

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
PROGRAMA DE GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO**

Uma Abordagem para Simulação de Comportamento Humano utilizando Modelos de Linguagem Natural

Carlo Smaniotto Mantovani

Orientador: Prof. Dr. Márcio Sarroglia Pinho

Porto Alegre

2024

AN APPROACH TO SIMULATING HUMAN BEHAVIOR UTILIZING NATURAL LANGUAGE MODELS

ABSTRACT

This project addressed challenges in simulating human behaviors in virtual environments and integrating natural large language models (LLMs) to guide avatar behavior, particularly in psychology. A system was developed to explore and understand psychological behaviors and conditions through interaction with a virtual avatar that responds emotionally with facial expressions to user questions. The specific objectives included implementing the avatar, integrating LLMs, and validating the system's effectiveness with feedback from experts in the field.

UMA ABORDAGEM PARA SIMULAÇÃO DE COMPORTAMENTO HUMANO UTILIZANDO MODELOS DE LINGUAGEM NATURAL

RESUMO

Este trabalho abordou desafios na simulação de comportamentos humanos em ambientes virtuais e na integração de modelos de linguagem natural de grande escala (LLMs) para guiar o comportamento de avatares, especialmente na psicologia. Assim, desenvolveu-se uma aplicação para explorar e entender comportamentos e condições psicológicas por meio da interação com um avatar virtual, que responde emocionalmente com expressões faciais a perguntas do usuário. Os objetivos específicos incluíram a implementação de um avatar, a integração de LLMs e a validação da eficácia do sistema por meio de professores da área.

SUMