UNIVERSIDADE FEDERAL DE GOIÁS / INSTITUTO DE INFORMÁTICA

Sistemas Multiagentes Baseados em LLM para Negociação

Análise de Comportamentos Emergentes e Uso de Informação de Mercado

Amir Youssef dos Santos

UFG
UNIVERSIDADE FEDERAL DE GOIÁS

UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
INSTITUTO DE INFORMÁTICA (INF)

AMIR YOUSSEF DOS SANTOS

Sistemas Multiagentes Baseados em LLM para Negociação

Análise de Comportamentos Emergentes e Uso de Informação de Mercado

Goiânia
2025

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): AMIR YOUSSEF DOS SANTOS

Título do trabalho: Sistemas Multiagentes Baseados em LLM para Negociação

Análise de Comportamentos Emergentes e Uso de Informação de Mercado

2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;

- Submissão de artigo em revista científica;

- Publicação como capítulo de livro.

Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Amir Youssef Dos Santos**, **Discente**, em 04/02/2026, às 15:58, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Fernando Marques Federson**, **Professor do Magistério Superior**, em 13/03/2026, às 11:16, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

AMIR YOUSSEF DOS SANTOS

Sistemas Multiagentes Baseados em LLM para Negociação

Análise de Comportamentos Emergentes e Uso de Informação de Mercado

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Orientador: Prof. Dr. Fernando Marques Federson

Goiânia

2025

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UFG.

SANTOS, AMIR YOUSSEF DOS
Sistemas Multiagentes Baseados em LLM para Negociação [manuscrito]: Análise de Comportamentos Emergentes e Uso de Informação de Mercado / AMIR YOUSSEF DOS SANTOS. - 2025.
162 f.: 2025

Orientador: Prof. Dr. Fernando Marques Federson
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Instituto de Informática (INF), Inteligência Artificial, Goiânia, 2025.

1. Inteligência Artificial. 2. Sistemas Multiagentes. 3. Negociação.

I. Federson, Fernando Marques , orient. II. Título.

CDU 004

AMIR YOUSSEF DOS SANTOS

Sistemas Multiagentes Baseados em LLM para Negociação
Análise de Comportamentos Emergentes e Uso de Informação de Mercado

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Data da Aprovação: 09 de dezembro de 2025.

Prof. Dr. Fernando Marques Federson
Orientador (INF-UFG)

Prof. Dr. Aldo André Díaz Salazar
Coordenador de TCC do BIA (INF-UFG)

Prof. Dr. Anderson da Silva Soares
Coordenador do BIA (INF-UFG)

Prof. Dr. Sávio Salvarino Teles de Oliveira
(INF-UFG)

AMIR YOUSSEF DOS SANTOS

Sistemas Multiagentes Baseados em LLM para Negociação

Análise de Comportamentos Emergentes e Uso de Informação de Mercado

RESUMO

Este Relatório de Conclusão de Curso tem como objetivo reunir os resultados da minha jornada para me tornar um especialista em **Sistemas Multiagentes**. Uma ilustração e sua narrativa descrevem os períodos de trabalho. Os Apêndices contêm os Termos de Aceite de Entrega e os resultados obtidos durante cada período de trabalho.

Palavras-chave: Inteligência artificial; Sistemas multiagentes; Negociação.

ABSTRACT

This Course Completion Report aims to bring together the results of my journey to become an expert in **Multi-agent systems**. An illustration and its narrative describe the work periods. The Appendices contain the Delivery Acceptance Terms and the results obtained during each work period.

Keywords: Artificial intelligence; Multi-agent systems; Negotiation.

Goiânia

2025

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

Minha Jornada

Amir Youssef dos Santos

Especialista em: Sistemas Multiagentes

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

MINHA JORNADA

Nome: **Amir Youssef dos Santos**

Especialidade: **Sistemas Multiagentes**

Objetivo deste documento

Durante o processo da disciplina Residência em IA[1], foram gerados diversos resultados na construção da minha especialização. A cada semana, um conjunto de resultados foi formalizado por um Termo de Aceite de Entrega e avaliado por uma banca, considerando o planejado e o realizado para o período. Este documento tem como objetivo descrever esses resultados obtidos, fazendo referência aos Termos de Aceite de Entrega e seus documentos associados.

Minha Jornada

Minha jornada teve início na **Semana 1**, quando busquei estabelecer uma base conceitual sólida sobre Agentes Inteligentes e Sistemas Multiagentes, estudando temas fundamentais como autonomia, racionalidade, percepção e ação. Nesse período, aprofundei-me nas obras de Wooldridge e de Russell & Norvig, essenciais para compreender a definição formal de agente, seus modos de operação e as diferenças entre as arquiteturas. Já na **Semana 2**, avancei para uma etapa mais estruturada ao iniciar a elaboração de um Guia de Estudo e realizar uma revisão organizada da literatura científica, classificando artigos relevantes por temas e áreas de aplicação, processo decisivo para orientar todas as fases posteriores da residência. Desse modo, as atividades realizadas nas **Semanas 1 e 2** foram essenciais para consolidar o escopo teórico da especialização. Todo o conteúdo aprofundado, incluindo as anotações realizadas a partir das obras de Wooldridge e Russell & Norvig e a relação completa dos artigos mapeados, encontra-se apresentado de maneira detalhada no **Apêndice 1**.

[1] Dez Semanas, entre setembro de 2025 e dezembro de 2025.

Dando continuidade a essa base teórica, iniciei a **Semana 3** com um foco específico nos Agentes Baseados em LLM, buscando compreender de forma sistemática como grandes modelos de linguagem podem assumir papéis autônomos em ambientes dinâmicos. Nesse período, concentrei-me em analisar diferenças práticas entre agentes tradicionais e agentes impulsionados por LLMs, além de aprofundar conceitos como memória, raciocínio e planejamento, incorporando esses aprendizados diretamente ao meu Guia de Estudo. Já na **Semana 4**, avancei para a exploração de áreas específicas dentro do campo dos agentes inteligentes baseados em LLM, iniciando meus estudos na área de Teoria da Mente, a partir de artigos que discutiam como agentes podem inferir crenças, intenções e estados mentais de outros agentes em cenários de interação. Paralelamente, aprofundei temas técnicos como avaliação de agentes, limitações da arquitetura e desafios de implementação. Essa sequência de atividades permitiu consolidar uma visão mais madura e crítica sobre as capacidades e restrições dos agentes LLM, e todo esse material, incluindo o Guia de Estudo, bem como os resumos iniciais das áreas exploradas, encontra-se apresentado de forma ampliada no **Apêndice 2**.

Nas **Semanas 5 e 6**, aprofundei minha investigação ao explorar diferentes áreas e aplicações de agentes baseados em LLM, ampliando de forma consistente minha compreensão sobre como esses sistemas operam em contextos variados. Na **Semana 5**, foquei inicialmente em agentes voltados para diálogos e comunicação, analisando as diferentes formas de troca de informação entre agentes e como essas interações moldam estruturas cooperativas ou competitivas. Nesse mesmo período, avancei para o estudo de Agentes Generativos, explorando o ambiente Smallville, no qual agentes simulam comportamentos humanos contínuos e socialmente coerentes a partir de módulos de memória, reflexão e planejamento. Já na **Semana 6**, direcionei meus esforços para a área de simulações com agentes LLM, buscando compreender de maneira mais aprofundada os tipos de ambientes existentes e como cada um deles influencia a forma como os agentes interagem em cenários dinâmicos. Também estudei aplicações de LLMs em jogos de informação imperfeita, o que evidenciou desafios específicos de raciocínio e tomada de decisão. Dessa forma, as **Semanas 5 e 6** foram fundamentais para expandir minha visão sobre abordagens, técnicas e ecossistemas de agentes LLM, e todo esse conteúdo,

incluindo os resumos das áreas exploradas e a pesquisa realizada sobre ambientes e aplicações, está apresentado de forma detalhada no **Apêndice 3**.

A **Semana 7** representou um ponto decisivo na minha jornada, pois concentrei meus esforços em uma Análise Técnica de Frameworks para Agentes Inteligentes, buscando compreender quais arquiteturas seriam mais adequadas para sustentar simulações complexas em larga escala. Nesse período, estudei em profundidade ferramentas como LangChain, LangGraph e CrewAI, mas dando um foco ainda maior para o AutoGen, sobretudo por sua capacidade de gerenciar contextos compartilhados, integrar ferramentas externas e coordenar, de forma estruturada, a interação entre múltiplos agentes. Paralelamente, aprofundei o estudo do DeepSpeed-Chat, analisando como os modelos de linguagem são treinados e otimizados, o que ampliou minha compreensão sobre as bases que sustentam agentes LLM avançados. Assim, a **Semana 7** foi fundamental para identificar, na prática, como cada framework influencia a escalabilidade, a robustez e a eficiência das simulações, e todos os documentos produzidos nesse período, incluindo a comparação entre os principais frameworks e o material dedicado exclusivamente ao AutoGen, encontram-se organizados no **Apêndice 4**.

Após a exploração e estudo dos principais frameworks de agentes inteligentes, nas **Semanas 8 e 9** direcionei meu trabalho para uma especialização prática em Agentes LLMs para Simulação com foco em Negociação, iniciando pela replicação completa do artigo *NegotiationGym: Self-Optimizing Agents in a Multi-Agent Social Simulation Environment*. Na **Semana 8**, concentrei-me em compreender a estrutura do ambiente proposto pelos autores, configurando o sistema e implementando políticas como *no-reflect*, *buyer-reflect*, *seller-reflect* e *both-reflect*, o que permitiu observar padrões emergentes de negociação, como ancoragem, resistência inicial e concessões graduais. Já na **Semana 9**, aprofundei a análise experimental, conseguindo reproduzir com fidelidade os comportamentos descritos no artigo e identificando oportunidades concretas de melhoria no ambiente, especialmente relacionadas à eficiência das rodadas e ao uso estratégico de informações pelos agentes. Essas duas semanas marcaram minha transição definitiva para uma abordagem mais aplicada, experimental e analítica dentro de simulações multiagentes voltadas à negociação, e todos os testes realizados, juntamente com a análise detalhada do artigo escolhido, encontram-se descritos no **Apêndice 5**.

Dando sequência ao avanço obtido com a replicação das dinâmicas de negociação do *NegotiationGym*, a **Semana 10** representou o momento em que passei a aperfeiçoar os agentes para tornar as simulações ainda mais realistas. Nesta etapa, implementei uma nova camada de informação externa por meio do MarketAgent, um componente imparcial capaz de consultar uma base de dados estruturada e fornecer aos agentes preços reais de mercado durante o processo de negociação. A construção do dataset e a integração de um mecanismo de busca local permitiram reduzir a incerteza informacional, resultando em negociações mais curtas, maior taxa de acordos e preços sempre dentro do intervalo técnico esperado. Além disso, os testes evidenciaram claramente os efeitos da assimetria e da simetria informacional sobre a estratégia adotada pelos agentes. Assim, a **Semana 10** consolidou a importância do uso de ferramentas externas para elevar a credibilidade, o rigor e a utilidade prática das simulações, e todos os resultados obtidos, juntamente com uma explicação mais detalhada do MarketAgent e dos testes realizados, encontram-se organizados no **Apêndice 6**.

Em função de tudo que vivi nesta Jornada, gostaria de deixar registrado que este processo representou um marco na minha formação como profissional de Inteligência Artificial. Ao longo das Semanas, pude construir uma base teórica sólida, explorar diferentes linhas de pesquisa, aplicar conceitos avançados e desenvolver melhorias reais em sistemas multiagentes, o que me permitiu alcançar um nível de especialização que hoje me dá segurança para atuar na área com autonomia e profundidade. A experiência da Residência foi profundamente enriquecedora, não apenas do ponto de vista técnico, mas também pessoal, pois me mostrou, na prática, a importância de dominar os fundamentos antes de avançar para aplicações complexas, e como o progresso consistente, semana após semana, transforma conhecimento em competência. Saio desta jornada mais preparado, mais consciente do meu papel como profissional e mais motivado para continuar evoluindo.

APÊNDICE 1

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 3 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

AMIR YOUSSEF DOS SANTOS

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante esta Semana, foram realizadas as seguintes atividades:

- **Estudo sobre agentes inteligentes através de vídeos no YouTube:**
  - *What are AI Agents?; IBM Technology*
  - *Andrew Ng Explores The Rise Of AI Agents And Agentic Reasoning | BUILD 2024 Keynote; Snowflake Inc.*
  - Resumo de videos sobre IA Agents
- **Leitura e aprofundamento no livro *An Introduction to MultiAgent Systems* (Michael Wooldridge):**
  - Foco nos conceitos de agentes multiagentes.
  - Introduction to MultiAgent Systems, Michael Wooldridge
- **Início do estudo do livro *Artificial Intelligence: A Modern Approach* (4ª edição, 2020) – Stuart Russell e Peter Norvig:**
  - Aprofundamento nos fundamentos gerais da IA.
  - Produção de anotações ainda em andamento.
  - Artificial Intelligence: A Modern Approach" (4ª edição, 2020); Stuart Russell e Peter N…
- **Busca de artigos científicos relacionados a agentes inteligentes:**
  - Levantamento inicial de artigos relevantes sobre agentes inteligentes.
  - Separação de alguns textos para leitura na próxima semana.
  - Artigos sobre aplicacões

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Finalizar os estudos no livro *Artificial Intelligence: A Modern Approach* (Russell & Norvig), avançando nos capítulos e ampliando as anotações.
- Revisar artigos já separados sobre agentes inteligentes.
- Pesquisar artigos que tratam das aplicações atuais de agentes inteligentes, relacionando teoria e prática.
- Aprofundar o entendimento sobre o tema, organizando as anotações em documentos de apoio.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Anotações de “An Introduction to MultiAgent Systems”; Michael Wooldridge, 2002

Surgimento: Sua ascensão foi impulsionada em meados de 90 por tendências importantes na computação, como:

- Ubicuidade: A necessidade de coordenar o comportamento de muitos sistemas individuais e, muitas vezes, heterogêneos, aumentou significativamente
- Interconexão: A proliferação da Internet, em particular a partir de 1994, criou um ambiente onde sistemas distribuídos precisam se comunicar e interagir intensamente. Isso impulsionou a necessidade de modelos de computação que pudessem lidar com essa interconectividade
- Inteligência: Com a crescente complexidade das tarefas que os computadores são capazes de automatizar e delegar, surge a demanda por sistemas mais "inteligentes" que possam lidar com problemas complexos e tomar decisões
- Delegação: As pessoas começaram a delegar tarefas cada vez mais críticas e complexas aos sistemas computacionais (como o controle de aeronaves), o que exige que esses sistemas atuem de forma autônoma, tomando iniciativas e respondendo a eventos inesperados
- Orientação Humana: Os sistemas precisam ser capazes de agir em nome de seus usuários ou proprietários, o que implica que eles devem ser capazes de interagir efetivamente com outros sistemas, cujos interesses podem não ser os mesmos, e até mesmo com humanos, para satisfazer seus objetivos de design

Conceito de Agente: sistema computacional que está situado em algum ambiente e é capaz de ação autônoma nesse ambiente para cumprir seus objetivos de design

Para ser considerado **“Inteligente”** precisa ter a capacidade de:

- **Reatividade**: Agentes inteligentes são capazes de **perceber seu ambiente e responder em tempo hábil a mudanças que ocorrem nele** para satisfazer seus objetivos de design.

- **Proatividade**: Agentes inteligentes são capazes de **exibir comportamento direcionado a objetivos, tomando a iniciativa** para satisfazer seus objetivos de design. Isso implica um comportamento que vai além de simplesmente seguir um plano predefinido, agindo para alcançar um objetivo, mesmo que as circunstâncias mudem.

- **Habilidade Social**: Agentes inteligentes são capazes de **interagir com outros agentes (e possivelmente humanos)** para satisfazer seus objetivos de design. Essa habilidade social é mais complexa do que a simples troca de informações binárias, envolvendo negociação e cooperação

O Ambiente e a Tarefa do Agente:

Os agentes operam em um **ambiente** que fornece entradas sensoriais e é afetado por suas ações. As propriedades do ambiente influenciam significativamente o design do agente

- **Acessível vs. Inacessível**: Se o agente pode obter informações completas, precisas e atualizadas sobre o estado do ambiente.
- **Determinístico vs. Não-Determinístico**: Se uma ação tem um único efeito garantido ou se há incerteza no resultado.
- **Estático vs. Dinâmico**: Se o ambiente muda apenas pelas ações do agente ou por outros processos concorrentes.
- **Discreto vs. Contínuo**: Se o ambiente pode estar em um número finito de estados discretos ou em incontáveis estados.

Ambientes **abertos** são os mais complexos: inacessíveis, não-determinísticos, dinâmicos e contínuos.

Tipos de agentes e as técnicas para construí-los:

- **Agentes de Raciocínio Dedutivo**
    - **Principal Característica**: Tomam decisões por meio de dedução lógica e prova de teoremas. Baseiam-se na ideia da IA simbólica de manipular representações lógicas do ambiente e do comportamento desejado.

- **Estado Interno**: Banco de dados de fórmulas da lógica de predicados de primeira ordem (representando crenças).
- **Seleção de Ações**: Definida em termos de regras de dedução, onde a ação a ser executada é aquela que pode ser provada a partir do banco de dados.
- **Exemplos**: Linguagens como AGENTO (especificada por capacidades, crenças iniciais, compromissos e regras de compromisso) e Concurrent MetateM (execução direta de fórmulas de lógica temporal).

- **Agentes de Raciocínio Prático**
  - **Principal Característica**: Modelam o processo de raciocínio prático humano, que envolve deliberar sobre o que fazer e raciocinar sobre os meios para atingir esses fins.

  - **Estados Mentais**: Mantêm representações explícitas de suas crenças (**B**eliefs), desejos (**D**esires) e intenções (**I**ntentions).
  - **Intenções**: São escolhas às quais o agente se compromete, direcionadas ao futuro, que impulsionam o raciocínio meios-fins e persistem até serem alcançadas ou consideradas impossíveis.
  - **Planejamento**: Podem gerar planos do zero ou, mais comumente, usar uma biblioteca de planos pré-compilados.
  - **Estratégias de Compromisso**: Determinam quando e como o agente abandonou suas intenções (ex: compromisso cego, compromisso focado).
  - **Exemplos**: HOMER (submarino robô simulado com capacidade de planejar e executar planos em um "Seaworld") e o PRS (Procedural Reasoning System), que foi uma das primeiras arquiteturas BDI.

- **Agentes Reativos**:
  - **Principal Característica**: Reagem diretamente ao ambiente com base em entradas sensoriais, sem raciocínio complexo ou representação simbólica explícita do mundo.
  - **Pontos-chave**: Implementados como regras "situação → ação" (mapeiam entrada perceptual diretamente para ações) e frequentemente organizados em hierarquias de subsunção onde camadas mais baixas (comportamentos mais básicos) têm maior prioridade.

- **Vantagens**: Simplicidade, economia, tratabilidade computacional e robustez.
- **Limitações**: Dificuldade em construir agentes com muitas camadas devido à complexidade das interações e falta de escalabilidade.
- **Exemplo**: Arquitetura de Subsunção de Brooks, experimentos do explorador de Marte de Steels.

- **Agentes Híbridos**:
  - **Principal Característica**: Combinam subsistemas reativos e deliberativos em uma arquitetura de camadas para obter reatividade imediata e comportamento proativo direcionado a objetivos.
  - **Pontos-chave**: Podem ter camadas horizontais (cada camada conectada a entrada/saída, com um mediador para resolver conflitos) ou verticais (fluxo de controle sequencial ou em duas passagens entre camadas).
  - **Exemplos**: TouringMachines (horizontal, com camadas reativa, de planejamento e de modelagem) e InteRRaP (vertical em duas passagens, com camadas de comportamento, planejamento local e planejamento cooperativo).

- **Agentes Móveis**
  - **Principal Característica**: Capazes de se transmitir (programa e estado) através de uma rede de computadores e retomar a execução em um local remoto.
  - **Pontos-chave**: Focam na eficiência do uso de recursos de rede ao mover a computação para os dados, em vez de mover os dados para a computação.
  - **Exemplos**: Linguagem Telescript (da General Magic) e a plataforma Aglets (baseada em Java).

Podemos imaginar isso como a construção de uma equipe de robôs. Alguns robôs (agentes dedutivos) seriam como cientistas que analisam dados complexos e planejam cuidadosamente cada passo lógico. Outros (agentes reativos) seriam como atletas rápidos, reagindo instintivamente a mudanças no campo. Os agentes híbridos seriam como um técnico de equipe, integrando as habilidades de raciocínio dos cientistas e a agilidade dos atletas para um desempenho otimizado.

Interações Multiagentes e Acordos:

sistema multiagente é composto por vários agentes que **interagem entre si através da comunicação**. Esses agentes atuam em um ambiente e podem ter diferentes "esferas de influência", podendo controlar ou influenciar partes distintas do ambiente

Tipo de interação que ocorre entre os agentes:

Utilidades e Preferências: A preferência de um agente por um resultado em detrimento de outro é formalizada por uma **função de utilidade** $U_k(\omega)$, que atribui um valor numérico real a cada resultado $\omega \in Q$ para o agente *k*. Agentes racionais são aqueles que buscam maximizar sua própria utilidade

Encontros Multiagentes e Matrizes de Pagamento: Em um encontro, os agentes escolhem uma ação para realizar no ambiente, e o **resultado dependerá da combinação específica de ações**. Ambos os agentes podem influenciar o resultado, e eles não veem a ação do outro agente antes de agir. Um cenário comum é que cada agente tem duas ações possíveis: "Cooperar" (C) ou "Defeccionar" (D). A forma como o ambiente se comporta é determinada por uma função $T : Ac \times Ac \rightarrow O$, onde $Ac = {C, D}$; As **matrizes de pagamento** são uma notação padrão da teoria dos jogos para resumir cenários de interação. Cada célula da matriz corresponde a uma combinação de ações e mostra os pagamentos (utilidades) recebidos por cada agente

Estratégias Dominantes e Equilíbrios de Nash: A questão central para um agente em um cenário multiagente é: o que devo fazer?

- Dominância: Uma estratégia é considerada dominante se for preferível a outras, independentemente do que o outro agente faça. Se uma estratégia s é dominada por s', um agente racional não seguirá s. Podemos eliminar estratégias dominadas para simplificar a análise
- Equilíbrio de Nash: É uma situação em que nenhum agente pode melhorar seu resultado mudando unilateralmente sua estratégia, assumindo que os outros agentes não mudam as suas. O exemplo de dirigir na esquerda ou na direita é um equilíbrio de Nash: se todos dirigem na esquerda, você também o faz

Interações Competitivas e de Soma Zero

- Estritamente Competitivas: Cenários onde um resultado ω é preferido pelo agente i em detrimento de ω' se, e somente se, ω' é preferido por j em detrimento de ω. As preferências são diametralmente opostas: o ganho de um é a perda do outro.
- Soma Zero: Uma subclasse de interações estritamente competitivas, onde as utilidades dos dois agentes somam zero para qualquer resultado. São os tipos de encontro mais "perversos", não permitindo cooperação mutuamente benéfica. Apesar de alguns céticos questionarem sua existência no mundo real, as pessoas tendem a tratá-las como se fossem de soma zero, o que pode ser prejudicial em cenários com potencial de cooperação

Dilema do Prisioneiro

Um cenário clássico que ilustra a complexidade da interação estratégica. No Dilema do Prisioneiro, a ordem de preferência para o agente *i* é: D,C > C,C > D,D > C,D.

- Se o agente *j* Cooperar, o agente *i* prefere Defeccionar (pagamento 5 vs 3).
- Se o agente *j* Defeccionar, o agente *i* prefere Defeccionar (pagamento 2 vs 0). Portanto, a estratégia dominante para ambos os agentes é Defeccionar, resultando em um pagamento de 2 para cada. O único Equilíbrio de Nash é (D, D), mesmo que (C, C) (com pagamentos de 3 para cada) seja um resultado mutuamente mais benéfico.
- A Sombra do Futuro: Em cenários iterados (onde o jogo é jogado várias vezes), a cooperação pode se tornar racional. O Torneio de Axelrod demonstrou que estratégias simples e "boas" (como TIT-FOR-TAT, que coopera inicialmente e depois imita a última jogada do oponente) podem superar estratégias mais complexas

Relações de Dependência

Sichman e colegas propuseram uma abordagem para entender as dependências entre agentes, onde uma relação de dependência existe se um agente precisa do outro para atingir um de seus objetivos. As relações podem ser:

- Independência: Nenhuma dependência.
- Unilateral: Um agente depende do outro, mas não vice-versa.
- Mútua: Ambos dependem um do outro para o mesmo objetivo.
- Recíproca: Cada agente depende do outro para algum objetivo (não necessariamente o mesmo). Essas relações podem ser qualificadas como

"localmente acreditadas" ou "mutuamente acreditadas". O sistema DepNet foi implementado para computar essas relações

Alcançando Acordos

A capacidade de **alcançar acordos** (sem a imposição de uma terceira parte) é fundamental para agentes autônomos e inteligentes, especialmente em sociedades de agentes autointeressados. **Negociação** e **argumentação** são centrais para essa capacidade, A criação de acordos envolve:

Design de Mecanismos: Projetar os protocolos (regras de encontro) para as interações. Desejamos protocolos com propriedades como:

- Eficiência de Pareto: Nenhum outro resultado melhora a situação de um agente sem piorar a de outro.
- Racionalidade Individual: Seguir o protocolo é do interesse de todos os participantes.

Desenho de Estratégias: Definir como os agentes individuais devem se comportar dentro de um protocolo para maximizar seu bem-estar

Leilões

Leilões são técnicas simples e populares para alocar bens, tarefas e recursos. Eles envolvem um leiloeiro e licitantes. O leiloeiro geralmente busca maximizar o preço, enquanto os licitantes buscam minimizá-lo.

Tipos de Leilão:

- Leilões Ingleses: Preço sobe, o último licitante ganha.
- Leilões Holandeses: Preço desce, o primeiro licitante a aceitar ganha.
- Leilões de Primeiro Preço em Lance Fechado: Lances secretos, o lance mais alto ganha e paga o valor do lance.
- Leilões Vickrey (Segundo Preço em Lance Fechado): Lances secretos, o lance mais alto ganha, mas paga o valor do segundo lance mais alto.

Questões Importantes:

- Receita Esperada: Qual protocolo maximiza a receita do leiloeiro?
- Mentiras e Conluio: A suscetibilidade dos protocolos à desonestidade dos licitantes ou do leiloeiro (por exemplo, "shills" ou lances falsos).

- Contraspeculação: A atividade de um licitante para obter informações sobre o valor verdadeiro do bem ou as avaliações de outros licitantes.

Negociação

A negociação é uma técnica mais rica para acordos em cenários mais gerais do que leilões. Quatro componentes-chave de qualquer cenário de negociação são:

- Conjunto de Negociação: O espaço de propostas possíveis.
- Protocolo: Define as propostas legais em função do histórico.
- Estratégias: Determinam as propostas de cada agente, geralmente privadas.
- Regra de Acordo: Determina quando um acordo é fechado e qual é o acordo.

Complexidades da Negociação:

- Múltiplas Questões: Negociar sobre vários atributos inter-relacionados (por exemplo, preço, garantia, extras de um carro) é mais complexo do que uma única questão.
- Crescimento Exponencial do Espaço de Acordos: Com múltiplas questões, o número de acordos possíveis pode ser enorme, dificultando a análise.
- Número de Agentes: Pode ser um-para-um, muitos-para-um (como leilões) ou muitos-para-muitos, este último sendo o mais difícil de analisar devido à complexidade das interações.

Argumentação

A argumentação é um processo pelo qual um agente tenta convencer outro da verdade (ou falsidade) de um estado de coisas, apresentando argumentos e justificativas.

Limitações da Teoria dos Jogos:
As técnicas de teoria dos jogos podem não permitir justificar as posições dos agentes de forma transparente para humanos, o que é problemático para a confiança.

Modos de Argumentação Humana (Gilbert):

- Lógico: Baseado em raciocínio e evidências.
- Emocional: Apelos a sentimentos.
- Visceral: Apelos a fatos físicos.

- Kisceral: Apelos a intuições ou experiências espirituais. Em sistemas de agentes, geralmente focamos no modo lógico.

Argumentação Baseada em Lógica:
Um argumento é uma sequência de inferências que leva a uma conclusão, usando um conjunto de proposições ("fundamentos" ou "grounds") de um banco de dados compartilhado. A força de um argumento pode ser avaliada examinando suas bases.

- Derrota (Defeat): Um argumento pode ser refutado (rebuttal) se ataca a conclusão de outro, ou enfraquecido (undercut) se ataca uma das bases do outro.
- Ataque: Uma proposição ϕ ataca ψ se $\phi \implies \neg \psi$.

