UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ARTUR CARVALHO DOS SANTOS

ANÁLISE COMPARATIVA DE FERRAMENTAS PARA AVALIAÇÃO DE AGENTES AUTÔNOMOS BASEADOS EM LLMS

Recife

2025

ARTUR CARVALHO DOS SANTOS

ANÁLISE COMPARATIVA DE FERRAMENTAS PARA AVALIAÇÃO DE AGENTES AUTÔNOMOS BASEADOS EM LLMS

Trabalho apresentado ao Curso de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Área de Concentração: Inteligência Artificial; Avaliação de Agentes Autônomos baseados em LLMs

Orientador (a): Filipe Carlos de Albuquerque Calegario

Recife

2025

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Santos, Artur Carvalho dos.
Análise Comparativa De Ferramentas Para Avaliação De Agentes Autônomos Baseados Em LLMs / Artur Carvalho dos Santos. - Recife, 2025.
42 p., tab.

Orientador(a): Filipe Carlos de Albuquerque Calegario
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Ciências da Computação - Bacharelado, 2025.
Inclui referências.

1. Inteligência Artificial. 2. Grandes Modelos de Linguagem. 3. LLMs. 4. Agentes Autônomos Baseados em LLMs. 5. Avaliação de Agentes Autônomos. I. Calegario, Filipe Carlos de Albuquerque. (Orientação). II. Título.

000 CDD (22.ed.)

ARTUR CARVALHO DOS SANTOS

ANÁLISE COMPARATIVA DE FERRAMENTAS PARA AVALIAÇÃO DE AGENTES AUTÔNOMOS BASEADOS EM LLMS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de bacharel em Ciência da Computação.

Aprovado em: 09/04/2025

BANCA EXAMINADORA

_____

Prof. Dr. Filipe Carlos de Albuquerque Calegario (Orientador)

Universidade Federal de Pernambuco

_____

Prof. Dr. Giordano Ribeiro Eulalio Cabral (Examinador Interno)

Universidade Federal de Pernambuco

Dedico a Patricia e Saulo, meus pais, que sempre me incentivaram a estudar, me motivaram nas horas difíceis e me deram todas as ferramentas e oportunidades para que eu chegasse até aqui.

AGRADECIMENTOS

Este trabalho e a conclusão do curso de Ciência da Computação são um marco do fim de uma importante fase. Gostaria de agradecer a algumas pessoas que, durante diversos momentos, me fortaleceram e são parte fundamental dessa conquista.

À minha mãe, Patricia e ao meu pai, Saulo, por serem meus maiores professores e protetores. Vocês me ensinaram o que é amor, gentileza, compaixão, empatia e dedicação. Agradeço por estarem sempre presentes e nunca me deixarem cair.

À Luiza, minha namorada, por me acompanhar durante toda a minha jornada universitária e ser meu principal alicerce durante os períodos mais difíceis. Você me inspira constantemente com sua dedicação e esforço. Agradeço por colorir minha vida com seu amor, alegria, companheirismo e carinho.

À Adriana, por ter cuidado de mim e estado presente ao longo de toda a minha vida. Agradeço por todo carinho, suporte e sei que sem você eu não teria condições de estar aqui.

Ao meu irmão, Mateus, por ser também um professor para a vida. Seu amor, presença, autenticidade, entusiasmo e, principalmente, conselhos despretensiosos de um irmão mais velho sempre foram uma luz na caminhada da vida e eu nunca os esqueci.

À Pogba, meu fiel companheiro, por me receber em êxtase independentemente da situação e encher a minha vida de alegria.

À todos os meus amigos, por serem parte tão fundamental na minha vida. Em especial, agradeço aos grandes amigos que fiz no CIn, Lucas, Gustavo, José Lucas, Bruna, Rodrigo, e Matheus, por dividirem comigo os momentos mais felizes e também mais desafiadores dos últimos quatro anos.

À minha psicanalista, Silvia Farias, por me ajudar na longa busca pelo autoconhecimento e por ter me escutado por todos esses anos.

Ao meu orientador e professor Filipe Calegario, por me mostrar que computação é empolgante e divertida. Agradeço por toda orientação e apoio durante este período.

Aos professores e colaboradores do Centro de Informática, por me apresentarem um universo de conhecimentos e despertaram minha curiosidade.

Ao Centro de Informática e à Universidade Federal de Pernambuco, onde me senti acolhido desde o primeiro dia. Agradeço por ser a minha escola para muito além da Ciência da Computação.

RESUMO

Este trabalho apresenta uma análise comparativa de ferramentas de avaliação desenvolvidas para agentes autônomos baseados em grandes modelos de linguagem (LLMs), com foco em sistemas de agente único. O estudo revisou e caracterizou seis frameworks, sendo eles Langsmith, Arize Phoenix, Vertex AI GenAI Evaluation, Agent-as-a-Judge, Agent-Eval-Refine e AgentBench. Essas ferramentas foram avaliadas de acordo com critérios de observabilidade, avaliação da saída e da trajetória, uso de recursos, versatilidade, compatibilidade externa, explicabilidade e usabilidade. Os resultados indicam que, embora plataformas como Langsmith e Arize Phoenix ofereçam ambientes abrangentes que integram observabilidade com diversas metodologias de avaliação, outras soluções fornecem abordagens inovadoras e eficazes, embora específicas para determinados contextos. De modo geral, os achados contribuem para o mapeamento do estado da arte atual em ferramentas de avaliação para agentes baseados em LLMs e sugerem direções promissoras para pesquisas futuras, especialmente no desenvolvimento de frameworks flexíveis e escaláveis que possam se adaptar às demandas rapidamente evolutivas da IA generativa.

Palavras-chaves: IA Generativa. Agentes Autônomos. Grandes Modelos de Linguagem (LLM). Avaliação de Agentes.

ABSTRACT

This work presents a comparative analysis of evaluation tools developed for autonomous agents based on large language models (LLMs), with a focus on single-agent systems. The study reviewed and characterized several frameworks, namely Langsmith, Arize Phoenix, Vertex AI GenAI Evaluation, Agent-as-a-Judge, Agent-Eval-Refine, and AgentBench. These tools were evaluated according to criteria such as observability, evaluation of output and trajectory, resource usage, versatility, external compatibility, explainability, and usability. The results indicate that, although platforms like Langsmith and Arize Phoenix offer comprehensive environments that integrate observability with diverse evaluation methodologies, other solutions provide innovative and effective approaches, albeit specific to certain contexts. Overall, the findings contribute to mapping the current state-of-the-art in evaluation tools for LLM-based agents and suggest promising directions for future research, especially in the development of flexible and scalable frameworks that can adapt to the rapidly evolving demands of generative AI.

Keywords: Generative AI; Autonomous Agents; Large Language Models (LLMs); Agent Evaluation.

LISTA DE TABELAS

Tabela 1 – Comparação de Ferramentas Segundo Critérios Específicos 24
Tabela 2 – Comparação de Ferramentas Segundo Critérios Específicos após conversão de critérios qualitativos 31

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
LLM	Large Language Model
VLM	Visual Language Model

SUMÁRIO

1 INTRODUÇÃO 12
1.1 MOTIVAÇÃO 12
1.2 OBJETIVOS 14
1.2.1 Objetivo Geral 14
1.2.2 Objetivos Específicos 14
2 CONCEITOS BÁSICOS 15
2.1 IA GENERATIVA 15
2.2 GRANDES MODELOS DE LINGUAGEM (LLM) 15
2.3 AGENTES BASEADOS EM LLMS 16
2.4 AVALIAÇÃO DE AGENTES BASEADOS EM LLMS 17
2.4.1 Abordagens de avaliação 17
2.4.2 Técnicas de Avaliação 18
2.4.2.1 Avaliação Objetiva 18
2.4.2.2 Avaliação Subjetiva 19
3 METODOLOGIA 20
3.1 FORMALIZAÇÃO DO ESCORE DE CAMPOS QUALITATIVOS 22
4 RESULTADOS E DISCUSSÃO 24
4.1 ANÁLISE INDIVIDUAL DAS FERRAMENTAS 25
4.1.1 Langsmith 25
4.1.2 Arize Phoenix 26
4.1.3 Vertex AI GenAI Evaluation 26
4.1.4 Agent-as-a-judge 27
4.1.5 Agent-Eval-Refine 28
4.1.6 AgentBench 29
4.2 COMPARAÇÃO GERAL 31
4.2.1 Principais diferenças 32
4.2.2 Níveis de avaliação 34
4.2.3 Técnicas de Avaliação 35
4.2.4 Compatibilidade e usabilidade 35
4.2.5 Uso de Recursos 36

4.3 DISCUSSÃO 37
5 CONCLUSÃO 39
5.1 CONSIDERAÇÕES FINAIS 39
5.2 TRABALHOS FUTUROS 39
REFERÊNCIAS 41

1 INTRODUÇÃO

O cenário mundial é marcado pela expansão da Inteligência Artificial Generativa. Segundo a Bloomberg Intelligence (2024), o mercado de inteligência artificial (IA) está projetado para alcançar US$1,3 trilhão em receita até 2032, partindo de um valor de cerca de US$64 bilhões em 2023.

Esse crescimento exponencial reflete não apenas o aumento do interesse comercial, mas também os avanços significativos na capacidade técnica dos modelos de IA generativa. Tais modelos estão emergindo com uma precisão sem precedentes na criação de conteúdos multimodais, frequentemente superando a performance humana. Eles já são amplamente utilizados por pessoas e empresas para resolver diversos desafios, como documentação de informações, escrita e reescrita de conteúdos, programação, raciocínio matemático, interpretação visual e até mesmo compreensão contextual e emocional (BUBECK et al., 2023).

Nesse contexto, os agentes autônomos baseados em Grandes Modelos de Linguagem (LLMs) representam um avanço significativo nas aplicações de inteligência artificial. Diferentemente de um LLM tradicional, esses agentes possuem habilidades que simulam capacidades humanas, como o planejamento estratégico, a memorização de informações relevantes e o uso de ferramentas para solucionar problemas complexos de maneira mais eficiente (WANG et al., 2024).

