UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO

Projeto de Inteligência Artificial- Q-Learning

Enzo Emanuel Costa Maia
Jackson Santana Carvalho Junior
Adam Guilherme Mendes Lima
Mateus Henrique Silva de Melo
Samuel Guimarães Silva

São Cristóvão, Sergipe
2026

Sumário

1. Descrição do Projeto

1.1 Introdução

A área de Inteligência Artificial tem avançado significativamente com a popularização dos Grandes Modelos de Linguagem (LLMs), sistemas capazes de gerar e processar linguagem natural de forma altamente coerente. Dentro desse contexto, a orquestração de chamadas e a gestão de contexto destacam-se por permitir que um agente converse de forma fluida, "lembrando-se" do que foi dito anteriormente.

Este trabalho tem como objetivo a implementação de um Agente Conversacional (Chatbot com memória), aplicando os conceitos de consumo de APIs e estruturação de pipelines a um problema prático e essencial: a manutenção do histórico de conversa. A escolha desse problema visa facilitar a compreensão dos desafios lógicos de uma aplicação baseada em LLM sem o uso de banco de dados, bem como permitir uma clara correspondência entre o pseudocódigo apresentado como base e a implementação prática desenvolvida pela equipe.

O foco principal do projeto é evidenciar como a manipulação do histórico e o truncamento de mensagens evitam o estouro de tokens do modelo, garantindo requisições eficientes. Dessa forma, o projeto prioriza a clareza conceitual, a fidelidade ao pseudocódigo e a facilidade de compreensão da orquestração.

1.2 Escopo do Projeto

O escopo deste projeto consiste no desenvolvimento de uma aplicação computacional completa, modular e didática que demonstre o uso de orquestração de APIs de LLM para a resolução do problema de manutenção de contexto em diálogos contínuos.

O foco principal é comprovar que a separação de responsabilidades (abstração de provedores) aliada a um algoritmo rigoroso de truncamento de memória permite o funcionamento contínuo do chatbot sem falhas de limite de processamento. Dentro do

escopo definido, o projeto contempla os seguintes tópicos:

- A implementação integral do Agente Conversacional, respeitando a estrutura de funções e passos do pseudocódigo utilizado como referência;
- A modelagem de uma arquitetura Agnóstica de Modelo (Model-Agnostic) baseada em interfaces orientadas a objetos, permitindo o consumo de múltiplas APIs (como OpenAI e Google Gemini) sem alteração da regra de negócio;
- O desenvolvimento de um gerenciador de contexto (ContextManager) que aplique o truncamento em formato de janela deslizante, com proteção estrita do *System Prompt*;
- O desenvolvimento de uma interface gráfica web que auxilie na interação com o agente e na validação do funcionamento da memória;
- A produção de documentação técnica detalhada, incluindo este relatório e configurações de versionamento para implantação em nuvem.

1.3 Descrição do Problema

O problema abordado neste trabalho consiste em permitir que um sistema interativo atue como um assistente virtual capaz de manter uma conversa contínua com o usuário, gerenciando dinamicamente o histórico de interações. O ambiente do sistema é estruturado em torno da comunicação com uma API externa (provedor de LLM), onde a natureza *stateless* (sem estado) dessas APIs exige que as mensagens anteriores sejam reenviadas a cada nova chamada para fornecer contexto.

O grande desafio lógico reside no fato de que essas APIs possuem limites rígidos de entrada (janela de contexto medida em tokens). Caso o histórico cresça indefinidamente, a requisição falhará por estouro de limite, ou gerará custos computacionais inviáveis.

O problema caracteriza-se, portanto, como um cenário de Engenharia de Prompt e Orquestração, no qual o algoritmo deve aplicar regras matemáticas e lógicas de truncamento — como a remoção das mensagens mais antigas em forma de fila (FIFO), mas excluindo rigorosamente as instruções de sistema do topo — garantindo que o volume de dados se mantenha dentro da capacidade de processamento do modelo, maximizando a coerência, a "persona" do agente e a utilidade das respostas ao longo da conversa.