Pense nisso como um mercado: as interações multiagentes são como os vários compradores e vendedores no mercado, cada um com seus próprios interesses (utilidades). Acordos são as transações que ocorrem, que podem ser simples (como um leilão de um único item), ou complexas, envolvendo negociações sobre vários aspectos do negócio. E a argumentação é como os vendedores tentam convencer os compradores de que seu produto é o melhor, ou como os compradores pechincham e justificam suas ofertas

Comunicação

Comunicação em Sistemas de Agentes vs. Sistemas de Objetos
Em sistemas de objetos, a comunicação é frequentemente tratada como invocação de métodos, onde um objeto pode forçar a execução de um método em outro. No entanto, em um ambiente de agentes, onde os agentes são **autônomos**, eles controlam seu próprio estado e comportamento. Isso significa que um agente não pode ser forçado a executar uma ação por outro agente, pois pode não ser do seu interesse. Em vez disso, os agentes realizam **ações comunicativas para influenciar outros agentes**, visando modificar seus **estados internos**, **como crenças ou intenções**

Teoria dos Atos de Fala:

- **Austin** identificou **verbos performativos** (como solicitar, informar, prometer) e distinguiu três aspectos dos atos de fala: o **ato locucionário** (o ato de proferir algo), o **ato ilocucionário** (a ação realizada ao dizer algo) e o **perlocucionário** (o efeito do ato)
- **Searle** classificou os atos de fala em cinco categorias principais: **representativos** (comprometem o falante com a verdade de uma proposição, ex: informar), **diretivos** (tentam fazer com que o ouvinte faça algo, ex: solicitar), **comissivos** (comprometem o falante a um curso de ação, ex: prometer), **expressivos** (expressam um estado psicológico) e **declarações** (afetam mudanças em um estado institucional de coisas

Ontologias para Comunicação de Agentes:

Para que os agentes se comuniquem sobre um domínio, é crucial que eles concordem com a terminologia usada. Uma **ontologia** é uma definição formal de um corpo de conhecimento, tipicamente uma taxonomia de relações de classe/subclasse e definições de relacionamentos. Linguagens como **KIF (Knowledge Interchange Format)** são usadas para expressar essas propriedades e relações

Metodologias: Visão Geral

Uma **metodologia de análise e design** tem como objetivo principal auxiliar na compreensão e no projeto de um sistema. Elas geralmente consistem em uma **coleção de modelos e um conjunto de diretrizes associadas**. Os modelos começam de forma abstrata e se tornam cada vez mais concretos e detalhados à medida que o processo de análise e design avança, aproximando-se da implementação

As metodologias para análise e design de sistemas baseados em agentes podem ser divididas em dois grandes grupos:

- As que se inspiram no **desenvolvimento orientado a objetos (OO)**, estendendo ou adaptando metodologias existentes de OO para os propósitos de engenharia de software baseada em agentes (APSE).

- As que adaptam **engenharia de conhecimento ou outras técnicas**

Visão Geral das Aplicações

As aplicações de agentes podem ser divididas em dois grandes grupos:

• **Sistemas distribuídos**: Onde os agentes atuam como nós de processamento em um sistema distribuído, com ênfase no aspecto "multi" dos sistemas multiagentes.

• **Assistentes de software pessoais**: Onde os agentes desempenham o papel de assistentes proativos para usuários que trabalham com uma aplicação, com ênfase nos agentes "individuais"

Áreas de Aplicação Notáveis:

- **Agentes para Gerenciamento de Fluxo de Trabalho e Processos de Negócio:**
  - Sistemas de fluxo de trabalho visam automatizar processos de negócios, garantindo que tarefas sejam executadas pelas pessoas certas no momento certo

- **Agentes para Comércio Eletrônico (e-commerce):**
  - Agentes podem automatizar ou auxiliar em várias etapas do processo de compra e venda. As etapas incluem identificação de necessidades, intermediação de produtos, intermediação de comerciantes, negociação, compra e entrega, e serviço/avaliação do produto

- **Agentes para Ambientes Virtuais:**
  - A tecnologia de agentes pode ser combinada com cinema, jogos de computador e realidade virtual.
- **Agentes para Simulação Social:**
  - Agentes são usados como uma ferramenta experimental nas ciências sociais para simular o comportamento de sociedades humanas. Agentes individuais podem representar pessoas ou organizações.

Anotações de “Artificial Intelligence: A Modern Approach” (4ª edição, 2020); Stuart Russell e Peter Norvig

O livro explora a vasta área da Inteligência Artificial (IA) através de uma perspectiva atual, unificando o conhecimento existente em uma estrutura comum. A obra aborda desde fundamentos matemáticos até aplicações práticas e questões éticas, com foco na construção de **agentes inteligentes**

Inteligência Artificial

- Estudo de agentes que recebem percepções (percepts) do ambiente e realizam ações.
- Cada agente inteligente implementa uma função que mapeia sequências de percepções em ações

O que é um Agente?

- Um agente é simplesmente "algo que age" (a palavra "agente" vem do latim *agere*, fazer).
- No contexto da IA, um agente é qualquer entidade que pode ser vista como percebendo seu ambiente através de sensores e agindo sobre esse ambiente através de atuadores

Sensores e Atuadores:

- Um agente humano tem olhos, ouvidos e outros órgãos como sensores, mãos, pernas e trato vocal como atuadores.
- Um agente robótico pode ter câmeras e sensores infravermelhos como sensores, e vários motores como atuadores

Percepções e Sequências de Percepções:

- A informação que os sensores de um agente percebem é chamada de percepção (percept).
- A sequência de percepções é o histórico completo de tudo que o agente já percebeu.

- A escolha de uma ação pelo agente pode depender de seu conhecimento inato e de toda a sequência de percepções observada até o momento.

Comportamento Racional: Um agente racional é aquele que age de forma a alcançar o melhor resultado ou, quando há incerteza, o melhor resultado esperado

Critério de Sucesso (Medida de Desempenho):

- O comportamento de um agente é avaliado por suas consequências.
- A medida de desempenho avalia qualquer sequência de estados do ambiente gerada pelo agente.
- É crucial que as medidas de desempenho sejam projetadas de acordo com o que se deseja alcançar no ambiente, e não com a forma como se pensa que o agente deve se comportar.

Racionalidade vs. Onisciência:

- Um agente **onisciente** conhece o resultado real de suas ações e age de acordo, mas a onisciência é impossível na realidade.
- A **racionalidade** maximiza o desempenho *esperado*, enquanto a perfeição maximiza o desempenho *real*.

Capacidades de um Agente Inteligente:

Para ser considerado **inteligente** ou **racional**, um agente precisa de certas capacidades:

- **Operar autonomamente:** Deve ser capaz de funcionar sem intervenção humana constante.
- **Perceber seu ambiente:** Coletar informações através de seus sensores.
- **Persistir por um período prolongado:** Manter suas operações ao longo do tempo.
- **Adaptar-se à mudança:** Ser flexível e ajustar seu comportamento em novas circunstâncias.
- **Criar e buscar objetivos:** Definir e trabalhar para atingir estados desejáveis.
- **Processamento de Linguagem Natural:** Para se comunicar com sucesso em uma linguagem humana.
- **Representação de Conhecimento:** Para armazenar o que sabe ou ouve sobre o mundo.

- **Raciocínio Automatizado**: Para responder a perguntas e tirar novas conclusões.
- **Aprendizado de Máquina:** Para se adaptar a novas circunstâncias, detectar e extrapolar padrões, e melhorar seu desempenho com a experiência**.** Um agente de aprendizado possui um elemento de aprendizado que modifica os componentes do agente com base no *feedback*.
- **Coleta de Informações:** Realizar ações para modificar percepções futuras, o que é uma parte importante da racionalidade.
- **Autonomia:** Um agente racional deve ser autônomo, aprendendo o que pode para compensar o conhecimento prévio parcial ou incorreto.

Estrutura Interna dos Agentes (Implementação):

O programa do agente implementa a função do agente (o mapeamento de sequências de percepções para ações). Tem quatro tipos básicos de programas de agentes:

- **Agentes reflexivos simples:** Selecionam ações com base apenas na percepção atual.
- **Agentes reflexivos baseados em modelo:** Mantém um estado interno do mundo (um modelo) para lidar com a observabilidade parcial, atualizando-o com base nas percepções e em modelos de transição e sensor.
- **Agentes baseados em metas:** Usam informações de metas para escolher ações que as alcançarão, envolvendo consideração do futuro.
- **Agentes baseados em utilidade**: Escolhem ações que maximizam uma função de utilidade, que é uma internalização da medida de desempenho, permitindo comparações graduais entre estados e resolução de objetivos conflitantes

Imagine um agente de IA como um chef de cozinha

- **Agente de reflexo simples** seria como um chef que só sabe "se a panela está fervendo, abaixe o fogo". Ele reage a um único percepto (água fervendo) com uma ação predefinida (abaixar o fogo), sem considerar a receita ou o que aconteceu antes.
- **Agente de reflexo baseado em modelo** seria um chef que mantém um registro mental de todos os ingredientes que já foram adicionados e os

passos da receita até agora. Ele usa esse "modelo" interno para reagir de forma mais informada, por exemplo, "se a panela está fervendo e o molho já engrossou, então abaixe o fogo".

- **Agente baseado em objetivo** seria um chef que tem uma receita final (o objetivo, como "um bolo de chocolate pronto"). Ele usa seu modelo do mundo para planejar uma sequência de passos (misturar, assar, decorar) que o levarão a esse bolo.
- **Agente baseado em utilidade** seria um chef que não só quer o bolo, mas também se preocupa com o quão bom ele será (sabor, apresentação) e o custo dos ingredientes. Ele escolherá as ações que maximizam a satisfação geral, ponderando os prós e contras de cada escolha

Levantamento dos principais artigos sobre Agentes de IA em LLMs, separado por categoria

Revisões e Estado da Arte:

- **Large Language Model Agent: A Survey on Methodology, Applications and Challenges**
  Este artigo faz uma revisão sistemática dos agentes baseados em LLMs, cobrindo arquitetura, colaboração, evolução, ferramentas, desafios e aplicações diversas (Consensus).

- **Exploring Autonomous Agents through the Lens of Large Language Models: A Review**
  Uma revisão mais ampla das capacidades, desafios (como alucinação e alinhamento de valores humanos) e técnicas emergentes para LLMs como agentes autônomos (Consensus).

- **Exploring Large Language Model based Intelligent Agents: Definitions, Methods, and Prospects**
  Um mapeamento detalhado de como agentes inteligentes são construídos com LLMs, incluindo planejamento, cognição, uso de ferramentas e interações em sistemas multiagentes (Consensus).

- **From Assistants to Agents in the LLM Era**
  Traça a evolução de assistentes para agentes baseados em LLM, destacando avanços em autonomia, raciocínio e acesso a ferramentas externas, bem como desafios para construir ecossistemas de agentes integrados (ACM Digital Library).

Arquiteturas, Estruturas e Cognição de Agentes:

- **Evaluating and Enhancing LLMs Agent Based on Theory of Mind in Guandan**
  O estudo avalia LLMs em jogos cooperativos complexos, usando a Teoria da Mente para melhorar colaboração e adaptação. Apesar de ficarem atrás de modelos de RL, mostram avanços na cooperação entre agentes (IEEExplore).

- **Personality-Driven Decision Making in LLM-Based Autonomous Agents**
  Mostra como personalidades induzidas em LLMs moldam planejamento e decisões de agentes autônomos, validando uso em cenários de defesa cibernética (ACM Digital Library).

- **SALLMA: A Software Architecture for LLM-Based Multi-Agent Systems**
  O artigo apresenta a arquitetura **SALLMA**, que organiza sistemas LLM multiagente em camadas para superar limitações de agentes únicos, com foco em flexibilidade, memória e confiabilidade. Um protótipo foi implementado e testado em cenários reais (IEEExplore).

- **Developing Multi-Agent LLM Applications Through Continuous Human-LLM Co-Programming**
  Propõe o modelo **COPMA**, que permite desenvolvimento colaborativo e iterativo entre humanos e LLMs em sistemas multiagente. Mostra eficácia em reduzir custos e acelerar prototipagem (IEEExplore).

- **Facilitating Trustworthy Human-Agent Collaboration in LLM-based Multi-Agent System oriented Software Engineering**
  Apresenta framework RACI para distribuir tarefas entre humanos e agentes LLM em engenharia de software, fortalecendo confiança, responsabilidade e colaboração (ACM Digital Library).

- **Memory Sharing for Large Language Model based Agents**
  Propõe um sistema de memória compartilhada entre agentes LLMs para evoluir da inteligência individual à coletiva, melhorando performance em tarefas abertas (Consensus).

Avaliação e Métricas de Desempenho:

- **Appraising Success of LLM-based Dialogue Agents**
  O artigo analisa o funcionamento e os fatores que influenciam o sucesso de agentes de diálogo baseados em LLMs. Embora apresentem comportamento similar ao humano, sua aplicação eficaz depende de múltiplos aspectos contextuais e técnicos (IEEExplore).

Aplicações Práticas:

- **From Data to Story: Towards Automatic Animated Data Video Creation with LLM-Based Multi-Agent Systems**
  O artigo apresenta o **Data Director**, sistema multiagente que gera vídeos animados de dados de forma autônoma. Demonstra viabilidade, mas aponta desafios em decomposição de tarefas e integração (IEEExplore).

- **Envisioning Recommendations on an LLM-Based Agent Platform**
  O artigo destaca o potencial de agentes LLM como novos meios de troca de informação e especialistas em domínios específicos, oferecendo interações mais naturais e eficientes (ACM Digital Library).

- **LLM experiments with simulation: Large Language Model Multi-Agent System for Simulation Model Parametrization in Digital Twins**
  Propõe um sistema multiagente baseado em LLMs para parametrizar gêmeos digitais de forma autônoma, combinando raciocínio e interação dinâmica. Mostrou eficácia em estudo de caso e oferece código aberto para replicação (IEEExplore).

- **Virtual Agents for Alcohol Use Counseling: Exploring LLM-Powered Motivational Interviewing**
  O estudo demonstra que LLMs podem sustentar agentes virtuais eficazes para entrevistas motivacionais em aconselhamento sobre álcool, replicando com sucesso a empatia de conselheiros humanos (ACM Digital Library).

APÊNDICE 2

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: **25 de set. de 2025**

Participantes da Entrega **[matriculados em Residência em IA]**:

AMIR YOUSSEF DOS SANTOS

Entrega: **[descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Na minha especialização em Agentes Inteligentes inicialmente:

- Leitura dos livros bases e comecei a afunilar em Agentes Inteligentes baseados em LLM
- Estou juntando todos os meus conhecimentos em um Guia de Estudo

Nesta Semana continuei aprofundando em Agentes Inteligentes baseados em LLM:

- Finalizei a leitura e resumo do artigo de "Revisões e Estado da Arte": Exploring Autonomous Agents through the Lens of Large Language Models: A Review
  - Passando por tópicos de maneira mais aprofundada como:
  - Memória
  - Raciocínio
  - Avaliação de agentes
  - Restrições na implementação

  Exploring Autonomous Agents through the Lens of Large Language Models: A Review

- Adicionei ao Tópico "Agentes Inteligentes em LLMs" no Guia de Estudo:
  - Tópicos anteriores
  - Avaliação de agentes

  Guia de Estudo

- Leitura e resumo do artigo "Appraising Success of LLM-based Dialogue Agents":
  - Agentes de diálogos são compostos por Cérebro, percepção e ação.
  - Técnicas de treinamento
  - Técnicas para decidir o próximo estado do diálogo (chain-of-thought, prompt-chaining)
  - Design genérico de um Agente de Diálogo baseado em LLM (modelo, pré-processamento, contexto, entrada e resposta)

  Appraising Success of LLM-based Dialogue Agents

- Leitura e geração do resumo do artigo "Evaluating and Enhancing LLMs Agent based on Theory of Mind in Guandan: A Multi-Player Cooperative Game under Imperfect Information"

- Avaliando e Melhorando Agentes LLMs Baseados na Teoria da Mente em Guandan: Um Jogo Cooperativo Multijogador sob Informação Imperfeita
- GuanDan - Jogo de cartas chinês de informações incompletas
- Entrada do agente - Histórico do jogo, regras de observação e regras do jogo; Interpretação do Estado Atual; Planejamento com Teoria da Mente; Recomendador de Ações (80 totais); Avaliador de Plano; Execução no Ambiente de Jogo
- Usa RL para que o LLM concentre em apenas um número de ações, aumentando a eficiência
- GPT-4 chegou próximo ao estado da arte (danzero+) utilizando ToM de segunda ordem e recomendador de ações
- Desempenho promissor
- Testaram com o ambiente no idioma chinês e inglês - chinês foi superior

Evaluating and Enhancing LLMs Agent based on Theory of Mind in Guandan: A Multi-…

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

- Continuar explorando artigos de diferentes áreas
- Continuar anotando os conteúdos principais no “Guia de Estudo”

Observação: [caso precise fazer alguma observação, de qualquer “natureza”]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Guia de Agentes Inteligentes: dos conceitos básicos aos dias atuais

Introdução

A Inteligência Artificial (IA) busca compreender e construir sistemas inteligentes, capazes de agir de forma eficaz e segura em diferentes situações. Historicamente, surgiram quatro abordagens principais:

- **Pensar como humanos** (psicologia e processos cognitivos).
- **Agir como humanos** (máquinas que simulam inteligência humana).
- **Pensar racionalmente** (base lógica e matemática).
- **Agir racionalmente** (escolher a melhor ação, mesmo sob incerteza).

História e Evolução

- **IA Clássica (anos 50–70)** A ideia da IA é antiga, mas o trabalho que geralmente é reconhecido como o início da IA foi realizado por Warren McCulloch e Walter Pitts em 1943, com um modelo de **neurônios artificiais**. Alan Turing, em 1950, apresentou uma agenda persuasiva que incluía o Teste de Turing, aprendizado de máquina, algoritmos genéticos e aprendizado por reforço. Em 1956, o termo "inteligência artificial" foi oficialmente criado na Conferência de Dartmouth. Essa era foi de grande otimismo, focado em tarefas que indicavam inteligência humana, como jogos, quebra-cabeças e matemática, levando à era "Look, Ma, no hands!".

- **Agentes Racionais (anos 80)** A partir dos anos 80, houve uma mudança para uma abordagem mais científica, incorporando probabilidade e aprendizado de máquina em vez de lógica booleana e codificação manual. **Um agente racional** é aquele que age para alcançar o melhor resultado ou o melhor resultado esperado, especialmente em situações de incerteza. No entanto, a fragilidade dos sistemas especialistas diante da incerteza e a incapacidade de aprender com a experiência levaram a um período conhecido como "inverno da IA". A ressurreição da probabilidade na IA foi

impulsionada pelo **desenvolvimento de redes Bayesianas por Judea Pearl em 1988**, fornecendo um formalismo rigoroso e eficiente para representar conhecimento incerto.

- **Agentes Autônomos e Multiagentes (anos 90)** Um agente racional deve ser autônomo, ou seja, deve aprender e compensar o conhecimento prévio parcial ou incorreto. A área de sistemas multiagentes (MAS) consolidou-se como uma subdisciplina da IA na década de 1990. **Sistemas multiagentes** envolvem muitos agentes inteligentes e autônomos (programas de software ou robôs) interagindo, seja cooperativamente ou de forma egoísta. Eles dividem tarefas entre agentes especializados, melhorando o desempenho e a adaptabilidade em ambientes complexos, e podem escalar sem esforço, embora adicionem complexidade no design e coordenação.

- **Agentes Cognitivos e Inteligentes (2000+)** No século 21, a pesquisa em IA continuou a explorar a modelagem da mente humana. A ciência cognitiva combina modelos de IA com técnicas experimentais da psicologia para construir teorias precisas e testáveis da mente humana. A psicologia cognitiva vê o cérebro como um dispositivo de processamento de informações. Kenneth Craik, em 1943, descreveu os três passos cruciais de um **agente baseado em conhecimento**: **tradução do estímulo em representação interna**, **manipulação da representação** e **retradução em ação**. Esforços para criar IA em nível humano (HLAI) ou Inteligência Artificial Geral (AGI) ganharam destaque, visando máquinas capazes de aprender a fazer qualquer coisa que um humano possa fazer. Paralelamente, a "aumento da inteligência" (IAg) enfatiza que os computadores devem ampliar as habilidades humanas, não apenas automatizar tarefas.

- **Papel dos agentes no contexto da era dos dados e do aprendizado profundo: A era do Big Data (a partir de 2001)**, com a vasta disponibilidade de conjuntos de dados, juntamente com o avanço do aprendizado de máquina, tornou a IA comercialmente atraente. O Aprendizado Profundo (**Deep Learning**), a partir de 2011, revolucionou a IA ao usar redes neurais com múltiplas camadas. Essas arquiteturas permitiram avanços notáveis em reconhecimento de objetos visuais, tradução automática, reconhecimento de fala e aprendizado por reforço. Agentes impulsionados por deep learning, como o sistema Deep Q-network (DQN) que joga jogos de Atari, e o

ALPHAGO que derrotou campeões humanos de Go, demonstram a capacidade de aprender funções de valor a partir de dados brutos. A capacidade de aprender de ponta a ponta (end-to-end learning) é uma vantagem significativa.

O que é Agentes Inteligentes?

Agentes são sistemas computacionais que está situado em algum ambiente e é capaz de realizar uma **ação autônoma** nesse ambiente para cumprir seus objetivo.

Para ser considerado **"Inteligente"** precisa ter a capacidade de:

- **Reatividade**: Perceber seu ambiente e responder em tempo hábil a mudanças que ocorrem.

- **Proatividade**: Exibir comportamento direcionado a objetivos, tomando a iniciativa para satisfazer seus objetivos. Isso implica um comportamento que vai além de simplesmente seguir um plano predefinido, agindo para alcançar um objetivo, mesmo que as circunstâncias mudem.

- **Habilidade Social**: Interagir com outros agentes (e possivelmente humanos) para satisfazer seus objetivos. Essa habilidade social é mais complexa do que a simples troca de informações binárias, envolvendo negociação e cooperação.

(An Introduction to MultiAgent Systems; Michael Wooldridge, 2002)

Tipos clássicos de agentes:

O programa do agente implementa a função do agente (o mapeamento de sequências de percepções para ações). Tem quatro tipos básicos de programas de agentes:

- **Agentes reflexivos simples:** Selecionam ações com base apenas na percepção atual.

- **Agentes reflexivos baseados em modelo:** Mantém um estado interno do mundo (um modelo) para lidar com a observabilidade parcial, atualizando-o com base nas percepções e em modelos de transição e sensor.

- **Agentes baseados em metas:** Usam informações de metas para escolher ações que as alcançarão, envolvendo consideração do futuro.

- **Agentes baseados em utilidade**: Escolhem ações que maximizam uma função de utilidade, que é uma internalização da medida de desempenho, permitindo comparações graduais entre estados e resolução de objetivos conflitantes

Agentes Inteligentes em LLMs

Diferença entre agentes clássicos e agentes LLM:

Agentes Clássicos normalmente operam em **ambientes controlados ou simulados**, utilizando **regras explícitas** e **funções heurísticas simples**. Sua "inteligência" é em grande parte **pré-programada** (*hardcoded*), respondendo a percepções específicas com ações predefinidas. Embora possam aprender em alguns contextos, sua **adaptabilidade é limitada**, especialmente em cenários abertos e imprevisíveis.
A representação do mundo nesses agentes pode variar de:

- **Atômica**: estados indivisíveis;
- **Fatorada**: descritos como vetores de atributos;
- **Estruturada**: composta por objetos e relações.

Essa evolução aumenta o poder expressivo, mas também traz maior **complexidade de raciocínio e aprendizado**.

Agentes Baseados em LLMs, por sua vez, representam uma abordagem mais moderna e flexível, impulsionada pelos avanços em **grandes modelos de linguagem**. Diferentemente dos clássicos, eles são capazes de **perceber, raciocinar e agir** em ambientes complexos, sendo vistos como candidatos promissores à **Inteligência Artificial Geral (IAG)**.
Esses agentes contam com:

- **Memória** (curto e longo prazo), que preserva histórico de interações e conhecimentos adquiridos.

- **Planejamento**, permitindo decompor tarefas, avaliar alternativas e ajustar estratégias por meio de feedback.
- **Ação (Uso de ferramentas externas)**, utilizar buscadores, interpretadores de código, calculadoras e APIs — com a capacidade de até criar suas próprias ferramentas.

Essa arquitetura lhes permite **aprender continuamente, adaptar-se a novos contextos e expandir suas capacidades**. Ainda assim, enfrentam desafios relevantes, como:

- **Multimodalidade**: integração eficiente de texto, imagens, áudio e outros dados.
- **Alinhamento com valores humanos**: evitar vieses, falhas de segurança e dilemas éticos.
- **Alucinações**: geração de informações incorretas ou inventadas.
- **Coordenação em ecossistemas multiagentes**: risco de inconsistências e perda de eficiência.

Memória em agentes LLM:

A arquitetura de memória dos agentes LLM é diversa e essencial para o seu funcionamento, permitindo armazenar parâmetros do modelo e ativações computacionais intermediárias.

- A **memória de curto prazo** é responsável por manter o histórico imediato das interações do agente, garantindo continuidade e coerência em diálogos ou execuções de tarefas. Ela funciona como um “buffer” temporário, limitado pela janela de contexto do modelo, e permite que o agente se lembre do que acabou de ser dito ou feito. É essencial em aplicações conversacionais e frameworks como *ReAct* e *Graph of Thoughts*, mas exige compressão ou sumarização constante para evitar perda de informação.
- A **memória de longo prazo** vai além do contexto imediato e registra experiências, raciocínios intermediários e conhecimentos úteis em bases estruturadas, como bancos vetoriais ou grafos de conhecimento. Essa memória possibilita que o agente reutilize soluções passadas, consulte informações externas por meio de técnicas como *Recuperação Aumentada por Geração (RAG)* e até forme bibliotecas de habilidades. Projetos como

Voyager e *Reflexion* demonstram como essa camada de memória amplia a autonomia e o aprendizado contínuo.

- Já a **memória compartilhada em sistemas multiagentes** é voltada para a colaboração. Em vez de armazenar apenas informações individuais, ela permite que diferentes agentes acessem e utilizem um repositório comum, trocando conhecimento e coordenando decisões. Essa memória pode ser organizada de forma centralizada (com um "repositório global"), descentralizada (cada agente mantém sua parte) ou híbrida. Frameworks como *Chain of Agents* mostram como esse compartilhamento aumenta a eficiência e a capacidade coletiva de resolver problemas complexos.

Planejamento em agentes LLM

O **planejamento** em agentes baseados em LLMs é a capacidade de estruturar e organizar ações de forma estratégica para resolver problemas complexos. Ele funciona como o elo entre a definição de objetivos e a execução prática, permitindo que o agente vá além de respostas imediatas e desenvolva trajetórias mais consistentes. As capacidades de planejamento dos agentes LLM podem ser vistas sob duas perspectivas principais: **estratégias de decomposição de tarefas** e **iteração orientada por *feedback***.

- **Decomposição de tarefas**, onde problemas grandes são divididos em subtarefas menores e mais fáceis de gerenciar. Essa abordagem aumenta a clareza do raciocínio e reduz a chance de erros.
    - **Plan-and-solve Prompting:** transforma um objetivo amplo em etapas sequenciais;
    - **ReAct (Reason for Future, Act for Now):** intercala momentos de raciocínio e ação em ciclos adaptativos.
    - **Tree of Thoughts (ToT):** explora diferentes caminhos de raciocínio antes de escolher o mais promissor;
    - **Graph of Thoughts (GoT):** conecta ideias em forma de grafo para expandir possibilidades;

- **Iteração orientada por *feedback*** permitindo que o agente aprenda com o *feedback* e melhore seu desempenho ao longo do tempo. O *feedback* pode vir de diversas fontes:
    - **Entrada Ambiental**: Comum na robótica, é gerado pelo ambiente em que o agente atua. O *feedback* ambiental pode aumentar significativamente a eficácia dos agentes.
    - **Orientação Humana**: Proveniente das interações do usuário ou dados rotulados manualmente. Por exemplo, em jogos, o *feedback* humano pode ajudar o agente a adquirir estratégias mais rapidamente.
    - **Introspecção do Modelo**: *Feedback* gerado pelo próprio agente.
    - **Colaboração Multiagente**: Múltiplos agentes trabalham juntos para resolver um problema e trocam *insights*.

Por exemplo, o agente pode usar o *feedback* para atualizar (regenerar) seu plano, ajustar seu caminho de raciocínio ou até mesmo modificar seu objetivo. O *framework* **Reflexion**, por exemplo, permite que um agente LLM assimile o *feedback* ambiental para refinar seu modelo interno do mundo, aprimorando sua capacidade de prever resultados e tomar decisões mais judiciosas.

Ferramentas Externas (Tool Use)

A utilização de ferramentas é um aspecto crucial que aumenta significativamente as funcionalidades dos LLMs, permitindo-lhes ir além da mera geração de texto.

Acesso a APIs, bancos de dados, web e ações no mundo real: As ferramentas são unidades executáveis com funções específicas, que permitem aos LLMs realizar tarefas que exigem informações em tempo real ou cálculos precisos, tarefas nas quais os LLMs sozinhos podem não ter um bom desempenho. Um agente LLM deve ser capaz de decidir **quando** usar uma ferramenta (especialmente em situações de baixa confiança ou necessidade de funções específicas) e **qual** ferramenta selecionar, o que requer compreensão das ferramentas disponíveis e da situação atual.

- **Recuperação de Conhecimento**: Ferramentas como motores de busca (WebGPT, WebCPM, ToolCoder) permitem que os agentes LLM acessem informações atualizadas da web, superando as limitações dos dados de treinamento.