1.1 MOTIVAÇÃO

Tão importante quanto o desenvolvimento de novos modelos, sistemas e aplicações na área de IA é a criação de ferramentas de avaliação acessíveis, intuitivas, flexíveis, explicativas e robustas, essenciais para a construção de IAs precisas, éticas e eficazes na resolução de problemas. Mohseni, Zarei e Ragan (2021) argumentam que, à medida que os sistemas de IA assumem tarefas mais sensíveis e com maior impacto social, a necessidade de sistemas interpretáveis e responsáveis cresce ainda mais. Além disso, destacam os principais benefícios da transparência e interpretabilidade:

- Conscientização dos usuários;
- Detecção de vieses e discriminação;

- Comportamento interpretável de sistemas inteligentes;
- Responsabilização perante os usuários

Nesse sentido, existe, hoje, uma numerosa e crescente quantidade de soluções voltadas para avaliação de LLMs, como LLMeBench (DALVI et al., 2024), OpenAI Evals (OpenAI, 2023), LMHarness (EleutherAI, 2020), Langsmith (LangChain, 2025), Parea AI (Parea AI, 2025), Prompt-Flow (MICROSOFT, 2025), DeepEval (Confident AI, 2025), Amazon Bedrock (AMAZON, 2025), entre outros, que dispõem de diversos critérios para medir suas capacidades.

No entanto, apesar do crescente interesse por *frameworks* de avaliação de modelos de linguagem, não foram encontradas muitas soluções voltadas para a avaliação modular de agentes autônomos baseados em LLMs. A avaliação de aplicações de agentes possui particularidades em relação à de sistemas que utilizam apenas LLMs tradicionais. É essencial avaliar o planejamento do agente e a sequência de ações até a resposta. Mesmo que a resposta esteja correta, um caminho excessivamente longo pode indicar ineficiência e aumentar os custos desnecessariamente. Ademais, por se tratar de um sistema que vai além de chamadas de API a LLMs, erros podem ocorrer no planejamento, na reflexão, em alguma das ferramentas (*tools*), ou na estruturação final da resposta. É fundamental entender como cada módulo participante da elaboração das respostas está se comportando. (WANG et al., 2024; LIU et al., 2023)

Adicionalmente, não foram encontrados estudos que agreguem as soluções de avaliação existentes e as comparem com base em critérios bem definidos para a avaliação de agentes. Este aspecto é especialmente relevante, pois compreender o estado atual dessas ferramentas é fundamental para identificar seus pontos fortes e fatores que demandam aprimoramento .

Este trabalho propõe uma análise comparativa das ferramentas de avaliação de agentes autônomos existentes, com foco em sistemas que possuem um único agente, pois também existem sistema multi-agentes, em que estes podem interagir entre si para potencializar suas capacidades. Também foi realizado um mapeamento do estado atual de desenvolvimento desses *frameworks* e o que pode ser desenvolvido na área. Esta análise se faz necessária para servir como base e orientar futuros projetos de avaliadores, fornecendo um ponto de partida sólido e um conjunto de indicadores relevantes.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Analisar e comparar ferramentas para avaliação de aplicações com Agentes Autônomos baseados em LLMs - em sistemas de agente único - e descrever o estado atual de desenvolvimento desses recursos.

1.2.2 Objetivos Específicos

- Identificar e documentar as ferramentas disponíveis para avaliação de agentes autônomos, fornecendo uma descrição técnica de seu funcionamento.
- Avaliar cada *framework* em oito critérios principais para determinar suas vantagens e limitações no uso prático;
- Elaborar uma síntese do estado atual do desenvolvimento dessas ferramentas de avaliação, identificando tendências, lacunas e desafios presentes no seu uso para a análise de agentes autônomos.

2 CONCEITOS BÁSICOS

2.1 IA GENERATIVA

Feuerriegel et al. (2024) definem o campo como o conjunto de técnicas computacionais capazes de gerar conteúdos aparentemente novos e significativos, como imagens, texto ou áudio a partir de dados de treinamento. Na prática, ferramentas de IA generativa se diferenciam de outras áreas da inteligência artificial por não possuírem uma faixa de respostas bem definida. Um modelo de IA generativa é uma arquitetura de aprendizado de máquina que usa algoritmos de IA para criar novas instâncias de dados, com base nos padrões e relacionamentos observados nos dados de treinamento.

2.2 GRANDES MODELOS DE LINGUAGEM (LLM)

LLMs são um tipo de IA generativa voltado para processamento de texto. São modelos de linguagem, o que significa que são treinados com dados textuais provindos de diversas fontes. São grandes, pois foram treinados com uma imensa quantidade de textos. Durante o treinamento, bilhões de parâmetros são ajustados, permitindo que o modelo aprenda padrões linguísticos complexos. Uma LLM pode determinar os termos mais adequados para seguir uma palavra ou frase com base em um cálculo probabilístico (DOUGLAS, 2023). Dessa forma, a LLM se torna capaz de realizar muito bem tarefas de *common reasoning* (raciocínio de senso comum), ou seja, compreender contextos, fazer inferências e gerar respostas coerentes em uma ampla variedade de situações (RADFORD et al., 2019). Ao aprender os padrões e nuances da linguagem, o modelo não apenas gera texto, mas também simula um nível de compreensão que se assemelha ao raciocínio humano, como argumenta Arcas (2022).

Observa-se que ela não escolhe exclusivamente as palavras mais prováveis, pois isso resultaria em uma estrutura excessivamente rígida e não natural. Para isso, há o parâmetro de temperatura, que vai definir uma taxa de aleatoriedade para as respostas. Esse parâmetro está associado à criatividade do modelo ao gerar respostas. (DOUGLAS, 2023)

As LLMs são construídas com uma arquitetura de Rede Neural Profunda denominada *Transformer*, introduzida por Vaswani et al. (2017). Essa arquitetura destaca-se de redes de recorrência e convolução por utilizar apenas mecanismos de atenção para processar as informações e gerar uma saída. Esses mecanismos envolvem a criação de *embeddings*, que são

vetores que armazenam informações semânticas sobre as palavras da entrada para estabelecer uma relação entre elas e facilitar a compreensão do sentido e obter uma resposta mais rápida e coesa. Os mecanismos de atenção ajudam na identificação de relevância e das relações entre os termos da entrada. Esse modelo de arquitetura permite o processamento paralelo das informações (tornando-o mais rápido), e com maior qualidade, pois a distância entre as palavras em uma frase não impede que elas possuam relação semântica (VASWANI et al., 2017).

2.3 AGENTES BASEADOS EM LLMS

Nesses sistemas, a LLM funciona como o cérebro do agente, planejando ações, guardando informações na memória, e utilizando ferramentas (*tools*) para realizar tarefas mais complexas com maior precisão (WENG, 2023; PARK et al., 2023). Um sistema de agentes baseados em LLMs geralmente possui os seguintes componentes:

- **Perfil**: é responsável por atribuir um perfil a um agente. Essas características são geralmente informadas via *prompt*. O perfil pode conter desde informações básicas como nome, gênero, e idade, até informações específicas sobre o comportamento e a forma como ele se relaciona com os demais agentes (WANG et al., 2024).

- **Planejamento**: atribui ao agente a característica de decompor um problema em partes menores e refletir sobre suas decisões, pois é assim que humanos se preparam para resolver uma tarefa complexa que lhes é atribuída (WANG et al., 2024). Existem várias técnicas para capacitar um agente com essa habilidade. *Chain-of-Thought* (YAO et al., 2023b) e *Tree of thoughts* (WEI et al., 2023) são as principais técnicas para decompor um problema em tarefas menores, enquanto *ReAct* (YAO et al., 2023c) e *Reflexion* (SHINN et al., 2023) são técnicas que permitem a correção de erros anteriores e o refinamento das ações futuras.

- **Memória**: O módulo de memória é responsável por guardar informações relevantes para o agente e as atividades a que se propõe. Auxilia no acúmulo de experiências, a evoluir e a se comportar de maneira mais consistente, razoável e eficaz (WANG et al., 2024). Para criar uma memória, geralmente a informação é salva em um armazenador de vetores em formato de *embeddings* e a busca é realizada através de algoritmos como o ANN (*approximate nearest neighbors*) (WENG, 2023).

- ***Tools***: são módulos especializados em ações específicas que podem ser utilizados pelo agente para expandir suas capacidades e resolver tarefas de maneira eficaz. Essas ferramentas permitem a realização de ações como consultar uma *API*, buscar na internet, consultar um banco de dados, gerar imagens e escrever códigos. (WENG, 2023)

2.4 AVALIAÇÃO DE AGENTES BASEADOS EM LLMS

Agentes baseados em LLMs possuem uma complexidade muito maior em sua estrutura em relação às LLMs tradicionais. A existência de vários módulos serve para elevar as capacidades da LLM ao máximo (WANG et al., 2024). Exatamente por esse fato é que avaliar um agente é, da mesma forma, uma tarefa consideravelmente mais complexa do que avaliar uma LLM. Zhuge et al. (2024) argumentam que a maior parte das técnicas de avaliação contemporâneas foca apenas na resposta final ou exige muito trabalho manual para ser realizada, tornando-as inadequadas para a avaliação de agentes. Um agente pode ser avaliado por meio de diferentes abordagens e técnicas de análise variadas.

2.4.1 Abordagens de avaliação

Observabilidade e AgentOps: Dong, Lu e Zhu (2024) defende que a inclusão de observabilidade e registro de *traces* é essencial para garantir a segurança da IA em agentes, permitindo que seja possível compreender o funcionamento interno dos agentes, detectar anomalias e prevenir potenciais falhas. Embora não se trate de uma forma de avaliação, este passo é essencial para a implementação de uma avaliação futura.

Avaliar a saída do agente: Essa é a maneira mais simples de se avaliar um agente, e se assemelha à maneira como são realizadas avaliações de LLMs e demais aplicações de software (XIA et al., 2024). Nessa categoria, a saída do agente é comparada com a tarefa passada na entrada - opcionalmente pode ser comparada com uma referência - para que então seja realizado algum cálculo ou interpretação acerca do resultado.

Apesar de fornecer importantes percepções sobre a performance do agente, esse método falha em detalhar o que ocorre internamente no agente. Xia et al. (2024) explicam que o escopo dos agentes é mais amplo, pois a avaliação da resposta de um agente pode indicar a necessidade de otimizar o prompt, aprimorar o planejamento e os fluxos, e/ou melhorar a tomada de decisão e execução, entre outros aspectos.

