustra a ação entre os componentes, detalhando o fluxo de uma mensagem do usuário desde a sua origem até a resposta final.

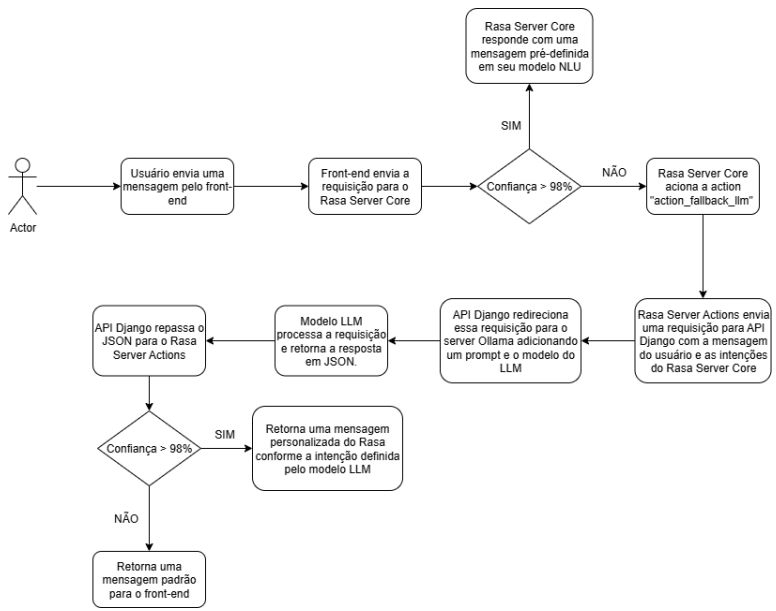

Figura 2 - Diagrama de Interação com sistema

Fonte: Autoria própria

Como mostrado na (Figura 2), o usuário pode receber dois tipos principais de retorno. No primeiro cenário, o usuário envia uma mensagem que já foi categorizada previamente pelo modelo do Rasa NLU, resultando em uma resposta instantânea.

O segundo cenário ocorre quando a mensagem é identificada como 'fora de escopo' ou 'nlu_fallback' pelo Rasa NLU. Após analisar as intenções com as quais foi treinado, o modelo detecta que a entrada do usuário não corresponde com um nível de confiança suficiente. Nessa situação, o servidor de ações é acionado e prepara uma requisição estruturada com o conteúdo da mensagem e a lista de intenções para a API Django.

A API Django, atuando como orquestradora, organiza essas informações em um prompt e as envia ao servidor do Ollama. Este servidor, por sua vez, armazena e executa o LLM localmente. Ao receber o prompt, o modelo de linguagem analisa a mensagem em relação às intenções fornecidas e determina qual delas é a mais correspondente, retornando um JSON com a sua decisão.

Se a intenção classificada for a de fallback ou se a confiança do LLM for baixa (por exemplo, abaixo de um limite de 90%), o intermediador envia essa decisão de volta para o servidor de ações do Rasa, que então gera uma resposta padrão de fallback para o usuário. Caso contrário, a API Django envia essa resposta de volta para o Rasa de forma customizada, que utiliza a intenção e a confiança para gerar a resposta apropriada para o usuário.

Tecnologias Utilizadas

Este software foi criado com o fim de aperfeiçoar o trabalho de uma framework muito robusto que é o Rasa, mas também todos os chatbots em geral. Para isso, foram necessárias algumas ferramentas que pudessem fazer isso ser possível. A primeira escolha feita para o projeto foi a parte visual (front-end). Após algumas pesquisas, foi perceptível que o Next.js era a opção mais ajustada para o serviço. Tem como foco o alto desempenho através da pré-renderização de páginas, o que é de vital importância em um projeto que a latência é importante.

A parte funcional do software (back-end) precisava ser feita para processar a mensagem do usuário. Neste caso, duas tecnologias foram utilizadas: Django e Rasa. O Django foi escolhido por sua escalabilidade e conhecimento prévio do autor sobre a ferramenta que tem domínio da linguagem.

O Rasa é insubstituível neste projeto, ele proporciona a criação de assistentes com lógica e fluxos de conversa personalizados, controle completo sobre dados de treino,

modelos e pipelines de implantação. Este repertório de junto a LLM local, é o que sustenta o conceito da proposta.