1.4 Tabela de Correspondência com o Pseudocódigo

Linha do Pseudocódigo	**Linha do Código Python (core/agent.py)**	**Explicação da Correspondência**
current_msg = CREATE-MESSAGE(role, content) APPEND(history,	current_msg = ContextManager.create_message(...) self.history.append(current_	Constrói o dicionário padrão de comunicação (role/content) e adiciona a nova entrada do usuário na lista de memória local.

current_msg)	msg)	
payload = TRUNCATE-HISTORY(history, limit=config.max_tokens)	payload = ContextManager.truncate_history(self.history, limit, ...)	Isola o *System Prompt* (índice 0) e executa um loop de conversation.pop(0) para remover as mensagens mais antigas até que o peso em tokens seja menor que o limite.
api_response = CALL-LLM-API(config, payload)	reply_text = self.provider.call_api(self.config, payload)	Abstração da chamada HTTP. No código real, esta interface delega a requisição para as classes implementadas OpenAIProvider ou GeminiProvider.
reply_text = EXTRACT-CONTENT(...) APPEND(updated_history, reply_msg)	reply_msg = ContextManager.create_message("assistant", reply_text) self.history.append(reply_msg)	Pós-processamento: Extrai o conteúdo útil da resposta da API e o adiciona ao histórico para fornecer contexto na próxima interação.

2. Fundamentação Teórica

2.1 Orquestração de LLMs e Gestão de Contexto

Diferente de algoritmos de aprendizado local, o uso de LLMs modernos na nuvem baseia-se no consumo de serviços RESTful assíncronos que não mantêm estado (*stateless*). Isso significa que a responsabilidade da "memória" recai inteiramente sobre a camada de orquestração do software cliente.

A técnica empregada neste trabalho baseia-se no conceito de **Janela de Contexto Deslizante (Sliding Window Context)** acoplada à proteção de *System Prompts*. Os tokens (unidades básicas de processamento semântico) são os limitadores físicos da inferência. Ao criar um gerenciador que estima a quantidade de tokens consumidos, o orquestrador age como um filtro, eliminando o passado distante da conversa para preservar o presente imediato, enquanto mantém as diretrizes de comportamento do assistente (o *System Prompt*) permanentemente ancoradas no topo do *payload* enviado. Essa orquestração

inteligente é o pilar que diferencia um simples script de chamada de API de um Agente Conversacional robusto e aplicável comercialmente.

2.2 Visualização e Resultados Obtidos

Para atender à proposta visual e interativa do projeto, a interface gráfica foi desenvolvida utilizando a biblioteca **Gradio**, e o sistema foi estruturado para implantação em nuvem através da plataforma Hugging Face Spaces. A interface expõe o agente de forma clara e interativa.

Durante os testes de uso contínuo, observou-se que o Agente foi capaz de dialogar retendo o contexto imediato da conversa de maneira impecável. Ao forçar o limite de tokens predefinido na classe de configuração, o algoritmo de truncamento provou sua eficácia matemática: a API continuou respondendo sem erros de limite, esquecendo gradativamente as saudações iniciais, mas mantendo perfeitamente o seu comportamento principal graças à blindagem do *System Prompt*, comprovando a solidez da lógica implementada.

3. Conclusão

3.1 Considerações Finais

O projeto proposto atingiu com êxito todos os objetivos delineados, sendo pautado na implementação de um Agente Conversacional a partir do pseudocódigo de referência focado em Engenharia de Prompt e Orquestração.

A atividade nos permitiu compreender, na prática, os desafios arquiteturais da Inteligência Artificial moderna baseada em APIs, especialmente a manipulação de estruturas de dados e estimativa de tokens. O resultado foi uma aplicação não apenas funcional, mas modular e agnóstica de provedor (comportando OpenAI e Gemini de forma alternável e transparente). Isso demonstra que o aprendizado transcendeu a simples chamada de um modelo, alcançando o nível de Engenharia de Software aplicada à IA, onde um orquestrador garante consistentemente o fluxo contínuo e à prova de falhas na comunicação humano-máquina.

REFERÊNCIAS

- GRADIO. *Gradio Documentation - ChatInterface*. 2026. Disponível em: https://www.gradio.app/docs. Acesso em: 19 fev. 2026.
- GOOGLE GENAI. *Google Generative AI SDK for Python*. 2026. Disponível em: https://github.com/google-gemini/generative-ai-python. Acesso em: 19 fev. 2026.
- OPENAI. *OpenAI API Documentation*. 2026. Disponível em: https://platform.openai.com/docs/api-reference. Acesso em: 19 fev. 2026.
- RUSSELL, S. J.; NORVIG, P. *Inteligência Artificial: Uma Abordagem Moderna*. 3. ed. Rio de Janeiro: Elsevier, 2013.