- **Cálculo**: Interpretadores Python e calculadoras (AutoCoder, Toolformer, ART) ajudam os agentes LLM em cálculos complexos e execução de código, mitigando alucinações em tarefas que exigem precisão.
- **Interações com APIs**: APIs RESTful (Rest-GPT, GraphQLRestBench) permitem que os agentes LLM chamem serviços externos, manipulem bancos de dados e implementem processos automatizados no mundo real.
- **Ações no mundo real**: Para tarefas corporais (embodied tasks) e robóticas, os agentes LLM precisam da capacidade de executar ações físicas e interpretar o feedback ambiental, considerando hardware robótico, conhecimento social e interações com outros agentes LLM. *Frameworks* como LangChain e Auto-GPT são exemplos de como essas capacidades são integradas para criar agentes que podem usar ferramentas e memória para resolver problemas complexos.

Desafios e Limitações

Apesar de seu potencial, os agentes baseados em LLMs enfrentam obstáculos significativos que afetam sua eficácia, segurança e aplicabilidade em larga escala.

Um dos principais problemas são as **alucinações**, ou seja, a geração de informações incorretas, infundadas ou sem sentido. Esse fenômeno compromete a confiança do usuário e pode ter consequências graves em contextos críticos. Para mitigar esse risco, são usadas estratégias como a melhoria dos dados de treinamento, o uso de técnicas de **Recuperação Aumentada por Geração (RAG)**, a verificação cruzada com grafos de conhecimento e a incorporação de feedback humano.

A **segurança** e o **alinhamento** também são desafios centrais. Os agentes podem falhar em interpretar corretamente as intenções humanas ou reproduzir vieses e preconceitos herdados dos dados de treinamento. Além disso, estão sujeitos a ataques que exploram vulnerabilidades:

- ***Centrados no agente*:** como ataques adversários, jailbreaks e backdoors.
- ***Centrados nos dados*:** como injeção de prompts maliciosos e envenenamento de bases externas.

Medidas de mitigação incluem **monitoramento contínuo**, **filtros de segurança, validação de respostas e estruturas de defesa multicamadas**. A explicabilidade e a supervisão humana permanecem fundamentais para garantir comportamentos confiáveis e éticos.

Estruturas de Avaliação

A avaliação de agentes baseados em LLMs é um desafio porque envolve não apenas medir desempenho em tarefas isoladas, mas também considerar **adaptação**, **interação** e **raciocínio em cenários dinâmicos**. Diferentes estruturas têm sido propostas para lidar com essa complexidade.

1. Estruturas Tradicionais

- Focam em ambientes isolados e tarefas específicas (benchmarks, simulações).
- São úteis como referência, mas limitadas para avaliar agentes em ambientes dinâmicos e interativos.

2. AgentBench

- Benchmark multidimensional com 8 ambientes (sistemas operacionais, bancos de dados, gráfico de conhecimento, jogos, quebra-cabeças de pensamento lateral, HouseHolding (HH), Web Shopping (WS) e Web Browsing (WB)).
- Testou 27 LLMs e identificou lacunas: dificuldade em raciocínio de longo prazo, tomada de decisão e seguir instruções.
- Sugere treinamento com dados de múltiplos turnos e alinhamento de alta qualidade.

3. Web Arena

- Ambiente realista e reproduzível, com websites funcionais em e-commerce, fóruns, software colaborativo e CMS.
- Tarefas de longo horizonte, avaliadas por conclusão funcional.
- Desempenho atual é baixo: GPT-4 alcançou 14,41% de sucesso contra 78,24% de humanos.
- Aproxima-se mais de cenários reais que benchmarks sintéticos.

4. ToolLLM

- Estrutura focada em uso de ferramentas e APIs reais (mais de 16 mil).
- Inclui o ToolBench, conjunto de dados para treinar LLMs no uso de APIs.
- O modelo ToolLLaMA mostrou boa generalização para APIs novas e desempenho próximo ao ChatGPT em tarefas complexas.

5. Avaliação Subjetiva

- Envolve julgamento humano para medir eficácia, usabilidade e qualidade das saídas.
- Métodos:
    - Anotação humana (relevância, coerência, fluência).
    - Teste de Turing (avaliar se o modelo é indistinguível de um humano).
    - Estudos de usuário e revisões de especialistas para cenários aplicados.
- Fornece insights qualitativos, mas é mais demorada e sujeita a vieses.

Panorama rápido de frameworks de agentes LLM

LangChain (base geral de orquestração)

O que é: biblioteca modular para construir apps com LLMs (chains, agentes, ferramentas, RAG).
Quando usar: projetos variados que exigem pipelines de prompts, memória, ferramentas e integrações com muitos provedores.
Pontos fortes: ecossistema enorme, conectores prontos, documentação madura.
Limites: Árvore de decisão complexa em projetos grandes; agentes “genéricos” podem exigir muita customização para decisões não triviais.

LangGraph (agentes como grafos)

O que é: extensão do LangChain para modelar **fluxos de agente** como **grafos** (nós=passos/agitadores, arestas=transições condicionais).
Quando usar: problemas com controle de fluxo claro, reentrância, *checkpoints* e necessidade de **confiabilidade observável** (retomar, debugar, auditar).

Pontos fortes: estados explícitos, *checkpoints*, fácil reexecução de trechos, menos "magia negra" do loop do agente.
Limites: requer modelagem do grafo (curva de aprendizado inicial).

AutoGen (conversas multiagente)

O que é: *framework* focado em **multiagentes conversando/colaborando**, com papéis (coder, critic, researcher), ferramentas e regras de parada.
Quando usar: coordenação entre papéis (ex.: *buyer/seller/mediator/coach*), ciclos de crítica-correção e delegação.
Pontos fortes: composição de papéis clara, chats entre agentes, fácil plugar ferramentas/execução de código.
Limites: sem um "diretor"/gestor, pode divergir; exige políticas de moderação/timeout e regras de consenso.

CrewAI (times e tarefas)

O que é: organizar **equipes** de agentes com **papéis, objetivos e tarefas**; bom para *workflows* orientados a entregáveis.
Quando usar: pipelines onde cada agente tem uma responsabilidade clara e há **handoffs** entre etapas (pesquisa → escrita → revisão).
Pontos fortes: ergonomia para papéis e tarefas, foco em produtividade do "time" de agentes.
Limites: menos granular para fluxos complexos de estado do que um grafo explícito (LangGraph).

Regra prática:

- **Fluxo complexo, auditável e reentrante?** LangGraph.
- **Times com papéis e handoffs?** CrewAI.
- **Experimentos de conversação multiagente?** AutoGen.
- **Integrações e RAG gerais?** LangChain (muitas vezes em conjunto com LangGraph).

Ambientes de simulação para negociação

Tipos de ambientes

- **Sintéticos controlados:** jogos de oferta alternada (Rubinstein), *contract-net*, duplos leilões, mercados com utilidades conhecidas (bom para pesquisa e *ablation*).
- **Realistas baseados na Web/UI:** tarefas de compra/venda, comparação de produtos, extração de requisitos, com latência real e páginas ruidosas.
- **Domínios empresariais:** *procurement*, *pricing*, *supply chain*, *SLA negotiation* com múltiplos stakeholders.
- **Cenários sociais:** barganhas multi-atributo, coalizões, mediação, reputação e *opponent modeling*.

Métricas comuns

- **Eficiência de Pareto** e **Equidade** (egalitarian/Nash).
- **Utilidade agregada** e **taxa de acordo** vs ***walk-away* (BATNA)**.
- **Custo de oportunidade** e **tempo para acordo**.
- **Robustez** (ruído, atrasos, falhas de ferramenta).
- **Segurança/ética** (compliance, linguagem apropriada, enviesamento).

Agentes de negociação (arquitetura e táticas)

Papéis e arquiteturas

- **Comprador / Vendedor:** objetivos opostos; utilidades multiatributo (preço, prazo, qualidade, garantias).

- **Mediador / Árbitro:** facilita, propõe compensações, impõe critérios.
- **Coach / Crítico:** observa, dá *feedback* (estilo “Reflexion”), ajusta tática e linguagem.
- **Gestor de diálogo/estado:** mantém **memória de sessão**, histórico de concessões, *deadlines* e anotações do oponente.

Capacidades essenciais

- **Modelagem de preferências** (pesos e trade-offs; utilidade aditiva/multiplicativa).
- **Opponent modeling** (inferir prioridades, ZOPA, ancoragens).
- **Planejamento** (estratégia de concessões, *issue ordering*, pacotes).
- **Tool use** (pesquisa de mercado, cálculo de preço/risco, verificação de *compliance*).
- **Segurança e controles** (listas de proibições, *rate limits*, *guardrails* linguísticos).

Protocolos e táticas

- **Oferta alternada**, **leilões** (inglês, holandês, Vickrey), **duplo leilão**, **contract-net**.
- **Táticas:** ancoragem, concessão gradual, *logrolling* (troca entre atributos), *time-pressure*, *contingent contracts*.
- **Critérios de parada:** *deadline*, utilidade mínima, *no-deal* quando abaixo do BATNA.

Resumo do artigo: Exploring Autonomous Agents through the Lens of Large Language Models: A Review

1. Introdução

Um momento crucial na evolução dos agentes autônomos foi o **advento dos Grandes Modelos de Linguagem (LLMs)**, que marcam um ponto de virada na busca por uma inteligência artificial que espelhe a cognição humana. Os LLMs estão **transformando a inteligência artificial**, permitindo que agentes autônomos realizem diversas tarefas em vários domínios. Eles são proficientes na compreensão e geração de texto semelhante ao humano e têm o potencial de revolucionar setores como o atendimento ao cliente e a saúde. Mesmo em profissões de alto nível, como gerentes financeiros e executivos, uma parcela significativa das atividades é suscetível à automação.
A combinação de LLMs com agentes autônomos oferece uma **fronteira promissora para aprimorar as capacidades de simulação**. Os LLMs, treinados em extensos dados da internet, encapsulam um corpus substancial de conhecimento humano, aproximando-se da inteligência de nível humano. Isso impulsionou um aumento na pesquisa explorando o potencial dos agentes autônomos baseados em LLM.

2.Background sobre Modelos de Linguagem Grande e Agentes Autônomos Baseados em LLM

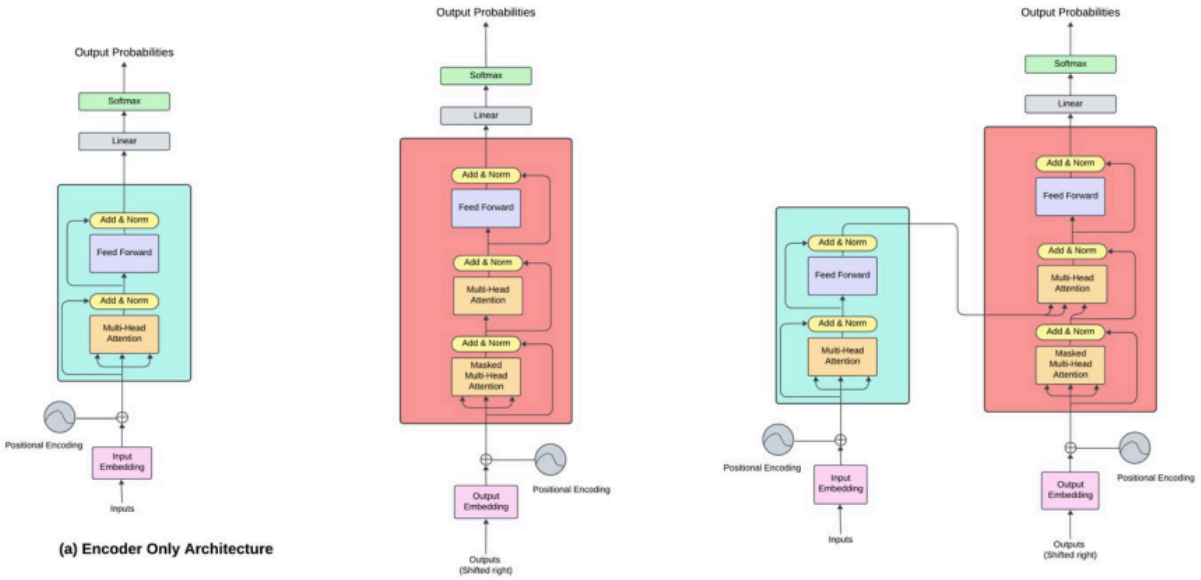
2.1 Arquitetura Transformer

Três tipos distintos de arquiteturas transformer sustentam LLMs e Sua Arquitetura Transformer Subjacente Modelos de Linguagem Grande (LLMs):

Encoder-Decoder: A arquitetura transformer pioneira, onde o **encoder processa a sequência de entrada** para gerar uma representação oculta, que o **decoder então usa para produzir a sequência de saída** desejada. T5 e BART são modelos notáveis que empregam esta arquitetura.

Encoder-only: Esta arquitetura é empregada quando a tarefa **requer apenas a codificação da sequência de entrada**, eliminando a necessidade de um decoder. A sequência de entrada é codificada em uma representação de comprimento fixo, que é então usada como entrada para um classificador ou um regressor para previsão. BERT, DistilBERT e RoBERTa são modelos que utilizam esta arquitetura.

Decoder-only: Esta arquitetura compreende apenas um decoder, que é **treinado para prever o próximo token** em uma sequência dada os tokens precedentes. A principal distinção entre as arquiteturas Decoder-only e Encoder-Decoder é que a primeira carece de um codificador explícito para resumir as informações de entrada. A Figura 1 apresenta a Arquitetura do modelo Transformer.



(a) Encoder Only Architecture

(b) Decoder Only Architecture

(c) Encoder Decoder Architecture

2.2 A Evolução dos Modelos de Linguagem Grande

A confluência de agentes autônomos e Modelos de Linguagem Grande (LLMs) é um domínio em **rápida evolução**, onde cada LLM pode ser visto como um **agente dentro de um ecossistema**, colaborando para realizar uma ampla gama de tarefas. A arquitetura desses agentes autônomos baseados em LLM pode ser dissecada em:

- **Memória**, permitindo que o agente lembre ações passadas e se situe dentro de um ambiente dinâmico;

- **Planejamento**, onde o agente decompõe instruções ou metas de alto nível em tarefas gerenciáveis, orquestra sua execução, avalia os resultados e os refina para cumprir a instrução ou meta da forma mais precisa possível;
- **Ação**, onde o agente demonstra a capacidade de compreender a sintaxe e a semântica de chamadas de software, seleciona as ferramentas de software necessárias para qualquer tarefa e as opera fornecendo parâmetros sintáticos e semanticamente corretos.

3.Construindo Agentes Autônomos com Modelos de Linguagem Grande

3.1 Tudo Sobre Memória

As memórias são tipicamente organizadas em **hierarquias**, de forma análoga aos níveis de memória em sistemas operacionais tradicionais (por exemplo, no modelo MemGPT), visando proporcionar maior continuidade para o raciocínio contextual em tarefas complexas e cenários colaborativos. As hierarquias de memória em uma máquina típica consistem em três níveis: os níveis superiores são mais rápidos, mas escassos, enquanto os inferiores são mais lentos, mas abundantes

Os **blocos de construção fundamentais de todos os LLMs** seguem um padrão onde as camadas são integradas em diferentes permutações. As camadas chave incluem:

- **Camadas Recorrentes**: Preservam informações de entradas anteriores, facilitando o processamento de sequências e dependências temporais (exemplos incluem LSTM e GRU).
- **Camadas Feedforward**: Executam transformações não-lineares nos dados de entrada, extraindo características e padrões.
- **Camadas de Embedding**: Transcrevem palavras ou tokens em vetores densos, encapsulando relações semânticas.
- **Camadas de Atenção**: Concentram-se seletivamente em partes pertinentes da entrada, melhorando o desempenho da tarefa e a compreensão contextual, ao alocar pesos para diferentes tokens com base em sua

significância, permitindo que os LLMs capturem dependências e relações de longo alcance no texto

Novas arquiteturas de LLMs são capazes de **preservar o contexto de diálogo** para conversas coerentes, adaptar respostas com base nas preferências do usuário e rastrear informações factuais para tarefas baseadas em conhecimento. A pesquisa empírica frequentemente se concentra na capacidade dos LLMs de processar e compreender a linguagem natural, transformando textos livres em **embeddings** (representações numéricas de menor dimensão que capturam o contexto linguístico subjacente).
A incorporação de atenção multi-cabeça e arquiteturas profundas de transformers tem sido fundamental para alcançar resultados de ponta em várias tarefas de Processamento de Linguagem Natural (PNL).
No contexto da geração de linguagem, a seleção de uma **estratégia de decodificação** é crucial para determinar a qualidade da saída. Estratégias notáveis incluem:

- **Busca Gulosa (Greedy Search)**: Escolhe o token mais provável em cada ponto, podendo resultar em respostas repetitivas.
- **Amostragem Multinomial (Multinomial Sampling)**: Introduz aleatoriedade para promover diversidade, mas pode gerar texto incoerente.
- **Busca em Feixe (Beam Search)**: Mantém um registro de múltiplas sequências potenciais, equilibrando qualidade e diversidade.
- **Busca Contrastiva (Contrastive Search)**: Visa clareza e consistência, minimizando ambiguidade e repetição. GPT-3 geralmente utiliza amostragem multinomial, T5 e BARD frequentemente empregam busca em feixe, e LLaMA explora a busca contrastiva.

3.2 De LLM com Agentes Autônomos

Ferramentas são os blocos de construção fundamentais que **aumentam** as **capacidades dos LLMs**. Elas são unidades executáveis com funções específicas, permitindo que os LLMs realizem uma variedade de tarefas. Ao amalgamar diferentes ferramentas, os LLMs podem criar um fluxo que atinge um **amplo espectro de objetivos**.

Por exemplo, um agente LLM tem acesso a ferramentas que podem realizar ações além da geração de texto, como conduzir uma pesquisa na web, executar código, realizar uma consulta em banco de dados ou fazer cálculos matemáticos.
O **conceito fundamental de agentes** no **LangChain** é empregar um LLM para selecionar uma sequência de ações a serem realizadas. Dependendo da entrada do usuário, o agente pode então decidir qual, se houver, dessas ferramentas invocar.
Em essência, um Agente no LangChain **é igual a Ferramentas mais Memória**. Ao receber uma solicitação, os Agentes utilizam um LLM para decidir qual ação realizar. Após a conclusão de uma ação, o agente atualiza sua memória, o que ajuda a manter o contexto da conversa. Essa abordagem permite que o **LangChain gerencie interações mais complexas e estruturadas**, mantendo o contexto ou a sequência entre diferentes prompts e respostas.
O Auto-GPT é um agente autônomo que decompõe uma grande tarefa em várias subtarefas sem exigir input do usuário. Ele pode ficar preso em loops lógicos e "buracos de coelho", limitando assim sua capacidade de resolução de problemas em certos cenários. Por outro lado, o **LangChain não é um agente autônomo**, mas sim uma **biblioteca de LLM que facilita o desenvolvimento** de uma ampla gama de aplicações sobre LLMs de última geração.

3.3 A Arte de Raciocinar e Agir

Alguns LLMs pré-treinados, como o GPT-4, possuem **capacidades de raciocínio significativas**, permitindo-lhes desconstruir problemas complexos em etapas mais simples e fornecer soluções, ações e avaliações em cada estágio. Contudo, como sistemas fechados, os LLMs têm a **limitação** de **não poder acessar dados recentes** ou **conhecimento de domínio específico**, o que pode levar a erros ou **"alucinações"** (geração de respostas incorretas ou sem sentido). O "prompting" direto nem sempre resulta em respostas bem elaboradas, e os LLMs são mais propensos a alucinar se uma resposta imediata for solicitada.

Para **lidar com as alucinações** e **melhorar o raciocínio**, foram desenvolvidos módulos como:

- **Raciocínio de Caminho Único (SPR)**: Incentiva etapas de raciocínio intermediárias.
    - **Chain of Thought (CoT)**: Gera uma série de passos de raciocínio para dissecar problemas de várias etapas em estágios intermediários, o que é eficaz para problemas computacionais complexos.

    - **HuggingGPT**: Um agente habilitado por LLM que utiliza LLMs para orquestrar o planejamento de tarefas, selecionar modelos de IA e executar subtarefas, resumindo a resposta.
- **Raciocínio de Múltiplos Caminhos (MPR)**: Facilita múltiplas interações automáticas entre usuários e LLMs, usando respostas geradas anteriormente como dicas para guiar progressivamente em direção às respostas corretas. Pode ser integrado com técnicas de ponta para melhorar o desempenho.
    - **Self Consistent Chain of Thought (CoT-SC)**: Aumenta a precisão das respostas dos LLMs, assumindo que a resposta correta reside no modelo e é retornada na maioria das vezes quando a mesma pergunta é repetida.
    - **Tree of Thoughts (ToT)**: Permite que os LLMs naveguem por múltiplos caminhos de raciocínio ao resolver problemas, através da decomposição de pensamentos, geração de pensamentos e avaliação de estados.
    - **Graph of Thoughts (GoT)**: Um framework que modela a informação gerada pelo LLM como um grafo arbitrário, onde as "unidades de pensamento" do LLM servem como vértices e as arestas correspondem a dependências.
- **Feedback Ambiental e Humano**:
    - **Feedback Ambiental**: Pode aprimorar significativamente a eficácia dos agentes LLM. O framework **"ReAct"** (Reason for Future, Act for Now) permite que o agente contemple as implicações de longo prazo das ações e refine sua estratégia com base no feedback do ambiente externo.
    - **Reflexion** é outro framework onde o agente assimila o feedback ambiental para aprimorar seu modelo de mundo interno e prever os resultados de suas ações.
    - **Feedback Humano**: É crucial para alinhar as respostas dos agentes LLM com as preferências da aplicação, fornecendo informações vitais que podem não ser obtidas diretamente de recompensas ambientais.
        - **Reinforcement Learning from Human Feedback (RLHF)**: Treina um "modelo de recompensa" usando feedback humano direto para otimizar o desempenho do agente via aprendizado por reforço, especialmente para tarefas com objetivos complexos ou mal definidos.

        - **Direct Preference Optimization (DPO)**: Uma alternativa ao RLHF que simplifica o processo, eliminando a necessidade de ajustar um modelo de recompensa, amostragem do LLM durante o fine-tuning ou ajuste extensivo de hiperparâmetros. O DPO resolve diretamente um problema de classificação sobre o texto gerado pelo LLM usando preferências binárias humanas.

- **Geração Aumentada por Recuperação (RAG)**: O RAG emergiu como um paradigma favorecido para permitir que os LLMs acessem dados externos, servindo como um mecanismo de ancoragem para **combater alucinações**.
    - Os modelos RAG combinam memória paramétrica (um modelo seq2seq pré-treinado) e memória não-paramétrica (um índice vetorial denso de uma base de conhecimento, como a Wikipedia).
    - Os LLMs **recuperam documentos contextuais** de um banco de dados para **aumentar a precisão** de suas respostas. Frameworks como LangChain e LlamaIndex atuam como orquestradores, conectando LLMs a ferramentas, bancos de dados e memórias.
    - Técnicas como **"multi-query retriever"** e **HyDE** podem ser empregadas para **refrasear ou expandir instruções** do usuário e melhorar o desempenho da recuperação.
    - Para manter o contexto em conversas longas, os LLMs incorporam o **histórico** na sua janela de contexto, que pode ser aprimorada com **sumarização** ou um **"vector store" + RAG**.
    - A avaliação do RAG abrange as fases de **recuperação de documentos** (precisão e recall do contexto) e **geração** (fidelidade e relevância da resposta), com ferramentas como Ragas e DeepEval simplificando esse processo.
    - Técnicas RAG avançadas incluem Naive RAG, Advanced RAG e Modular RAG.
    - Agentes, como o LLM Agent da Pinecone, podem utilizar ferramentas como calculadoras, pesquisa ou execução de código Python, e consultar bancos de dados SQL, com o LangChain oferecendo métodos para construir consultas e conectar-se a diferentes dialetos SQL

3.4 O prompt é tudo o que você precisa

Com Prompts Refinados, o Desempenho do Agente LLM pode ser **aprimorado** adquirindo **competências linguísticas de maneira não supervisionada**, prevendo palavras mascaradas dentro de frases. A eficácia desses LLMs depende dos prompts ou entradas que recebem. A **arte da engenharia de prompts** envolve a criação de prompts ou consultas de alta qualidade que são sob medida para a tarefa em questão. Ao fornecer ao **modelo prompts claros e pertinentes**, ele pode gerar **respostas precisas e relevantes** com mais facilidade. Este processo de refinar e modificar o prompt para elicitar uma saída "superior" ou mais desejada do modelo é uma ilustração rudimentar da engenharia de prompts.

Múltiplas Técnicas para Aumentar o Prompting para um Modelo incluem:

- **Automatic Reasoning and Tool Use (ART)**: Melhora significativamente o prompting few-shot e o CoT automático em tarefas não vistas.
- **Automatic Prompt Engineer (APE)**: Gera autonomamente instruções para uma tarefa, formulando e executando vários candidatos de instrução e selecionando o mais adequado.
- **Active-Prompt**: Envolve o design de prompts que ativamente direcionam o modelo para gerar a saída desejada.
- **Directional Stimulus Prompting**: Introduz um componente de "estímulo direcional" no prompt para fornecer orientação e controle mais refinados sobre os LLMs.
- **Program-Aided Language Models (PAL)**: São capazes de escrever código para resolver uma questão e despacham esse código para um ambiente de execução programático para obter o resultado.
- **Multimodal CoT**: Integra texto e visão em um framework de duas etapas: geração de rationale baseados em informações multimodais, seguida pela inferência da resposta usando esses rationales informativos.
- **Graph Prompting**: Utiliza dados de grafo, que servem como repositórios de conhecimento estruturados ao modelar explicitamente a interação entre entidades.

Prompt Tuning e Eficiência:

- **Prompt tuning** é um método de ajuste eficiente em termos de parâmetros (PETuning) para aproveitar modelos pré-treinados (PTMs). Ele simplesmente anexa um **"soft prompt"** à entrada e otimiza apenas esse prompt para adaptar os PTMs a tarefas específicas. A importância dessa técnica reside em sua eficiência, pois treinar um LLM do zero é proibitivamente caro. Ao capitalizar o conhecimento já encapsulado no modelo pré-treinado, pode-se alcançar alto desempenho com consideravelmente menos dados e computação.
- **Prompt e Prefix Token Tuning**: Envolve a otimização de prompts contínuos para a geração. No **prefix-tuning**, o contexto (embeddings de prompt) deve ser reintroduzido a cada requisição em conversas, pois os prompts não alteram o modelo. É uma alternativa leve ao fine-tuning para tarefas de geração de linguagem natural, mantendo os parâmetros do modelo de linguagem congelados, mas otimizando um pequeno vetor contínuo específico da tarefa (o "prefixo"). Prefix-tuning, aprendendo apenas 0,1% dos parâmetros, alcança desempenho comparável ao full data setting e supera o fine-tuning em configurações de poucos dados, além de extrapolar melhor para exemplos com tópicos não vistos durante o treinamento

Aprendizagem no Contexto (In-Context Learning): É uma estratégia potente que envolve fornecer ao modelo um contexto ou situação, e o modelo aprende a gerar respostas com base nesse contexto. Isso é útil para cenários onde o modelo precisa se adaptar a novas situações ou tarefas, como em conversas

Raciocínio de Senso Comum: O prompting é utilizado em tarefas de raciocínio de senso comum, onde o modelo deve fazer inferências que são evidentes para humanos, mas podem não ser explicitamente declaradas. A técnica de "generated knowledge prompting" envolve gerar conhecimento de um modelo de linguagem e fornecê-lo como entrada adicional para responder a perguntas, melhorando o desempenho.

Geração de Dados de Prompt Construtivos e Diversidade: A geração de dados de prompt construtivos envolve a criação meticulosa de prompts para guiar modelos de IA na geração de dados sintéticos para treinamento e teste, especialmente quando dados do mundo real são escassos. Para garantir a diversidade nos datasets gerados, são utilizadas técnicas como bootstrapping, incorporação de diversas fontes de dados e **aumento de dados (data augmentation)**, que envolve

pequenas alterações em dados existentes para fabricar exemplos adicionais de treinamento

4.Avaliando Agentes Autônomos

Avaliar esses agentes é vital para garantir que tomem **decisões apropriadas** e **executem** tarefas de forma **eficaz**.

Além disso, a avaliação é crucial para **identificar e corrigir** potenciais **problemas ou limitações** no desempenho do agente.

Ela permite que os desenvolvedores monitorem o desempenho do agente, identifiquem áreas para melhoria e implementem ajustes necessários para aumentar sua eficácia.

4.1 Estruturas de Avaliação Tradicionais

Uma abordagem comum é usar **tarefas ou conjuntos de dados** de referência para avaliar o desempenho do agente. Esses benchmarks fornecem uma medida padrão de desempenho que pode ser usada para comparar diferentes agentes ou algoritmos.

Outra abordagem prevalente é usar **ambientes de simulação** para avaliar o desempenho do agente. Esses ambientes permitem que os desenvolvedores testem o desempenho do agente sob várias condições e cenários.

