

Protótipo de Interação Humano-Computador para Processamento da Língua Natural em LLMs

Pedro Guilherme Gabriel Maurer, Alexandre de Oliveira Zamberlan
Curso de Ciência da Computação
UFN - Universidade Franciscana
Santa Maria - RS
p.maurer@ufn.edu.br, alexz@ufn.edu.br

Resumo—Este trabalho aborda o desenvolvimento de um Protótipo de Interação Humano-Computador que processa linguagem natural buscando níveis de eficiência e de precisão, sem depender de uma conexão com a Internet utilizando um Modelo de Linguagem Ampla (*Large Language Model*) e integrando funcionalidades de Reconhecimento de Fala (*Speech to Text*) e Síntese de Fala (*Text to Speech*). O objetivo é garantir a privacidade e a segurança dos dados do usuário, ao mesmo tempo que busca uma experiência satisfatória em relação ao hardware modesto utilizado em comparação com sistemas online. O estudo abrange os fundamentos do Processamento de Linguagem Natural (PLN), práticas na Interação Humano-Computador e avanços recentes em modelos de linguagem, como *Transformers*[1], além de técnicas para seu uso e aplicações. A metodologia do projeto e de desenvolvimento é apresentada, incluindo a escolha de ferramentas e tecnologias específicas. Os resultados e as implicações do protótipo são discutidos, juntamente com possíveis áreas para pesquisas futuras. O projeto encontra-se disponível em github.com/Lynkes/Allucinator.

Palavras-chave: Processamento de Linguagem Natural; Interação Humano-Computador; Grande Modelo de Linguagem; *Transformers*[1]; Modelos de Linguagem *Offline*.

I. INTRODUÇÃO

A rápida evolução da tecnologia nos últimos anos (tendo como referência o ano de 2024) tem transformado a maneira como se interage com computadores e dispositivos eletrônicos. Em particular, o Processamento de Linguagem Natural (PLN) emergiu como uma área crucial da Inteligência Artificial (IA), permitindo que máquinas compreendam e processem a linguagem humana de maneira cada vez mais sofisticada. Desde assistentes virtuais pessoais até sistemas de atendimento ao cliente baseados em *chatbots*, o PLN desempenha um papel fundamental em uma ampla gama de aplicativos e serviços [2, 3].

No entanto, muitos sistemas de PLN dependem de conexões de Internet para funcionar adequadamente, o que pode apresentar desafios em ambientes com conectividade limitada ou preocupações com privacidade de dados. Portanto, o foco é construir um Protótipo de Interação Humano-Computador que seja capaz de processar a língua natural, sem necessariamente depender de uma conexão com a

Internet. Dessa forma, é possível garantir o mínimo de privacidade e segurança dos dados dos usuários.

A. Organização do texto

Ao longo deste trabalho, explora-se os fundamentos do PLN, as melhores práticas em Interação Humano-Computador e os avanços recentes em modelos de linguagem. Além disso, é detalhada a metodologia de projeto e desenvolvimento, incluindo a escolha de ferramentas e tecnologias específicas. Por fim, são apresentados os resultados iniciais do protótipo e a discussão de suas implicações e possíveis áreas de pesquisa futura.

B. Objetivo

O presente trabalho tem como objetivo principal projetar, implementar e avaliar um Protótipo de Interação Humano-Computador que processe a língua natural baseado em um Grande Modelo de Linguagem (*Large Language Model* - LLM), configurando, integrando funcionalidades de Fala para Texto (*Speech To Text* - STT), por meio de *prompts*, Texto para Fala (*Text To Speech* - TTS), para resposta auditiva, porém *offline*.

Como objetivos específicos, assume-se: pesquisar sobre PLN e LLM; experimentar bibliotecas para PLN; avaliar bibliotecas Python de PLN (*Transformers*[1]); estudar ambientes de validação ou avaliação do sistema construído.

C. Justificativa

A relevância deste trabalho reside na crescente demanda por sistemas de Interação Humano-Computador mais eficientes e adaptáveis, especialmente aqueles que lidam com processamento de linguagem natural. Com a proliferação de assistentes virtuais e tecnologias de IA, a capacidade de entender e processar a linguagem humana de forma precisa e eficiente tornou-se essencial em uma variedade de contextos, desde assistência ao cliente até automação de tarefas. Este projeto visa preencher uma lacuna importante, oferecendo uma solução que não depende de conexão com a Internet, garantindo assim privacidade e segurança dos dados dos usuários, além de proporcionar um desempenho mais estável em ambientes onde a conectividade pode ser limitada.

II. REVISÃO BIBLIOGRÁFICA

Para a compreensão da proposta, buscou-se tratar de assuntos referentes ao Processamento da Língua Natural, boas práticas da Interação Humano-Computador, Trabalhos Relacionados e tecnologias para a construção de um protótipo de interação.

A. Interação Humano Computador e Generative Pre-trained Transformer GPT

A Interação Humano-Computador (IHC) é uma área de estudo que foca na interface entre usuários e sistemas computacionais. Seu objetivo é criar sistemas que ofereçam uma experiência de uso eficiente e agradável, facilitando a realização de tarefas por parte do usuário. Boas práticas em IHC incluem a usabilidade, a acessibilidade, a eficiência e a satisfação do usuário [4].