Avaliar o fluxo planejado e a execução: Esta categoria irá permitir uma avaliação mais profunda dos agentes, pois todas as etapas de construção de solução por parte do agente serão avaliadas. Agentes são sistemas com vários componentes além da LLM em si, e a sua avaliação deve ser a nível de sistema, englobando todos os componentes (XIA et al., 2024).

1. Avalia-se se o planejamento da solução foi bem feito em relação à tarefa passada. Nessa etapa, é importante verificar se o agente está criando um fluxo de execução eficiente, ou se realiza chamadas desnecessárias à ferramentas.

2. Em seguida, deve-se analisar a execução desse fluxo, observando se os resultados de cada *tool* chamada durante o processo estão corretos. Afinal, uma resposta errada no meio do caminho pode levar a uma saída inusitada ao fim. Adicionalmente, é importante verificar se o agente está utilizando a sintaxe correta para a chamada dessas ferramentas.

3. Observar se o agente está estruturando a resposta corretamente ao final.

2.4.2 Técnicas de Avaliação

Existem inúmeras abordagens bem estabelecidas para a avaliação de sistemas tradicionais. Contudo, aplicações com agentes baseados em LLMs possuem uma estrutura significativamente diferente, podendo exibir comportamentos menos consistentes. Sua capacidade de realizar planejamento autônomo e acionar ferramentas externas pode gerar resultados que se desviam do objetivo principal (XIA et al., 2024).

Portanto, estas são algumas técnicas que podem ser utilizadas para avaliar agentes baseados em LLMs.

2.4.2.1 Avaliação Objetiva

Envolve o teste do agente em relação à verdades absolutas estabelecidas, métricas bem definidas, que não dependem de interpretação.

1. **Métricas**: Estão inclusas aqui avaliações relacionadas ao sucesso na realização de determinadas tarefas. Para verificar a validade da operação, utiliza-se meios práticos, como a executabilidade de um programa (em caso de geração de código), ou definindo uma estrutura de saída que permita a verificação (WANG et al., 2024).

2. **Protocolos**: Os protocolos de avaliação incluem: **Capacidade de multi-tarefa**, onde o agente é avaliado com tarefas de propósitos diferentes; **Teste de software**, em que se testa a capacidade de realizar tarefas relacionadas a software (WANG et al., 2024).

3. **Benchmarks**: são conjuntos de dados ou *frameworks* construídos para testar alguma habilidade específica, como compras *online* (WebShop (YAO et al., 2023a)) ou capacidade de uso de *tools* e APIs (ToolBench (QIN et al., 2024)) (WANG et al., 2024).

2.4.2.2 Avaliação Subjetiva

Envolve a avaliação interpretativa de um resultado. É útil para casos em que não há *benchmarks* ou não é possível avaliar o critério desejado em métricas quantitativas (WANG et al., 2024). Além disso, também pode ser vantajoso para quando se deseja uma avaliação mais explicada.

1. **Human-as-a-judge**: Esse método consiste em avaliações realizadas por pessoas, geralmente desenvolvedores ou especialistas. Essas avaliações podem ser feitas analisando as saídas do agente ou revisando seu planejamento e fluxo de trabalho através de plataformas de observabilidade. Para diminuição do erro, uma solução é propor um debate entre os avaliadores posteriormente. Embora essa abordagem proporcione avaliações de alta qualidade, ela possui limitações significativas, como altos custos e baixa eficiência (ZHUGE et al., 2024).

2. **LLM-as-a-judge**: De acordo com Gu et al. (2025), esse método utiliza Grandes Modelos de Linguagem (LLMs) como avaliadores em diversas tarefas, verificando se um resultado está alinhado ao escopo definido de uma tarefa. Por sua capacidade de imitar o raciocínio humano, as LLMs conseguem realizar com sucesso grande parte das avaliações anteriormente feitas por humanos, oferecendo uma alternativa mais econômica e escalável.

3. **Agent-as-a-judge**: Conforme apresentado por Zhuge et al. (2024), essa abordagem representa uma evolução do *LLM-as-a-judge*. Por possuírem habilidades de planejamento, raciocínio e execução, agentes atuando como avaliadores podem fornecer julgamentos mais próximos aos de especialistas humanos, mantendo, no entanto, a escalabilidade como uma vantagem significativa.

3 METODOLOGIA

Foi realizada uma revisão exploratória das principais soluções voltadas para avaliação de agentes autônomos baseados em LLMs. As fontes de pesquisa incluíram buscas em repositórios *online*, blogs, artigos e outras publicações relevantes. Posteriormente, cada *framework* foi descrito em detalhes da sua estrutura e funcionamento, submetido a uma avaliação elaborada no *Google Forms* e testado em funcionamento. A avaliação foi embasada em oito critérios:

1. Observabilidade: Presença de registro de traces e ferramentas que permitam o monitoramento da aplicação;

2. Avaliação da saída do agente: Capacidade de avaliar o resultado produzido pelo agente de diferentes maneiras.

    - Avaliação com métricas;
    - Avaliação com protocolos;
    - Avaliação com *benchmarks*;
    - LLM-as-a-judge ou Agent-as-a-judge.

3. Avaliação do fluxo de planejamento e execução de cada etapa: Capacidade de avaliar a capacidade de planejamento e cada etapa executada pelo agente de diferentes formas.

    - Avaliação com métricas;
    - Avaliação com protocolos;
    - Avaliação com *benchmarks*;
    - LLM-as-a-judge ou Agent-as-a-judge.

4. Uso de Recursos: Recursos computacionais necessários para realizar a avaliação de um sistema. Foi estabelecida uma pontuação de 1 a 5, onde 5 indica baixo uso de recursos e 1 indica alto uso de recursos;

5. Versatilidade: Capacidade de testar os sistemas em diferentes ambientes, tarefas e critérios. Foi estabelecida uma pontuação de 1 a 5, onde 1 indica um agente pouco versátil e 5 indica um agente bastante versátil;

6. Integração/compatibilidade externa: Capacidade de incorporação a ambientes ou pipelines de outras aplicações. Foi estabelecida uma pontuação de 1 a 5, onde 1 indica pouca capacidade de integração e 5 indica bastante capacidade de integração.

7. Explicabilidade: Capacidade da ferramenta de explicar os resultados de cada avaliação de forma simples e compreensível.

8. Facilidade de uso: Mede o quão intuitiva e acessível é a ferramenta, considerando fatores como documentação, processo de configuração e curva de aprendizado. A avaliação será baseada em três aspectos: (i) tempo necessário para realizar um teste, (ii) número de etapas exigidas para concluir o processo de teste, e (iii) qualidade da documentação, classificada em uma escala de 1 a 5.

Foram selecionados os seguintes frameworks para a análise comparativa, considerando diferentes abordagens de avaliação de agentes autônomos baseados em LLMs:

- Agent-Eval-Refine (PAN et al., 2024)
- Vertex AI GenAI Evaluation (Google Cloud, 2025)
- Agent-as-a-judge (ZHUGE et al., 2024)
- Arize Phoenix (Arize AI, 2025)
- Langsmith (LangChain, 2025)
- AgentBench (LIU et al., 2023)

A escolha das ferramentas se baseou na necessidade de apresentar as diferentes abordagens em torno da avaliação de aplicações de agentes autônomos baseados em LLMs. Foram selecionadas soluções recentes que oferecem desde observabilidade completa (como LangSmith e Phoenix), abordagens inovadoras de avaliação (como Vertex AI GenAI Evaluation, Agent-as-a-Judge, Agent-Eval-Refine) até benchmarking (como AgentBench). Essa combinação permite abarcar vários modelos de avaliação e entender as vantagens e desvantagens da aplicação de cada caso.

Os resultados obtidos foram discutidos e comparados quantitativa e qualitativamente, destacando os pontos fortes e as limitações de cada solução. A análise culminou em uma visão geral consolidada, apresentada em formato de tabela e discussões textuais, para sintetizar o que já foi alcançado e as lacunas a serem preenchidas.

3.1 FORMALIZAÇÃO DO ESCORE DE CAMPOS QUALITATIVOS

Para possibilitar a representação dos resultados qualitativos por meio de uma curva de valor, foi desenvolvido um sistema de pontuação que visa traduzir de forma objetiva e consistente as características avaliadas em cada critério. Nos campos de Observabilidade e Explicabilidade, por se tratarem de variáveis binárias com respostas "Sim" ou "Não", adotou-se uma pontuação fixa: atribuiu-se o valor 1 para respostas "Não" e 5 para respostas "Sim".

Por sua vez, os critérios relacionados à avaliação da saída, das trajetórias e das etapas exigiram uma abordagem mais elaborada, considerando a diversidade e complexidade das técnicas utilizadas. Para esses campos, foram definidas variáveis específicas e atribuídas pontuações diferenciadas conforme o tipo e o grau de sofisticação das abordagens adotadas:

1. Variedade (0 a 2 pontos)
2. Qualidade (0 a 2 pontos)
3. Inovação (0 a 1 ponto)

Cálculo da Nota Final da Dimensão

A nota final para cada dimensão será dada por:

$$S_{\text{dim}} = V + Q + I, \quad 0 \leq S_{\text{dim}} \leq 5,$$

onde:

- V é a pontuação da Variedade (0 a 2),
- Q é a pontuação da Qualidade (0 a 2),
- I é a pontuação da Inovação (0 a 1).

Cálculo do Escore Final

Após a atribuição de notas individuais para cada critério, foi definido um **escore final** para cada ferramenta com o objetivo de facilitar a comparação geral entre as soluções analisadas.

Esse escore é calculado por meio de uma **média simples** entre todos os critérios considerados. Cada ferramenta recebe uma nota de 1 a 5 em nove dimensões: Avaliação da saída,

Avaliação da trajetória, Avaliação de etapa, Observabilidade, Uso de Recursos, Versatilidade, Compatibilidade Externa, Explicabilidade e Usabilidade.

Assim, a fórmula utilizada é:

$$S_{\text{final}} = \frac{1}{n} \sum_{i=1}^{n} S_i,$$

onde:

- S_{final} representa o escore final da ferramenta;
- S_i representa a nota da ferramenta no critério i;
- $n = 9$ corresponde ao número total de critérios avaliados.