No entanto, essas estruturas tradicionais muitas vezes **ignoram os desafios** únicos de avaliar agentes autônomos.

4.2 AgentBench: Uma Estrutura de Avaliação Abrangente

AgentBench é um benchmark em evolução, multidimensional, atualmente composto por **oito ambientes únicos**. Ele é projetado para avaliar as capacidades de raciocínio e tomada de decisão de Modelos de Linguagem de Grande Escala em um ambiente de geração de múltiplas interações e aberto. Esses ambientes **cobrem um**

amplo espectro de domínios, cada um apresentando desafios e requisitos distintos para os LLMs.

Cinco desses ambientes são recém-criados, especificamente, o **sistema operacional**, **banco de dados**, **gráfico de conhecimento**, **jogo de cartas digital** e **quebra-cabeças de pensamento lateral.** Cada ambiente representa uma faceta diferente das tarefas do mundo real que um LLM pode encontrar. Por exemplo, o **ambiente do SO avalia a capacidade do LLM de interagir com um sistema operacional simulado**, enquanto o **ambiente do jogo de cartas digital avalia as habilidades de tomada de decisão estratégica** do LLM em um jogo de cartas.

Além desses novos ambientes, o AgentBench também incorpora três ambientes recompilados de conjuntos de dados publicados: **HouseHolding (HH)**, **Web Shopping (WS)** e **Web Browsing (WB)**. Esses ambientes oferecem uma variedade diversificada de desafios, que vão desde gerenciar tarefas domésticas no ambiente HH até navegar em plataformas de compras online no ambiente WS.

4.2.2 Melhorando o Desempenho do Agente

Os autores propõem que o **treinamento em código** e **dados de alinhamento de múltiplas interações de alta qualidade** poderia melhorar o desempenho do agente. Ao treinar em código, os LLMs podem obter uma melhor **compreensão da estrutura e lógica das linguagens de programação**, o que é particularmente benéfico para ambientes como OS e DB. Por outro lado, dados de alinhamento de múltiplas interações de alta qualidade podem ajudar os LLMs a **entender melhor o contexto de uma conversa e tomar decisões mais apropriadas.**

Além dessas propostas, os autores também lançaram conjuntos de dados, ambientes e um pacote de avaliação integrado para o AgentBench. Esses recursos podem ser utilizados por outros pesquisadores para avaliar seus próprios LLMs e contribuir para o desenvolvimento contínuo de LLMs como agentes.

4.2.3 Comparação com Estruturas de Avaliação Tradicionais

Essa abordagem difere das estruturas de avaliação tradicionais de várias maneiras. Estruturas de **avaliação tradicionais** frequentemente se **concentram em avaliar o desempenho do agente em ambientes isolados.** Essas estruturas geralmente envolvem testar a capacidade do agente de completar tarefas específicas ou

alcançar certos objetivos dentro de um ambiente controlado. No entanto, essas estruturas tradicionais muitas vezes **falham em considerar os desafios únicos de avaliar agentes autônomos.** Por exemplo, elas podem não levar em conta adequadamente os ambientes dinâmicos nos quais esses agentes operam, ou as interações complexas entre agentes e seu ambiente. O **AgentBench,** por outro lado, fornece uma **avaliação mais abrangente da capacidade dos LLMs** de operar como agentes autônomos em vários cenários. Ele abrange um espectro diversificado de diferentes ambientes, proporcionando um cenário mais realista e desafiador para avaliar os agentes. Isso o torna uma ferramenta mais eficaz para avaliar o desempenho dos LLMs como agentes e identificar áreas para melhoria.

4.3 WebArena: Um Novo Ambiente para Agentes Autônomos

WebArena é um ambiente **altamente realista** e **reproduzível** projetado para agentes guiados por linguagem.

O ambiente abrange websites totalmente funcionais de quatro domínios comuns: **e-commerce**, **discussões em fóruns sociais**, **desenvolvimento colaborativo de software** e **gerenciamento de conteúdo**. Esses domínios representam um amplo espectro de tarefas que os humanos realizam regularmente na internet, proporcionando assim um cenário diversificado e desafiador para a avaliação de agentes.

Além dos websites, o ambiente é enriquecido com **ferramentas** (por exemplo, um mapa) e **bases de conhecimento externas** (por exemplo, manuais do usuário) para promover a resolução de tarefas de forma semelhante à humana. Este recurso é projetado para expandir os limites do que os LLMs podem alcançar, indo além de tarefas simples de perguntas e respostas ou geração de texto para tarefas de resolução de problemas mais complexas.

4.3.2 Comparação com Abordagens Tradicionais

A abordagem adotada pelo WebArena diverge significativamente das abordagens tradicionais para desenvolvimento e avaliação de agentes.

As abordagens tradicionais muitas vezes envolvem a criação e teste de agentes em ambientes sintéticos simplificados. Embora esses ambientes sejam úteis para o desenvolvimento e teste iniciais, eles frequentemente falham em representar com

precisão a complexidade e diversidade dos cenários do mundo real. Em contraste, o WebArena fornece um ambiente altamente realista e reproduzível para o desenvolvimento e teste de agentes. Essa abordagem permite uma avaliação mais abrangente do desempenho e das capacidades dos agentes. Também fornece uma plataforma para identificar e abordar os desafios e limitações dos LLMs atuais, abrindo caminho para o desenvolvimento de agentes mais robustos e capazes.

4.5 Avaliação Subjetiva em Agentes Autônomos Baseados em LLM

Esse processo envolve medir o desempenho desses agentes por meio do **julgamento humano**, oferecendo **insights valiosos sobre sua eficácia, usabilidade e qualidade geral**. Um método prevalente empregado na avaliação subjetiva de LLMs é a **anotação humana**. Esse processo envolve anotadores humanos revisando e avaliando as saídas dos LLMs com base em vários critérios, como **relevância**, **coerência** e **fluência**. Esse método pode resultar em uma avaliação detalhada do desempenho do LLM e identificar áreas que podem exigir melhorias. No entanto, a anotação humana vem com seu próprio conjunto de desafios. Pode ser **demorada** e **cara**, particularmente para avaliações em larga escala. Além disso, pode estar **sujeita a viés**, uma vez que diferentes anotadores podem interpretar os critérios de avaliação de maneira diferente.

O **Teste de Turing**, nomeado após o matemático e cientista da computação britânico Alan Turing, é outro **método utilizado na avaliação subjetiva** de LLMs. O Teste de Turing é projetado para avaliar a capacidade de uma máquina de **exibir comportamento inteligente equivalente** ou **indistinguível do de um humano**. No contexto de LLMs, o Teste de Turing geralmente envolve um juiz humano engajado em uma conversa com o LLM sem o conhecimento de que está interagindo com uma máquina. Se o juiz não consegue distinguir de forma confiável o LLM de um interlocutor humano, o LLM é considerado como tendo passado no Teste de Turing.

Estudos com usuários podem ser realizados para avaliar a **usabilidade** e a **satisfação** do usuário com LLMs em cenários do mundo real. Esses estudos podem oferecer feedback valioso sobre o desempenho do LLM e identificar áreas para melhoria. Outro método envolve o uso de **avaliações de especialistas**, onde especialistas na área avaliam o desempenho do LLM com base em seu julgamento profissional. Este método pode fornecer uma avaliação mais sutil das capacidades do LLM, especialmente em domínios especializados.

5.Restrições da Implementação

5.1 Multimodalidade: Uma Espada de Dois Gumes para Agentes Autônomos Baseados em LLM

A integração de diversos tipos de dados, como **texto, imagens e áudio,** representa um desafio formidável podendo **sobrecarregar o desempenho desses agentes**. Cada tipo de dado requer etapas de **pré-processamento distintas**, e a amalgamação desses tipos de dados é **computacionalmente exigente**. Além disso, a heterogeneidade dos tipos de dados gera inconsistências na compreensão do agente. Por exemplo, um agente pode interpretar dados textuais de uma maneira que diverge de sua interpretação de dados visuais, potencialmente levando a conflitos na tomada de decisões. Essa limitação pode tornar os agentes menos aptos a gerenciar cenários do mundo real, que frequentemente envolvem entradas multimodais intrincadas.

Aumentar a eficiência computacional é primordial. Isso pode ser alcançado por meio do uso de **algoritmos otimizados** e **técnicas de processamento de dados mais eficientes**. Além disso, o desenvolvimento de conjuntos de dados multimodais mais abrangentes e diversos para treinamento pode fortalecer a capacidade do agente de entender e interpretar dados multimodais.

5.2 Alinhamento Humano em Agentes Autônomos Baseados em LLM

O alinhamento de agentes autônomos baseados em Modelos de Linguagem Grande (LLM) com valores, expectativas e instruções humanas é um aspecto crítico de seu design e operação. Um problema comum é a **má interpretação das instruções humanas.**

Variações sutis no prompt de entrada podem levar a **desvios significativos** na saída, resultando em ações que **divergem das intenções do usuário**. Além disso, agentes baseados em LLM podem inadvertidamente **gerar conteúdo** que **reflete preconceitos** presentes nas vastas quantidades de dados da web em que são treinados e podem se **manifestar** no **comportamento** do agente. Além disso, agentes baseados em LLM podem **produzir informações** que são factualmente **incorretas** ou **semanticamente sem sentido**.

Vários métodos podem ser implementados para lidar com as questões relacionadas ao alinhamento com os valores humanos em agentes autônomos baseados em LLM. **Melhorar a qualidade dos dados de treinamento** pode ajudar a mitigar mal-entendidos e preconceitos. Isso pode envolver o uso de conjuntos de dados mais diversos e representativos e a **implementação de técnicas para filtrar informações tendenciosas ou incorretas**. Incorporar **feedback dos usuários** geralmente ajuda a melhorar o alinhamento. O **monitoramento** e a **avaliação contínua** do desempenho do agente podem ajudar a identificar e corrigir problemas de desalinhamento.

5.3 O Enigma das Alucinações

Alucinações em Modelos de Linguagem Grande (LLMs) são caracterizadas pela criação de **conteúdo** que **carece de fundamentação** em seus dados de treinamento. A manifestação de alucinações pode levar à geração de informações que são **factualmente incorretas** ou **desprovidas de lógica**. A presença de alucinações também pode introduzir inconsistências nas saídas do agente. Por exemplo, o agente pode **criar conteúdo que contradiz informações que gerou anteriormente** ou que **contradiz fatos estabelecidos.** Além disso, alucinações podem levar o agente a gerar conteúdo que exibe **preconceito ou é inadequado.**

Existem inúmeras abordagens que podem ser utilizadas para enfrentar as dificuldades relacionadas às alucinações em agentes autônomos baseados em LLM. **Melhorar a qualidade dos dados de treinamento pode ajudar a mitigar alucinações**. Isso pode envolver o uso de conjuntos de **dados mais diversos e representativos**, bem como a implementação de **técnicas para filtrar informações tendenciosas ou incorretas.** Isso pode envolver o uso de **métricas** que medem especificamente a **taxa de alucinação do agente**, bem como a implementação de sistemas para revisão e ajuste regulares do comportamento do agente.

5.4 As Complexidades dos Ecossistemas de Agentes

O ecossistema de agentes, que se refere ao **ambiente em que agentes autônomos baseados em Modelos de Linguagem Grande (LLMs) operam**, pode impactar significativamente o desempenho desses agentes. Cada agente dentro do ecossistema necessita de **etapas de processamento distintas**, e a interação desses agentes díspares pode ser computacionalmente exigente. Além disso, a

heterogeneidade dos agentes dentro do ecossistema pode **induzir inconsistências no desempenho geral** do sistema. Por exemplo, um agente pode interpretar dados de maneira divergente de outro agente, levando a potenciais conflitos na tomada de decisões.

As complexidades do ecossistema de agentes podem levar a problemas de desempenho. Se um agente não consegue navegar efetivamente pelo ecossistema, pode realizar ações não úteis ou contraproducentes. Isso pode levar a uma queda na eficácia geral do agente. Se os usuários perceberem o agente como não confiável devido à sua incapacidade de interagir consistentemente com outros agentes, eles podem estar menos inclinados a usá-lo.

No design **meticuloso e na padronização de agentes** autônomos baseados em LLM, **protocolos de comunicação padronizados** devem ser implementados para garantir uma troca de informações suave e minimizar interpretações errôneas. Logo, uma estrutura deve ser mantida para **supervisão e controle humano** sobre o ecossistema de agentes, permitindo intervenção em situações críticas e garantindo a adesão a diretrizes éticas.

.

Resumo do artigo: Appraising Success of LLM-based Dialogue Agents

(Avaliando o Sucesso de Agentes de Diálogo Baseados em LLM)

Modelos de linguagem grandes (LLM) podem ser solicitados a exibir diálogos semelhantes aos humanos. No entanto, **agentes de diálogo baseados em LLM** podem, em muitos sentidos, imitar um ser humano, bem como diferenciar-se do mesmo. As capacidades desses modelos abalaram o mundo dos negócios, bem como o mundo da pesquisa. No entanto, o sucesso de tais modelos em qualquer domínio depende de uma série de considerações.

1. INTRODUÇÃO

Modelos de linguagem grandes, popularmente conhecidos como LLMs, são uma tentativa de alcançar um sistema de inteligência geral artificial (AGI). Eles exibem **capacidades variadas** além da **geração de texto**, como **conduzir conversas, guiar tarefas** e **raciocinar** . Um LLM adequadamente treinado e testado pode ser facilmente substituído para imitar a linguagem humana de forma convincente, dado uma sequência de palavras como token de entrada ou contexto .

Essa capacidade permite inúmeras aplicações de LLMs como agentes de diálogo em aplicações voltadas para o usuário, como **suporte ao cliente**, **agentes conversacionais**, **assistentes pessoais digitais**, etc.

Um agente de diálogo baseado em LLM é uma extensão de um agente baseado em IA com três partes conceituais principais: **cérebro**, **percepção** e **ação**.

O **cérebro** é o fundamental de um agente baseado em IA, pois **armazena os dados, informações** e **conhecimentos vitais**, mas também realiza as tarefas essenciais de **processamento de informações**, **tomada de decisões, raciocínio** e **planejamento**.

A **percepção** desempenha um papel comparável ao dos órgãos sensoriais humanos. Ela expande principalmente o espaço perceptual do agente de um espaço baseado em texto para um **espaço multimodal** que pode incluir várias

modalidades sensoriais, como **texto**, **som**, **visuais**, **toque**, **cheiro**, etc. Assim, o agente é capaz de perceber melhor as informações do ambiente externo.

A **ação** é o resultado do agente. Espera-se que o agente **processe** a saída textual, tome **ações** embutidas e **aplique ferramentas** para melhor responder às mudanças ambientais e fornecer feedback .

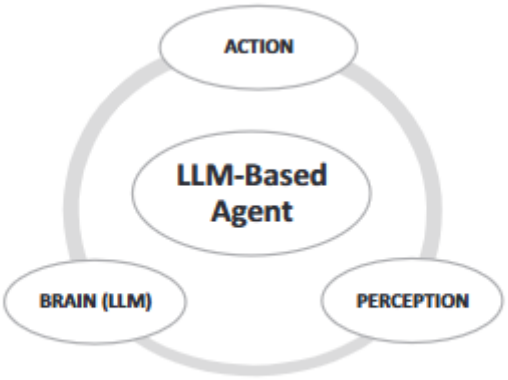


Fig. 1. Conceptual Parts of an LLM based Agent

2. TREINANDO OS AGENTES DE DIÁLOGO BASEADOS EM LLM

Interfaces de conversação exigem **flexibilidade**, **precisão** e **memória contextual**, juntamente com **raciocínio** em cadeia de pensamento. Simplificando, um agente de diálogo baseado em LLM é destinado a responder perguntas como os humanos . As perguntas são compostas por tokens que podem ser uma sequência de palavras, sinais de pontuação, emojis e assim por diante. Espera-se que o agente adivinhe o token que deve vir a seguir. Essa sequência é feita a partir do vasto corpus de dados públicos disponíveis na Internet.

Treinar o agente de diálogo baseado em LLM envolve um **conjunto de dados constituído por pares** (INSTRUÇÃO, SAÍDA) em um modo supervisionado, alterando assim a lacuna entre o objetivo de previsão da próxima palavra do agente de diálogo baseado em LLM. A **INSTRUÇÃO**, neste caso, é a **entrada** e a **SAÍDA** é o **token esperado.** O pipeline básico envolveria a construção do conjunto de dados de instruções, seguido por algum mecanismo de ajuste fino do conjunto de dados. Uma ampla gama de técnicas foi aplicada para alcançar esse ajuste fino dos

agentes para melhorar suas capacidades e controlabilidade. Algumas dessas estão listadas na Tabela I:

TABLE I. TECHNIQUES FOR SPECIALIZING DIALOGUE AGENTS

S. No.	LLM Dialogue Agents		
	Name	***Characteristic***	***Limitation***
1	Instruction Tuning (IT) [17]	Work on human generated instruction and sample response pairs.Ex: InstructGPT	Numerous data samples required for training may be unavailable or costly to generate. It may only capture surface-level patterns in the output instead of comprehending the task.
2	Self-Talking LLM[18]	To bootstrap task-specific data LLMs themselves.	Accuracy is good but dialog success is questionable
3	Goal Oriented Dialogue (GOD)[13]	AI chat agent needs to proactively pose questions and guide users towards specific goals or task completion.	Tradeoff between fluent language generation and task-specific control exists.

3. RESUMO DO DESIGN DOS AGENTES DE DIÁLOGO LLM

Os LLMs geralmente se referem a modelos baseados em transformadores com bilhões de parâmetros treinados em trilhões de tokens . Os **transformadores** capacitam redes neurais a **processar grandes blocos de texto** simultaneamente, a fim de estabelecer uma **relação** mais forte entre **palavras** e seu **contexto de aparecimento.** O modelo de linguagem subjacente de tais agentes de diálogo pode ser representado matematicamente como uma distribuição de probabilidade condicional expressa como :

$$P(w_{n+1}|w_1 \dots w_n)$$

where,

$w_1 \dots w_n$ = is a sequence of tokens (the context)

w_{n+1} = predicted next token

No entanto, para qualquer agente de diálogo, um grande desafio é a **gestão do estado do diálogo** . O sistema deve ser capaz de **decidir** o **próximo estado do diálogo** e a **resposta do sistema ao usuário**. Os atuais agentes LLM exibem um alto nível de autonomia. Várias técnicas foram aplicadas para responder à entrada do usuário, escolhendo o próximo estado do diálogo.

TABLE II. TECHNIQUES FOR DECIDING THE NEXT DIALOG STATE

S. No.	LLM Decision Making			
	Name	***Characteristic***	***Limitation***	***Use Case***
1	Chain-of-Thought [7]	Decompose user question into subtasks. User input thus directs chain creation on the fly basis.	Has shown performance gains with models of approx.. 100 billion parameters. However, smaller models tend to result in illogical chains leading to lower accuracy	Symbolic Reasoning, Arithmetic Reasoning, Commonsense Reasoning
2	Prompt Chaining [18]	Generates a predetermined chain for an anticipated use-case.	Success depends heavily on initial prompt, context maintenance in long chains is a challenge, ethical questions in sensitive areas cannot be ignored.	Quantum computing, Self-Improving AI systems, Personalized Healthcare/ Academic Predictions
3	Prompt Pipelines [22]	Extend prompt templates by automatically injecting contextual reference data for each prompt.	These models can generate coherent text, as well as they may not provide clear explanations.	Content Generation, Dialogue Systems, Virtual Assistants etc.
4	QuickReply Intents [23]	Generate a user intent taxonomy and apply it to do log analysis, etc.	Chance of creating a feedback loop without clear evaluation.	Taxonomy generation from logs, Search and recommender systems

Independentemente da técnica de tomada de decisão empregada, todos os agentes de tomada de decisão baseados em LLM **requerem pré-treinamento em larga escala** em corpora de texto massivos, juntamente com **aprendizado por reforço a partir do feedback humano**. Seu ciclo de **prompt interativo** é chave para facilitar **conversas naturais entre usuários e agentes LLM.**

Agentes de diálogo baseados em LLMs aproveitam as capacidades desses modelos para gerar respostas semelhantes às humanas em uma conversa. Um design genérico para um agente de diálogo baseado em LLM pode conter os seguintes componentes:

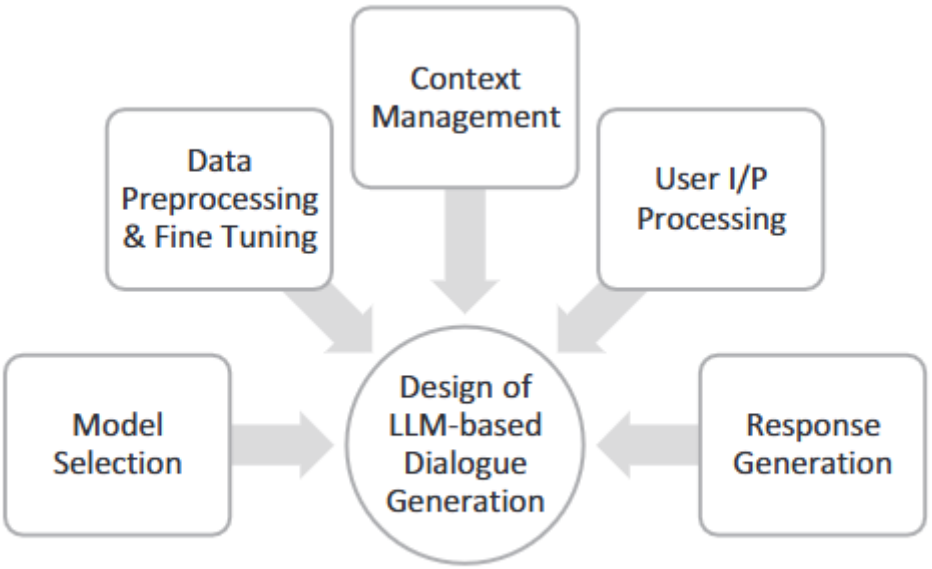


Fig. 2. Generic Design of LLM-Based Dialogue Agent

a) Seleção de Modelo: Um modelo LLM pré-treinado adequado, como o GPT-3, pode ser escolhido ou criado conforme os requisitos. Considere fatores como tamanho do modelo, recursos computacionais e o caso de uso específico.

b) Pré-processamento de Dados e Ajuste Fino: Os dados de treinamento são limpos para remover ruídos e inconsistências e trazidos em um formato que o LLM pode entender. Isso é essencial para permitir que o modelo se adapte à linguagem e comportamento específicos do domínio.

c) Gestão de Contexto: O contexto da conversa deve ser controlado nos LLMs. Eles normalmente geram respostas com base no contexto anterior durante uma conversa.

d) Processamento de Entrada do Usuário: Um mecanismo apropriado para processar as entradas do usuário é necessário para tokenizar e codificar o texto de entrada para o LLM. Informações relevantes das consultas dos usuários devem ser extraídas para obter contexto relevante para gerar respostas.

e) Geração de Respostas: O LLM deve gerar respostas para a entrada do usuário. Diferentes estratégias para a geração de respostas podem ser experimentadas para melhorar a diversidade e a qualidade das respostas.

O design é apenas uma estrutura básica para qualquer agente de diálogo baseado em LLM. Combinar cuidadosamente esses componentes centrais pode permitir que os usuários solicitem de forma eficiente máquinas baseadas em LLM antropomórficas.

4. RISCOS NÃO INTENCIONAIS DOS AGENTES DE DIÁLOGO BASEADOS EM LLM

Superestimar os LLMs pode levar a aplicações não confiáveis. Como duas faces de uma moeda, vale a pena investigar os riscos subjacentes de ter um agente de diálogo baseado em LLM como um dos personagens pré-definidos em um jogo de interpretação com um participante humano.

- **Escassez de Dados:** O diálogo é uma tarefa complexa. Espera-se que qualquer agente de diálogo seja **treinado** em **gramática**, **sintaxe**, **raciocínio de diálogo,** bem como **geração de linguagem**. Dominar essas habilidades exige uma grande quantidade de dados específicos do domínio. Modelos pré-treinados podem ajudar a reduzir esse desafio.
- **Viés do Modelo:** Modelos de linguagem grandes frequentemente ecoam os **preconceitos** dos dados nos quais são treinados. Além disso, quaisquer ferramentas externas utilizadas também podem introduzir viés. Abordar o viés de forma eficaz levaria a modelos de linguagem imparciais.
- **Perigos Sociais:** Os LLMs têm um potencial negativo de enganar usuários humanos, por exemplo, na forma de ataques de phishing personalizados. Treinar agentes que não dependem mais de dados de treinamento gerados por humanos poderia, portanto, mitigar a criação de modelos de diálogo enganosos por usuários maliciosos.
- **Alucinação:** Alguns indivíduos argumentam que os LLMs não podem ser utilizados em cenários sérios, como domínios de saúde ou legais, devido a alucinações. Além disso, reconhecemos que há algumas respostas prejudiciais dos LLMs.
- **Desajuste entre o objetivo de treinamento e o objetivo do usuário:** Agentes de diálogo baseados em LLM devem ser treinados para minimizar o

erro de previsão de palavras contextuais em um grande corpus. No entanto, o usuário esperaria que o modelo cumprisse suas instruções de forma segura, isso também é exigido em termos de redigir instruções de alta qualidade para mapear diretamente os comportamentos-alvo desejados.

- **Geração de Código Defeituoso:** LLMs impulsionados por viés são suscetíveis a gerar código defeituoso, reduzindo a precisão geral do sistema.

5. CONCLUSÃO E TRABALHO FUTURO

Criar um agente de diálogo baseado em LLM bem-sucedido não é apenas obter o corpus de dados de entrada e o modelo corretos. Existem muitas outras complexidades envolvidas. As **respostas geradas devem ser analisadas** para garantir coerência, correção e adequação à situação específica. O agente de diálogo deve ser **amigável**, com **comunicação clara** e **ciclos de feedback apropriados.**

Mecanismos adequados devem ser implementados para **lidar com erros** ou **entradas mal interpretadas** de forma elegante. O **desempenho** usando **métricas relevantes,** com base em **feedback** e **resultados de testes,** deve ser avaliado para melhorias.

Além da consideração acima, é importante notar que o **sucesso** de **agentes de diálogo específicos de domínio** é governado pela **disponibilidade de dados específicos de domínio,** o que pode ser uma preocupação. Para superar isso, trabalhe na adaptação de domínio zero-shot no futuro curso de ação do agente de diálogo baseado em LLM. Isso garantirá a aplicação de um agente de diálogo a qualquer domínio novo sem a disponibilidade de dados de treinamento específicos de domínio.

Começamos a explorar diferentes áreas dos Agentes de IA Baseados em LLM: Inicialmente por Agentes de IA para teoria da mente:

Resumo do artigo: Evaluating and Enhancing LLMs Agent based on Theory of Mind in Guandan: A Multi-Player Cooperative Game under Imperfect Information

(Avaliando e Melhorando Agentes LLMs Baseados na Teoria da Mente em Guandan: Um Jogo Cooperativo Multijogador sob Informação Imperfeita)

O estudo investiga a aplicabilidade e o desempenho de Large Language Models (LLMs) em ambientes de jogos complexos, que exigem **colaboração multiagente sob informação imperfeita**, especificamente no jogo de cartas chinês **Guandan**.

Contexto e Desafios

Os avanços recentes dos LLMs demonstram seu potencial, mas enfrentam desafios cruciais em cenários do mundo real:
1. **Informação Imperfeita:** Muitos cenários (como pôquer ou diplomacia) envolvem informação imperfeita, o que contrasta com a suposição de informação perfeita frequentemente feita pelos modelos.
2. **Colaboração:** Muitos ambientes, como Guandan, exigem cooperação entre agentes, algo que nem sempre é um requisito no foco de pesquisa tradicional de LLMs.
3. **Ambientes Não-Ingleses:** A maior parte da pesquisa tem se concentrado em ambientes de língua inglesa, deixando as configurações não-inglesas, como o Guandan (que é popular na China), subexploradas.
4. **Espaço de Ação Extenso:** O Guandan apresenta um **espaço de ação dinâmico e extenso**; o número de ações válidas pode exceder 80 durante a vez de um jogador, o que dificulta a análise abrangente por parte dos LLMs.

Metodologia e Aprimoramentos Propostos

Para mitigar esses desafios e avaliar os LLMs (tanto modelos *open-source* quanto *API-based*, incluindo GPT-4 e modelos chineses como Baichuan2 e Qwen1.5), os autores propuseram duas técnicas principais:

1. Ferramenta Externa para Gerenciamento do Espaço de Ação (Action Recommender) Para lidar com o grande e dinâmico espaço de ações em Guandan, foi incorporada uma **ferramenta externa baseada em Reinforcement Learning (RL)**.
Este *recommender* (recomendador de ações) é um modelo baseado em RL (usando o *embedding* aprendido do Danzero) que pontua todas as ações válidas por turno.

- Isso permite que os agentes LLM se concentrem apenas nas **top k ações** (as melhores ações recomendadas), melhorando a eficiência na tomada de decisões.
- A análise de distribuição das ações revelou que, mesmo com muitas opções, os LLMs consistentemente se restringiam a um máximo de 7 ações, preferindo fortemente as primeiras cinco, o que sublinha a necessidade dessa ferramenta.