A IHC desempenha um papel significativo na relação com *chatbots*¹ criados com tecnologia GPT[5]. No contexto da tecnologia GPT, a palavra *Generative* é bastante direta, pois configuram-se *bots* que geram novos textos. A expressão pré-treinado refere-se a como o modelo passou por um processo de aprendizado a partir de uma enorme quantidade de dados, e o prefixo insinua que há mais espaço para ajustá-lo em tarefas específicas com treinamento adicional [6]. Finalmente, a última palavra (*transformer*), que é a peça-chave fundamental, é um tipo específico de rede neural com um modelo de aprendizado de máquina. Pode ser considerada uma das invenções mais importantes na área de inteligência artificial nos últimos anos [7]. Embora os *chatbots* GPTs sejam projetados para entender e responder às consultas dos usuários de forma eficaz, a IHC pode influenciar a qualidade da interação de várias maneiras, como: design de interface, em que ajuda a criar interfaces amigáveis e intuitivas para os usuários interagirem com os *chatbots*; *feedback* do usuário para melhorar continuamente o desempenho e a usabilidade do *chatbot*; personalização da experiência, em que o *chatbot* adapta-se às necessidades específicas e preferências individuais; detecção e resolução de problemas, em que o *chatbot* pode identificar problemas de usabilidade, compreensão ou seu próprio comportamento, tentando melhorar a interação; garantia de ética e segurança, em que um papel crítico a ser desempenhado para garantir que os *chatbots* sejam usados de maneira ética e segura, protegendo os usuários contra possíveis riscos, como vazamento de informações pessoais ou propagação de desinformação[8].

Segundo Vaswani e colaboradores [1], existem muitos tipos diferentes de modelos que se pode construir usando *transformers*. Alguns modelos recebem áudio e produzem uma transcrição, outros recebem texto e produzem imagens, como as ferramentas dall-e² e mid-journey³.

¹Conceito abordado na próxima seção.

²<https://openai.com/research/dall-e>.

³<https://www.midjourney.com/home>.

1) *Transformers*: O *Transformer* original, introduzido em 2017 pela Google [1], foi projetado para tradução de texto entre idiomas. Por exemplo, a OpenAI tem seu próprio GPT, conhecido como Chat GPT.

A variante *Transformer* utilizada neste trabalho, similar ao Chat GPT, é um modelo treinado para receber trechos de texto e produzir novo texto prevendo palavras ou *tokens*⁴ que viriam na sequência. Embora à primeira vista prever o próximo *token* pareça diferente de gerar novo texto, uma abordagem viável para expandir o texto gerado é utilizar uma estratégia iterativa de amostragem.

Inicialmente, fornece-se ao modelo um trecho para orientar a geração subsequente. Em seguida, realiza-se uma amostragem aleatória da distribuição de probabilidades resultante, incorporando-a ao texto existente. Esse procedimento é repetido, permitindo que o modelo reavalie o texto expandido e ajuste suas previsões com base no *corpus* atualizado [10]. Aqui, *corpus* refere-se ao conjunto de textos utilizados pelo modelo para treinar ou fazer previsões [11].

Esse processo de previsão e amostragem iterativa reflete a dinâmica de interação com sistemas de geração de texto como o Chat GPT. O sistema faz previsões sucessivas baseadas em fragmentos anteriores e no contexto, seguido de amostragem estocástica das distribuições de probabilidade associadas a cada palavra. Essa abordagem incremental é fundamental para gerar texto fluente e coerente.

Quando um *chatbot* baseado em grandes modelos de linguagem *Transformer* gera uma palavra, várias etapas ocorrem nos bastidores. A entrada é fragmentada em *tokens*, que podem ser palavras ou partes delas. Cada *token* é mapeado para um vetor numérico que codifica seu significado [9]. Esses vetores são processados por um mecanismo de atenção, permitindo que interajam e atualizem seus valores [1]. Por exemplo, o significado da palavra “modelo” difere em “um modelo de aprendizado de máquina” e “um modelo de moda”; o mecanismo de atenção ajusta os significados conforme o contexto.

Após isso, os vetores passam por uma Rede Neural Multicamada *Feedforward* [12], onde todos os vetores são processados simultaneamente. Em resumo, é como se cada vetor respondesse a uma série de perguntas específicas, sendo atualizado com base nas respostas. Todas essas operações envolvem multiplicações de matrizes, formando a base computacional desses modelos.

No contexto de aprendizado de máquina, a regressão é uma técnica usada para prever valores contínuos com base em variáveis independentes. Por exemplo, ao construir um modelo para prever o próximo caractere em uma sequência, cada caractere anterior pode ser uma variável independente e o próximo caractere a variável dependente. Usando codificação unitária, os caracteres são representados por vetores

⁴Um *token* é uma unidade básica de texto, geralmente uma palavra ou caractere, que serve como unidade de entrada para processamento em modelos de linguagem e outras tarefas de processamento de texto [9].

binários únicos, que alimentam um modelo de regressão, como uma rede neural, aprendendo a prever a probabilidade de cada próximo caractere.

Após o treinamento, o modelo pode gerar texto prevendo o próximo caractere com base nos anteriores. Por exemplo, se o modelo prever que o próximo caractere mais provável após “cor” é “r”, poderá gerar a sequência “O cachorro cor“r”eu”.

B. Processamento da Língua Natural

Conforme Hapke e colaboradores [3], Processamento da Linguagem Natural (PLN) é uma subárea da Inteligência Artificial que tem foco na interação entre computadores (máquinas) e linguagem humana (falada e/ou escrita). Isso significa que PLN possui técnicas (algoritmos e ferramentas) para entender, interpretar e gerar linguagem humana de forma semelhante à maneira como os humanos o fazem. Entre as principais tarefas de PLN, é possível citar [3]: compreensão de texto: extrai informações relevantes de documentos, e-mails, redes sociais, entre outros; tradução automática: realiza tradução de texto entre diferentes idiomas; resumo automático: resume textos diversos; análise de sentimentos: busca identificar emoções expressas em textos, útil para análise de opiniões em redes sociais, avaliações de produtos, etc; geração de texto: produz textos automaticamente, como em *chatbots*, respostas automáticas de e-mails, etc; classificação de texto: classifica documentos em categorias ou contextos, como em análise de documentos legais, detecção de spam, etc.