4 RESULTADOS E DISCUSSÃO

Nesta seção, são apresentados os resultados e uma discussão derivada da análise comparativa das ferramentas de avaliação de agentes autônomos baseados em LLMs. Inicialmente, os dados coletados foram organizados em formato de tabela, para permitir uma análise detalhada em função dos critérios previamente estabelecidos.

Critérios/ Ferramentas	**Langsmith**	**Phoenix**	**VertexAI**	**Agent-as-a-judge**	**Agent-Eval-Refine**	**AgentBench**
Observabilidade	Sim	Sim	Não	Não	Não	Não
Avaliação de saída	Métricas; Benchmarks; LLM e Human-as-a-judge	Métricas; Benchmarks; LLM e Human-as-a-judge	Métricas; LLM-as-a-judge	Agent-as-a-judge	Não realiza	Métricas; Protocolos
Avaliação de trajetórias	Benchmarks; LLM-as-a-judge	Benchmarks; LLM-as-a-judge	Métricas; LLM-as-a-judge	Agent-as-a-judge	LLM-as-a-judge	Não realiza
Avaliação de etapa	Benchmarks; LLM-as-a-judge	Benchmarks; LLM-as-a-judge	Não realiza	Agent-as-a-judge	Não realiza	Não realiza
Uso de Recursos	5	5	5	4	3	1
Versatilidade	4	5	3	2	2	3
Compatibilidade externa	4	5	3	2	2	1
Explicabilidade	Sim	Sim	Não	Sim	Sim	Não
Usabilidade	5	5	3	2	2	4

Tabela 1 – Comparação de Ferramentas Segundo Critérios Específicos

Em uma primeira análise dos dados, é notório que há soluções que são mais completas com relação aos critérios pré-estabelecidos. Em todo caso, é importante ressaltar que foram selecionadas ferramentas com diferentes propósitos para este trabalho, com o intuito de comparar diferentes abordagens e discutir vantagens e desvantagens de cada uma. Assim, antes de realizar a discussão dos resultados da avaliação das ferramentas, faz-se necessária a apresentação de cada solução, destacando seu propósito, funcionamento e principais características.

4.1 ANÁLISE INDIVIDUAL DAS FERRAMENTAS

4.1.1 Langsmith

Langsmith é uma plataforma de observabilidade, avaliação e otimização criada pela Langchain para ser um ambiente de controle geral de aplicações baseadas em LLMs ou agentes. A ferramenta é agnóstica a frameworks, funcionando tanto com soluções da própria Langchain, como Langchain e LangGraph, quanto com outras abordagens independentes. Para projetos menores, a plataforma é hospedada pela própria Langchain, o que reduz drasticamente o uso de recursos, ao mesmo tempo em que garante uma infraestrutura robusta e confiável (LangChain, 2025).

No que se refere à observabilidade, o Langsmith oferece registro nativo de logs e *traces*, permitindo um acompanhamento detalhado das interações dos agentes — incluindo entradas, saídas, consumo de tokens, tempos de resposta, entre outros. Além disso, para agentes desenvolvidos com Langchain ou LangGraph, o rastreamento das etapas ocorre de forma nativa, exigindo apenas a configuração de algumas variáveis de ambiente, sem necessidade de adição de código extra. Já para agentes desenvolvidos com outras implementações, é necessário incluir alguns trechos de código para comunicação com a plataforma. Além do monitoramento, o LangSmith incorpora fluxos de avaliação que podem usar bases de dados de teste e avaliadores customizados ou embutidos para medir a qualidade das saídas do agente. Por exemplo, é possível definir métricas de acurácia ou coerência, utilizar um *LLM-as-a-judge* e integrar avaliações humanas (LangChain, 2025).

A principal vantagem do Langsmith é sua abrangência, pois combina observabilidade, múltiplas metodologias de avaliação e mecanismos de otimização do modelo em uma única plataforma. Adicionalmente, embora apresente uma curva de aprendizado para compreender todas as suas funcionalidades, a ferramenta conta com uma documentação extensa, suporte ativo dos desenvolvedores e uma comunidade engajada, o que facilita sua adoção. Como possível desvantagem, por ser uma ferramenta voltada a desenvolvedores, exige configuração prévia de definição de critérios de sucesso para avaliação – não sendo uma solução de *benchmark* pronta, mas sim um arcabouço flexível que o usuário adapta às necessidades da sua aplicação (LangChain, 2025).

4.1.2 Arize Phoenix

Phoenix, da Arize AI, é uma ferramenta **open-source** abrangente de observabilidade e avaliação de LLMs. Assim como o LangSmith, a ferramenta fornece visibilidade detalhada sobre o comportamento de agentes em execução, registrando rastros completos de cada etapa do agente. A ferramenta é agnóstica em relação a frameworks, oferecendo instrumentadores para bibliotecas populares – como LangChain, smolagents, OpenAI, LlamaIndex e Haystack – o que possibilita instrumentar toda a execução de um agente com um único comando e facilita a integração em sistemas já existentes. Além disso, o Phoenix não se restringe a um único fornecedor ou linguagem, podendo receber dados de qualquer fonte que siga o protocolo OpenTelemetry[1] (Arize AI, 2025).

Quanto à avaliação, o Phoenix permite que o usuário crie e implemente fluxos de avaliação conforme a necessidade de sua aplicação. A ferramenta tem um foco claro em avaliações utilizando LLM-as-a-judge, apesar de permitir avaliações utilizando métricas definidas em código. Além disso, também há funções de avaliação disponibilizadas pela própria ferramenta para identificar problemas comuns, como alucinação, toxicidade e acurácia de resposta. Ademais, a Arize também disponibiliza a biblioteca LLM Evals para avaliação de saídas, trazendo templates pré-testados e calibrados com diferentes modelos. Para a avaliação do planejamento do agente e execução das etapas, o Phoenix também permite o emprego de várias abordagens, além de oferecer funções pré-definidas, como a *Agent Function Calling Evaluation*, que permite avaliar a habilidade de escolha de funções e ferramentas do agente (Arize AI, 2025).

O Arize Phoenix oferece uma solução versátil, integrável e bem documentada, destacando-se pelas suas vastas capacidades de avaliação. Entretanto, uma limitação possível é a ausência, na configuração inicial, de uma opção de hospedagem própria, deixando essa responsabilidade ao usuário.

4.1.3 Vertex AI GenAI Evaluation

O Vertex AI Gen AI Evaluation é a solução de avaliação de aplicações de LLMs ou agentes autônomos baseados em LLMs dentro da plataforma Google Cloud (Vertex AI). Diferentemente das ferramentas anteriores, o foco aqui está em **avaliar modelos ou agentes via**

[1] **OpenTelemetry (OTel)** é um conjunto de ferramentas e padrões que auxiliam na coleta, processamento e envio de dados sobre o desempenho de sistemas e aplicações.

métricas automatizadas e serviços gerenciados, integrando-se ao ecossistema do Google. Por padrão, o serviço disponibiliza métricas acadêmicas clássicas como BLEU, ROUGE (comumente usadas para avaliar qualidade de texto por similaridade com referência) e também possui a opção de utilizar um LLM-as-a-Judge tanto por pontos (avaliação de uma única saída) quanto por pares (comparação de duas saídas). No LLM-as-a-judge, há métricas qualitativas pré-definidas, como fluência, coerência e segurança. Além disso, o usuário pode definir métricas qualitativas ou quantitativas customizadas de acordo com as necessidades da aplicação. Para avaliação da trajetória planejada pelo agente, fornece métricas quantitativas, com a avaliação de trajetória por comparação com uma trajetória esperada, permitindo o cálculo de métricas como precisão, *recall*, se uma *tool* específica foi usada, ou mesmo se uma ordem foi seguida (Google Cloud, 2025).

Entre os principais pontos positivos do Vertex AI GenAI Evaluation, destaca-se a ampla variedade de funções de avaliação pré-definidas, que agilizam o desenvolvimento e permitem iniciar as avaliações objetivas das aplicações de forma rápida. Além disso, o fato de ser um serviço em nuvem integrado ao ecossistema Google facilita sua adoção e contribui para um baixo consumo de recursos (Google Cloud, 2025). Por outro lado, embora a ferramenta ofereça visualizações em forma de tabelas, gráficos de barras e radar, ela ainda apresenta limitações na sua capacidade explicativa. Ademais, a documentação encontra-se bastante dispersa e, mesmo contando com funções para avaliação de trajetórias, a abordagem para análise de agentes ainda deixa a desejar. É essencial destacar, entretanto, que a avaliação de agentes com o Vertex AI GenAI Evaluation ainda está em fase prévia, o que justifica suas limitações (Google Cloud, 2025).

4.1.4 Agent-as-a-judge

O Agent-as-a-Judge é uma solução, criada por Zhuge et al. (2024), para avaliação de agentes autônomos baseados em LLMs, que introduz uma inovação em cima do conceito de LLM-as-a-judge. Enquanto LLM-as-a-judge refere-se a usar um modelo de linguagem para pontuar ou julgar a resposta final de outro modelo, o Agent-as-a-Judge propõe que um sistema agêntico avalie outro sistema agêntico em todas as etapas. Ou seja, coloca-se um agente (com habilidades de raciocínio, acesso a ferramentas, etc.) para observar a execução de outro agente e fornecer feedback em cada estágio da tarefa, não apenas no resultado final. Essa abordagem oferece feedback intermediário durante todo o processo de resolução de tarefas pelo agente

avaliado (ZHUGE et al., 2024).

Na prática, foi demonstrado que a avaliação conduzida por agentes pode ser muito eficaz: ao avaliar agentes de geração de código, por exemplo, o Agent-as-a-Judge superou a abordagem tradicional de avaliação apenas com LLM (sem agente) e atingiu um nível de confiabilidade comparável ao de uma avaliação humana. Isso indica que um agente avaliador, por compreender contexto e passos intermediários, consegue identificar erros ou sucessos parciais que passariam despercebidos em uma checagem somente do resultado final (ZHUGE et al., 2024).