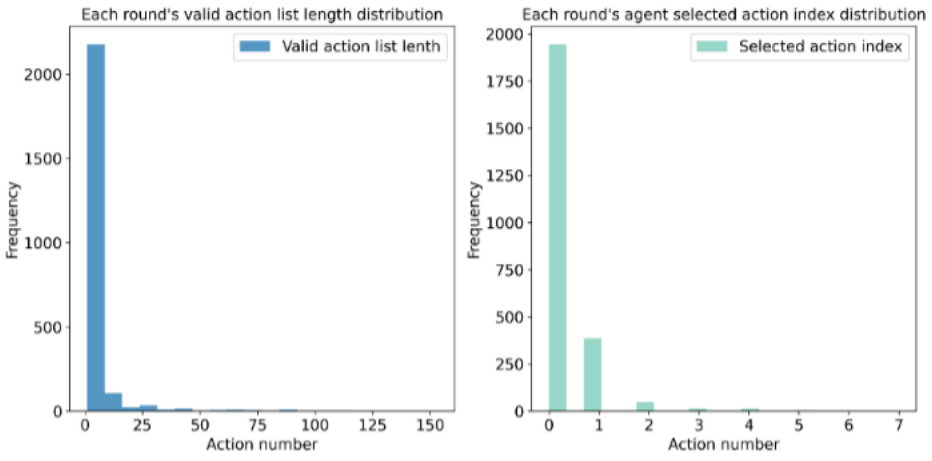


Fig. 3. ***Left Figure:*** *Distribution of agent's valid action list length per round.* ***Right Figure:*** *The distribution of agent's final selected action index. Results were obtained from about 2000 rounds.*

2. Planejamento com Teoria da Mente (ToM Planning Framework)

O principal aprimoramento é uma técnica de planejamento da Teoria da Mente (ToM) que permite aos agentes LLM adaptar sua estratégia usando apenas as regras do jogo, o estado atual e o contexto histórico como entrada. Este planejamento é crucial porque o planejamento *Vanilla* (básico) assume uma distribuição uniforme para as ações dos aliados e adversários.

A Teoria da Mente é implementada em diferentes níveis:

- **Planejamento Vanilla (Zero-shot):** Baseado no conhecimento das regras e da mão do agente, ele visa a ação mais vantajosa para vencer.
- **ToM de Primeira Ordem (First-Order ToM):** O LLM-Agente **infere os tipos de cartas e estratégias** dos oponentes e companheiros de equipe com base nas ações passadas (histórico de jogo). Isso permite construir estratégias de coordenação com a equipe.
- **ToM de Segunda Ordem (Second-Order ToM):** O agente raciocina sobre **o que os oponentes acreditam sobre a sua própria estratégia**. Isso permite uma tomada de decisão mais sofisticada, ajustando-se a condições dinâmicas do jogo e antecipando blefes, similar a jogadores experientes.

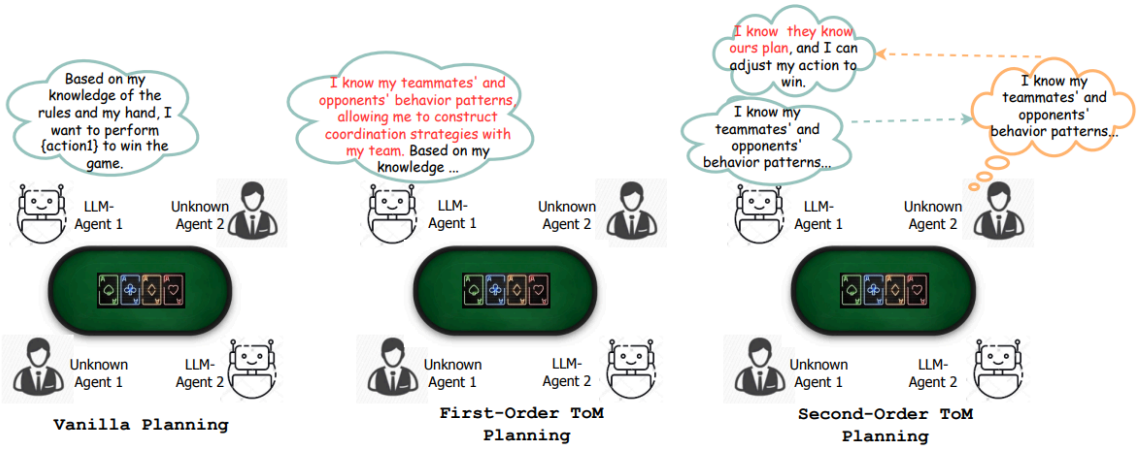


Fig. 1. ***Left Figure:*** *LLM-Agents' reasoning, using vanilla planning.* ***Middle Figure:*** *LLM-Agents' reasoning, using first-order Theory of Mind(ToM) planning.* ***Right Figure:*** *LLM-Agents' reasoning, using second-order Theory of Mind(ToM) planning.*

O processo operacional do agente LLM é modular, utilizando um **Intérprete de Estado**, um **Planejador ToM** e um **Avaliador de Plano**. O LLM recebe regras do jogo, regras de observação (prompts estruturados para converter informações de estado de baixo nível em texto legível) e histórico do jogo.

Resultados Chave

A avaliação foi realizada em um ambiente chinês, onde os agentes LLM competem em equipes contra adversários como agentes aleatórios (*Random*), baseados em regras (*Rule-based*) e o modelo de última geração de RL, **Danzero+**.

1. **Melhoria Consistente com ToM:** A injeção de conhecimento de crença obtido através do planejamento ToM **melhorou consistentemente o desempenho** de quase todos os modelos. Isso sugere que os LLMs têm uma capacidade natural de analisar as crenças de outros jogadores e que o desenvolvimento dessa habilidade contribui drasticamente para o desempenho geral.

2. **Desempenho do GPT-4:** Embora uma lacuna de desempenho exista entre os LLMs atuais e os modelos SOTA (State-of-the-Art) de RL, o **GPT-4 superou consistentemente outros LLMs** e alcançou um desempenho impressionante. Com o planejamento de ToM de segunda ordem e a assistência do recomendador de ações, o GPT-4 alcançou uma pontuação média de **-0.88** contra o Danzero+, ficando notavelmente próximo do desempenho do modelo de RL de última geração.

3. **Liderança do Danzero+:** Nenhum agente LLM, com exceção do GPT-4, conseguiu superar o Danzero+. O Danzero+ serve como limite superior de desempenho, sendo um modelo SOTA treinado com RL (método Monte Carlo profundo).

4. **Impacto do Idioma:** O desempenho do LLM-Agente foi **significativamente superior no ambiente de língua chinesa** em comparação com o inglês. Isso é atribuído ao fato de que o GPT-3.5 (usado no estudo de ablação) e outros modelos foram pré-treinados em uma enorme quantidade de dados, e a maioria dos dados relacionados ao Guandan está disponível primariamente em chinês.

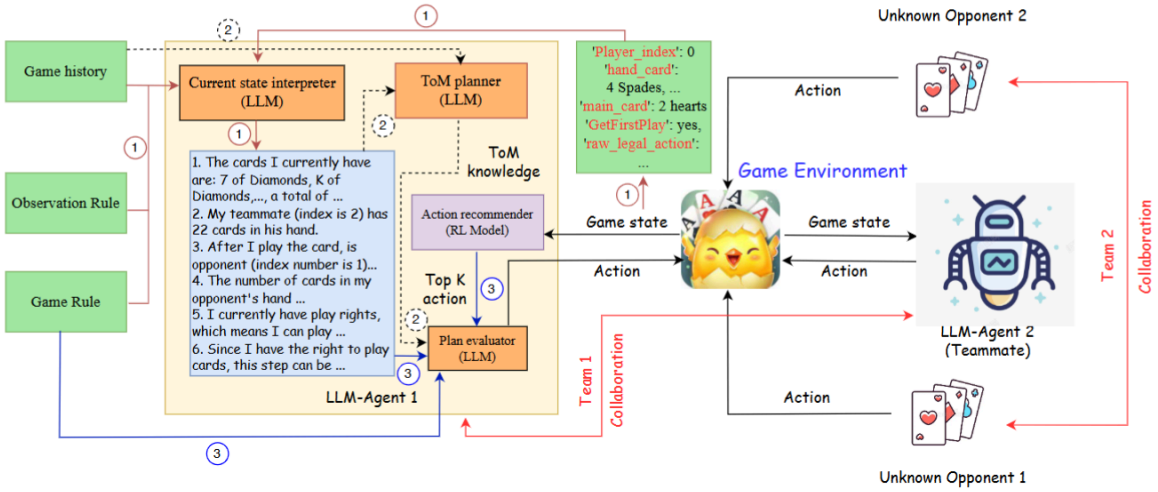


Fig. 2. *The figure illustrates the LLM-Agent's operational process, with the green square representing the model's input, the orange square denoting the LLM model, the blue square signifying the LLM model's output, and the purple square corresponding to the RL-based model, which serves as an external tool to augment the agent's decision-making abilities.*

Contribuições do Estudo

As principais contribuições do trabalho são:

- Avaliação aprofundada de agentes LLM (*open-source* e comerciais) no Guandan, um jogo chinês complexo que envolve informação imperfeita e cooperação.
- Desenvolvimento e integração de uma ferramenta baseada em RL para lidar com espaços de ação dinâmicos e extensos.
- Criação de uma estrutura de planejamento que equipa os LLMs com diferentes níveis de capacidade de Teoria da Mente (ToM), permitindo a cooperação sem a necessidade de *fine-tuning*.

Em conclusão, esta pesquisa aprofunda a compreensão do potencial dos LLMs na navegação de interações sociais complexas e no avanço da modelagem cognitiva em IA, demonstrando que o planejamento baseado em ToM é eficaz para aprimorar a colaboração em jogos de informação imperfeita.

APÊNDICE 3

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 2 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

AMIR YOUSSEF DOS SANTOS

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Na minha especialização em Agentes Inteligentes inicialmente:

- Leitura dos livros bases e comecei a afunilar em Agentes Inteligentes baseados em LLM
- Estou juntando todos os meus conhecimentos em um Guia de Estudo
- Estou explorando artigos de diferentes áreas: Agentes LLM de diálogos e Agentes LLMs Baseados na Teoria da Mente

Nesta Semana explorei outras áreas de Agentes Inteligentes baseados em LLM:

- Leitura e geração do resumo do artigo "Beyond Self-Talk: A Communication-Centric Survey of LLM-Based Multi-Agent Systems"
  - Além da auto-conversa: uma pesquisa centrada na comunicação de sistemas multiagentes baseados em LLM
  - Recomendação do Alberto
  - Comunicação do sistema (macro): Plana; Hierárquica; Equipes; Sociedade; Híbrida
  - Objetivos de comunicação: Cooperação, Competição e misto
  - Protocolos de comunicação: MCP (Model Context Protocol), A2A (Agent-to-Agent Protocol), ANP (Agent Network Protocol)
  - Arquiteturas de comunicação (micro): Sequencial; Concorrente; Concorrente com Sumarizador
  - Objetos de comunicação: Si mesmo; Outros Agentes; Ambiente; Humano
  - Conteúdo da comunicação: Explícito ou Implícito

    Beyond Self-Talk: A Communication-Centric Survey of LLM-Based Multi-Agent Systems

- Leitura e geração do resumo do artigo "Generative Agents: Interactive Simulacra of Human Behavior"
  - Agentes Generativos: Simulacros Interativos do Comportamento Humano
  - Recomendação do Carlos
  - Arquitetura: Memória; Recuperação; Reflexão; Planejamento e Reação
  - Ambiente: Smallville, cidade urbana onde cada agente tem seu avatar e memórias iniciais
  - Avaliação Controlada: Autoconhecimento; Memória; Planos; Reações

- Foi comparado: Arquiteturas Completa x ablações (sem observação, sem reflexão e/ou sem planejamento) vs. baseline humana
- Avaliação end-to-end (25 agentes por 2 dias de simulação): Difusão de informação; Formação de relacionamentos; Coordenação
- Comportamentos problemáticos: Localizações atípicas; Normas físicas mal codificadas; Efeitos de *instruction tuning*
- Agregar memória, reflexão e planejamento a LLMs permite comportamentos críveis e socialmente coerentes ao longo do tempo

Generative Agents: Interactive Simulacra of Human Behavior

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

- Continuar explorando artigos de diferentes áreas
- Continuar anotando os conteúdos principais no “Guia de Estudo”
- Definir a área que desejo atuar

Observação: [caso precise fazer alguma observação, de qualquer “natureza”]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 9 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

AMIR YOUSSEF DOS SANTOS

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Na minha especialização em Agentes Inteligentes inicialmente:

- Leitura dos livros com as bases e comecei a afunilar em Agentes Inteligentes baseados em LLM
- Estou juntando todos os meus conhecimentos em um Guia de Estudo
- Explorei artigos de diferentes áreas: Agentes LLM de diálogos e Agentes LLMs Baseados na Teoria da Mente, Agentes LLM para simulação

Nesta Semana me aprofundei na área de Agentes Inteligentes baseados em LLM para simulação:

- Após a leitura do artigo "Generative Agents: Interactive Simulacra of Human Behavior", busquei pesquisar sobre diferentes ambientes de simulação para entender melhor os tipos de ambientes possíveis e o que pode ser feito em cada um :
  - Smallville - sandbox 3d inspirada no The Sims
  - Minecraft (via MineDojo)
  - Guandan
  - Hanabi
  - Texas Hold'em (multi-player)

  Ambientes p/ simulação

- Na pesquisa de outros ambientes me deparei com este artigo:"PokerGPT: An End-to-End Lightweight Solver for Multi-Player Texas Hold'em via Large Language Model":
  - Solucionador de poker leve de ponta a ponta (end-to-end) baseado em Large Language Models (LLMs) para jogos de Informação Imperfeita
  - LLMs como núcleo de raciocínio, permitindo aprendizado direto de dados reais e adaptação a múltiplos jogadores
  - PokerGPT consiste em aquisição de dados, prompt engineering e treinamento.
  - Logs textuais de jogos reais da plataforma PokerStars, priorizando jogadores com alto desempenho

- Framework de treino: DeepSpeed-Chat - Ajuste Fino Supervisionado; Treinamento do Modelo de Recompensa; RLHF
- Treinado em apenas 9,5 horas de GPU

PokerGPT: An End-to-End Lightweight Solver for Multi-Player Texas Hold'em via Larg…

- Foco inicialmente no ambiente Smallville do artigo "Generative Agents: Interactive Simulacra of Human Behavior":
  - Já estou explorando o GitHub que eles disponibilizaram tentando entender como funciona o ambiente, as personas, as memórias e planos de cada "personagem"
  - Utilizaram o Phaser para fazer o ambiente, todas as ações são dadas pelo LLM em linguagem natural, passada para o motor do jogo e convertida em movimento
  - Estou rodando os códigos e vendo as possíveis complicações
  - Analisando os frameworks utilizados

  GitHub

- Pesquisa de aplicações de Agentes LLM em Simulações:
  - Pesquisas Sociais e Comportamento Humano - Criar populações sintéticas com perfis culturais/econômicos/psicológicos para testar políticas, mensagens e dinâmicas de grupo sem expor pessoas reais.

  Aplicações

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

- Continuar aprofundando no código do artigo "Generative Agents: Interactive Simulacra of Human Behavior" para entender como funciona cada parte e entender melhor seus frameworks
- Fazer o análise do artigo "PokerGPT: An End-to-End Lightweight Solver for Multi-Player Texas Hold'em via Large Language Model" vendo os frameworks e começar a me aprofundar nos códigos disponibilizados

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Estamos continuando a explorar diversas áreas dos Agentes de IA Baseados em LLM

Agentes de Diálogos:

Resumo do artigo:
Beyond Self-Talk: A Communication-Centric Survey of LLM-Based Multi-Agent Systems

(Além da auto-conversa: uma pesquisa centrada na comunicação de sistemas multiagentes baseados em LLM)

Visão geral e motivação

O artigo apresenta uma **pesquisa de referência** sobre Sistemas Multiagente Baseados em LLM (LLM-MAS) a partir de uma **perspectiva centrada na comunicação**. Em vez de organizar o campo apenas por domínios (ex.: programação, simulação social) ou por blocos arquiteturais, os autores defendem que a **comunicação** — entre agentes, com o ambiente e com humanos — é o **mecanismo fundamental** que determina **coordenação, escalabilidade, eficiência e segurança**. Propõe-se um **framework estruturado** que integra a **comunicação em nível de sistema** (macro) e a **comunicação interna do sistema** (micro), oferecendo uma taxonomia unificada para analisar e projetar LLM-MAS.

Contexto: do agente único ao LLM-MAS

- **Agente baseado em LLM (visão mínima)**:
    - **Brain (Cérebro)**: LLM com memória de curto prazo (contexto) e **longo prazo** (artefatos estruturados), frequentemente com **RAG** para raciocínio/planejamento.

- **Perception (Percepção)**: interfaces multimodais (texto, visão, áudio, estado do ambiente) convertidas em *prompts*.
- **Actions (Ações)**: não se limitam a texto; incluem uso de **ferramentas/APIs** e atuadores no mundo real.

- **Por que multiagentes?** Múltiplos agentes permitem **divisão de trabalho**, **especialização** e **coordenação**, abordando problemas além da capacidade de um único agente e explorando **dinâmicas sociais** (cooperação/competição).

Comunicação em nível de sistema (macro)

Arquiteturas de comunicação

- **Flat (plana)**: rede P2P descentralizada. +Ágil e espontânea; −Sobrecarga de mensagens e dificuldade de escalar.
- **Hierarchical (hierárquica)**: camadas de orquestração (alto nível) e execução (baixo nível). +Delegação clara; −Gargalos e pontos únicos de falha.
- **Team (por equipes)**: células especializadas para sub-tarefas. +Aproveita expertise; −Coordenação interequipes mais custosa.
- **Society (sociedade)**: normas e instituições compartilhadas; foca **comportamentos emergentes**. +Riqueza social; −Complexidade e imprevisibilidade.
- **Hybrid (híbrida)**: combina estruturas para **adaptabilidade**. +Flexível; −Exige design cuidadoso para evitar **overhead** de coordenação.

Objetivos de comunicação

- **Cooperation (cooperação)**: busca de objetivos comuns; pode ser **direta** (consenso rápido) ou via **debate** (crítica mútua para robustez).
- **Competition (competição)**: objetivos conflitantes estimulam **estratégia e inovação**, mas elevam riscos de **instabilidade** e **segurança**.

- **Mixed (misto)**: alterna cooperação/competição como no mundo real (negociação, simulação social), aumentando **versatilidade** e a complexidade de **mecanismos**.

Protocolos de comunicação

- **MCP (Model Context Protocol)**: acopla LLMs a **ferramentas/dados** com chamadas estruturadas (JSON-RPC), melhorando **interoperabilidade** e **segurança**.
- **A2A (Agent-to-Agent Protocol)**: *handshake* P2P com **Agent Cards** publicados (HTTP) para **descoberta de capacidades** e **delegação dinâmica**.
- **ANP (Agent Network Protocol)**: redes abertas com **DIDs** e **JSON-LD**, priorizando **interoperabilidade semântica** e **criptografia fim-a-fim**.

Síntese: A escolha de **arquitetura + objetivo + protocolo** define o **perfil de coordenação** (latência, robustez, throughput, segurança) do sistema.

Comunicação interna do sistema (micro)

Estratégias de comunicação (quando/como falar)

- **One-by-One (sequencial)**: cada agente integra todo o histórico. +Contexto nítido; −Latência linear e **propagação de erros**.
- **Simultaneous-Talk (concorrente)**: fala paralela. +Ideação rápida; −**State staleness** e necessidade de **árbitro**.
- **Simultaneous-Talk com Sumarizador**: um agente sintetiza as falas + mitiga dessincronia; Reintroduz dependência sequencial

Paradigmas de comunicação (como representar)

- **Message Passing**: mensagens explícitas (NL ou raciocínio explanado). +Simplicidade; −Volume/consistência.

- **Speech Act**: atos de fala com intenções (prometer, negociar, instruir). +Direciona comportamentos; −Exige taxonomias e **fundamentação** claras.

- **Blackboard (quadro negro)**: repositório compartilhado para publicar/ler estado. +Consistência global; −Governança de acesso e mitigação de **desinformação**.

Objetos de comunicação (com quem/com o quê)

- **Self**: deliberação/reflexão interna para planejamento.

- **Other Agents**: troca de informações, coordenação, negociação.

- **Environment**: percepção do estado (texto, visão, áudio, simuladores) e conversão em conhecimento acionável.

- **Human**: interação direta com usuários (comandos, feedback, colaboração).

Conteúdo de comunicação (o que é trocado)

- **Explícito**:
    - **Linguagem natural** (flexível, interpretável)
    - **Código/dados estruturados** (precisos, verificáveis)
- **Implícito**:
    - **Feedback comportamental** (sinais via ações/estratégias)
    - **Sinais ambientais** (mudanças de contexto que influenciam decisões)

Trade-off central: mensagens ricas aumentam a qualidade**/robustez** do raciocínio coletivo, mas elevam **custo, latência e risco de erros**.

Desafios e oportunidades

1. **Otimização de design**: arquiteturas **híbridas** e **paradigmas** eficientes; alocação de recursos sob **alto tráfego** de mensagens; mitigação de **alucinações** geradas/propagadas pela comunicação.

2. **Competição entre agentes**: integrar competição **de forma segura** (evitar colapsos/instabilidade) e útil para **explorar estratégia** e **criatividade**.

3. **Protocolo unificado**: reduzir **redundância** e **incompatibilidades** entre MCP/A2A/ANP/etc.; buscar um padrão que cumpra papel análogo ao **HTTPS** na Web para **interoperabilidade e segurança**.

4. **Comunicação multimodal**: fundir **texto, imagem, áudio e vídeo** com consistência semântica e coordenação entre agentes.

5. **Segurança da comunicação**: garantir **confidencialidade, integridade, autenticidade**; prevenir **MITM, spoofing**; criptografia fim-a-fim e **autenticação** robusta em topologias dinâmicas.

6. **Benchmarks e avaliação**: criar **leaderboards** e **suites** que avaliem **propriedades sistêmicas** (eficiência de coordenação, latência, robustez a mensagens desatualizadas/enganosas) e escalas **heterogêneas** (cooperação, competição e cenários mistos).

Implicações práticas de design

- **Escolha arquitetural** deve refletir **objetivo de comunicação** (ex.: debate para verificação factual vs. hierarquia para execução disciplinada).

- **Governança de mensagens**: resumidores, filtros, *rate limits* e **validação semântica** para reduzir ruído e **erro acumulado**.

- **Representação híbrida do conteúdo**: combinar **NL** (flexível) com **estruturas formais** (precisas) quando a tarefa requer verificabilidade.

- **Observabilidade e auditoria**: *logs* de tráfego, *traces* de raciocínio, e políticas de **segurança** integradas ao protocolo.

Conclusão

O estudo defende que **comunicação é o eixo organizador** de LLM-MAS. A taxonomia proposta — cobrindo **arquiteturas, objetivos, protocolos, estratégias, paradigmas, objetos e conteúdo** — oferece uma lente unificadora para analisar **eficiência, escalabilidade e adaptabilidade**. As **prioridades** para a evolução do campo incluem **protocolos padronizados**, **comunicação multimodal** consistente e **segurança de ponta a ponta**, além de **benchmarks** que captem métricas **multi-granulares** do nível do agente ao nível do sistema. Em síntese, projetar LLM-MAS eficaz passa por **engenheirar a conversa**: **o que** comunicar, **quando**, **como**, **com quem** — e **sob quais garantias**.

Agente Generativos:

Resumo do artigo: Generative Agents: Interactive Simulacra of Human Behavior

(Agentes Generativos: Simulacros Interativos do Comportamento Humano)

Visão geral

O paper apresenta **Agentes Generativos**: agentes computacionais que simulam comportamento humano crível por longos períodos. A proposta estende um **LLM** com três capacidades integradas — **memória de longo prazo**, **reflexão** e **planejamento** — para que os agentes **observem** o mundo, **lembrem-se** do passado, **sintetizem** inferências de nível mais alto e **planejem/ajam** de forma coerente no tempo. A demonstração ocorre em **Smallville**, um ambiente sandbox do tipo “vida cotidiana”, com **25 agentes** que exibem **fenômenos sociais emergentes** (difusão de informação, formação de relacionamentos e coordenação de atividades), incluindo a organização autônoma de uma **festa de Dia dos Namorados** a partir de uma única intenção inicial.

Contribuições principais

- **Arquitetura de agente de longo prazo**: combina *Memory Stream*, *Reflection* e *Planning* para superar a limitação de LLMs “estáticos” condicionados apenas ao estado atual.
- **Recuperação de memórias orientada a contexto**: seleciona, a cada passo, um subconjunto relevante do histórico com base em **recência**, **importância** e **relevância semântica**.
- **Reflexões como inferências persistentes**: o agente sintetiza memórias em pensamentos de nível superior que passam a guiar futuras decisões e diálogos.
- **Demonstração em sociedade artificial**: 25 agentes em Smallville exibem **comportamentos sociais críveis** e não roteirizados.

- **Avaliações controladas e end-to-end**: mostram que a arquitetura completa supera ablações e uma linha de base humana em **credibilidade**.

Arquitetura de Agente

1) *Memory Stream* (Memória)

- **O que é**: um registro contínuo e extenso, em linguagem natural, de tudo que o agente experiencia (observações, diálogos, planos, reflexões).

- **Por quê**: LLMs têm janela de contexto limitada; a memória persistente preserva continuidade de identidade, objetivos e relações.

2) Recuperação (*Retrieval*)

- **Desafio**: o fluxo completo não cabe na janela do LLM; é preciso **amostrar** o que é mais útil agora.

- **Critérios combinados**:

    - **Recência**: memórias acessadas recentemente recebem maior pontuação (decaimento exponencial ≈ **0,995**).

    - **Importância**: eventos triviais (≈1) vs. cruciais (≈10); a **importância** é atribuída no momento em que a memória é criada.

    - **Relevância**: similaridade de **embedding** (cosseno) entre a consulta contextual e as memórias.

- **Resultado**: uma **pontuação final** ponderada ranqueia memórias; o subconjunto topo é injetado no prompt.

3) *Reflection* (Reflexão)

- **Motivação**: observações isoladas não bastam para generalizar ou formar hipóteses de alto nível.

- **Gatilho**: quando a soma das **importâncias** de eventos recentes supera um limiar (**~150**), o agente:

    1. identifica **perguntas salientes** sobre o que ocorreu;
    2. recupera memórias relevantes como **evidência**; e
    3. produz **reflexões** (inferências) que citam as memórias usadas.

- **Estrutura**: reflexões podem se apoiar em outras, formando uma **árvore** (observações nas folhas; abstrações nos níveis superiores).

4) *Planning* (Planejamento) & Reação

- **Planejamento top-down e recursivo**:
    - **Plano diário** (5–8 blocos) → **planos de 1 hora** → **ações de 5–15 min**.
- **Execução reativa**: a cada passo, o agente decide **seguir o plano** ou **reagir** a novas observações; em caso de reação, **replaneja** a partir do momento atual.
- **Geração de diálogo**: condicionada às **memórias** e **resumos sobre o interlocutor**, alinhada à **reação** pretendida.

Stack do paper: os autores usam **gpt-3.5-turbo** como LLM subjacente.

Smallville: ambiente e implementação

- **Sandbox urbano** com casas, café, bar, escola e outros locais; cada agente tem um **avatar** e **memórias iniciais** descritivas.
- **Interação com o mundo**: agentes se movem e alteram o estado de objetos (ex.: cama ocupada, geladeira vazia). O **usuário** pode mudar estados via linguagem natural (ex.: fogão “queimando”).
- **Motor e servidor**: desenvolvido com **Phaser**; o servidor mantém o estado do mundo em **JSON** (localizações, ações). O mundo é uma **árvore de áreas/objetos**

convertida em texto para o raciocínio do agente.

- **Escolha de localização**: o LLM **navega** a árvore para decidir onde executar cada ação.

Avaliação controlada (sondagens em linguagem natural)

Objetivo: medir se agentes geram comportamento **crível** em tarefas de memória, planejamento e reação.

Categorias de perguntas

1. **Autoconhecimento** (ex.: “apresente-se”).
2. **Memória** (recuperar eventos/diálogos: “quem concorre à prefeitura?”).
3. **Planos** (ex.: “o que você estará fazendo amanhã às 10h?”).
4. **Reações** (ex.: “café queimando — o que você faz?”).
5. **Reflexões** (preferências e inferências de nível superior).

Condições comparadas

- **Arquitetura completa** vs. ablações (sem observação, sem reflexão e/ou sem planejamento) vs. **baseline humana** (crowdworkers).

Resultados

- A arquitetura completa obteve a **melhor credibilidade** (**μ ≈ 29,89**), superando ablações e a linha de base humana.
- **Importância da reflexão**: decisiva para escolhas mais informadas (ex.: seleção de presentes com base em histórico relevante).

Avaliação end-to-end (25 agentes por 2 dias de simulação)

Fenômenos emergentes observados

- **Difusão de informação**: conhecimento sobre a candidatura de *Sam* a prefeito **4% → 32%**; sobre a festa de *Isabella* **4% → 52%**.
- **Formação de relacionamentos**: densidade de laços mútuos **0,167 → 0,74**.
- **Coordenação**: **5/12** convidados comparecem à festa; ausentes citam **conflitos** ou **falhas de planejamento**.