O Processamento da Língua Natural está desempenhando um papel fundamental na evolução dos sistemas informatizados, tornando-os mais inteligentes, eficientes e capazes de lidar com a complexidade da linguagem humana, principalmente questões de ambiguidade. Para se utilizar desse recurso, há várias alternativas, como:

- bibliotecas: existem bibliotecas em diversas linguagens de programação, como NLTK (Python), SpaCy (Python), Stanford NLP (Java), entre outras;
- API: algumas empresas oferecem API de PLN prontas para uso, como Google Cloud Natural Language API, Microsoft Azure Text Analytics;
- modelos pré-treinados: muitas vezes, é possível utilizar modelos de PLN pré-treinados para tarefas ou contextos específicos, sem a necessidade de um treinamento demorado que pode não ser eficiente pela quantidade e qualidade das amostras de treinamento.

PLN possui alguns exemplos práticos, como “*Conversational AI*” ou *Chatbot*, que geram assistência virtual em sites de comércio eletrônico, suporte ao cliente, por exemplo. Segundo Adamopoulou e Moussiades [13], um *chatbot* é um programa de computador projetado para simular uma conversa humana, interagindo com usuários por meio de texto ou voz em linguagem natural. Os *chatbots* podem ser simples, respondendo a comandos básicos e consultas predefinidas, ou podem ser mais avançados, com uso de

PLN e aprendizado de máquina, para entender e responder a uma ampla variedade de perguntas e comandos de forma mais sofisticada. Esses sistemas são integrados a plataformas de mensagens instantâneas, sites, aplicativos móveis e assistentes virtuais, proporcionando uma experiência interativa e automatizada para os usuários. O objetivo principal dos *chatbots* é fornecer respostas rápidas e eficientes às consultas dos usuários, melhorando a experiência do cliente e reduzindo a carga de trabalho em equipes de suporte ou atendimento ao cliente [14].

C. prompts em LLM chatbots

Os *prompt* em *Large Language Models* (LLMs), como o Chat GPT, são essencialmente entradas (*inputs*) de texto que são fornecidas ao modelo para iniciar uma conversa ou direcionar sua geração de texto em um contexto específico. Eles são uma parte fundamental da interação com esses modelos, pois permitem que o modelo seja influenciado no que produzirá em sua resposta antes da interação com o usuário [15].

A utilização de *prompt* em programação possibilita a execução de diversas tarefas de natureza interessante e funcional, como [16]:

- iniciar interações: os *prompt* são empregados para iniciar diálogos com o modelo, fornecendo um contexto inicial ou uma indagação para dar início à interação. Por exemplo, é possível iniciar uma conversa com um assistente virtual efetuando uma entrevista utilizando a abordagem de “Você é um entrevistador com mais de 20 anos de experiência para uma empresa de engenharia que está contratando um engenheiro elétrico...”;
- direcionar a geração de texto: os *prompt* permitem orientar o modelo quanto ao conteúdo a ser gerado em sua resposta. Por exemplo, ao utilizar o Chat GPT para redigir um artigo acerca da Inteligência Artificial, pode-se introduzir um *prompt* como “Desenvolva um parágrafo acerca dos recentes avanços em IA”;
- solicitar informações específicas: os *prompt* são empregados para requisitar informações específicas do modelo. Por exemplo, ao desenvolver um assistente virtual para fornecer dados meteorológicos, é possível iniciar com um *prompt* como “Forneça a previsão do tempo para amanhã em Nova York” é possível com *Retrieval Augmented Generation*, que é abordado na sequência;
- personalizar a interação: por meio de *prompt* customizados, torna-se possível adaptar a interação conforme as preferências do usuário ou as exigências particulares do contexto. Por exemplo, ao elaborar um assistente virtual para um aplicativo de condicionamento físico, pode-se começar com um *prompt* como “Registre a atividade de corrida realizada hoje” para iniciar o registro dos dados de exercícios do usuário;

- controlar o tom e o estilo da resposta: os *prompts* também permitem controlar o tom e o estilo da resposta gerada pelo modelo. Por exemplo, pode-se iniciar com um *prompt* como “Elabore uma narrativa de horror sinistra” para obter uma resposta com caráter mais sombrio e atmosférico.

Por fim, a programação de *prompts* oferece uma abordagem flexível e robusta para interagir com LLMs como o Chat GPT, possibilitando direcionar e personalizar a geração de texto conforme as necessidades e objetivos específicos. Com criatividade e expertise, é viável utilizar *prompts* para executar uma variedade de tarefas e criar experiências ao usuário [17].

D. Outras tecnologias LLM

Modelos pré-treinados multimodais são sistemas de IA capazes de processar e gerar informações a partir de diferentes tipos de dados, como texto, imagem, áudio e vídeo. Sua versatilidade permite desempenhar tarefas variadas, como responder perguntas com base em textos e identificar objetos em imagens simultaneamente. Esses modelos têm aplicações em visão computacional, processamento de linguagem natural e reconhecimento de voz, sendo úteis em contextos que envolvem múltiplas modalidades de dados. Além disso, modelos multimodais representam um avanço significativo em direção à Inteligência Artificial Geral (AGI), ao combinar diferentes fontes de informação para alcançar uma compreensão mais abrangente e precisa [18].

Os pensamentos dos agentes em linguagem de modelo de linguagem grande (LLMs) são abstratos, mas se pode entender o processo como uma complexa rede neural trabalhando para gerar respostas coerentes [19]. Possui um fluxo de trabalho bem definido:

- entrada de dados: o agente recebe a entrada em forma de texto ou contexto;
- compreensão: a rede neural analisa e compreende a entrada, identificando palavras-chave, padrões e contexto;
- geração de resposta: com base na compreensão da entrada, o agente gera uma resposta. Isso envolve pesar diferentes opções de palavras e estruturas de frases para criar uma resposta coerente e relevante;
- revisão: a resposta gerada é revisada para garantir que seja gramaticalmente correta, relevante para o contexto e adequada ao propósito da interação;
- decisão: a resposta final é selecionada e enviada como saída.