A escolha pelo Ollama é a possibilidade de executar os modelos LLM localmente, o que foi fundamental para os seus testes de latência, eliminando a dependência de APIs externas. Sem falar na possível redução de custos devido a estar consumindo poder computacional localmente e pelo fato dos modelos serem *open-source*.

A avaliação dos modelos foi analisada a partir de três modelos que possuem código aberto: o Phi-3 mini:3.8b, o Qwen3:4b e o DeepSeek-r1:1.5b. A seleção desses modelos foi estratégica, pois todos eles representam um baixo consumo computacional devido a sua quantidade de parâmetros. Segundo (GHOLAMI; OMAR, 2023, p. 9, tradução própria) "Um tamanho maior de modelo normalmente permite que ele aprenda representações mais complexas, mas também aumenta o risco de sobreajuste e requer mais recursos computacionais".

TESTES E RESULTADOS

Neste capítulo, são apresentados os resultados quantitativos e qualitativos obtidos após a implementação e concretização do projeto. O objetivo central é validar a viabilidade da arquitetura proposta, especificamente em relação à integração de Modelos de Linguagem de grande porte (LLMs) de código aberto. A avaliação é focada na acurácia e na latência dos modelos, a partir de um prompt definido para extrair informações no formato JSON. Para garantir a consistência e a validade dos resultados, todos os modelos foram submetidos a testes sob as mesmas condições e com a mesma base de dados de entrada e saída esperada.

Configuração Experimental

Os experimentos foram conduzidos para verificar a viabilidade deste projeto, e, com o objetivo de criar um ambiente o mais estável e otimizado possível, algumas ações específicas foram tomadas. Devido ao uso de hardware com arquitetura AMD, uma versão do Ollama com suporte à tecnologia ROCm (uma pilha de software aberto que oferece um conjunto de otimizações para cargas de trabalho de IA) foi obtida diretamente do repositório do projeto no GitHub (ollama-for-amd). Esta medida foi crucial para garantir a aceleração por GPU, otimizando a velocidade de inferência e, consequentemente, diminuindo a latência, um dos focos centrais deste estudo. Para que a versão do Ollama reconhecesse a placa de vídeo, foi necessária a instalação de arquivos do HIP (Heterogeneous-computing Interface for Portability) correspondentes à placa de vídeo utilizada.

Para realizar os testes, foi disponibilizado um setup com as seguintes especificações:

- Hardware

- Processador: AMD Ryzen 7 3700x;
- Mémoria RAM: 32gb/DDR4 3200mhz;
- Placa de vídeo: AMD Radeon RX 6600m ;

- Software

- Sistema Operacional: Windows 10 Home;
- Ollama Setup v0.11.6 (versão específica com suporte ROCm);
- Python: 3.9.13;
-Django: 5.2.5;
-django-cors-headers: 4.7.0;
-djangorestframework: 3.16.1;
-rasa: 3.6.21;
-rasa-sdk: 3.6.2;

Medição dos Resultados

A medição será pautada sobre duas variáveis, acurácia e latência dos modelos. A acurácia foi mensurada pela porcentagem de respostas corretas em relação aos dados de saída esperados no formato JSON (Figura 3). Foram enviadas cinco frases com a intenção esperada. A latência foi calculada pelo tempo de resposta que o modelo teve para responder a requisição da API Django (Figura 4).

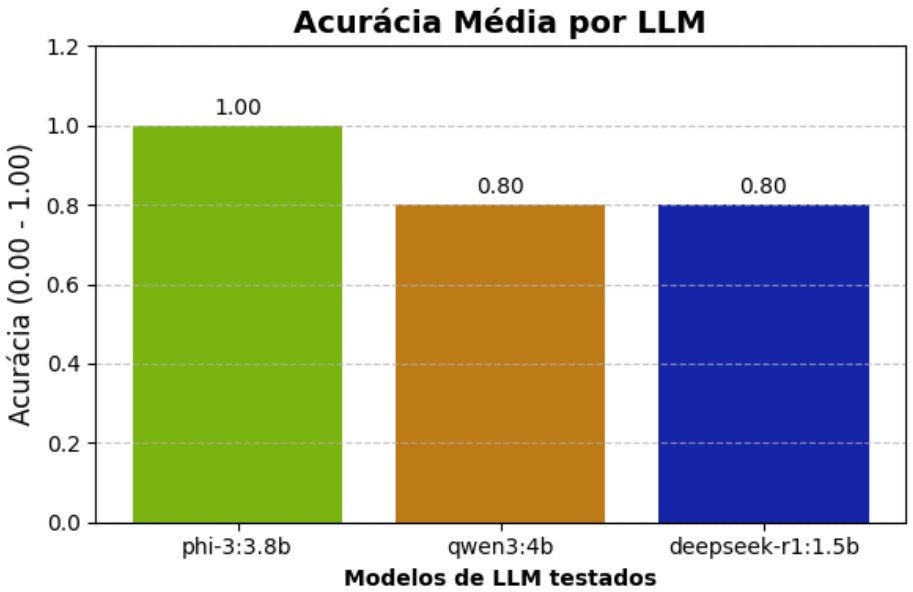

Figura 3 - Acurácia Média por Modelo

Fonte: Autoria Própria

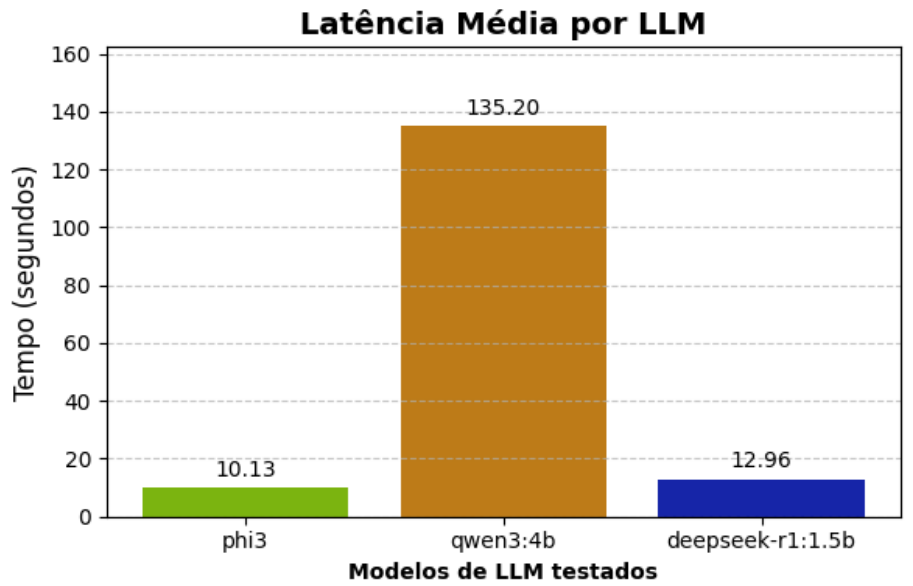

Figura 4 - Latência Média por Modelo

Fonte: Autoria Própria

Conforme demonstrado nos resultados, o modelo phi-3 destacou-se em ambas as métricas. Em termos de acurácia (Figura 3), ele apresentou a maior média, indicando uma performance superior na extração correta de informações em formato JSON. Já em relação à latência (Figura 4), o phi-3 obteve o menor tempo de resposta, confirmando sua maior velocidade de inferência, seguido de perto pelo deepseek-r1.

CONCLUSÃO

Com base nos testes e nos resultados obtidos, este trabalho demonstra a viabilidade da aplicação de uma arquitetura híbrida de chatbot, que integra o Rasa a uma Large Language Model (LLM) local. Os resultados indicam um potencial promissor para diversos cenários, como ambientes industriais e comerciais, principalmente por oferecer um modelo de custo fixo previsível. Apesar da latência considerável observada durante os testes, é crucial contextualizar que este resultado está diretamente ligado às limitações do ambiente computacional utilizado, que não foi otimizado para um cenário de produção. Assim, os dados corroboram que a abordagem é tecnicamente possível.

Tendo em vista a continuidade deste estudo, sugere-se que trabalhos futuros foquem na otimização da arquitetura proposta para aprimorar o desempenho e a experiência do usuário. Experimentos com um maior volume de dados seriam necessários para obter uma acurácia mais fidedigna. Testes utilizando hardwares mais robustos, como GPUs dedicadas de alto desempenho, na redução da latência observada, visando um ambiente de produção. É a se considerar a variação de hardwares para outras marcas como: NVidia, devido ao suporte que o Ollama oferece ao CUDA (API destinada a computação paralela). Adicionalmente, seria relevante explorar a integração com uma variedade de outros modelos de linguagem de código aberto, como Llama 3, Mistral, Gemma, para verificar suas acurácias e latências. Por fim, a avaliação da arquitetura sob a perspectiva de usabilidade e impacto no negócio, com testes de usabilidade e métricas de satisfação do usuário, poderiam validar a proposta em um cenário prático. A dedicação de tempo para estudar, testar e analisar as possibilidades será essencial para maturação da proposta e confirmar, assim, o veredito da viabilidade do projeto.

REFERÊNCIAS

BEATTY, Sally. **Tiny but mighty: The Phi-3 small language models with big potential**. Microsoft Source, 23 abr. 2024. Disponível em: https://news.microsoft.com/source/features/ai/the-phi-3-small-language-models-with-big-potential/. Acesso em: 5 set. 2025.

GARCIA, Mathias Brem. **LLMs On-Premises: Vale a Pena Rodar Modelos de IA Localmente?** Rox Partner, 13 mar. 2025. Disponível em: https://roxpartner.com/llms-on-premises-vale-a-pena-rodar-modelos-de-ia-localmente/.

GHOLAMI, A., & OMAR, S. (2023). **Do Generative Large Language Models need billions of parameters?**. Disponível em: https://arxiv.org/pdf/2309.06589

GMBH, R. T. **Architecture Overview l Rasa Open Source Documentation**. 2025. Disponível em: https://legacy-docs-oss.rasa.com/docs/rasa/arch-overview.

KARANIKOLAS, Nikitas; MANGA, Eirini; SAMARIDI, Nikoletta; TOUSIDOU, Eleni; VASSILAKOPOULOS, Michael. **Large Language Models versus Natural Language Understanding and Generation**. In: PAN-HELLENIC CONFERENCE ON PROGRESS IN COMPUTING AND INFORMATICS, 27., 2023, Lamia, Greece. *Proceedings…* New York: Association for Computing Machinery, 2024. p. 278-290. DOI: https://doi.org/10.1145/3635059.3635104.

LIKELOVEWANT. Disponível em: <https://github.com/likelovewant/ollama-for-amd>. Acesso em: 17 ago. 2025.

ROTIROTI, Rafael. **Rodando Modelos LLM Localmente com Ollama**. DEV Community, 29 jan. 2024. Disponível em: https://dev.to/rotirotirafa/rodando-modelos-llm-localmente-com-ollama-273d. Acesso em: 9 ago. 2025.

SHARMA, Rakesh Kumar; JOSHI, Manoj. **An Analytical Study and Review of Open Source Chatbot Framework, RASA**. International Journal of Engineering Research & Technology (IJERT), v. 9, n. 6, p. 1009-1012, 2020. Disponível em: https://www.ijert.org/an-analytical-study-and-review-of-open-source-chatbot-framework-rasa.

SHASHIDHAR, Sumuk; CHINTA, Abhinav; SAHAI, Vaibhav; WANG, Zhenhailong; JI, Heng. **Democratizing LLMs: An Exploration of Cost-Performance Trade-offs in Self-Refined Open-Source Models**. In: FINDINGS OF THE 2023 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING (EMNLP 2023), 2023. Proceedings… [S. l.]: Association for Computational Linguistics, 2023. DOI: https://doi.org/10.48550/arXiv.2310.07611

SOARES, Jéferson do Nascimento. **Uma Metodologia para o Desenvolvimento de Chatbots de Domínio Fechado Baseado em Rasa**. 2023. 127 f. Dissertação (Mestrado Acadêmico em Ciência da Computação) – Universidade Estadual do Ceará, Fortaleza, 2023. Disponível em: https://siduece.uece.br/siduece/report?id=110249&tipo=3

VLASOV, Vladimir; NICHOL, Tom; BOCKLISCH, Tobias. **Dialogue Management with Rasa Core**., 2018. Disponível em: https://arxiv.org/abs/1811.00271.