Assim, como vantagem o Agent-as-a-Judge captura a natureza processual dos agentes autônomos, alinhando-se melhor à forma como eles operam (passo a passo). Ele permite detectar onde, em uma sequência de ações, o agente tomou uma decisão errada ou se desviou do objetivo, viabilizando correções mais direcionadas. Para obter o mesmo resultado utilizando LLM-as-a-judge, é preciso aplicar vários métodos de avaliação ao longo do processo, e ainda assim a avaliação não terá todo o contexto disponível. Utilizar um agente na avaliação pode ser mais lento do que utilizar um LLM como juiz, mas, observando os resultados do artigo de Zhuge et al. (2024), é importante notar que a verdadeira concorrência do Agent-as-a-judge é com avaliadores humanos. Nessa perspectiva, o Agent-as-a-judge apresenta uma solução de alta confiabilidade e muito mais rápida do que a avaliação humana (ZHUGE et al., 2024).

Entre os pontos negativos, cabe mencionar que essa solução ainda é fruto de pesquisa (com código aberto disponível, mas não um produto pronto), exigindo que o desenvolvedor modifique seu agente para publicar seus outputs e logs em um diretório em um formato específico, o que evidencia a baixa versatilidade da ferramenta. Além disso, a ferramenta não possui uma boa usabilidade. Apesar desses desafios, o Agent-as-a-Judge destaca-se como um paradigma promissor para avaliação de agentes, especialmente em tarefas de múltiplas etapas, apontando uma direção de avaliação mais alinhada à dinâmica interna desses sistemas do que as métricas tradicionais (ZHUGE et al., 2024).

4.1.5 Agent-Eval-Refine

Agent-Eval-Refine não é exatamente uma ferramenta com interface pronta, mas sim um framework proposto por Pan et al. (2024) para avaliação e refinamento autônomo de agentes. A ideia central é utilizar modelos avaliadores para tanto avaliar quanto refinar automaticamente o desempenho de agentes digitais em tarefas complexas. Em outras palavras, o próprio sistema

executa ciclos de análise e aperfeiçoamento: primeiro o agente atua em um ambiente (por exemplo, navegando na web ou controlando um dispositivo móvel), então um LLM (Evaluator) julga seu desempenho, e por fim um processo de refinamento ajusta o agente com base nesse feedback.

A ferramenta implementa duas abordagens para a avaliação:

1. **Abordagem ponta-a-ponta (end-to-end):** Um modelo de visão de linguagem (VLM) como o GPT-4V é usado para avaliar diretamente a trajetória, combinando instruções, ações e estados do ambiente. A trajetória é capturada por meio de capturas de tela.
2. **Abordagem modular:** Primeiro, um VLM transcreve as capturas de tela em descrições textuais. Em seguida, um LLM usa essas descrições, juntamente com as ações e instruções do usuário, para avaliar a trajetória.

Uma vantagem do Agent-Eval-Refine é que ele introduz uma forma de treinamento contínuo do agente em produção, fechando o laço entre avaliação e ação: problemas detectados pelo avaliador podem ser corrigidos imediatamente em uma próxima iteração do agente. Isso é particularmente útil em cenários onde é inviável obter avaliações humanas constantes ou quando o ambiente muda rapidamente exigindo adaptação. Além disso, ao automatizar a avaliação, essa técnica pode explorar critérios complexos de desempenho definidos pelos pesquisadores (por exemplo, sucesso em sub-tarefas, eficiência de navegação, etc.) de maneira consistente.

Por outro lado, há desvantagens claras: a implementação é complexa e computacionalmente custosa – basicamente rodando múltiplas execuções do agente (para avaliar e refinar) para cada tarefa. Também é necessário dispor de um modelo avaliador bem estruturado e instruído; caso contrário, o sistema pode se auto-ajustar de forma incorreta. Além disso, apesar de a proposta do Agent-Eval-Refine ser a apenas a avaliação de trajetórias de forma automatizada, o fato de ele não oferecer uma cobertura de todo o agente na avaliação indica que ele precisaria ser aplicado com outra solução. Por fim, esse método ainda é recente e nichado, o que significa que carece da facilidade de uso e é pouco versátil, sendo aplicado apenas no WebArena, e em ambientes *mobile* (Android e IOS). (PAN et al., 2024).

4.1.6 AgentBench

O AgentBench distingue-se das demais ferramentas por ser um benchmark abrangente para avaliar, de forma padronizada, a capacidade de LLMs agirem como agentes em diversos

cenários. Esta solução possui um propósito diferente das demais do trabalho, pois não é utilizada para avaliar agentes em si, mas sim o desempenho de LLMs com agentes autônomos. Trata-se do primeiro esforço sistemático focado em medir o desempenho de LLMs como agentes em um espectro amplo de ambientes.

Atualmente, o AgentBench consiste em oito ambientes distintos de teste, cobrindo tarefas interativas em domínios variados. São eles:

1. Sistema operacional
2. Banco de dados
3. Knowledge Graph
4. Jogo de cartas digital
5. Quebra-cabeças de pensamento lateral
6. Tarefas domésticas
7. Compras *online*
8. Navegação na internet

Nessas arenas, avalia-se as habilidades de raciocínio, tomada de decisão e seguimento de instruções dos modelos ao longo de interações de múltiplos turnos. Em outras palavras, o AgentBench coloca os agentes em situações complexas e mensura se eles conseguem planejar ações, executar comandos adequadamente e atingir objetivos de longo prazo. A avaliação é feita via métricas de sucesso em cada ambiente (por exemplo, percentual de objetivos concluídos, pontos marcados, acertos em perguntas, etc.), permitindo comparar uma variedade de modelos sob os mesmos desafios.

Em termos de vantagens, o AgentBench fornece à comunidade uma linha de base unificada para avaliar progressos. Com a ferramenta, os desenvolvedores são apoiados na fase anterior à implementação do agente, que é a escolha do modelo. Essa fase é crucial e o AgentBench permite a visualização de desempenho de uma LLM como agente em muitos casos de uso.

Quanto a pontos de atenção, é importante notar que executar os testes do AgentBench pode ser custoso – os ambientes simulados e a necessidade de diversas rodadas de interação (um único teste pode requerer milhares de turnos de geração no total) implicam alto uso de recursos computacionais, conforme refletido na Tabela 1. Portanto, não é trivial incorporá-lo

continuamente no desenvolvimento de um produto, servindo melhor como avaliação pontual ou em estudos de novas arquiteturas.

Assim, o AgentBench pode atuar como um termômetro do estado-da-arte para LLMs como agentes, oferecendo uma visão clara de quão longe (ou quão perto) estamos de ter agentes plenamente eficazes em problemas complexos do mundo real.

No cálculo do escore dos campos qualitativos, o AgentBench atingiu **4** na **avaliação de saída**, pois abrange vários cenários de teste (1 ponto em variedade) e possui boa qualidade de *benchmark* (2 pontos), e foi um dos primeiros benchmarks para agentes (1 ponto). Já a **avaliação de trajetória** e a **avaliação de etapa** ficaram em **1**, pois a ferramenta não se propõe a avaliar o processo intermediário, focando apenas no resultado final em cada ambiente.

4.2 COMPARAÇÃO GERAL

Após a análise individual de cada ferramenta, foi realizada uma conversão dos campos qualitativos para valores quantitativos, que culminou na criação da Tabela 2. Esta visualização permite que os resultados das pesquisas e testes das ferramentas sejam apresentados em gráficos que permitam uma visualização mais ampla, possibilitando uma análise mais aprofundada e embasada.

Critérios/ Ferramentas	**Langsmith**	**Phoenix**	**VertexAI**	**Agent-as-a-judge**	**Agent-Eval-Refine**	**AgentBench**
Observabilidade	5	5	1	1	1	1
Avaliação de saída	4	5	4	3	1	4
Avaliação de trajetórias	4	4	3	3	3	1
Avaliação de etapa	4	4	1	3	1	1
Recursos	5	5	5	4	3	1
Versatilidade	4	5	3	2	2	3
Compatibilidade externa	4	5	3	2	2	1
Explicabilidade	5	5	1	5	5	1
Usabilidade	5	5	3	2	2	4

Tabela 2 – Comparação de Ferramentas Segundo Critérios Específicos após conversão de critérios qualitativos

A Tabela resume as notas atribuídas a cada ferramenta com base nos critérios definidos

na Seção 3.1. Para os campos de **avaliação de saída, trajetórias, e etapas**, os resultados foram os seguintes: O Langsmith recebeu pontuações altas em todos os tipos de avaliação (saída: 4,5; trajetória e etapa: 4), demonstrando boa robustez com leve perda em variedade. O Phoenix obteve destaque na avaliação de saída (nota 5), graças à diversidade de métodos bem integrados, e nota 4 para trajetória e etapa. O VertexAI teve notas 4 em saída e trajetória, por sua boa oferta de métricas quantitativas, mas com pouca inovação; a avaliação de etapa é inexistente. O Agent-as-a-judge recebeu 3,5 em todos os aspectos, refletindo inovação e qualidade, mas sem variedade. O Agent-Eval-Refine possui apenas na avaliação de trajetória (3,5). Por fim, o AgentBench obteve nota 4 apenas em avaliação de saída, por sua variedade e pioneirismo na criação de benchmarks para agentes.

4.2.1 Principais diferenças

Analisando as ferramentas em conjunto, a partir da tabela 2, fica evidente a existência de distinções em relação aos enfoques de avaliação e capacidades oferecidas. Langsmith e Arize Phoenix possuem foco em serem plataformas para depuração e melhoria iterativa em ambiente de produção, permitindo identificar exatamente onde e por que um agente falhou ou tomou determinada decisão. Essas duas ferramentas são as únicas que cumprem o critério de Observabilidade, que foi marcado como critério "zero", pois não está diretamente ligado à avaliação, mas é um facilitador e potencializador. Em contrapartida, métodos como Agent-as-a-Judge e Agent-Eval-Refine, apesar de não proverem um “painel” de observabilidade, trazem conceitos inovadores de avaliação intrínseca ao processo (por exemplo, avaliação a cada etapa ou ciclos de refinamento). Além disso, o Vertex AI GenAI Evaluation se posiciona como uma solução que agrega funções pré-definidas e permite a rápida avaliação de agentes, enquanto o AgentBench se localiza em uma etapa precedente à construção do modelo.