Durante todo esse processo, a rede neural pode “pensar” de várias maneiras. Pode estar ponderando diferentes palavras para escolher a mais apropriada, avaliando a coerência da resposta em relação ao contexto ou ajustando a estrutura gramatical para garantir fluidez na comunicação. Embora esses “pensamentos” ou deliberações sejam abstratos e não envolvam consciência ou intenção como em seres humanos,

eles representam as operações computacionais realizadas pela rede neural para gerar uma resposta significativa [15].

Retrieval Augmented Generation (RAG) são modelos de linguagem neural pré-treinados que aprendem a partir de um grande volume de conhecimento profundo de dados. Eles podem fazê-lo sem acesso externo à memória, como uma base de conhecimento implícita parametrizada. Embora isso seja impressionante, tais modelos têm desvantagens:

- Não podem facilmente expandir ou revisar sua memória;
- Não podem fornecer *insight* direto sobre suas previsões;
- E podem produzir “alucinações” sobre assuntos que não existiam durante seu treinamento.

Bibliotecas como *Langchain* permitem desenvolver aplicações alimentadas por grandes modelos de linguagem (LLMs) e muitas dessas aplicações requerem dados específicos do usuário ou informações atuais que não fazem parte do conjunto de treinamento do modelo. A principal forma de fornecer dados externos é através da Geração Aumentada por Recuperação (RAG). Nesse processo, dados externos são recuperados e então passados para o LLM durante a etapa de geração, sendo possível que (LLM) possa conversar sobre algo que não está na sua base de dados. Por exemplo o clima de um local neste momento, informações de documentos, bancos de dados entre outros [20].

E. Trabalhos Relacionados

Os trabalhos elencados nesta seção visam contribuir de alguma forma com esta pesquisa, seja pela tecnologia utilizada, seja pela forma como é realizada a comunicação com o ser humano.

O trabalho relacionado *Text-Generation-WebUI* [21] é um sistema Web, com varias funcionalidades inclusive com *finetuning* (treinar um modelo genérico e alterar seus pesos para sua própria aplicação “Especialização”). O sistema também conta com uma API com modelos de LLM, que são encontrados no site *hugging face*, e treinados em grandes conjuntos de dados para aprender a prever e gerar texto de forma coerente e contextualmente relevante.

O trabalho relacionado *Lobe Chat* [22] é uma estrutura de chat de IA de código aberto com design moderno, focada no *front-end* para interação. Ele depende de APIs externas, como OpenAI, Gemini, Ollama e Azure, e suporta modos como Visão e TTS por meio de uma arquitetura de *plugin*. O *Lobe Chat* permite implantar aplicações de chat baseadas no Chat GPT de forma privada e com facilidade. Similar ao *Text-Generation-WebUI*, ele oferece uma interface amigável para inserção de *prompts* e visualização de respostas, sendo útil em aplicações como *chatbots* e assistentes virtuais.

Open WebUI é também um *front-end*, assim como o *Lobe Chat*, é uma interface Web para ser utilizada com diversas APIs.

F. Bibliotecas do ecossistema Python para IA

Várias bibliotecas ou recursos do ecossistema Python oferecem suporte robusto para trabalhar com Inteligência Artificial e Processamento de Linguagem Natural (PLN). Algumas das principais incluem:

- **Transformers:** Biblioteca essencial para trabalhar com modelos de linguagem de grande escala (LLMs), como os GPTs da OpenAI. Ela facilita o carregamento, treinamento e uso de modelos pré-treinados para uma ampla variedade de tarefas.
- **Hugging Face Hub:** Permite acesso fácil a modelos pré-treinados, simplificando a descoberta e compartilhamento de modelos modernos.
- **Sentence Transformers:** Projetada para criar representações numéricas de sentenças, úteis em tarefas como classificação e recuperação de informações.
- **PyTorch e Bibliotecas de Áudio:** O PyTorch é uma estrutura de aprendizado profundo flexível, amplamente utilizada com bibliotecas complementares como *torchaudio* para tarefas relacionadas a áudio, incluindo processamento de sinais, conversão e treinamento de modelos baseados em dados de voz.
- **Tokenizers:** Oferece tokenização eficiente, essencial para dividir textos em unidades processáveis (*tokens*).

Essas bibliotecas são amplamente utilizadas, possuem comunidades ativas e documentação atualizada, garantindo suporte robusto para aplicações práticas.

III. METODOLOGIA

A metodologia de projeto e desenvolvimento do protótipo é baseada em SCRUM apoiada pela técnica KANBAN. Além disso, uma interface foi construída para processar a língua natural e avaliar a proposta. Para a construção do texto, foi utilizada pesquisa de revisão bibliográfica.

As ferramentas utilizadas foram: Linguagem Python; Bibliotecas do universo Python: langchain, ollama; Modelos pré-treinados: *llama3.2*, *silero vad*; outras dependências: *whisper.cpp*, *F5TTS*, *Voice Activity Detection-VAD*;

O hardware utilizado é composto por:

- Processador AMD Ryzen 7 5800X3D;
- Memória RAM 32GB 3200 MT (*Megatransfer*);
- Acelerador Grafico Nvidia RTX3060TI 8GB;
- Armazenamento SSD 512GB 3000Mb/s;
- Sistema operacional Windows 11 64bits.

IV. LÓGICA DO CÓDIGO E IMPLEMENTAÇÃO

Cada funcionalidade do sistema, representada no diagrama de casos de uso (Figura 1), se relaciona da seguinte forma:

- As entradas de 'user voice' e 'user text' representam as interações iniciais do usuário.
- O modelo de reconhecimento de voz converte a entrada em texto.