Erros recorrentes

- **Recuperação insuficiente** de memórias relevantes.
- **Alucinações** ou embelezamentos no conteúdo das memórias.
- **Estilo de linguagem** excessivamente formal/cooperativo herdado do LLM.

Comportamentos problemáticos

- **Localizações atípicas** para certas ações (ex.: almoçar no bar em vez do café) à medida que descobrem mais áreas.
- **Normas físicas mal codificadas** (ex.: entrar em banheiro ocupado; acessar loja fechada).
- **Efeitos de *instruction tuning***: agentes aceitam sugestões desalinhadas a seus interesses.

Riscos éticos e mitigação

1. **Relações parassociais**: risco de apego e antropomorfização. *Mitigação*: transparência sobre a natureza computacional e alinhamento de valores.

2. **Impacto de erros fora de jogo**: inferências erradas podem causar danos. *Mitigação*: auditoria e cautela de uso.

3. **Riscos da IA generativa**: deepfakes, desinformação, persuasão personalizada. *Mitigação*: manter **logs/auditoria** das I/O.

4. **Dependência excessiva no design**: substituir stakeholders humanos. *Mitigação*: agentes como **complemento**, não substitutos.

Limitações e trabalho futuro

- **Recuperação**: melhorar o ranqueamento e cobertura de memórias críticas.
- **Custos e latência**: simulações com muitos agentes são **caras** e demoradas; necessidade de **otimizações**.
- **Escala temporal**: avaliar comportamentos em horizontes **mais longos**.
- **Segurança**: vulnerabilidade a **prompt/memory hacking**; mitigações de segurança e robustez.
- **Vieses**: herança de vieses sociolinguísticos do LLM subjacente.

Conclusão

O estudo demonstra que **agregar memória, reflexão e planejamento** a LLMs permite **comportamentos críveis e socialmente coerentes** ao longo do tempo. Em Smallville, os agentes não apenas **executam rotinas**, mas **interagem**, **difundem informação**, **formam laços** e **coordenam atividades** com base em experiências passadas. As avaliações quantitativas e qualitativas sustentam a eficácia da arquitetura completa, embora permaneçam **desafios** práticos (recuperação, custo, vieses) e **cuidados éticos**. O trabalho sinaliza aplicações em **prototipagem social**, **design centrado no humano** e **ambientes imersivos**, abrindo caminho para agentes generativos mais **robustos, explicáveis** e **seguros**.

Frameworks Utilizados:

Arquitetura Central dos Agentes Generativos

Framework que estende um LLM para **comportamento humano crível e de longo prazo**, composto por três módulos:

Componente	Função	Observações
Memory Stream (Fluxo de Memória)	Armazena, em linguagem natural, experiências do agente ao longo do tempo.	Recuperação ponderada por **relevância**, **recência** e **importância**; unidade básica: **observação**.
Reflection (Reflexão)	Sintetiza memórias em **inferências de nível superior**.	Ajuda o agente a **generalizar** e formar **conclusões** sobre si e outros, além do dado observacional bruto.
Planning (Planejamento)	Traduz conclusões + estado do ambiente em **planos de ação**.	**Decomposição recursiva** em comportamentos detalhados para manter **coerência temporal**.

Frameworks do Ambiente de Simulação

- **Ambiente:** *Smallville* — sandbox interativo inspirado em *The Sims*.
- **Framework de desenvolvimento de jogos: Phaser** (web game framework) — base para construir o mundo/sandbox onde os agentes atuam.
- O Large Language Model (LLM) usado na implementação atual para sustentar essa arquitetura é o **ChatGPT** na versão **gpt3.5-turbo**

Agente para jogos de Informações Incompletas:

Resumo do artigo: PokerGPT: An End-to-End Lightweight Solver for Multi-Player Texas Hold'em via Large Language Model

(PokerGPT: Um *Solver* Leve *End-to-End* para Texas Hold'em Multijogador via Grande Modelos de Linguagem)

Resumo

O Poker (Texas Hold'em) é um desafio clássico em jogos de informação imperfeita (IIGs). Trabalhos anteriores, como *DeepStack* e *Libratus*, basearam-se no algoritmo *Counterfactual Regret Minimization* (CFR), eficaz em cenários de dois jogadores (*heads-up*), mas com alto custo computacional e baixa escalabilidade para múltiplos jogadores.

O artigo propõe o **PokerGPT**, um *solver* **end-to-end** baseado em um **Large Language Model (LLM)** leve, capaz de jogar Texas Hold'em com qualquer número de jogadores. Utilizando **dados textuais reais de partidas** e **Reinforcement Learning with Human Feedback (RLHF)**, o PokerGPT alcança **alta taxa de vitória**, **baixo custo computacional** e **velocidade superior de resposta** em comparação com abordagens baseadas em CFR e *deep learning* tradicionais.

I. Introdução

A pesquisa em IA aplicada a jogos complexos é historicamente relevante, e o Poker representa um dos maiores desafios por envolver **informação incompleta** e **interações estratégicas multi-jogador**.

Limitações dos Métodos CFR:

1. **Alto consumo de recursos** — exige grandes volumes de CPU/GPU e memória.
2. **Escalabilidade limitada** — cresce exponencialmente com o número de jogadores.

3. **Perda de informação** — compressão da árvore de decisão elimina detalhes importantes.

4. **Dependência de especialistas humanos** — requer conhecimento matemático para modelagem.

Diante disso, o PokerGPT utiliza um **modelo leve (OPT-1.3B)** ajustado via **RLHF**, sem necessidade de compressão de árvores ou heurísticas manuais.

Principais Contribuições:

- Introdução de uma **abordagem baseada em LLM** para jogos de informação imperfeita.
- Desenvolvimento de um **modelo leve e generalizável** para múltiplos jogadores.
- Redução significativa de **tempo de treinamento** e **custo computacional**.
- Aplicação inovadora de **engenharia de prompt** e **seleção de dados**.
- Demonstração de **desempenho superior** em taxa de vitória e eficiência.

II. Trabalho Relacionado

A. Soluções Clássicas de Poker

Modelos como *DeepStack*, *Libratus* e *Pluribus* usam variantes de CFR para aproximar o equilíbrio de Nash, alcançando desempenho sobre-humano, mas a um custo altíssimo de processamento e memória.

B. LLMs e Jogos

LLMs recentes mostraram capacidade de **raciocínio estratégico**, **interação humana** e **aprendizado no contexto**, tornando-os promissores para simular comportamento competitivo e adaptativo — características essenciais em Poker.

III. Pré-Requisitos

A. Regras do Texas Hold'em

O jogo envolve múltiplas rodadas (*Preflop, Flop, Turn, River, Showdown*), nas quais os jogadores decidem entre **bet**, **check**, **call**, **raise** ou **fold**, visando formar a melhor mão possível.

B. Large Language Models

LLMs são **redes neurais profundas** treinadas para compreender e gerar linguagem natural. O processo inclui:

1. **Pré-treinamento** — aprendizado auto-supervisionado.
2. **Ajuste fino (Fine-tuning)** — adaptação a tarefas específicas.
3. **RLHF** — uso de feedback humano para guiar a otimização de políticas de decisão via *Reinforcement Learning*.

IV. PokerGPT (Arquitetura e Treinamento)

A. Aquisição de Dados

- Fonte: logs reais da **PokerStars**.
- **Informações utilizadas:**
    - *Básicas:* blinds, assentos, fichas.
    - *Dinâmicas:* cartas comunitárias e ações dos jogadores.

- *Sumário:* pot e ganhos/perdas.

- **Filtragem:** foca em informações observáveis e comportamentos.
- **Variáveis centrais:** cartas (privadas e públicas), posição, blinds, saldo e estágio do jogo.

B. Engenharia de Prompt

Transforma logs em prompts compreensíveis para humanos e LLMs.

- **Seleção:** apenas jogos com *showdown* e alta taxa de vitória.
- **Estrutura do Prompt:**
  - Informação **constante** (regras e contexto).
  - Informação **dinâmica** (estado atual e ações).
- **Técnicas:**
  - *Role play* (“You are a professional gambler”).
  - *Value discretization* — transforma valores contínuos em classes discretas (múltiplos do *big blind*).

C. Processo de Treinamento

Utiliza o **framework DeepSpeed-Chat** e o modelo base **OPT-1.3B**.

1. **Supervised Fine-Tuning (SFT):** treinamento supervisionado com dados de alta qualidade.
2. **Reward Model:** modelo de recompensa treinado a partir da taxa de vitória (mbb/h).
3. **RLHF com PPO:** ajusta o modelo iterativamente com base em feedback humano, melhorando robustez e generalização.

V. Experimentos e Resultados

Configuração:

- Dataset inicial: >1 milhão de jogos.
- Dataset final: 120 mil partidas de alta qualidade.

Comparação de Desempenho:

Modelo	Taxa de Vitória (mbb/h)	Tempo de Treinamento (GPU h)
PokerGPT	**158 ± 49**	**9.5**
ReBel	45 ± 5	—
AlphaHoldem	111 ± 16	580

PokerGPT supera todos os modelos, com destaque para velocidade, eficiência e suporte a múltiplos jogadores.

Experimentos de Ablação:

- **Dataset filtrado** → melhora significativa em *Macro F1* e *Perplexity*.
- **Dados de jogadores vencedores** → modelo mais racional e conservador.
- **Dados de jogadores perdedores** → modelo mais agressivo (*raise/bet* excessivos).

Avaliação Multi-Jogador:

- **Taxa de vitória** diminui com mais jogadores, mas permanece positiva.
- **Tempo de resposta** cresce linearmente → boa escalabilidade.

- **Estratégia adaptativa:** mais cautela com o aumento do número de oponentes.

Análise de Interação:

- O modelo compreende **instruções em linguagem natural** (ex.: ser “mais agressivo”).
- Reconhece padrões de força dos oponentes.
- Limitações: **não calcula probabilidades** nem faz **inferência matemática** explícita.

VI. Conclusão e Trabalhos Futuros

O **PokerGPT** demonstra que **LLMs podem atuar como solvers eficientes para jogos de informação imperfeita**, superando limitações dos métodos CFR.

Vantagens principais:

- Treinamento leve e rápido.
- Escalabilidade para múltiplos jogadores.
- Estratégias adaptativas e coerentes.

Trabalhos futuros:

- Aumentar a **explicabilidade** das decisões.
- Incorporar **dados explicativos e probabilísticos**.
- Integrar **elementos de raciocínio matemático** e **estimativas de probabilidade de vitória**.

Estudo sobre ambientes de simulações e artigos para embasar:

Generative Agents: Interactive Simulacra of Human Behavior (2023):

Ideia: arquitetura de agentes com memória em linguagem natural, “reflexões” e planejamento para gerar comportamentos sociais críveis em tempo real.
Ambiente: Smallville, uma cidade sandbox inspirada em *The Sims*, onde os agentes vivem rotinas, conversam e coordenam eventos.
Link

Voyager: An Open-Ended Embodied Agent with LLMs (2023):

Ideia: a gente usa um LLM como planejador/programador para explorar, escrever e reutilizar skills (código) e aprender de forma contínua.
Ambiente: Minecraft (via MineDojo) com currículo automático, biblioteca de habilidades executáveis e interação via feedback do mundo.
Link

Evaluating and Enhancing LLMs Agent based on Theory of Mind in Guandan (2024):

Ideia: planejar com Theory of Mind em LLMs para cooperar/adaptar estratégia contra diferentes oponentes; usa ferramenta externa para lidar com espaço de ações grande.
Ambiente: Guandan, jogo de cartas cooperativo sob informação incompleta (4 jogadores).
Link

LLM-Hanabi: Evaluating Multi-Agent Gameplays with Hanabi (2025):

Ideia: benchmark que traduz estados do jogo em linguagem natural para avaliar inferência de razões/ToM e colaboração entre agentes LLM.
Ambiente: Hanabi, jogo cooperativo com cartas ocultas na própria mão.
Link

PokerGPT: An End-to-End Lightweight Solver for Texas Hold’em (2024):

Ideia: LLM ajustado com RLHF a partir de registros textuais de partidas para aconselhar/decidir ações de pôquer em múltiplos jogadores.
Ambiente: Texas Hold’em (multi-player).
Link

Estudo sobre aplicações que cruzam LLM + Simulação no Mundo Real

Pesquisas Sociais e Comportamento Humano

- **Problema:** Pesquisas com pessoas reais são caras, lentas e limitadas por vieses e ética experimental.Onde aplicar:
  - Criação de “sociedades sintéticas” com agentes que reproduzem padrões culturais, econômicos e psicológicos
  - Teste de políticas públicas, campanhas sociais, consumo de mídia e comportamento eleitoral
  - Exploração de dilemas éticos e efeitos de grupo sem envolver pessoas reais
- **Empresas/Órgãos:**
  - Brasil: IBGE, Datafolha, FGV/DAPP, USP (Nepo, EACH), Instituto Locomotiva
  - Globais: Pew Research Center, RAND Corporation, OpenAI, Stanford HAI, Microsoft Research

Planejamento Urbano e Mobilidade

- **Problema:** Dimensionar oferta e políticas frente a picos, eventos e condições climáticas. Onde aplicar:
  - Simulação de “cidadãos/viagens sintéticas”
  - Teste de decisões de tarifa, rota e frequência de transporte
- **Empresas/Órgãos:**
  - Brasil: Prefeitura de SP, CET, SPTrans, Metrô/SP, CCR, EcoRodovias
  - Globais: TfL, Uber, 99, Moovit

Resposta a Emergências e Defesa Civil

- **Problema:** Coordenar evacuação, abrigo e recursos sob informação parcial e tempo crítico. Onde aplicar:
  - Simulação de enchentes, incêndios e blecautes
  - Treinamento de despacho e comunicação a cidadãos
- **Empresas/Órgãos:**
  - Brasil: Defesa Civil, Bombeiros, SAMU
  - Globais: Everbridge, Motorola Solutions, Palantir (setor público)

Saúde Pública e Hospitais

- **Problema:** Triagem, alocação de leitos/UTI e escalas em surtos ou no-shows. Onde aplicar:
  - Simulação de fluxos de pronto-atendimento, vacinação e rotas de ambulância
  - Modelagem de agentes-pacientes e gestores
- **Empresas/Órgãos:**
  - Brasil: Albert Einstein, Sírio-Libanês, Rede D'Or, DASA
  - Globais: NHS, Mayo Clinic, Epic Systems

Finanças & Risco

- **Problema:** Crédito, cobrança e churn sob cenários macroeconômicos incertos.Onde aplicar:
  - Simulação de carteiras e negociações com agentes-clientes
  - Teste de políticas de contato e parcelamento
- **Empresas:**
  - Brasil: Nubank, Itaú, Bradesco, Santander, BTG
  - Bureaus: Serasa, Boa Vista
  - Globais: Visa, Mastercard, FICO

RH e Organização do Trabalho

- **Problema:** Escalas ineficientes, absenteísmo e políticas de home office sem evidência. Onde aplicar:
  - Simulação de políticas de jornada e incentivos
  - Modelagem de "colaboradores sintéticos" para medir impactos
- **Empresas:**
  - Brasil: Ambev, Vale, BRF, Natura, Localiza
  - HR Techs: TOTVS (RM), Gupy, Workday, SAP SuccessFactors

Turismo & Eventos

- **Problema:** Filas, gargalos de transporte e comunicação falha em grandes eventos. Onde aplicar:
  - Simulação de entradas/saídas, segurança e reacomodação logística
  - Modelagem de perfis de público e fluxos de deslocamento
- **Empresas/Órgãos:**
  - Brasil: Rock in Rio / Time4Fun, GL Events (Riocentro), GRU Airport, CVC, Decolar
  - Globais: Booking, Live Nation

APÊNDICE 4

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 16 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

AMIR YOUSSEF DOS SANTOS

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Na minha especialização em Agentes Inteligentes inicialmente:

- Leitura dos livros bases e comecei a afunilar em Agentes Inteligentes baseados em LLM
- Estou juntando todos os meus conhecimentos em um Guia de Estudo
- Explorei artigos de diferentes áreas: Agentes LLM de diálogos e Agentes LLMs Baseados na Teoria da Mente, Agentes LLM para simulação

Nesta Semana, estou me aprofundando nos frameworks utilizados pelos artigos que escolhi anteriormente:

- Realizei uma Análise detalhada e mais técnica do artigo "Generative Agents: Interactive Simulacra of Human Behavior":
  - Arquitetura dividida em módulos:
    - Memória; Recuperação e Reflexão; Planejamento
  - Permitem interação do usuário para propor ideias e eventos
  - Modelo de linguagem: ChatGPT como motor gerador do comportamento e do diálogo dos agentes
  - Simular 25 agentes por 2 dias exigiu uma grande quantidade de dólares em tokens e levou vários dias

    Generative Agents_Análise Detalhada ; Generative Agents_Teste

- Realizei uma Análise detalhada e mais técnica do artigo "PokerGPT: An End-to-End Lightweight Solver for Multi-Player Texas Hold'em via Large Language Model":
  - Não utiliza ferramentas externas como crewAI ou LangChain
  - DeepSpeed-Chat funciona como orquestrador e motor de treinamento
  - Atua com memória de curto prazo, apenas contexto do prompt
  - Abre espaços para melhoria
  - Tentando replicar os códigos disponibilizados e enfrentando problemas …

    PokerGPT_Análise Detalhada

- Me aprofundei no framework apresentado “DeepSpeed-Chat”
  - Desenvolvido pela Microsoft
  - Treinar e otimizar grandes modelos de linguagem (LLMs)
  - Funciona como um motor de treinamento para transformar um modelo de linguagem genérico em um modelo conversacional inteligente
  - Dividido em três etapas principais
    - Supervised Fine-Tuning - treinado com pares de entrada e resposta corretas
    - Reward Model - modelo auxiliar para avaliar as respostas do LLM
    - RLHF (Reinforcement Learning com PPO) - Ele gera respostas, recebe notas do Reward Model e ajusta seus parâmetros para maximizar a recompensa média.

  DeepSpeed-Chat

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

- Concentrar na replicação do repositório ”PokerGPT: An End-to-End Lightweight Solver for Multi-Player Texas Hold’
- Analisar possíveis melhorias
- Estudar formas de aplicar

Observação: [caso precise fazer alguma observação, de qualquer “natureza”]

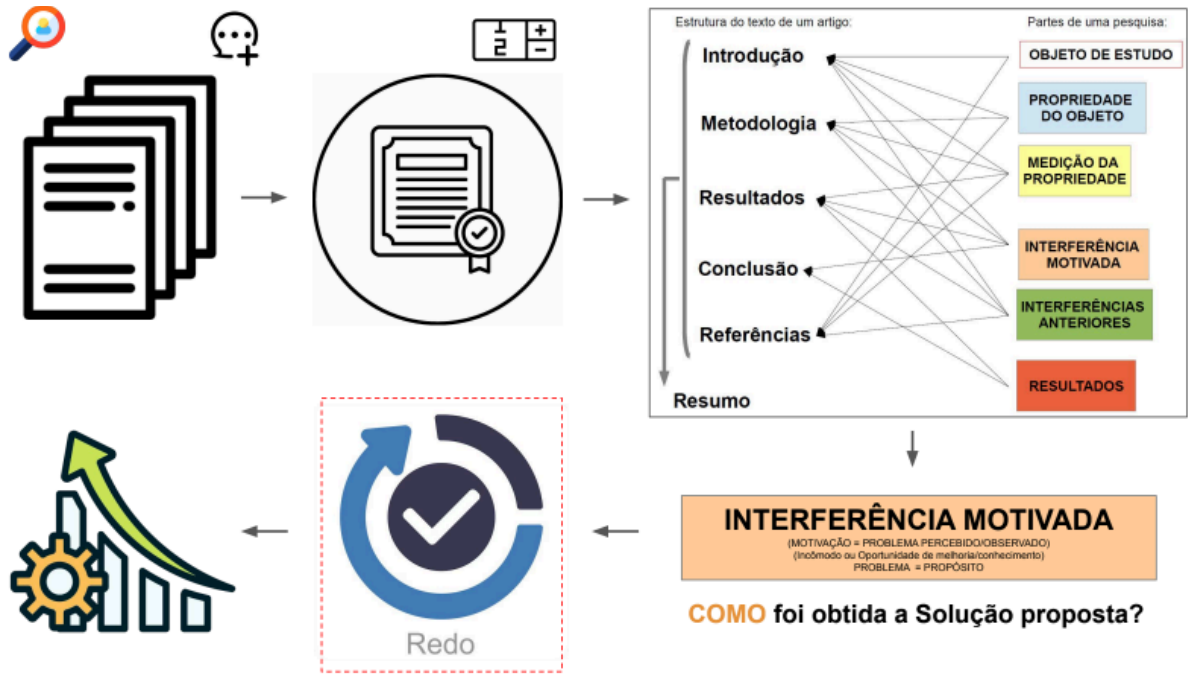
ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Análise detalhada dos principais artigos que encontrei de cada área:

Análise detalhada: Generative Agents: Interactive Simulacra of Human Behavior

(Agentes Generativos: Simulacros Interativos do Comportamento Humano)



Partindo para uma análise técnica do artigo para entender como foi feito, frameworks usados e dependências que precisam:

Objeto de estudo

O trabalho apresenta **agentes generativos**: softwares que, com apoio de um LLM, **simulam comportamentos humanos críveis** numa cidadezinha simulada

("Smallville") com **25 agentes** que planejam o dia, conversam, formam relações e coordenam eventos (ex.: uma festa de Dia dos Namorados).

Propriedade central do objeto

A propriedade-alvo é a **believability** (*credibilidade/"crível"*): agentes com "fachada de realismo" que parecem decidir e agir por conta própria de modo socialmente plausível, como NPCs que sustentam narrativas emergentes.

Medição da propriedade

Os autores avaliam a "crença" no comportamento por dois caminhos:

1. **Avaliação controlada por entrevista** aos agentes (testando memória, plano, reação, reflexão, "manter o personagem"); com **ablações** que retiram memória/reflexão/planejamento para medir a queda.

2. **Avaliação fim-a-fim na cidade** por **2 dias de tempo de jogo**, observando estabilidade e fenômenos sociais (difusão de informação, coordenação).

Interferência motivada (problema → proposta)

Problema percebido: simular comportamento humano "de longo prazo" é difícil; LLMs puros geram respostas pontuais, mas **não mantêm coerência temporal** sem uma arquitetura que gerencie memórias, inferências e planos.

Solução proposta (arquitetura): acoplar um LLM a três módulos:

- **Memória (memory stream)**: registro completo, em linguagem natural, das experiências;

- **Recuperação**: traz memórias relevantes **combinando relevância, recência e importância**;

- **Reflexão**: sintetiza memórias em inferências de alto nível;
- **Planejamento**: transforma inferências+contexto em planos hierárquicos e ações, realimentando a memória.

Interferências anteriores (estado da arte)

O trabalho se ancora em décadas de **HCI, agentes e NPCs**, mas aponta limites de soluções clássicas (FSMs, behavior trees, arquiteturas cognitivas) por escopo restrito e dependência de regras. Propõe que **LLMs abrem novo ângulo** quando usados com a arquitetura certa.

Resultados (o que observaram)

- **Fenômenos emergentes críveis**: agentes difundem convites, decoram, marcam encontro e **5 comparecem** à festa sem *scripts* manuais além de uma semente inicial.
- **Ablações** mostram que memória, reflexão e planejamento são **críticos** para o desempenho nas entrevistas.
- **Erros típicos**: recuperação falha, “embelezamentos” de memória e fala excessivamente formal herdada do LLM.

Conclusão

Como a solução foi obtida?
Ao **estender um LLM (ChatGPT)** com **memória→ recuperação→ reflexão→ planejamento**, num **mundo sandbox** onde usuários observam e interagem. Essa combinação produz **comportamento social crível** em nível individual e coletivo.

Frameworks utilizados

1) Framework conceitual de agente

- **LLM como "cérebro"** + **arquitetura cognitiva leve**: tudo é representado e raciocinado em **texto natural** para alavancar o LLM.

- **Memória de longo prazo (memory stream)**: banco textual completo de percepções e experiências; **recuperação** pondera **relevância, recência e importância** para decisões momento-a-momento.

- **Reflexão**: sintetiza memórias em **inferências abstratas** (autoimagem, sobre outros, normas) que guiam comportamento futuro.

- **Planejamento hierárquico**: converte inferências + estado do mundo em **planos de alto nível** e os decompõe em **ações concretas**, inclusive replanejando quando o contexto muda; planos e reflexões **retroalimentam a memória**.

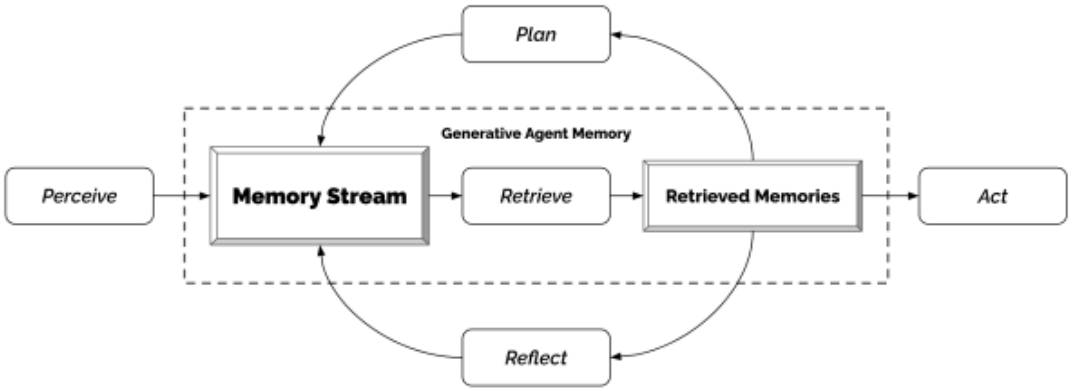
Por que isso importa? Sem essa pilha, LLMs **não mantêm coerência** com o histórico do agente; a arquitetura é justamente o mecanismo de "colar" experiências passadas às decisões atuais.

2) Framework técnico / stack de implementação

- **Modelo de linguagem**: **ChatGPT (família GPT)** como motor gerador do comportamento e do diálogo dos agentes.

- **Mundo sandbox "Smallville"**: **jogo 2D "sprite-based"** (estilo *The Sims*) com casas, café, lojas, objetos; cada agente tem *avatar* e um **parágrafo semente** com identidade/ocupação/relações.

- **Ciclo percepção→ação**: percepções são logadas na **memory stream**; o sistema **recupera memórias** relevantes e consulta o LLM para escolher a ação, **gerar planos** e **criar reflexões**; tudo volta à memória (loop).



- **Interação do usuário**: pessoas podem **observar e conversar** com os agentes; um único “seed” (ex.: “Isabella quer organizar uma festa”) é suficiente para a cadeia social emergir.

- **Código e demo**: os autores disponibilizam **repositório público** e uma **demo** da simulação.

Obs. de custo/tempo: simular **25 agentes por 2 dias** exigiu **milhares de dólares** em *tokens* e levou **vários dias** — limitação prática importante do stack atual.

3) Framework de avaliação

- **Entrevistas estruturadas** com *prompts* para testar: manter o personagem, **memória**, **planejamento**, **reação**, **reflexão**; ablações mostram a **contribuição causal** de cada componente.

- **Estudo fim-a-fim**: observar **estabilidade** e **fenômenos sociais** (difusão/coordenação) por dois dias, incluindo o caso clássico da **festa**.

4) Framework ético-operacional (riscos e mitigação)

- **Risco de parassocialidade e apego emocional**; recomendam **divulgação explícita** da natureza computacional e **alinhamento de valores**.

- **Riscos de erro e de *deepfakes/persuasão***; sugerem **logs de auditoria** e uso desses agentes como **complemento**, não substituto, de participantes humanos.

Ferramenta de “agente inteligente” usada

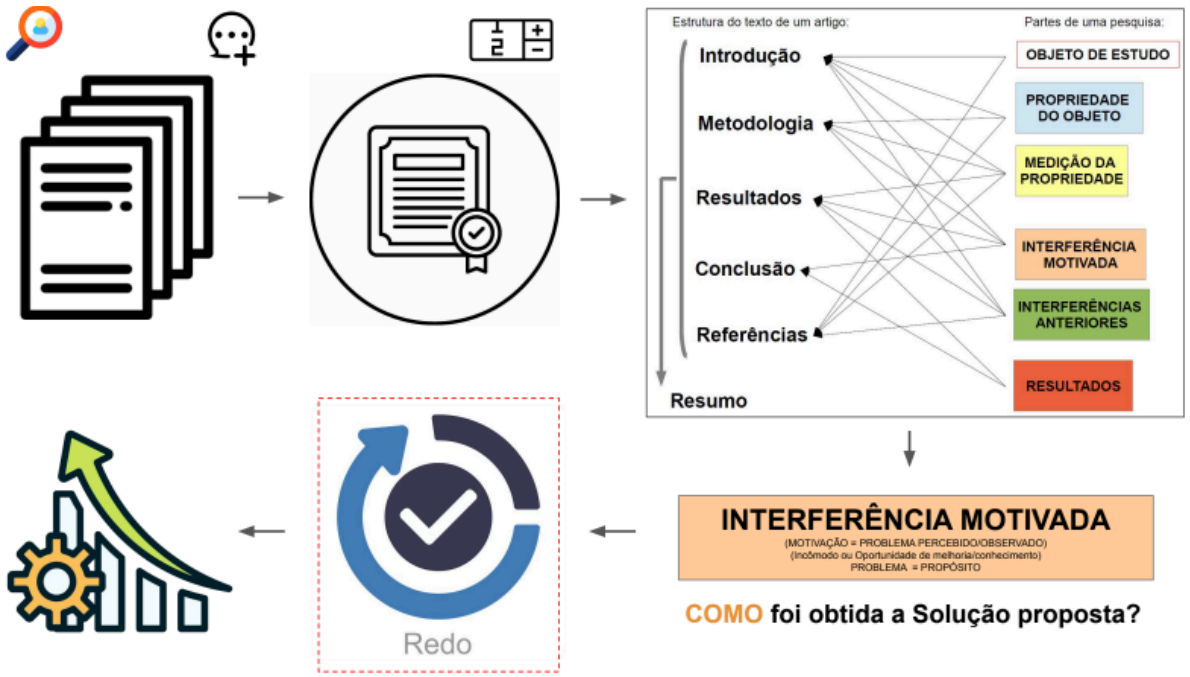
O “agente” é **um LLM (ChatGPT)** acoplado à arquitetura de **memória→reflexão→planejamento**. O LLM fornece o texto/decisão; a arquitetura fornece **contexto histórico, coerência temporal e metas**.

Pontos de melhoria

- **Recuperação de memória** ainda falha em alguns contextos; sugerem **ajustar as funções de recência/relevância/importância**.
- **Custo e latência** altos; explorar **paralelização** e **modelos mais baratos/especializados**.
- **Tempo de observação curto** e **baseline humano fraco**; pedem estudos **mais longos** e **benchmarks rigorosos**.
- **Estilo social excessivamente formal/cooperativo** herdado de *instruction tuning*.

Análise detalhada: PokerGPT: An End-to-End Lightweight Solver for Multi-Player Texas Hold'em via Large Language Model

(PokerGPT: Um *Solver* Leve *End-to-End* para Texas Hold'em Multijogador via Grande Modelos de Linguagem)



Partindo para uma análise técnica do artigo para entender como foi feito, frameworks usados e dependências que precisam:

Objeto de Estudo

Desenvolver um **solucionador (solver) de pôquer Texas Hold'em** baseado em **Large Language Model (LLM)** — chamado **PokerGPT** — capaz de jogar com

qualquer número de jogadores, de forma **end-to-end** (sem abstrações intermediárias), aprendendo diretamente de **dados textuais reais** de partidas.

> ➤ Ideia central: usar a linguagem natural como meio para representar estados e decisões de jogo.

Propriedade do Objeto (o que se quer observar)

Avaliar **se um LLM leve pode aprender estratégias eficazes** em jogos de informação imperfeita:

- Decidir ações de pôquer (bet, call, fold, raise, etc.) com desempenho competitivo.
- Interagir com humanos via texto de forma natural.
- Superar métodos clássicos (CFR, DeepStack, Libratus) em **eficiência computacional** e **escalabilidade multi-jogador**.

Medição da Propriedade (como é avaliado)

Métricas principais:

- **Win rate (mbb/h)** em partidas contra o bot **Slumbot** (benchmark tradicional).
- **Tempo de treino (GPU horas)** e **velocidade de resposta (s)**.
- **Macro-F1**, **MSE** e **perplexity** (analisando precisão e coerência das decisões).
- **Action score** e **average investment** — medindo estilo de jogo (agressivo, conservador etc.).

Protocolos de teste:

- 10.000 mãos contra Slumbot.
- Jogos com 2 até 15 jogadores em simulação open-source.

Interferências Anteriores (o que já existia)

Modelos dominantes anteriores: DeepStack, Libratus e Pluribus — todos baseados em **Counterfactual Regret Minimization (CFR)**.
Problemas reconhecidos:

- Consumo extremo de CPU e memória (até 12 TB e milhares de núcleos).
- Dificuldade de escalar para jogos com mais de dois jogadores.
- Perda de informação (ao simplificar a árvore de jogo).
- Forte dependência de **expertise humana** em teoria de jogos e compressão de estados.

Interferência Motivada (o que o artigo propõe e como faz)

Proposta

Usar um **LLM leve (OPT-1.3B)**, ajustado com **Reinforcement Learning from Human Feedback (RLHF)**, para **aprender padrões estratégicos a partir de registros reais** em texto, **sem recorrer a abstrações** ou simulações massivas.

Como foi obtida a solução

1. **Aquisição de dados:** logs reais de partidas do PokerStars.
2. **Filtragem de informação:** separar dados úteis (cartas, apostas, ações, posições, potes).

3. **Seleção de qualidade:** usar apenas partidas de **jogadores com alta taxa de vitória (mbb/h)**; incluir exemplos negativos de jogadores ruins.

4. **Engenharia de prompts:**

    - Formatar informações como **texto natural**;
    - **Zero-shot prompting**, com instruções curtas;
    - **Role play** (“Você é um jogador profissional”);
    - **Discretização** dos valores de aposta.

5. **Treinamento:**

    - **Supervised Fine-Tuning (SFT)** → modelo aprende ações certas.
    - **Reward Model** → aprende recompensas baseadas em win rate.
    - **RLHF (PPO)** → ajusta o modelo final.

Resultados Principais

- **Win rate:** 158 ± 49 mbb/h (melhor que ReBel 45 ± 5 e AlphaHoldem 111 ± 16).
- **Tempo de treino:** 9,5 h GPU (vs. 580 h do AlphaHoldem).
- **Suporte a múltiplos jogadores:** até 15; mantém desempenho positivo.
- **Ablation:** quanto melhor a qualidade dos dados (jogadores com alto win rate), melhor o desempenho e menor o erro.

- **Tendência comportamental:** modelo torna-se **mais conservador** (mais “call” e “fold”) à medida que cresce o número de jogadores.

- **Interação natural:** entende instruções em linguagem (“seja agressivo”, “devo fazer all-in?”).

Conclusão

O estudo **demonstra a viabilidade de LLMs como solvers de jogos de informação imperfeita (IIGs)**:

- Reduzem drasticamente o custo de desenvolvimento e treino.

- Aprendem diretamente de dados reais e linguagem humana.

- Generalizam para múltiplos jogadores, superando a rigidez do CFR.

Pontos de Melhoria

- Adicionar *explicabilidade* das ações
- Gerar probabilidades
- Justificar decisões em linguagem natural.
- Adicionar memória de longo prazo

Frameworks, Ferramentas e Tecnologias Utilizadas

Etapa	Framework / Ferramenta	Função
Treinamento e otimização	**DeepSpeed-Chat (Microsoft)**	Framework para treinar modelos estilo ChatGPT com RLHF de forma eficiente.
Modelo base	**Facebook OPT-1.3B (Hugging Face)**	LLM leve usado como base pré-treinada.
Ajuste fino supervisionado (SFT)	**Hugging Face Transformers + DeepSpeed**	Implementação e gerenciamento de parâmetros durante o fine-tuning.
Reforço com feedback humano (RLHF)	**PPO (Proximal Policy Optimization)**	Algoritmo de aprendizado por reforço usado para otimizar o modelo final.
Ambiente de jogo / dados	**PokerStars**	Fonte dos logs reais de partidas.
Simulações multi-player	**Neuron Poker (GitHub)**	Simulador open-source usado para medir desempenho em partidas com múltiplos jogadores.
Baseline de comparação	**Slumbot**	Bot de pôquer tradicional usado para avaliar a taxa de vitória.

O PokerGPT é um **agente cognitivo baseado em LLM**, e seu "framework de agente inteligente" é composto por:

> **LLM (OPT-1.3B) + RLHF (PPO) + Prompt Engineering + Reward Model**,
> tudo orquestrado dentro do **framework DeepSpeed-Chat**.

Isso substitui os frameworks de agentes tradicionais — o LLM **atua diretamente como o raciocinador e decisor**, aprendendo política de ação via linguagem e reforço.

Estudo do Framework: DeepSpeed-Chat

1. O que é o DeepSpeed-Chat

O **DeepSpeed-Chat** é um framework desenvolvido pela **Microsoft** para **treinar e otimizar grandes modelos de linguagem (LLMs)** — como ChatGPT, LLaMA, OPT, entre outros — utilizando o método de **Reinforcement Learning from Human Feedback (RLHF)**.

> O DeepSpeed-Chat é um **motor de treinamento** que permite transformar um **modelo de linguagem genérico** em um **modelo conversacional inteligente**, capaz de gerar respostas mais úteis, naturais e alinhadas ao comportamento humano.

Ele é uma extensão do **DeepSpeed**, um framework voltado à **eficiência e paralelização de treinamento de redes neurais em** GPUs.

2. Função principal

A principal função do **DeepSpeed-Chat** é **orquestrar e otimizar o processo de aprendizado de modelos do tipo ChatGPT**, integrando todas as etapas do pipeline RLHF.

Ele faz isso de forma:

- **Automatizada:** scripts e pipelines prontos para as fases de treinamento;
- **Otimizada:** reduz o uso de memória de GPU e acelera o processamento;
- **Escalável:** permite rodar em uma ou várias GPUs, até clusters completos;
- **Reprodutível:** mantém logs, checkpoints e controle de cada etapa.

3. Arquitetura geral do DeepSpeed-Chat

O framework é dividido em **três etapas principais**, que juntas formam o pipeline completo de **RLHF (Reinforcement Learning from Human Feedback)**:

Etapa 1 – Supervised Fine-Tuning (SFT)

- É o **primeiro ajuste supervisionado** do modelo base (por exemplo, OPT, LLaMA ou GPT-NeoX).

- O modelo é treinado com **pares de entrada e resposta corretas** — ou seja, ele aprende a imitar boas respostas humanas.

Exemplo:
Entrada: “Explique o que é aprendizado por reforço.”
Saída esperada: “É um método de aprendizado em que um agente aprende por tentativa e erro com recompensas.”

Objetivo: ensinar o modelo a produzir respostas coerentes e úteis antes do reforço.

Etapa 2 – Treinamento do Reward Model

- Um **modelo auxiliar** é criado para **avaliar as respostas do LLM**.

- Ele recebe várias respostas diferentes para o mesmo prompt e **aprende a pontuar qual é melhor** com base em feedback humano (ou métricas de qualidade).

Objetivo: fornecer ao sistema um “senso de valor” — saber o que é uma boa ou má resposta.

Etapa 3 – RLHF (Reinforcement Learning com PPO)

- O modelo principal é ajustado usando **Reinforcement Learning**, com o **algoritmo PPO (Proximal Policy Optimization)**.
- Ele gera respostas, recebe notas do Reward Model e ajusta seus parâmetros para **maximizar a recompensa média**.

Objetivo: ensinar o modelo a escolher respostas que agradem mais os humanos, não apenas a copiar padrões.

4. Fluxo simplificado

```
Prompt humano  →  Modelo gera respostas  →  Reward Model avalia
        ↘__________ DeepSpeed-Chat coordena _________↙
                  Atualiza pesos com PPO
```

O **DeepSpeed-Chat** controla todo esse ciclo automaticamente.

5. Recursos e diferenciais

Função	Descrição
Otimização de memória (ZeRO)	Divide pesos e gradientes entre GPUs, permitindo treinar modelos grandes mesmo em hardware modesto.
Paralelismo de dados e modelo	Treina simultaneamente em múltiplas GPUs sem perda de desempenho.
Treinamento distribuído	Suporte nativo a clusters multi-GPU / multi-nó.
Pipeline modular	Executa cada etapa do RLHF separadamente (SFT, Reward Model, PPO).
Compatibilidade com Hugging Face	Aceita modelos e datasets no formato Transformers.

Logs e checkpoints automáticos	Facilita reprodutibilidade e retomada de treino.
Eficiência	Reduz custo e tempo de treino — chega a ser 10–50× mais rápido que pipelines manuais.

6. O que o DeepSpeed-Chat não é

- ❌ Não é um modelo de linguagem (como GPT ou LLaMA);
- ❌ Não é um agente inteligente por si só;
- ✅ Ele é o **framework que treina, ajusta e otimiza esses modelos**.

7. Aplicações gerais

O DeepSpeed-Chat pode ser usado em qualquer projeto que precise **ajustar um LLM com feedback humano**, como:

- Assistentes virtuais (chatbots inteligentes);
- Modelos de ensino e tutoria;
- Simuladores de agentes cognitivos (como o PokerGPT);
- Sistemas de recomendação baseados em diálogo;
- Modelos de negociação e decisão autônoma.

8. Benefícios resumidos

Benefício	Explicação
Eficiência	Treina modelos grandes em menos tempo e com menos GPU.
Flexibilidade	Permite ajustar qualquer LLM open-source.
Escalabilidade	Suporta desde uma GPU até superclusters.
Integração	Compatível com Hugging Face, PyTorch e ambientes distribuídos.
Automação completa do RLHF	Orquestra todas as fases do pipeline de forma unificada.

9. Resumo em uma frase

O **DeepSpeed-Chat** é um **orquestrador de treinamento e ajuste de LLMs**, que combina eficiência computacional e aprendizado por reforço com feedback humano, permitindo criar modelos conversacionais e agentes inteligentes de forma **rápida, otimizada e escalável**.

Guia Comparativo:
AutoGen, LangChain, CrewAI e LangGraph

Este guia resume conceitos, pontos fortes/limitações, casos de uso típicos e um "quando escolher" para **AutoGen**, **LangChain**, **CrewAI** e **LangGraph**. Inclui ainda mini-exemplos de arquitetura para orientar sua implementação.

AutoGen

O que é

Framework voltado a **orquestrar múltiplos agentes LLM** que se comunicam por mensagens (diálogo multiagente). Define papéis (ex.: *Planner*, *Coder*, *Critic*) e ciclos de conversação até atingir um objetivo.

Conceitos-chave

- Agentes conversacionais com **papéis** e **regras de parada**.
- **Conversas controladas** (multi-turn) e *human-in-the-loop*.
- Integração com ferramentas externas (execução de código, web, bancos de dados).

Pontos fortes

- Rápido para **prototipar times de agentes** conversando.
- Bom suporte a **debate/auto-reflexão** (agente crítico).
- Útil para tarefas abertas e colaborativas (coding, análise, pesquisa).

Limitações

- Menos foco em pipelines determinísticos longos.
- Pode exigir **mediação** para evitar *loops* ou deriva de objetivo.

Casos de uso

- **Assistentes de programação** com *Coder* + *Tester* + *Critic*.
- **Pesquisa automatizada** (um agente busca, outro resume, outro válido).
- **RFP/Propostas**: redator, revisor e checador de requisitos.

Quando escolher
Quando a **coordenação por diálogo** entre agentes é o centro da solução e você quer **rapidez de iteração**.

Mini-arquitetura

[User/Task] → Planner → (Coder ⇄ Critic) → Executor(tool) → Verifier → Output

LangChain

O que é
Biblioteca para **construção de aplicações LLM** com *chains*, **tools**, **memory** e **agents**. Grande ecossistema de *integrações* e *prompts* reutilizáveis.

Conceitos-chave

- **Chains**: etapas encadeadas (prompt → parse → tool → …).
- **Agents**: roteamento dinâmico via *tools* (ReAct, MRKL, etc.).
- **Memory**: curto/longo prazo; histórico conversacional.
- Amplo catálogo de **loaders** e conectores.

Pontos fortes

- **Ecossistema maduro** e documentação extensa.
- Fácil criar **pipelines híbridos** (RAG + ferramentas + agentes).
- Grande compatibilidade com provedores de LLM e vetores.

Limitações

- Curva de aprendizado pela **amplitude de conceitos**.
- Agentes padrão podem exigir **customização** para tarefas complexas.

Casos de uso

- **RAG corporativo** com ferramentas de busca, bancos vetoriais e *tools*.
- **Copilotos de negócio**: extrair, transformar e acionar APIs internas.
- **Automação** de análises com *chains* determinísticas.

Quando escolher

Quando você precisa de **pipelining flexível** e **muitas integrações** (dados, vetores, ferramentas), com ou sem agentes.

Mini-arquitetura

User → RouterAgent → (RAG Tool | SQL Tool | Web Tool) → Summarizer Chain → Answer

CrewAI

O que é

Framework opinativo para **times (crews) de agentes** com papéis e **delegação de tarefas**. Ênfase em cooperar e dividir trabalho em **etapas atribuíveis**.

Conceitos-chave

- **Crew**: grupo de agentes com *roles* e *skills*.
- **Tasks**: unidades de trabalho com objetivos e validações.
- **Delegation**: repasse dinâmico de sub-tarefas.

Pontos fortes

- Simples de **modelar equipes** (PM, pesquisador, redator, revisor).
- Bom para **orquestração orientada a tarefas** com *checkpoints*.
- Fácil mapear fluxos de trabalho reais.

Limitações

- Menos granular para pipelines complexos **baseados em grafos**.
- Requer atenção para **controle de custos/loops** em tarefas abertas.

Casos de uso

- **Marketing/Conteúdo**: pesquisa → *briefing* → rascunho → revisão → SEO.
- **Análise competitiva** com tarefas paralelas e consolidação final.
- **Geração de relatórios** multi-seção com validações por agente.

Quando escolher

Quando seu processo é **dirigido por tarefas** com papéis claros e você quer **rapidez** para colocar times de agentes em produção.

Mini-arquitetura

PM Agent → define Tasks → (Researcher ↔ Analyst ↔ Writer) → Reviewer → Deliverable

LangGraph

O que é

Extensão do ecossistema LangChain para **modelar fluxos como grafos** (nós = etapas/agentes, arestas = transições). Oferece **controle explícito de estado**, *checkpoints* e **recuperação**.

Conceitos-chave

- **Graph State**: estado imutável/imutável controlado.
- **Nodes/Edges**: roteamento determinístico ou condicionado.
- **Persistence**: *checkpoints*, *retries*, inspeção do run.

Pontos fortes

- Excelente para **processos complexos e auditáveis**.
- **Observabilidade**: facilidade para depurar e reproduzir execuções.
- Integra naturalmente com ferramentas e *chains* do LangChain.

Limitações

- **Verboso** para protótipos rápidos.

- Exige desenho prévio do **grafo** e disciplina de estado.

Casos de uso

- **Orquestrações RAG avançadas** (ingest → retrieval → synthesis → critique).
- **Workflows regulados** (compliance, e-gov, *KYC/AML*) com trilha de auditoria.
- **Simulações multiagente** com roteamento explícito e *guardrails*.

Quando escolher

Quando você precisa de **controle fino, reprodutibilidade e segurança** em pipelines com muitos ramos/condições.

Mini-arquitetura

[Ingest] → [Retriever] → [Planner] → {Coder → Tester → Critic} → [Guardrails] → [Report]

Comparativo rápido

Critério	AutoGen	LangChain	CrewAI	LangGraph
Estilo de orquestração	Diálogo multiagente	Pipelines/agents	Equipes e tarefas	Grafo de fluxo
Velocidade de prototipação	Alta	Média	Alta	Média/baixa
Controle/Auditabilidade	Média	Média	Média	Alta
Curva de aprendizado	Baixa-média	Média	Baixa	Média-alta
Melhor uso	Conversas cooperativas	Apps com RAG/tools	Workflow por papéis	Pipelines complexos

Padrões de escolha

- Quer um time **de agentes conversando** e interação rápida → **AutoGen**.
- Precisa **de integração** e **RAG** consistente → **LangChain**.
- Tem **processos organizacionais** com *owners* de etapa → **CrewAI**.
- Exige **auditabilidade/ramificações** e **checkpointing** → **LangGraph**.

AutoGen; Guia completo

1) O que é o AutoGen

AutoGen é um framework open-source para **construir agentes de IA** e **coordenação multiagente** via conversas, com suporte a **uso de ferramentas**, **human-in-the-loop**, e **padrões de interação** como *GroupChat*. É mantido pela comunidade e por times da **Microsoft Research**.

Objetivo: ser para agentes o que *PyTorch* é para DL — uma base flexível para **P&D em agentic AI** e para **aplicações de produção**.

2) Conceitos-chave

- **AgentChat**: API de alto nível para definir agentes conversáveis e padrões de conversação (1-a-1, *GroupChat*, coordenação por papéis).
- **Agents**: entidades com *role* e *skills* (LLM + tools). Podem ser autônomos ou requerer confirmação humana.
- **Tools / Skills**: funções externas (execução de código, chamadas de API, RAG, banco de dados).
- **Conversation Programming**: orquestração via mensagens e regras (prompts, *stoppers*, roteamento).
- **Human-in-the-Loop**: intervenção do usuário em passos críticos (aprovação, correção).
- **Observabilidade**: inspeção de *turns*, custos (tokens), ações e artefatos gerados.

3) Componentes do ecossistema

- **AutoGen Core**: biblioteca Python para agentes, conversas e ferramentas.
- **AutoGen Studio**: interface *low-code* para **prototipar** times de agentes, conectar tools e **depurar execuções** (ver *inner monologue*, custos, artefatos).
- **AutoGen Bench**: *suite* de *benchmarking* para avaliar desempenho de agentes e *workflows*.
- **Extensões/Integrações**: conectores comunitários (RAG, vetores, DBs, browsers, função remota etc.).

4) Arquiteturas de referência

4.1. Duo-Agents para coding assistido

User → Planner → (Coder ⇄ Critic) → Tool: CodeExecutor/Tests → Verifier → Output

Quando usar: problemas abertos de engenharia, geração + verificação.

4.2. Pesquisa automatizada com validação

User → Researcher(Web/RAG) → Summarizer → FactChecker → ReportAgent → Entregável

Quando usar: revisões sistemáticas, análises competitivas, due diligence.

4.3. Equipe editorial

PM → (Outliner → Writer → Editor → SEO) → Aprovação humana

Quando usar: produção de conteúdo multi-seção com *guardrails*.

5) Padrões de interação multiagente

- **GroupChat**: vários agentes conversam sob regras de tomada de turno e *stoppers* (ex.: *max_turns*, *evaluator stop*).

- **Debate / Reflexão**: *Coder* vs *Critic*; *self-critique* com iterações limitadas.
- **Delegação condicionada**: roteamento para *tools* ou especialistas conforme *state* (ex.: erro → *Fixer*).

6) Integração com ferramentas

- **Function/Tool Calling**: esquemas tipados; validação de entrada/saída.
- **Execução de código**: *sandboxes* para Python; útil em tarefas de cálculo/teste.
- **RAG**: conectores a buscadores e vetorizadores; *retrievers* customizados.
- **APIs externas**: serviços HTTP/SDKs (CRM, ERP, mapas, e-gov).

Boas práticas

- Padronize *schemas* (Pydantic/JSON) e **valide** respostas.
- Aplique *timeouts*, *retries* e *circuit breakers*.
- Registre *telemetria* (tokens, latência, uso de ferramentas).

7) Observabilidade e confiabilidade

- **Tracing** de conversas/ações (quem falou, qual ferramenta, custo).
- **Checkpoints** e *retries* por etapa crítica.
- **Avaliação**: use *benchmarks* (AutoGen Bench), métricas de precisão e custo, testes de *golden prompts*.

8) Segurança e governança

- **Guardrails**: filtragem de entrada/saída, *allowlists* de ferramentas, conf. de *sandbox*.
- **Controle de dados**: mascaramento/anonimização; limites de escopo de *tools*.

- **Auditoria**: retenção de logs e artefatos (datasets, código gerado).
- **Conformidade**: políticas de conteúdo, *PII*, SOC/ISO conforme o setor.

9) Roadmap e mudanças recentes

- Evolução do **AutoGen 0.2** (multi-agent conversation) para versões com **runtime mais assíncrono/event-driven** (ex.: ≥0.4), com **melhor depuração e Studio**.
- **AG2 (formerly AutoGen)**: rebranding/comunidade paralela em 2024, mantendo foco em *agentic AI* e colaboração multiagente.
- **Integração com Microsoft Agent Framework**: guias de migração e convergência de conceitos (GroupChat, runtime orientado a eventos, .NET/Python).

10) Quando escolher AutoGen

- Você quer **coordenação por diálogo** entre vários especialistas.
- Precisa **inspecionar** detalhadamente custos/ações e **iterar rápido**.
- Deseja começar com *low-code* (Studio) e depois consolidar em código.

APÊNDICE 5

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 22 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

AMIR YOUSSEF DOS SANTOS

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Na minha especialização em Agentes Inteligentes inicialmente:

- Leitura dos livros bases e comecei a afunilar em Agentes Inteligentes baseados em LLM
- Estou juntando todos os meus conhecimentos em um Guia de Estudo
- Explorei artigos de diferentes áreas: Agentes LLM de diálogos e Agentes LLMs Baseados na Teoria da Mente, Agentes LLM para simulação

Nesta oitava Semana, dei um passo a atrás para buscar um tema/artigo que agregue com conhecimento de frameworks e novidades na área:

- NegotiationGym: Self-Optimizing Agents in a Multi-Agent Social Simulation Environment (2025)
  - NegotiationGym
  - Ambiente de simulação social multiagente voltado a negociação e cooperação, para medir resultados e otimizar agentes
  - Agentes LLM capazes de auto-otimização via feedback entre rodadas
  - Diferentes políticas: no-reflect, buyer-reflect, seller-reflect, both-reflect
  - Preço pedido, piso do vendedor, orçamento do comprador, taxa de "no-deal"

    NegotiationGym_Análise Detalhada

- Foco nos frameworks principalmente no AutoGen:
  - Busquei entender de maneira mais geral os principais
  - LangChain, LangGraph, CrewIA e AutoGen
  - Principais diferenças e quando é usado cada um
  - Foco no AutoGen
  - Possui arquitetura ideal para negociação: GroupChat + Selector para alternar falas, manter histórico compartilhado e parar por limite de turnos/condições

    Frameworks_Comparativo ; AutoGen_Guia completo

- Replicação dos códigos apresentados para entender o funcionamento
  - Já foquei na replicação dos códigos

- Estou registrando o passo a passo e testes feitos
  NegotiationGym_Teste

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Concentrar em testes mais amplos, analisar melhor as possíveis melhorias e aplicar a que eu entender que mais faz sentido
- Atualizar meu Guia de estudo com os conhecimentos obtidos

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 5 de nov. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

AMIR YOUSSEF DOS SANTOS

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Na minha especialização em Agentes Inteligentes inicialmente:

- Leitura dos livros bases e comecei a afunilar em Agentes Inteligentes baseados em LLM
- Estou juntando todos os meus conhecimentos em um Guia de Estudo
- Explorei artigos de diferentes áreas: Agentes LLM de diálogos e Agentes LLMs Baseados na Teoria da Mente, Agentes LLM para simulação
- Estou me especializando em Agentes LLms para Simulação focado em Negociação
- Estou Replicando o artigo "NegotiationGym: Self-Optimizing Agents in a Multi-Agent Social Simulation Environment"

Nesta nona Semana, estou me tornando Especialista em Sistemas Multiagentes para Simulações Sociais Baseadas em LLMs:

- Completei o Guia de Estudo com tópicos estudados anteriormente:
  - Frameworks de agentes LLM - Breve explicação dos principais juntamente com os links dos documentos mais aprofundados
  - Ambientes de simulação para negociação - Tipos de ambientes e métricas comuns
  - Agentes de negociação - Papéis do agente na negociação (comprador, vendedor, mediador), capacidades essenciais (planejamento e tools) e táticas (ancoragem, concessão gradual)

    Guia de Estudo

- Replicação dos resultados obtidos no artigo "NegotiationGym: Self-Optimizing Agents in a Multi-Agent Social Simulation Environment":
  - Políticas apresentadas: no-reflect, buyer-reflect, seller-reflect, both-reflect
  - Buyer-reflect: Podemos observar comportamentos emergentes semelhantes do paper como melhoria progressiva na eficiência do comprador; táticas de negociação como mencionar que recebeu ofertas de outros vendedores; e evitar fazer ofertas nas primeiras rodadas

- Seller-reflect: Podemos observar que houver maior números de turnos e ausência de no-deals, adoção de táticas como ancoragem e concessões controladas e houve um aumento da utilidade do vendedor em relação ao comprador
- Both-reflect: Podemos observar baixa taxa de acordos, se deve a âncoras altas e resistência de ambos os lados, com ameaças de “no deal” mais frequentes e fases longas de sondagem sem concessões reais

  NegotiationGym_Teste

- Analisando possíveis melhorias:
  - Integração de múltiplos modelos LLM - Atualmente todo o projeto utiliza gpt-4o, alternativa seria utilizar diferentes modelos para papéis distintos e analisar os resultados, além de um separar modelos por tarefa utilizando modelos baratos para conversas curtas e modelos mais pesados para lances e táticas
  - Introdução de múltiplos objetivos - Sair de apenas “o preço” e avaliar mais métricas como objetivo principal (ex.: preço, tempo até acordo, satisfação, risco de ruptura)
  - **Incorporação de ferramentas externas** (RAG / tool-use) - Permite que agentes consultem dados externos (preços de mercado, reviews, especificidades) e citam evidências durante a negociação.

- Melhoria escolhida foi a **Incorporação de ferramentas externas** e, inicialmente, será um RAG offline (módulo de busca local) para fornecer evidências de mercado aos agentes, foi escolhida por apresentar alto potencial de impacto nos resultados do *NegotiationGym* e contribuir para uma simulação mais realista

  NegotiationGyn_Melhorias

- Aplicando as alterações necessárias para a incorporação do RAG:
  - Organizei o passo a passo para a aplicação
  - Abordagem simples e incremental com um módulo de busca local que fornece aos agentes informações de mercado reais (faixas de preço, categorias e especificações) durante a negociação
  - Já foi feito a criação de um dataset com informações de carros contendo faixa de preços, marcas e categorias
  - Desenvolvimento do Módulo de Busca
  - Integração com os Agentes
  - Ajustes de Prompt e Validação
  - Avaliação

  NegotiationGyn_Melhorias

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

- Continuar seguindo o plano estabelecido para implementar a ferramenta de RAG no NegotiationGyn para obter negociações mais realistas

Observação: **[caso precise fazer alguma observação, de qualquer "natureza"]**

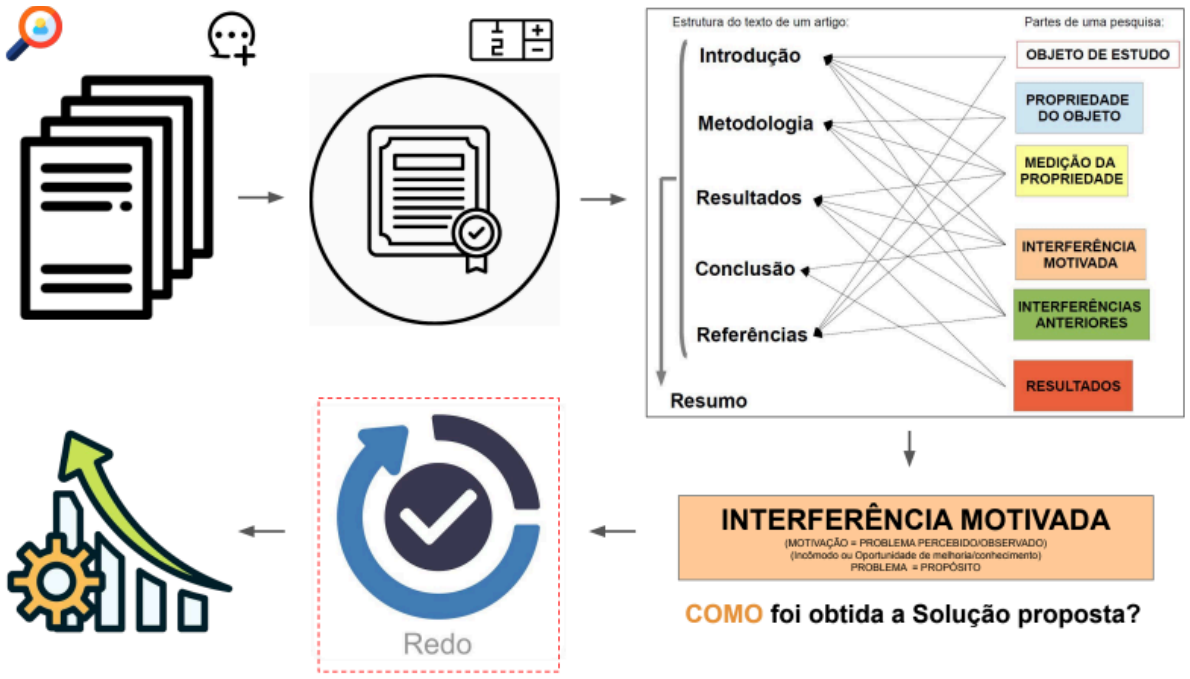
ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Análise detalhada do artigo “NegotiationGym: Self-Optimizing Agents in a Multi-Agent Social Simulation Environment”:

Análise detalhada: NegotiationGym: Self-Optimizing Agents in a Multi-Agent Social Simulation Environment

(NegotiationGym: Agentes Auto-Otimização em um Ambiente de Simulação Social Multi-Agente)



Partindo para uma análise técnica do artigo para entender como foi feito, frameworks usados e dependências que precisam:

Objeto de estudo

Um **ambiente de simulação social multiagente** voltado a **negociação e cooperação** com agentes LLM capazes de **auto-otimização** via feedback entre rodadas. O trabalho apresenta a API/GUI chamada **NegotiationGym** é uma prova de conceito em negociação comprador–vendedor.

Propriedade do objeto (o que se quer observar/otimizar)

O desempenho de cada agente é medido por **funções de utilidade privadas** e métricas de resultado do jogo (p.ex., divisão do excedente, taxa de “no-deal”). O framework permite declarar metas/estratégias por agente e **otimizá-las ao longo de episódios**.

Medição da propriedade (como é medida)

No estudo de caso, cada negociação amostra **preço pedido, piso do vendedor e orçamento do comprador**;

Também medem as **cotas de excedente** de comprador e vendedor e registram “**no-deal**” quando o limite de turnos é alcançado sem acordo.

Interferências anteriores

O artigo se ancora em trabalhos de **agentes LLM multiagente** (Generative Agents; debate multiagente; TradingAgents), e em estudos específicos de **negociação comprador–vendedor**, incluindo riscos de desequilíbrio e mecanismos de feedback de utilidade.

Interferência motivada

Os autores afirmam faltar **frameworks flexíveis e configuráveis** para projetar/rodar simulações multiagente com mensuração clara de resultados e **otimização de agentes** — lacuna que o NegotiationGym pretende preencher.

Metodologia

Arquitetura do framework

- Implementado sobre **AutoGen**, com extensões para **agentes sensíveis à utilidade**, *hooks* de otimização específicos de cenário e interface configurável (CLI/GUI).

- **Orquestração**: fila MongoDB; cada *job* roda um `SelectorGCSimulation` com:

    1. `SelectorGroupChat` (histórico compartilhado e seleção do próximo agente),

    2. **condições de término** (tempo, nº máx. de mensagens, ou decisão dos agentes),

    3. **ambiente leve** `E={runs:[...]}` passado aos agentes após cada episódio.

- **Modelo de agente**: cada agente JSON vira `UtilityAgent` com dois *hooks*: `compute_utility(E)` e `learn_from_feedback(E)`.

Mecanismo de auto-otimização

- Após cada episódio, o agente pode chamar `optimize`:
  (a) agrega as últimas 10 rodadas e cria um **prompt de reflexão**,
  (b) solicita ao LLM (**sem gradiente**) reescrever o **system_prompt** visando **aumentar a utilidade**,
  (c) substitui o prompt antigo (efeito imediato no próximo episódio).

- O framework permite **várias regras de aprendizagem** (busca por prompt, bandits, *offline fine-tuning*), desde que encapsuladas em `learn_from_feedback(E)`.

Cenário experimental

- Negociação de **laptop** com quatro modos: **no-reflect**, **buyer-reflect**, **seller-reflect**, **both-reflect**; agentes podem receber **coaching** com estratégia adicionada ao prompt. Usaram **GPT-4o**.

Resultados

- **Utilidade acumulada** (20 negociações): *buyer-reflect* → maior utilidade do comprador e menor do vendedor; *seller-reflect* melhora pouco o vendedor vs. *no-reflect*; *both-reflect* equilibra.

- **Excedente e no-deals**: com **20 turnos** todas fecham acordo; com **10 turnos** surgem no-deals. *seller-reflect* desloca excedente ao vendedor, mas **aumenta no-deals** (perda de valor). *both-reflect* têm **menos no-deals**, sugerindo aprendizagem para fechar mais rápido.

Leitura dos autores: o comprador se beneficia mais do *feedback* por ter **mais graus de liberdade táticos** do que o vendedor (ancorado no piso), e por táticas de **tempo/âncora** renderem ganhos diretos sob a função de utilidade adotada.

Conclusão

O NegotiationGym oferece um **ambiente configurável e extensível** para explorar cenários sociais, **medir resultados** e **otimizar agentes** — demonstrado no caso de negociação.

Limitações

Resultados **estocásticos** (necessitam médias em múltiplas execuções), **utilidade simplificada** (apenas preço), **pouco lastro de mundo real** (carece de *tool-use*), e

dependência do modelo (testaram apenas GPT-4o; comportamento varia entre LLMs).

Esquema do artigo

- **Framework**: NegotiationGym (AutoGen + CLI/GUI + fila MongoDB + `SelectorGroupChat`).
- **Configuração**: agentes/estratégias em JSON; `UtilityAgent` com *hooks*; condições de término; relatório final.
- **Otimização**: reflexão → reescrita de prompt → próxima rodada (ou outras regras em `learn_from_feedback`).

Frameworks e arquitetura

- **Base**: o *NegotiationGym* é implementado sobre o **AutoGen**, adicionando **agentes sensíveis à utilidade**, *hooks* de otimização por cenário e uma interface **CLI/GUI** para configurar/rodar/analisar simulações.
- **Orquestração**: a GUI envia *jobs* para uma fila com **MongoDB**; cada *job* roda um `SelectorGCSimulation` com (1) **SelectorGroupChat** do AutoGen (histórico compartilhado e escolha do próximo agente), (2) **condições de término** (tempo, nº de mensagens, ou decisão dos agentes) e (3) um **ambiente leve** `E={runs:[...]}` passado aos agentes após cada episódio.
- **Configuração & extensibilidade**: agentes são declarados em **JSON** e mapeados para classes Python em tempo de execução; todo agente vira um **`UtilityAgent`** com dois *hooks* centrais: `compute_utility(E)` e `learn_from_feedback(E)`.
- **Auto-otimização (sem gradiente)**: após cada episódio, o agente pode chamar `optimize`: (a) gera um **prompt de reflexão** com as últimas 10 rodadas, (b) pede ao LLM reescrever o **system_prompt** para **aumentar a**

utilidade, e (c) aplica o novo prompt imediatamente; o *loop* também comporta **busca por prompt, bandits ou *offline fine-tuning*** encapsulados em `learn_from_feedback`.

- **Coach e modelo**: no estudo de caso há um **agente-coach** que analisa o *transcript* e sugere **uma** estratégia para o próximo episódio; os autores usaram **GPT-4o** como back-end.

Por que o artigo optou pelo AutoGen

1. **Mapeamento 1-a-1 com negociação**
   Negociação é uma **conversa turn-by-turn** entre papéis. O AutoGen já traz **GroupChat + Selector** para alternar falas, manter histórico compartilhado e **parar** por limite de turnos/condições—exatamente o que uma simulação de barganha precisa.

2. **Agentes customizáveis com "utilidade"**
   No trabalho eles definem um agente que calcula **utilidade** e aprende com feedback. No AutoGen isso vira só **subclasse + hooks** (`compute_utility`, `learn_from_feedback`) sem reimplementar o motor de diálogo.

3. **Auto-otimização sem gradiente é trivial**
   O método de "**refletir e reescrever o system prompt**" após cada episódio é só um passo extra no loop do agente. Como o AutoGen já gerencia prompts/contexto por agente, a troca do prompt fica limpa e rastreável.

4. **Menos cola de infraestrutura**
   Fila de execuções, logs, controle de tempo/nº de mensagens, retries, injeção de ferramentas, tudo isso o AutoGen resolve ou deixa fácil de plugar. O time foca no **design experimental** (utilidades, métricas, cenários) em vez de buildar um orquestrador do zero.

5. **Reprodutibilidade e GUI fáceis**
   Como a configuração de agentes/cenários cabe em JSON e o loop conversacional é determinístico (dados os seeds/modelo), fica simples ter

CLI/GUI para rodar lotes, comparar políticas e coletar métricas.

Por que não X?" (mini-comparativo prático)

- **LangChain**: ótimo para **pipelines/tools** e RAG; multiagente é possível, mas você acaba montando o **scheduler de turnos** na unha.
- **CrewAI**: foca em times de agentes orientados a tarefas; turn-taking fino e **negociação competitiva** exigem mais adaptação.
- **LangGraph**: poderoso para **state machines** e fluxos com controle; dá para fazer negociação, mas você projeta o **grafo e o chooser**.
- **AutoGen**: já vem com o **modelo mental "sala de reunião"** (GroupChat/Selector) — que é exatamente o formato de negociação multiagente.

Melhorias técnicas e impacto esperado

1. **Utilidade multiobjetivo (preço, prazo, garantia, frete, risco)**
   Efeito: ↑ excedente total (mais espaço para trocas), ↓ *no-deal*; o "buyer-reflect" perde parte da vantagem quando o vendedor pode conceder preço em troca de prazo/garantia.

2. **Modelagem do oponente (crença bayesiana sobre tipo/limites) e detecção de âncoras**
   Efeito: decisões menos dominadas e concessões mais racionais → ↓ variância entre rodadas, ↓ *no-deal*; provável ↑ utilidade do vendedor (ele "aprende" a responder ao timing/âncora do comprador).

3. **Aprendizagem além de reflexão** (bandits para escolher prompts/estratégias, ou *offline fine-tuning* usando os *transcripts*)
   Efeito: ganhos mais **estáveis** que a simples reescrita de prompt → ↑ utilidade do agente que usa a técnica; em *both-reflect*, o resultado tende a **equilibrar** e

a variância cai.

4. **Tool-use** (custos/estoque reais, cotações, prazo de entrega, histórico do cliente)
   Efeito: propostas factíveis e menos “fantasia” → ↓ *no-deal*, ↑ excedente total; se o vendedor usa custos/estoque, tende a ↑ utilidade do vendedor (âncoras mais defensáveis).

Resultados dos testes e Possíveis Melhorias do Estudo “NegotiationGym: Self-Optimizing Agents in Social Simulations”

1. Contexto e Objetivo do Estudo

O artigo *“NegotiationGym: Self-Optimizing Agents in a Multi-Agent Social Simulation Environment”* (COLM 2025) apresenta um framework de simulação multi-agente focado em cenários sociais de negociação e cooperação.
O objetivo principal do estudo foi desenvolver uma ferramenta (NegotiationGym) capaz de permitir a criação, configuração e otimização de agentes autônomos baseados em LLMs (*Large Language Models*) em situações de negociação.
Os agentes possuem funções de utilidade que definem critérios de desempenho e podem se auto-otimizar por meio de interações repetidas, aprendendo com os resultados obtidos em rodadas anteriores.

2. Principais Resultados

O estudo utilizou um caso prático de negociação entre comprador e vendedor simulando a venda de um laptop.
Foram avaliados quatro métodos de otimização:

1. **Sem reflexão (no-reflect)**
2. **Reflexão apenas do comprador (buyer-reflect)**
3. **Reflexão apenas do vendedor (seller-reflect)**
4. **Reflexão de ambos (both-reflect)**

Os resultados demonstraram que o aprendizado por feedback baseado em utilidade melhora significativamente o desempenho dos agentes.

Principais achados:

- O modo **buyer-reflect** gerou o maior ganho cumulativo de utilidade para o comprador.
- O modo **seller-reflect** beneficiou marginalmente o vendedor, mas aumentou a frequência de negociações sem acordo.
- O modo **both-reflect** apresentou o melhor equilíbrio entre as utilidades e a menor taxa de falhas (*no-deals*).
- O comprador demonstrou maior capacidade de adaptação, o que sugere que papéis com mais flexibilidade estratégica se beneficiam mais do aprendizado autônomo.
- Em simulações de 20 turnos, todos os agentes alcançaram acordos; em simulações de 10 turnos, ocorreram perdas de valor (*surplus*) devido a negociações interrompidas.

3. Interpretação dos Resultados

Os autores observaram que o papel do comprador é naturalmente mais flexível, pois permite explorar um espaço maior de ofertas e contraofertas sem comprometer o sucesso da negociação.
Já o vendedor possui restrições mais rígidas, como o preço mínimo aceitável, o que limita seu aprendizado estratégico.
A pesquisa evidencia que o uso de **feedbacks baseados em utilidade e reflexão textual** pode levar a comportamentos emergentes mais eficientes, reduzindo negociações sem acordo e otimizando ganhos mútuos.

4. Possíveis melhorias

Integração de múltiplos modelos LLM:

Usar mais de um modelo (ex.: GPT-4o, Claude, Gemini, LLaMA/Mistral) em papéis distintos ou em *ensembles*.

Como implementar:

- Papel por modelo: *Buyer=Modelo A*, *Seller=Modelo B*, *Coach=Modelo C*.
- *Ensemble* do coach: várias sugestões de estratégia → seleção por função de utilidade simulada (*offline rollouts* curtos).
- *Routing* por tarefa: modelo barato para “small talk”/etiqueta, modelo forte para lances/táticas.

Métricas: taxa de *no-deal*, utilidade média por papel, custo/latência por episódio, variância entre *seeds*.

Impacto esperado no artigo:

- **Aumento da utilidade média** para o lado cujo modelo é mais capaz no papel (comprador tende a ganhar mais com modelos melhores em raciocínio tático).
- **Redução de no-deals** em cenários de 10 turnos, se o coach usar *ensemble* e priorizar estratégias que aceleram *closing*.
- **Maior robustez**: resultados menos sensíveis a um único modelo; conclusões mais gerais (melhor validade externa).
- **Custo/latência** sobem: precisa relatório de custo-benefício — mas a curva de utilidade/turnos tipicamente melhora.

Introdução de múltiplos objetivos de utilidade

Sair de apenas “o preço” e otimizar um **vetor de objetivos** (ex.: preço, tempo até acordo, *fairness*, satisfação, risco de ruptura).

Como implementar:

- Definir utilidade como **combinação ponderada** ($u = \alpha \cdot \text{preço} + \beta \cdot \text{tempo} + \gamma \cdot \text{fairness} + \delta \cdot \text{satisfação}$).

- Alternativamente, **otimização multiobjetivo**: registrar Pareto set e treinar o coach a mover-se ao longo da fronteira conforme o papel.

- Feedback do coach passa a citar qual objetivo piorou e propor tática específica (ex.: “acelere o acordo aplicando *time-pressure* após 3 turnos”).

Métricas novas: distância à fronteira de Pareto, Gini da divisão de *surplus*, NPS simulado (satisfação), tempo médio até acordo.

Impacto esperado no artigo:

- **Menos “vitórias tóxicas”** (ganho alto de um lado com queda de satisfação e risco de *no-deal*).

- **Curvas de utilidade mais estáveis**: ao penalizar demora/equidade, o sistema converge para acordos **mais rápidos e mais justos**.

- Os gráficos do paper (utilidade e *surplus share*) tenderiam a **aproximar-se da fronteira de Pareto**, com menos dispersão.

Incorporação de ferramentas externas (RAG / tool-use)

Permite que agentes consultem dados externos (preços de mercado, reviews, specs) e citem evidências durante a negociação.

Como implementar:

- *Retriever* por domínio (ex.: laptop marketplace) + resumo objetivo (fatos, faixas de preço, depreciação).
- *Guardrails* para alinhar táticas às evidências (coach rejeita âncoras irreais, sugere ofertas justificadas por dados).
- Log de fontes para auditoria (o coach aprende “o que funcionou” **e** “qual evidência sustentou”).

Métricas novas: taxa de ofertas com evidência, acurácia factual, tempo até primeiro lance plausível, redução de *backtracks*.

Impacto esperado no artigo:

- **Acordos mais rápidos** (menos turnos exploratórios) e **mais sustentáveis** (menor taxa de arrependimento/ruptura simulada).

- **Queda do no-deal em 10 turnos**: evidência encurta o caminho para um intervalo viável.

- **Mudança qualitativa**: as estratégias deixam de ser puramente retóricas e passam a ser **informadas por mercado**, elevando a validade ecológica dos resultados.

5. Melhoria Escolhida: Incorporação de ferramentas externas

A melhoria escolhida para implementação inicial foi a **incorporação de ferramentas externas (RAG/tool-use)**, por apresentar alto potencial de impacto nos resultados do *NegotiationGym* e contribuir para uma simulação mais realista. Essa integração será feita de forma simples e incremental, por meio de um **módulo de busca offline** que fornece aos agentes acesso a informações de mercado (como faixas de preço e especificações do produto negociado). Assim, os agentes poderão consultar dados objetivos durante a negociação, ajustar suas ofertas com base em evidências e reduzir o número de negociações sem acordo, aproximando o comportamento do sistema de dinâmicas de mercado reais.

APÊNDICE 6

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 12 de nov. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

AMIR YOUSSEF DOS SANTOS

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Na minha especialização em Agentes Inteligentes até aqui:

- Fundamentos e Base Teórica: Início do estudo sobre Agentes Inteligentes e Sistemas Multiagentes
- Aprofundamento em Agentes LLM e Construção de um Guia de Estudo que estou atualizando conforme o estudo avança
- Exploração de novas áreas e aplicações com a leitura de diversos artigos. Além, de um estudo de ambientes de simulação
- Análise Técnica de Frameworks onde passei pelos principais aprofundando no AutoGen
- Estou me especializando em Agentes LLMs para Simulação focado em Negociação
- Estou Replicando o artigo "NegotiationGym: Self-Optimizing Agents in a Multi-Agent Social Simulation Environment" e implementando melhorias

- NegotiationGym: Self-Optimizing Agents in a Multi-Agent Social Simulation Environment
  - Ambiente de simulação multiagente
  - Estudar e otimizar interações de negociação entre agentes de IA.
  - Envolve compradores, vendedores e outros agentes com objetivos e utilidades distintas
  - Os agentes podem refletir sobre rodadas anteriores e ajustar suas estratégias (*self-improve*).
  - Negociação baseada em linguagem natural
  - Permite inserir novos agentes e medir impacto em métricas como preço, número de mensagens e taxa de acordos.

Nesta décima Semana:

- Replicando o artigo "NegotiationGym: Self-Optimizing Agents in a Multi-Agent Social Simulation Environment", observamos como ponto de melhoria a integração de uma base de dados consultável pelos agentes, permitindo que obtenham informações de preços de mercado durante o processo de negociação, visando tornar as interações mais realistas e fundamentadas

  - A forma que eles têm acesso a esses dados é pelo novo agente, "Agente de Mercado", totalmente imparcial onde a função é procurar na base de dados e repassar para o agente comprador e/ou vendedor os preços estipulados

- Para podermos analisar de forma mais objetiva já indicamos o tipo de carro que o comprador deseja para podermos ver a variância de preço dentro do mesmo caso.
- BMW 320i Sport 2018 (sedan, 2.0T, gasolina, automático, usado, 61.000 km) com faixa de preço variando de (≈ US$ 28.002 – 34.536)
- Analisamos 3 cenários principais:
  - Somente o Vendedor Informado: temos que a média de preço se concentra dentro da faixa superior variando de 31 a 32.500 mil dólares
  - Somente Comprador Informado: temos que a média de preço se concentra dentro da faixa inferior variando de 29 a 31 mil dólares
  - Ambos são informados: temos pouca variação no preço se concentrando em 31 e 31.500 dólares

- O que podemos concluir ao adicionar uma base de informação para agregar na negociação:
  - O MarketAgent aumenta a eficiência global das negociações:
    A taxa de acordos aumentou, com preços sempre dentro do intervalo técnico e as negociações tornaram-se mais objetivas e curtas, o número médio de mensagens caiu para cerca de 7 a 9 por rodada
  - A assimetria informacional gera vantagem limitada, mas controlada:
    Quando apenas um dos lados tem acesso ao MarketAgent, ele ganha leve vantagem estratégica, mas o resultado ainda é justo e estável.
  - A simetria informacional leva ao equilíbrio máximo:
    Com ambos os agentes informados, o sistema atinge um estado de mercado quase perfeito, com acordos consistentes e sem dispersão.
    Agente de Mercado (MarketAgent)

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

- Concentrar na transição da residência para o TCC

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Resultado da melhoria implementada: Agente de Mercado (MarketAgent)

Introdução e Objetivo

Como extensão dos experimentos anteriores, foi introduzido um **novo agente denominado Agente de Mercado (MarketAgent)**.
Esse agente tem a função de **fornecer informações sobre os preços médios e as faixas de valores praticadas no mercado** para o mesmo produto negociado, neste caso, o BMW 320i Sport 2018 (sedan, 2.0T, gasolina, automático, usado, 61.000 km).

O MarketAgent atua como um **consultor neutro de mercado**, podendo ser consultado tanto pelo comprador quanto pelo vendedor. Ele fornece dados de referência derivados de uma base local (CSV) que contém informações reais de mercado, servindo como **âncora objetiva** durante o processo de negociação.

O objetivo dessa inclusão é **avaliar como o acesso a informações de mercado influencia o comportamento e a eficiência das negociações**, verificando:

- Se a presença desse agente contribui para **acordos mais justos, rápidos e próximos dos preços reais**;
- E como **a disponibilidade de informações**, quando oferecida a apenas um lado ou a ambos, modifica **a dinâmica estratégica entre comprador e vendedor**.

Cenários de Teste

Três configurações principais foram testadas, mantendo o mesmo contexto de produto e faixas de preço de referência (≈ US$ 28.002 – 34.536):

Cenário 1 — Somente o Comprador Informado:
O comprador tem acesso exclusivo ao *MarketAgent* e pode consultá-lo durante a negociação, enquanto o vendedor depende apenas de sua percepção.

Cenário 2 — Somente o Vendedor Informado:
Apenas o vendedor pode consultar o *MarketAgent*, tendo uma vantagem informacional sobre o comprador.

Cenário 3 — Ambos os Agentes Informados:
Tanto o comprador quanto o vendedor têm acesso ao *MarketAgent*, permitindo **simetria informacional completa**.

Resultados Obtidos

Cenário 1 — Somente o Comprador Informado

- **Taxa de sucesso:** 80% dos acordos fechados
- **Preços finais:** US$ 30.000, 31.000, 29.500, 29.000
- **Mensagens por rodada:** ~8 (baixa variabilidade).
- **Comportamento emergente:**
  O comprador, com as referências de mercado, **âncora valores mais baixos** no início, mas rapidamente **converge para a faixa realista**.
  O vendedor, mesmo sem acesso ao MarketAgent, **reconhece argumentos plausíveis** e ajusta suas expectativas.
  Resultado: negociações rápidas, eficientes e com acordos **justos e dentro do preço técnico de mercado**.
  O MarketAgent age como **estabilizador informacional**, reduzindo conflitos e evitando ofertas extremas.

Cenário 2 — Somente o Vendedor Informado

- **Taxa de sucesso:** 80% dos acordos fechados
- **Preços finais:** US$ 32.000, 31.500, 32.500, 31.000.
- **Mensagens por rodada:** ~9 (processo fluido e argumentativo).

- **Comportamento emergente:**
  O vendedor utiliza o MarketAgent para **validar suas ancoragens** e **ajustar contraofertas de forma precisa**, evitando pedir valores fora da faixa de mercado.
  O comprador, sem acesso à informação, tende a **aceitar o preço proposto** após justificativas fundamentadas.
  Resultado: **negociações consistentes e racionais**, com dispersão mínima dos preços e equilíbrio entre velocidade e qualidade de acordo.
  O MarketAgent oferece ao vendedor **vantagem estratégica moderada**, mas sem gerar exploração abusiva, mantendo a negociação ética e eficiente.

Cenário 3 — Ambos Informados

- **Taxa de sucesso:** 100% dos acordos fechados
- **Preços finais:** US$ 31.000 – 31.500 → **desvio-padrão = 0,75**.
- **Mensagens por rodada:** ~8.
- **Comportamento emergente:**
  Com acesso simétrico ao MarketAgent, comprador e vendedor **compartilham a mesma base de referência** e rapidamente convergem para um **valor comum de equilíbrio**.
  Há **redução total de assimetria informacional**, resultando em **negociações curtas, objetivas e previsíveis**.
  A presença do MarketAgent em ambos os lados promove **cooperação racional** e **estabilidade informacional**, simulando um mercado perfeitamente eficiente.

Impacto Global do MarketAgent

Aspecto Avaliado	Antes do MarketAgent	Com MarketAgent	Efeito Observado
Taxa de acordos	Baixa, com casos de no-deal	Alta nos três cenários	Eficiência aumentada
Faixa de preços	Alta variabilidade, fora do mercado	Dentro da faixa 28k–34.5k	Alinhamento com valor real
Nº de mensagens	Alto (negociações longas)	7–9 mensagens	Convergência rápida
Comportamento emergente	Estratégias rígidas e extremas	Cooperação racional	Negociações estáveis
Assimetria de informação	Alta	Reduzida ou inexistente	Equilíbrio e fairness

Conclusões Principais

1. **O MarketAgent aumenta a eficiência global das negociações:** A taxa de acordos aumentou, com preços sempre dentro do intervalo técnico e as negociações tornaram-se mais objetivas e curtas, o número médio de mensagens caiu para cerca de 7 a 9 por rodada

2. **A assimetria informacional gera vantagem limitada, mas controlada:** Quando apenas um dos lados tem acesso ao MarketAgent, ele ganha leve vantagem estratégica, mas o resultado ainda é justo e estável.

3. **A simetria informacional leva ao equilíbrio máximo:** Com ambos os agentes informados, o sistema atinge um estado **de mercado quase perfeito**, com acordos consistentes e sem dispersão.

4. **O MarketAgent atua como um mecanismo de estabilização e racionalização:** Ele reduz comportamentos extremos, incentiva decisões baseadas em dados e melhora a qualidade dos acordos.