O Gráfico de Linhas na Figura 1 apresenta visualmente como Langsmith e Phoenix se sobressaem na maioria dos critérios analisados. As demais ferramentas tendem a se concentrar nas partes inferiores do gráfico, o que reforça sua proposta mais experimental ou de escopo reduzido. A curva evidenciada reflete as especializações de cada ferramenta, indicando que algumas priorizam profundidade em poucos aspectos, enquanto outras visam abrangência.

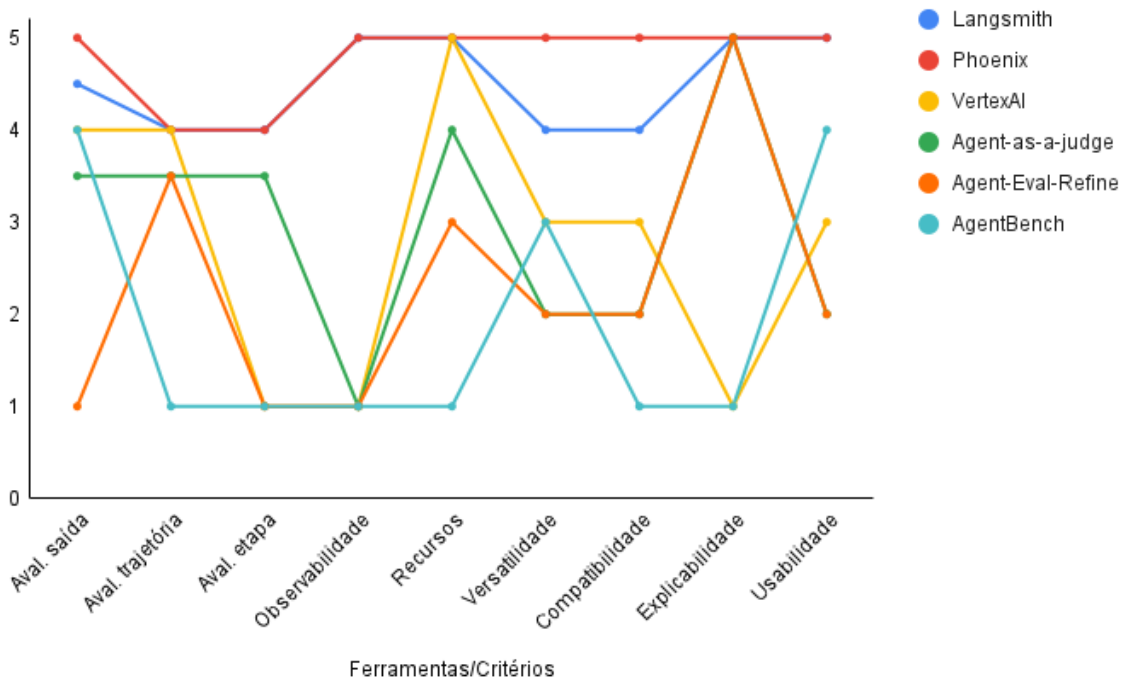

Figura 1 – Desempenho comparativo das ferramentas de avaliação de agentes autônomos baseados em LLMs, segundo nove critérios definidos.

Além disso, após a apresentação do desempenho de cada um na Figura 1, os resultados foram consolidados e um escore foi calculado por média simples, facilitando a visualização geral do desempenho de cada ferramenta na Figura 2. Este escore não está relacionado diretamente à qualidade das ferramentas, mas sim à sua abrangência em relação aos critérios de avaliação estabelecidos. Assim, ferramentas que possuem maior nota são aquelas cuja proposta é avaliar todo o fluxo dos agentes.

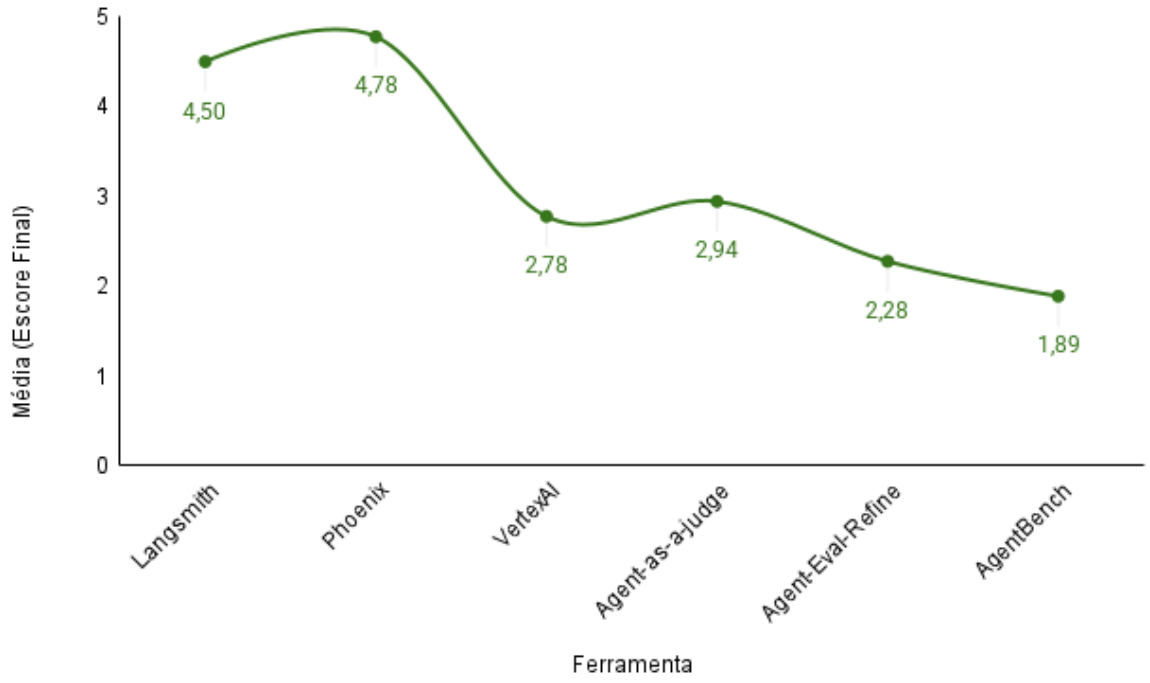

Figura 2 – Comparação do escore final das ferramentas calculado por média simples da soma das pontuações.

4.2.2 Níveis de avaliação

Dentre os critérios de avaliação listados, os níveis de avaliação mostraram as maiores diferenças. O Vertex AI fornece métricas e julgamentos sobre a performance final, mas também oferece algumas métricas pré-estabelecidas para avaliação de trajetória. Entretanto, a documentação não menciona a avaliação de etapas intermediárias, o que gerou uma queda na avaliação final dessa ferramenta, como mostra a figura 2. Já o Agent-as-a-Judge foca explicitamente na avaliação granular por etapas, fornecendo feedbacks em cada sub-tarefa dentro da execução do agente. Isso o alinha mais com frameworks de observabilidade no sentido de acompanhamento do processo etapa a etapa, porém com a diferença de emitir julgamentos automatizados sobre cada passo. LangSmith e Phoenix fazem uma ponte entre essas abordagens – eles registram todo o processo e permitem aplicar avaliadores tanto no final quanto em pontos intermediários (por exemplo, é possível configurar avaliações de cada ação ou escolha do agente dentro dessas plataformas). De fato, LangSmith e Phoenix suportam a incorporação de avaliações em mais níveis, mas se diferenciam, pois o Arize Phoenix possui algumas implementações de maneira nativa, enquanto o Langsmith necessita de implementação, o que

colocou o Arize Phoenix na frente, como indicam as figuras 1 e 2. Quanto ao AgentBench, este avalia apenas a resposta final do agente, com o intuito de avaliar a precisão da LLM.

4.2.3 Técnicas de Avaliação

As **técnicas de avaliação** empregadas também variam. Tanto Langsmith, quanto Arize Phoenix e Vertex AI dão suporte a avaliações com métricas e uso de LLM-as-a-judge. Entretanto, há diferenças na abordagem: enquanto o Vertex AI apoia-se em métricas quantitativas (cálculo de acurácia, BLEU/ROUGE, taxas de sucesso, etc.), uso de LLM-as-a-judge para métricas como coerência e fluência e comparações com gabaritos, ferramentas como LangSmith e Phoenix adicionam avaliações qualitativas automatizadas (via LLMs avaliadores) às métricas tradicionais. Elas dão mais liberdade e suporte ao desenvolvedor quanto ao uso de LLM-as-a-judge e criação de métodos de avaliação customizados. Além disso, também permitem a avaliação humana nas suas plataformas. Já o Agent-as-a-Judge traz um novo conceito para vencer a pouca eficiência da avaliação humana sem grandes perdas de qualidade. Ele utiliza agentes completos como avaliadores (fornecendo avaliações contextuais ricas). O Agent-Eval-Refine realiza apenas a avaliação de trajetória, mas traz ganhos significativos pela capacidade de refinamento contínuo do agente. Em termos de abrangência de critérios, LangSmith, Phoenix novamente se destacam por suportarem múltiplos modos simultaneamente. No AgentBench, os critérios são específicos por ambiente (e normalmente binários ou numéricos, como sucesso/falha em uma tarefa). Na avaliação geral, os frameworks que são mais nichados perderam pontuação no critério de variedade, mas ganharam no critério de inovação.

4.2.4 Compatibilidade e usabilidade

A **facilidade de integração e uso** tende a ser maior nas ferramentas com foco de produto (LangSmith, Phoenix, Vertex) do que nos frameworks de pesquisa (AgentBench, Agent-as-a-judge, Agent-Eval-Refine). LangSmith e Phoenix possuem SDKs e documentação visando desenvolvedores de aplicações, com integração relativamente direta a pipelines existentes (especialmente se já usam LangChain, no caso do LangSmith, ou se podem rodar a biblioteca Phoenix em seu ambiente). Entretanto, o Arize Phoenix se destaca ainda mais, por sua capacidade de integração sem grandes alterações de código com muitas soluções e seus instrumentadores para bibliotecas populares. O Vertex AI, sendo um serviço gerenciado, expõe

interfaces de alto nível via API/SDK do Google Cloud, simplificando a adoção para quem já está na plataforma – embora possa ser inacessível fora dela. Um problema do Vertex AI é a documentação pouco explicativa. Já o Agent-as-a-Judge e Agent-Eval-Refine requerem esforço adicional de implementação, configuração de agentes avaliadores, etc., o que os posiciona como ferramentas mais experimentais. O AgentBench, por sua vez, apesar de vir com uma suíte pronta, demanda que se adapte os modelos em teste aos ambientes e se disponha dos recursos para executá-lo.

4.2.5 Uso de Recursos

Por fim, em relação ao **uso de recursos** e **custo**, as abordagens variam de muito leves a pesadas. Soluções incorporadas em ferramentas de desenvolvimento como LangSmith e Phoenix tendem a adicionar sobrecarga mínima à aplicação, como mostrado nos resultados dos testes na Tabela 1 – tipicamente apenas logam informações e fazem chamadas extras de avaliação com um LLM quando necessário, o que é administrável. Já métodos como Agent-as-a-Judge e Agent-Eval-Refine introduzem agentes ou modelos adicionais no ciclo, implicando um custo moderado (é como rodar o agente "duas vezes", uma para agir e outra para julgar, no caso do Agent-as-a-Judge, ou múltiplas vezes no caso do refine). O Vertex AI Evaluation pode incorrer em custos variáveis dependendo do volume de dados e do modelo de julgamento utilizado (usar o Gemini Pro como juiz, por exemplo, consome créditos da plataforma), mas em termos de computação não demanda infraestrutura local do usuário, e por isso foi classificado como uso baixo na análise. O AgentBench, pelo seu caráter de benchmark intensivo, é a opção mais custosa em recursos: executar os oito ambientes contra diversos modelos envolve um grande número de interações (na ordem de dezenas de milhares de gerações no total), além de potenciais componentes de simulação. Assim, ele é o menos viável para uso frequente, sendo mais adequado a avaliações pontuais e bem planejadas (dada a necessidade de infraestrutura robusta). Assim, observa-se que ferramentas comerciais tendem a ter um overhead[2] mais baixo, enquanto soluções que introduzem novos conceitos são menos otimizadas e soluções como o AgentBench, que realizam múltiplos testes simultaneamente e em ambiente local, tem um custo de performance mais alto (Figura 1).

[2] Overhead refere-se a qualquer custo adicional (computacional, de tempo ou de recursos) necessário para a execução de uma tarefa, além da operação principal.

4.3 DISCUSSÃO

Assim, a partir da avaliação que gerou a Tabela 1 e Figura 2, a comparação realizada evidencia Langsmith e Arize Phoenix como as ferramentas mais completas em relação às abordagens mencionadas na seção 2.4. Ambas fornecem observabilidade com uma boa usabilidade, permitem a avaliação de todas as etapas do agente por meio de diversas técnicas, explicam suas respostas, utilizam poucos recursos e são versáteis quanto aos agentes que utilizam. Apesar de ambas soluções serem muito versáteis, o Arize Phoenix se destacou nesse aspecto, por permitir a instrumentação da execução de várias bibliotecas (não só o Langchain) com um único comando. Essa característica torna a ferramenta particularmente valiosa em um cenário onde novas bibliotecas para criação e gerenciamento de agentes autônomos estão surgindo continuamente. A possibilidade de incorporar facilmente um framework de observabilidade e avaliação sem a necessidade de adaptações extensas confere ao Phoenix um papel estratégico no ecossistema de IA, garantindo que desenvolvedores possam rapidamente instrumentar agentes em diferentes ambientes e acompanhar sua performance em tempo real.

Um outro aspecto interessante a ser mencionado é que plataformas de observabilidade - como o Langsmith - oferecem os meios para realização das avaliações, mas não possuem funções prontas pré-implementadas e prontas para uso para avaliar aspectos comuns a agentes, como toxicidade e alucinação. O Arize Phoenix se destaca nesse sentido, por já incorporar templates de avaliação para diferentes cenários, incluindo avaliação da escolha de ferramentas pelo agente, detecção de alucinações, geração de *SQL*, comparação de diferentes versões de *prompts* e modelos, entre outras abordagens. Essa característica se alinha com a proposta do Vertex AI GenAI Evaluation, que oferece uma variedade ainda maior de funções de avaliação, seja para avaliação da resposta final, seja para avaliação da trajetória do agente. Isso é extremamente útil, pois permite que desenvolvedores e times de desenvolvimento possam iniciar os testes de seus agentes com mais agilidade. Para casos em que se deseja ter as avaliações mais comuns de agentes prontas para uso, a solução do Vertex AI pode se encaixar mais. Caso também se deseje mais técnicas de avaliação e observabilidade, o Phoenix pode ser a melhor escolha.

Embora Langsmith e Arize Phoenix apresentem-se como as soluções mais robustas e completas para avaliação completa de agentes, é fundamental entender que ferramentas menos generalistas também possuem grande valor no processo de avaliação de agentes autônomos baseados em LLMs. O AgentBench, por exemplo, está posicionado para a avaliação de LLMs

quando utilizadas como agentes, e pode ser fundamental, para aplicações de agentes, realizar testes no AgentBench ocasionalmente para avaliar o desempenho da LLM que está em uso. Já o Agent-Eval-Refine traz inovação na avaliação das trajetórias planejadas pelo agente, por ser capaz de interpretar trajetórias enviadas por imagem, avaliar com um LLM-as-a-judge e refinar o agente incluindo a sua avaliação no contexto. O Agent-as-a-judge, por sua vez, traz uma nova abordagem de avaliação de agentes, agregando a um único avaliador todas as etapas do agente. Ao avaliar todas as etapas com capacidade de reflexão e raciocínio, a ferramenta mostra que pode se aproximar de uma avaliação humana sendo muito mais escalável (ZHUGE et al., 2024). Essas soluções atuam, muitas vezes, como laboratórios de ideias que, obtendo sucesso nos seus testes, podem se tornar o estado da arte e eventualmente serem incorporadas também em plataformas mais amplas. Portanto, apesar de não serem utilizadas no dia a dia da mesma forma que uma plataforma de observabilidade, seu valor estratégico é alto – impulsionando avanços e servindo de complemento para se atingir uma avaliação verdadeiramente robusta e abrangente. A combinação de diferentes ferramentas e técnicas de avaliação é a abordagem ideal para representar com fidelidade cada caso, especialmente em sistemas complexos como os sistemas baseados em agentes.

Nesse sentido, dado o valor do surgimento de novas soluções para o aumento da qualidade das avaliações de agentes autônomos baseados em LLMs, torna-se essencial destacar o papel da iniciativa Open Source (código aberto) (Open Source Initiative, 2024) e do movimento Software Livre, idealizado por Stallman (1985). Apesar de divergências filosóficas, ambos promovem a disponibilização pública do código-fonte, permitindo que os usuários utilizem, estudem, modifiquem e redistribuam o software. Essa abordagem fortalece toda a comunidade ao permitir que haja um crescimento colaborativo, possibilitando que as pessoas e organizações trabalhem cooperativamente para o desenvolvimento de IAs mais poderosas, seguras e explicáveis.

5 CONCLUSÃO

5.1 CONSIDERAÇÕES FINAIS

O presente trabalho realizou uma análise comparativa de ferramentas para a avaliação de agentes autônomos baseados em LLMs, com ênfase em sistemas de agente único. A pesquisa explorou diversas soluções que abordam a avaliação de diferentes formas e as avaliou segundo critérios que considerou relevantes para a avaliação de agentes, como observabilidade, avaliação da saída e do fluxo, uso de recursos, versatilidade, compatibilidade externa, explicabilidade e usabilidade.

Os resultados indicam que plataformas como Langsmith e Arize Phoenix se destacam por oferecer um ambiente integrado que combina observabilidade e avaliação, permitindo tanto a aplicação de métricas quantitativas quanto a incorporação de avaliações qualitativas. Já o Vertex AI GenAI Evaluation, não possui as funções de observabilidade, mas se destaca pela sua vasta quantidade de métricas e funções de avaliação pré-estabelecidas, tornando-a a ferramenta mais indicada para aqueles que desejam começar a avaliar rapidamente ou não possuem tempo para implementação. Outras ferramentas apresentaram propostas mais direcionadas e inovadoras, que podem indicar para onde as avaliações de agentes autônomos baseados em LLMs deve se direcionar no futuro. Assim, a escolha da ferramenta ideal depende dos requisitos específicos de cada aplicação, e é possível que, para certos casos, a solução ideal é a união de ferramentas.

Além disso, a discussão ressaltou a importância de unir métodos quantitativos e qualitativos para se obter uma avaliação mais completa e contextualizada do desempenho dos agentes autônomos. A pesquisa evidenciou que, para se alcançar avaliações eficazes, é essencial integrar diferentes abordagens, combinando métricas objetivas com feedbacks interpretativos, de modo a proporcionar uma análise robusta e adaptável às novas demandas impostas pelo cenário da IA generativa.

5.2 TRABALHOS FUTUROS

O campo de estudos de agentes autônomos baseados em LLMs é recente, e diversas ferramentas, soluções e abordagens de avaliação estão surgindo em grande velocidade. Dessa forma, estudos como este, que fazem um mapeamento do estado da arte das ferramentas de

avaliação, devem ser realizados periodicamente, de forma a contribuir para a compreensão do direcionamento que a área está tomando e indicar possíveis correções. Ademais, para próximos estudos, é interessante que o campo de explicabilidade seja avaliado de maneira mais aprofundada, realizando uma descriminação entre tipos de abordagens de explicabilidade. Isso pode ser essencial, pois a qualidade da explicação de uma avaliação é uma etapa central para compreender as ações a serem tomadas para melhorar a qualidade dos agentes.

Além disso, o presente trabalho teve seu escopo fechado a agentes de sistema único, mas sistemas multi-agentes oferecem uma arquitetura ainda mais complexa, exigindo que se avalie todos os agentes não só individualmente, mas também sua capacidade de colaboração, planejamento conjunto e tomada de decisões coordenada. Assim, é fundamental que seja feito um estudo semelhante no contexto de sistemas multi-agentes. Por fim, o estudo não só contribui para o mapeamento do estado atual dessas ferramentas, mas também abre caminho para futuras pesquisas que visem a integração de métodos híbridos de avaliação, possibilitando a evolução contínua e a melhoria dos processos de monitoramento e refinamento de agentes baseados em LLMs.

REFERÊNCIAS

AMAZON. *Amazon Bedrock*. 2025. Disponível em: <https://aws.amazon.com/bedrock/>.

ARCAS, B. A. y. Do large language models understand us? *Daedalus*, v. 151, n. 2, p. 183–197, 05 2022. ISSN 0011-5266. Disponível em: <https://doi.org/10.1162/daed_a_01909>.

Arize AI. *Arize Phoenix*. 2025. Acessado em: 18 de janeiro de 2025. Disponível em: <https://docs.arize.com/phoenix>.

Bloomberg Intelligence. *Generative AI 2024 Report*. 2024. Acesso em: 7 jan. 2025. Disponível em: <https://assets.bbhub.io/promo/sites/16/Bloomberg-Intelligence-NVDA-Gen-AIs-Disruptive-Race.pdf>.

BUBECK, S.; CHANDRASEKARAN, V.; ELDAN, R.; GEHRKE, J.; HORVITZ, E.; KAMAR, E.; LEE, P.; LEE, Y. T.; LI, Y.; LUNDBERG, S.; NORI, H.; PALANGI, H.; RIBEIRO, M. T.; ZHANG, Y. *Sparks of Artificial General Intelligence: Early experiments with GPT-4*. 2023. Disponível em: <https://arxiv.org/abs/2303.12712>.

Confident AI. *DeepEval (Confident AI)*. 2025. DeepEval é um framework open-source para avaliação de aplicações LLM, proporcionando uma abordagem detalhada para analisar o desempenho e a eficiência de modelos generativos. Disponível em: <https://confident.ai/deep-eval/>.

DALVI, F.; HASANAIN, M.; BOUGHORBEL, S.; MOUSI, B.; ABDALJALIL, S.; NAZAR, N.; ABDELALI, A.; CHOWDHURY, S. A.; MUBARAK, H.; ALI, A.; HAWASLY, M.; DURRANI, N.; ALAM, F. *LLMeBench: A Flexible Framework for Accelerating LLMs Benchmarking*. 2024. Disponível em: <https://arxiv.org/abs/2308.04945>.

DONG, L.; LU, Q.; ZHU, L. *AgentOps: Enabling Observability of LLM Agents*. 2024. Disponível em: <https://arxiv.org/abs/2411.05285>.

DOUGLAS, M. R. *Large Language Models*. 2023. Disponível em: <https://arxiv.org/abs/2307.05782>.

EleutherAI. *Language Model Evaluation Harness*. 2020. GitHub repository. Disponível em: <https://github.com/EleutherAI/lm-evaluation-harness>.

FEUERRIEGEL, S.; HARTMANN, J.; JANIESCH, C.; ZSCHECH, P. Generative ai. *Business & Information Systems Engineering*, v. 66, n. 1, p. 111–126, 2024. Disponível em: <https://doi.org/10.1007/s12599-023-00834-7>.

Google Cloud. *Evaluate Gen AI agents*. 2025. Acessado em: 18 de janeiro de 2025. Disponível em: <https://cloud.google.com/vertex-ai/generative-ai/docs/models/evaluation-agents>.

GU, J.; JIANG, X.; SHI, Z.; TAN, H.; ZHAI, X.; XU, C.; LI, W.; SHEN, Y.; MA, S.; LIU, H.; WANG, Y.; GUO, J. *A Survey on LLM-as-a-Judge*. 2025. Disponível em: <https://arxiv.org/abs/2411.15594>.

LangChain. *Get started with LangSmith*. 2025. Acessado em: 18 de janeiro de 2025. Disponível em: <https://docs.smith.langchain.com/>.

LIU, X.; YU, H.; ZHANG, H.; XU, Y.; LEI, X.; LAI, H.; GU, Y.; DING, H.; MEN, K.; YANG, K.; ZHANG, S.; DENG, X.; ZENG, A.; DU, Z.; ZHANG, C.; SHEN, S.; ZHANG, T.; SU, Y.; SUN, H.; HUANG, M.; DONG, Y.; TANG, J. *AgentBench: Evaluating LLMs as Agents*. 2023. Disponível em: <https://arxiv.org/abs/2308.03688>.

MICROSOFT. *Prompt Flow (Microsoft)*. 2025. Prompt Flow oferece ferramentas para otimizar o ciclo de vida completo de desenvolvimento de IA generativa, desde a ideação até a produção, facilitando a prototipagem, teste e implantação de aplicações baseadas em LLM. Disponível em: <https://azure.microsoft.com/en-us/services/ai-studio/prompt-flow/>.

MOHSENI, S.; ZAREI, N.; RAGAN, E. D. A multidisciplinary survey and framework for design and evaluation of explainable ai systems. *ACM Trans. Interact. Intell. Syst.*, Association for Computing Machinery, New York, NY, USA, v. 11, n. 3–4, set. 2021. ISSN 2160-6455. Disponível em: <https://doi.org/10.1145/3387166>.

Open Source Initiative. *The Open Source Definition*. 2024. <https://opensource.org/osd>. [Acessado em: 14-mar-2025].

OpenAI. *Evals*. 2023. GitHub repository. Disponível em: <https://github.com/openai/evals>.

PAN, J.; ZHANG, Y.; TOMLIN, N.; ZHOU, Y.; LEVINE, S.; SUHR, A. *Autonomous Evaluation and Refinement of Digital Agents*. 2024. Disponível em: <https://arxiv.org/abs/2404.06474>.

Parea AI. *Parea AI*. 2025. Parea oferece ferramentas para engenheiros de IA desenvolverem, testarem e monitorarem aplicações baseadas em LLM, com foco em confiabilidade e prontidão para produção. Disponível em: <https://parea.ai/>.

PARK, J. S.; O'BRIEN, J. C.; CAI, C. J.; MORRIS, M. R.; LIANG, P.; BERNSTEIN, M. S. *Generative Agents: Interactive Simulacra of Human Behavior*. 2023. Disponível em: <https://arxiv.org/abs/2304.03442>.

QIN, Y.; HU, S.; LIN, Y.; CHEN, W.; DING, N.; CUI, G.; ZENG, Z.; HUANG, Y.; XIAO, C.; HAN, C.; FUNG, Y. R.; SU, Y.; WANG, H.; QIAN, C.; TIAN, R.; ZHU, K.; LIANG, S.; SHEN, X.; XU, B.; ZHANG, Z.; YE, Y.; LI, B.; TANG, Z.; YI, J.; ZHU, Y.; DAI, Z.; YAN, L.; CONG, X.; LU, Y.; ZHAO, W.; HUANG, Y.; YAN, J.; HAN, X.; SUN, X.; LI, D.; PHANG, J.; YANG, C.; WU, T.; JI, H.; LIU, Z.; SUN, M. *Tool Learning with Foundation Models*. 2024. Disponível em: <https://arxiv.org/abs/2304.08354>.

RADFORD, A.; WU, J.; CHILD, R.; LUAN, D.; AMODEI, D.; SUTSKEVER, I. et al. Language models are unsupervised multitask learners. *OpenAI blog*, v. 1, n. 8, p. 9, 2019.

SHINN, N.; CASSANO, F.; BERMAN, E.; GOPINATH, A.; NARASIMHAN, K.; YAO, S. *Reflexion: Language Agents with Verbal Reinforcement Learning*. 2023. Disponível em: <https://arxiv.org/abs/2303.11366>.

STALLMAN, R. The gnu manifesto. *Dr. Dobb's Journal of Software Tools*, v. 10, n. 3, p. 30–35, 1985. Disponível em: <https://www.gnu.org/gnu/manifesto.en.html>.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. *Attention Is All You Need*. 2017. Disponível em: <https://arxiv.org/abs/1706.03762>.

WANG, L.; MA, C.; FENG, X.; ZHANG, Z.; YANG, H.; ZHANG, J.; CHEN, Z.; TANG, J.; CHEN, X.; LIN, Y.; ZHAO, W. X.; WEI, Z.; WEN, J. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, Springer Science and Business Media LLC, v. 18, n. 6, mar. 2024. ISSN 2095-2236. Disponível em: <http://dx.doi.org/10.1007/s11704-024-40231-1>.

WEI, J.; WANG, X.; SCHUURMANS, D.; BOSMA, M.; ICHTER, B.; XIA, F.; CHI, E.; LE, Q.; ZHOU, D. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. Disponível em: <https://arxiv.org/abs/2201.11903>.

WENG, L. Llm-powered autonomous agents. *lilianweng.github.io*, Jun 2023. Disponível em: <https://lilianweng.github.io/posts/2023-06-23-agent/>.

XIA, B.; LU, Q.; ZHU, L.; XING, Z.; ZHAO, D.; ZHANG, H. *An Evaluation-Driven Approach to Designing LLM Agents: Process and Architecture*. 2024. Disponível em: <https://arxiv.org/abs/2411.13768>.

YAO, S.; CHEN, H.; YANG, J.; NARASIMHAN, K. *WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents*. 2023. Disponível em: <https://arxiv.org/abs/2207.01206>.

YAO, S.; YU, D.; ZHAO, J.; SHAFRAN, I.; GRIFFITHS, T. L.; CAO, Y.; NARASIMHAN, K. *Tree of Thoughts: Deliberate Problem Solving with Large Language Models*. 2023. Disponível em: <https://arxiv.org/abs/2305.10601>.

YAO, S.; ZHAO, J.; YU, D.; DU, N.; SHAFRAN, I.; NARASIMHAN, K.; CAO, Y. *ReAct: Synergizing Reasoning and Acting in Language Models*. 2023. Disponível em: <https://arxiv.org/abs/2210.03629>.

ZHUGE, M.; ZHAO, C.; ASHLEY, D.; WANG, W.; KHIZBULLIN, D.; XIONG, Y.; LIU, Z.; CHANG, E.; KRISHNAMOORTHI, R.; TIAN, Y.; SHI, Y.; CHANDRA, V.; SCHMIDHUBER, J. *Agent-as-a-Judge: Evaluate Agents with Agents*. 2024. Disponível em: <https://arxiv.org/abs/2410.10934>.