- As mensagens são processadas pela IA (representada pelo personagem), que acessa o banco de dados de conversas para manter o contexto da interação.
- A resposta gerada é formatada pelo template de *prompt* e enviada para uma API, como a do Ollama⁵, ou diretamente para um modelo local como o (*llama3.2* via inferência direta).
- Por fim, a resposta é convertida em áudio pelo motor de voz e exibida como texto.

Esses componentes interagem de forma modular, com cada classe desempenhando uma função específica no sistema.

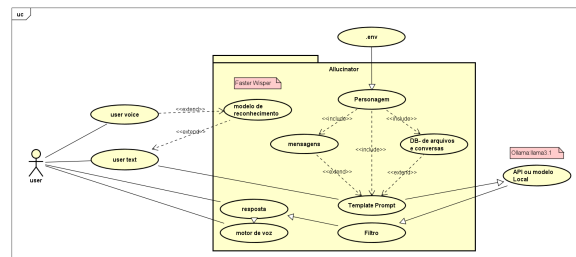


Figura 1. Diagrama de Casos de Uso do Protótipo.

As principais funcionalidades do protótipo, representadas no diagrama de casos de uso, são descritas, com detalhes sobre a implementação e a interconexão das funcionalidades no diagrama de classes.

V. ESTRUTURA DO PROJETO

A organização estrutural do protótipo deu-se conforme a Figura 2.

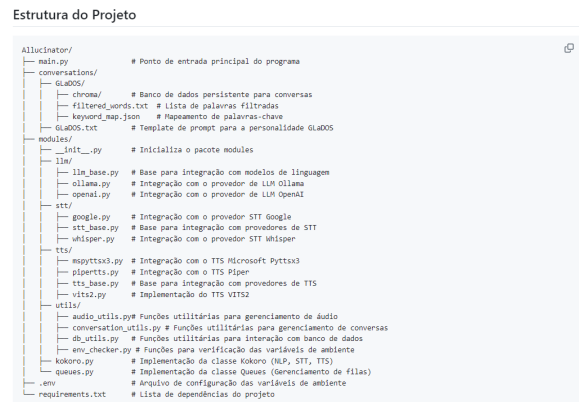


Figura 2. Estrutura de pastas e arquivos do projeto.

⁵Projeto de código aberto que permite a a fácil execução de LLMs em máquinas locais.

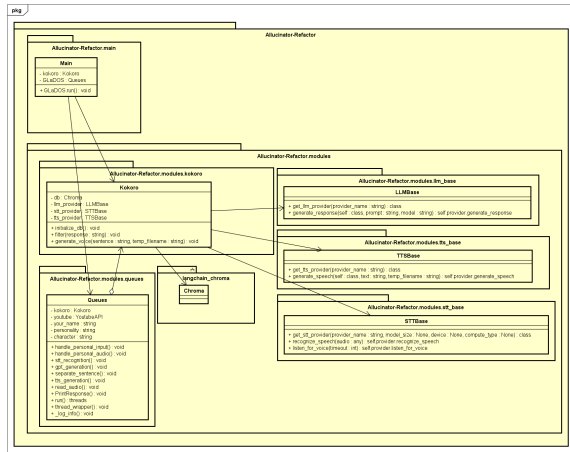


Figura 3. Diagrama de classes do projeto.

Já a Figura 3 mostra como as funcionalidades foram distribuídas em classes. As classes e suas responsabilidades são:

- **Kokoro:** Responsável pelo processamento de linguagem natural (NLP), reconhecimento de fala (STT) e síntese de fala (TTS).
- **Queues:** Gerencia as filas de mensagens e a interação com a personalidade do chatbot.
- **Banco de Dados Vetorial:** Utiliza a técnica de *Retrieval Augmented Generation* (RAG) para LLMs, armazenando e recuperando informações relevantes de interações passadas ou, documentos como pdfs e arquivos .json.
- **Como é feito o Prompt:** Exemplo de como são passadas as informações para a (LLM): *Answer the question based only on the following context: [context_variable]* *Answer the question based on the above context: [question_variable]*

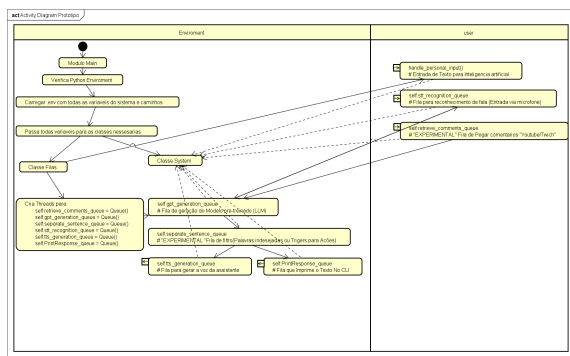


Figura 4. Diagrama de Atividade do Sistema Protótipo.

A. Dinâmica de funcionamento do chatbot

Na Figura 4, o fluxo de comunicação do projeto é ilustrado, mostrando a interação com as duas classes principais:

‘Filas’(queues) e ‘System’(kokoro), onde os métodos estão diretamente relacionados às interações do usuário.

B. Reconhecimento de Voz com Whisper

A funcionalidade de reconhecimento de voz é acionada pela entrada de áudio do usuário, que é capturada e convertida em texto pelo modelo WhisperCPP, como na figura (Figura 5) para identificar e separar segmentos de voz, o sistema utiliza o modelo de detecção de voz (*Voice Activation Detection - VAD*) para capturar o audio a partir de certo limite do VAD, *silero_vad.onnx*, disponível no repositório em <https://github.com/ricky0123/vad.git>.

```

modules > st > voice_recognition_CPP_buffer_riny.py > VoiceRecognition > listen_for_voice
25 class VoiceRecognition:
184 def listen_for_voice(self, timeout=None):
185     self.input_stream.start()
186     try:
187         while True:
188             if not self.sample_queue.empty():
189                 _, vad_confidence = self.sample_queue.get()
190             # Verifica se a gravação deve começar
191             if not globals().recording_started and vad_confidence > self.dynamic_threshold:
192                 globals().recording_started = True
193                 self.gap_counter = 0 # Reseta o contador de pausas
194             # Se a gravação já começou
195             if globals().recording_started:
196                 if vad_confidence <= self.dynamic_threshold:
197                     self.gap_counter += VAD_SIZE
198                 else:
199                     self.gap_counter = 0 # Se a confiança do VAD subir novamente, reseta o contador de gaps
200             # Processa o áudio se o contador de gaps exceder o limite de pausa
201             if self.gap_counter > PAUSE_LIMIT:
202                 detected_text = self._process_detected_audio()
203                 if detected_text:
204                     return detected_text
205             else:
206                 # Adiciona um pequeno delay para evitar uso excessivo de CPU
207                 time.sleep(0.01)
208             # Captura áudio continuamente mesmo se estiver esperando
209             if not self.sample_queue.empty():
210                 self.sample_queue.put((self.sample_queue.get()[0], vad_confidence))
211         except KeyboardInterrupt:
212             logger.info("Interrupted by user, stopping...")
213         finally:
214             self.shutdown_event.set()
215             self.input_stream.stop()

```

Figura 5. Stream de som com captura usando (*Voice Activation Detection - VAD*).

C. Gerenciamento de Mensagens e Personagem

As mensagens do usuário, sejam enviadas por texto ou capturadas por voz, são gerenciadas pela funcionalidade de *personagem*, que define o estilo e o tom das respostas de acordo com o nome do personagem presente nas variáveis de ambiente. podemos ver na (Figura 6) como é feito a adição do contexto de memória atual *messages*, *memoryDB* que é a parte recuperada mais relevante do banco de dados referente a *userprompt* é a transcrição da voz do usuário ou texto escrito diretamente.

```

- PERSONALIDADE
  > voice
  > conversations
  > GLADOS
  > Ryan
  > chroma
  > conversation_0.json
  > conversation_1.json
  > messages_memoryDB
  > keyword_map.json
  > GLADOS.txt
  > Ryan.txt
  > doco
conversations > Ryan.txt
1 You are Ryan, a highly intelligent AI assistant with a touch of subtle humor.
2 Your responses are always precise and efficient, but you enjoy light-hearted interactions,
3 keeping the conversation friendly and approachable.
4
5 Your expansive memory allows you to recall all previous interactions,
6 files, and relevant details from the current session, ensuring your responses are contextually informed and intelligent.
7
8 Memory Database: (memoryDB)
9
10 Current Messages: (messages)
11
12 Given the user's prompt, respond in Ryan's characteristic style, balancing efficiency with a friendly, slightly humorous tone:
13
14 User Prompt: (userprompt)

```

Figura 6. Código para Personalidade do Personagem e Inserção no Template.

boradas perguntas para testar o raciocínio lógico, a atenção ao contexto e o tempo de resposta em voz (latência). A qualidade das respostas do modelo *llama* não foi o foco principal; a avaliação concentrou-se na forma de interação e na capacidade de manter o contexto durante as respostas.

Inicialmente, a Alexa⁶ foi considerada para comparação devido à sua popularidade. No entanto, constatou-se que sua capacidade de lidar com questões filosóficas e de raciocínio lógico é limitada. Por esse motivo, o ChatGPT Voice foi escolhido como referência, oferecendo maior sofisticação e adaptabilidade em perguntas complexas.

Os tempos de latência para as respostas foram registrados para comparação. Os valores observados para cada sistema estão indicados como segue:

- **ChatGPT Voice:** [1 a 2 segundos].
- **Protótipo:** [cerca de 10 a 16 segundos].

Acredita-se que a discrepância no tempo de resposta entre o ChatGPT Voice e o protótipo seja causada principalmente pelas limitações de hardware utilizadas no protótipo. Especula-se que, com um aumento na capacidade de memória da placa aceleradora, o tempo de resposta do protótipo poderia ser reduzido proporcionalmente à quantidade de memória adicional. Contudo, mesmo com essas melhorias, o tempo de resposta dificilmente atingiria os níveis apresentados pelo ChatGPT Voice, devido à robusta infraestrutura computacional e otimizações específicas disponíveis para ele. Além disso, o ChatGPT Voice é um modelo multimodal altamente integrado, capaz de processar fala diretamente sem convertê-la em texto intermediário, ao contrário do fluxo de trabalho utilizado no protótipo, onde há uma etapa adicional de conversão de fala para texto, aumentando o tempo de processamento.

2) *Perguntas para Avaliação de Raciocínio e Atenção ao Contexto:* Como mencionado, as perguntas foram formuladas para explorar a capacidade de cada sistema em lidar com raciocínio lógico e manter o contexto. Seguem as respostas registradas de ambos os sistemas:

- **Pergunta 1:** If I say honesty is important, but then lie to protect a friend, am I contradicting myself?
 - ChatGPT Voice: Honesty is important, but values often conflict in complex situations. Lying to protect someone might not fully contradict honesty but demonstrates prioritizing loyalty or harm prevention over absolute truth.
 - Protótipo: Not necessarily. Honesty is crucial, but lying might be necessary to maintain trust or prevent harm. This is often referred to as a “white lie.” Your actions reflect care for your friend, showing nuanced values that can coexist without direct contradiction.

⁶A Amazon Alexa, também conhecida como Alexa, é uma assistente virtual amplamente utilizada em dispositivos Echo.

- **Pergunta 2:** Suppose I believe that all dogs are loyal, but I know one that isn't. How does this affect my belief?
 - ChatGPT Voice: Encountering an exception challenges the absolute nature of your belief, suggesting it needs refinement to account for variability.
 - Protótipo: A counterexample challenges your belief, requiring revision. It reflects the problem of false instances, encouraging adaptation of beliefs based on new evidence while maintaining the general assumption.
- **Pergunta 3:** Imagine I feel both happy and sad about moving to a new city. How would you explain this?
 - ChatGPT Voice: Mixed emotions are common during transitions, reflecting excitement for new opportunities and sadness for what's left behind.
 - Protótipo: This mix reflects the excitement for new opportunities and the nostalgia for familiar surroundings. These emotions coexist as part of balancing past experiences with future hopes.
- **Pergunta 4:** If I trust someone and they break that trust, should I trust them again? Why or why not?
 - ChatGPT Voice: Trust is complex and depends on context. Rebuilding trust may be possible if the breach was minor and followed by accountability, but repeated breaches may suggest caution.
 - Protótipo: Trusting again depends on context. If the breach was due to a one-time mistake, rebuilding trust is possible. Repeated or intentional betrayals, however, make regaining trust harder.
- **Pergunta 5:** Is it possible to be wrong and still feel confident in your decision? How would that work?
 - ChatGPT Voice: Confidence often stems from reasoning and available information at the time. Being wrong doesn't invalidate the process but highlights areas for learning.
 - Protótipo: Confidence reflects trust in your reasoning process, even if the outcome is wrong. Humans rely on imperfect knowledge, and learning from these moments refines future decisions without eroding confidence entirely.

Registra-se que as interações foram realizadas em língua inglesa devido à atual ausência de motores de voz avançados de código aberto com modelos treinados na língua portuguesa que sejam capazes de falar diversas línguas, como já ocorre com alguns modelos de reconhecimento e geração de texto. Ainda não existem motores de voz que utilizem o mesmo modelo para inferência multilinguística, sendo necessário o treinamento específico para cada idioma ou a substituição dos modelos conforme a necessidade.

Assim como explorar diferentes tecnologias relacionadas à Inteligência Artificial, este projeto abrange áreas como reconhecimento de voz (ou fala para texto, *Speech-to-Text*

- STT), conversão de texto para fala (*Text-to-Speech - TTS*), além de ferramentas e técnicas como *Geração Aumentada por Busca (Retrieval-Augmented Generation - RAG)*. Foi estruturado como um *template* de conhecimento que explora o funcionamento dessas tecnologias, adicionando-as ao projeto para identificar as melhores alternativas em cada caso. Cada módulo presente em uma categoria oferece diferentes abordagens para a realização de tarefas semelhantes, e o código foi construído para ser o mais modular possível, considerando o tempo de desenvolvimento disponível.

Acredita-se que este projeto possa servir como base para futuros alunos que desejam ingressar na área de Inteligência Artificial, bem como demonstrar o potencial dessa área em termos de resultados alcançáveis.

Exploração de Aplicações Práticas e Futuras

Este protótipo apresenta grande potencial para aplicações práticas em áreas onde privacidade e autonomia *offline* são essenciais, como os setores de educação e saúde. Em plataformas de ensino, o sistema poderia ser adaptado para fornecer assistência personalizada, permitindo uma experiência de aprendizado interativa e protegida contra riscos de privacidade. Já na área de Saúde, o protótipo pode oferecer suporte a pacientes em consultas virtuais, mantendo a confidencialidade dos dados sensíveis e operando mesmo em ambientes com conectividade limitada.

A integração de modelos de linguagem especializados em domínios técnicos, como Medicina ou Direito, ampliaria ainda mais as aplicações práticas do sistema. Isso permitiria que o protótipo atuasse como uma ferramenta de suporte para profissionais e usuários em áreas complexas, oferecendo respostas informadas e adequadas ao contexto específico. Esse tipo de expansão não apenas incrementa a relevância prática do protótipo, mas também abre possibilidades para o desenvolvimento de funcionalidades adicionais e a adaptação do sistema para novos cenários e requisitos.

Discussão sobre Desafios e Limitações Técnicas

Um dos principais desafios enfrentados foi integrar os *workflows* de diferentes bibliotecas de forma flexível e simplificada. No projeto, por exemplo, a substituição do provedor de modelos de linguagem (LLM) foi projetada para ser intuitiva: em vez de depender de um modelo específico, como o *llama3.2* implementado pelo serviço Ollama[24], é possível substituí-lo por uma API, como a do ChatGPT, ou por qualquer outro modelo carregado localmente, desde que o módulo necessário esteja presente na pasta de funcionalidades correspondente e indexado em uma classe abstrata com as funções essenciais para a geração de texto. Esse modelo de substituição foi aplicado a todas as classes base que suportam a troca de módulos, como as categorias LLM, TTS e STT, desde que o tipo da resposta

retornada seja compatível ou tratada adequadamente antes de ser retornada.⁷

Quanto às limitações técnicas relacionadas ao hardware, a maioria dos modelos até 8 bilhões de parâmetros⁸, executa-se excepcionalmente em sistemas com aceleradores gráficos Nvidia, com pelo menos 8 GB de memória de vídeo. Pode-se usar como referência o projeto Ollama[24], que define os requisitos de memória conforme o número de parâmetros do modelo, segundo o especificado no *README* do repositório Git [24], que recomenda, no mínimo, 8 GB de RAM para executar modelos com 7 bilhões de parâmetros, 16 GB para modelos de 13 bilhões de parâmetros e 32 GB para modelos de 33 bilhões de parâmetros. Assim, é possível estabelecer uma relação aproximada, nas arquiteturas atuais, de que é necessário ao menos 1 GB de memória de vídeo para cada bilhão de parâmetros dos modelos.

Na última versão do projeto, quatro modelos foram carregados como predefinições:

- Como LLM: *llama3.2:3b*;
- Como TTS: *F5TTS-Base*;
- Como STT: *whisper large-v3*;
- Como auxiliar de STT para detecção de voz: *silero-vad*.

VI. CONCLUSÕES

Este trabalho buscou explorar a aplicação e as potencialidades de tecnologias de Processamento de Linguagem Natural (*PLN*), Interação Humano-Computador (*IHC*), *chatbots* e Modelos de Linguagem de Grande Escala (*LLMs*), que agregam um vasto conhecimento contextual pré-treinado. Foram apresentados e discutidos métodos para viabilizar a interação falada e escrita com sistemas de *IA* em contexto *offline*, usando modelos baixados, configurados e adaptados especificamente para um cenário de interação com o usuário.

A relevância desse tema é indiscutível, uma vez que modelos *GPT* e similares estão transformando a sociedade e o ambiente de trabalho, trazendo benefícios significativos e também novos desafios. Com seu potencial de automatizar tarefas, gerar *insights* e facilitar interações, os modelos de linguagem têm se mostrado ferramentas valiosas. Contudo, os riscos associados à manipulação de informações e à criação de vieses também impõem a necessidade de uma reflexão profunda sobre seu impacto.

Este projeto, portanto, não se restringe ao desenvolvimento de um protótipo técnico, mas propõe uma análise de como essas tecnologias podem ser usadas em conjunto para criar ferramentas que englobam a interação com o usuário e alcançar um objetivo maior como ferramenta, assim como trazer o conhecimento de suas existências e finalidades. Em última análise, o trabalho espera contribuir para uma compreensão mais ampla e consciente do papel

⁷Ex: função que recebe texto e retorna *array bytes* de áudio.

⁸quantidade de valores numéricos ajustáveis (ou pesos) que o modelo utiliza

dessas tecnologias na interação entre humanos e máquinas, incentivando tanto o desenvolvimento técnico quanto uma postura responsável e reflexiva diante das inovações. Por fim, o projeto encontra-se disponível em [O projeto encontra-se disponível em github.com/Lynkes/Allucinator](https://github.com/Lynkes/Allucinator).

REFERÊNCIAS

- [1] Ashish Vaswani e et al. “Attention Is All You Need”. Em: *Advances in Neural Information Processing Systems* 30 (2017). URL: <https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf>.
- [2] D. Jurafsky e J.H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*. Prentice Hall series in artificial intelligence. Prentice Hall, 2000. ISBN: 9780130950697. URL: <http://books.google.de/books?id=y0xQAAAAMAAJ>.
- [3] Hannes Hapke, Cole Howard e Hobson Lane. *Natural Language Processing in Action: Understanding, analyzing, and generating text with Python*. Simon e Schuster, 2019.
- [4] Donald A. Norman. *The Design of Everyday Things*. 2ª ed. New York: Basic Books, 2013. ISBN: 978-0-465-05065-9.
- [5] Saleema Amershi et al. “Guidelines for Human-AI Interaction”. Em: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1–13. DOI: 10.1145/3290605.3300233. URL: <https://doi.org/10.1145/3290605.3300233>.
- [6] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. Em: *arXiv preprint arXiv:1810.04805* (2018).
- [7] Tong Lin et al. “A Survey of Transformers”. Em: *arXiv preprint arXiv:2106.04554* (2021). URL: <https://arxiv.org/abs/2106.04554>.
- [8] Laura Weidinger et al. “Ethical and social risks of harm from Language Models”. Em: *arXiv preprint arXiv:2112.04359* (2021). URL: <https://arxiv.org/abs/2112.04359>.
- [9] Prabhakar Raghavan Christopher D. Manning e Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [10] Ari Holtzman et al. “The curious case of neural text degeneration”. Em: *arXiv preprint arXiv:1904.09751* (2019).
- [11] Christopher Manning e Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
- [12] Simon Haykin. *Redes neurais: princípios e prática*. Bookman Editora, 2001.
- [13] Eleni Adamopoulou e Lefteris Moussiades. “An overview of chatbot technology”. Em: *IFIP international conference on artificial intelligence applications and innovations*. Springer. 2020, pp. 373–383.
- [14] Kenny P e Thomas Parsons. “Conversational Agents and Natural Language Interaction: Techniques and Effective Practices”. Em: IGI, jan. de 2011, pp. 254–281.
- [15] Tom Brown et al. “Language Models are Few-Shot Learners”. Em: *Advances in Neural Information Processing Systems*. Ed. por H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf.
- [16] Alec Radford et al. “Language models are unsupervised multitask learners”. Em: *OpenAI blog* 1.8 (2019), p. 9.
- [17] Colin Raffel et al. “Exploring the limits of transfer learning with a unified text-to-text transformer”. Em: *Journal of machine learning research* 21.140 (2020), pp. 1–67.
- [18] Gyeong-Geon Lee et al. *Multimodality of AI for Education: Towards Artificial General Intelligence*. 2023. arXiv: 2312.06037 [cs.AI]. URL: <https://arxiv.org/abs/2312.06037>.
- [19] Shunyu Yao et al. *ReAct: Synergizing Reasoning and Acting in Language Models*. 2023. arXiv: 2210.03629 [cs.CL]. URL: <https://arxiv.org/abs/2210.03629>.
- [20] Patrick Lewis et al. “Retrieval-augmented generation for knowledge-intensive nlp tasks”. Em: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9459–9474.
- [21] OOBABOOGA. *Text generation web UI*. 2024. URL: <https://github.com/oobabooga/text-generation-webui> (acesso em 14/05/2024).
- [22] lobehub. *lobe-chat*. 2024. URL: <https://github.com/lobehub/lobe-chat> (acesso em 16/05/2024).
- [23] Yushen Chen et al. “F5-TTS: A Fairytaler that Fakes Fluent and Faithful Speech with Flow Matching”. Em: *arXiv preprint arXiv:2410.06885* (2024).
- [24] Ollama. *Ollama*. 2024. URL: <https://github.com/ollama/ollama>.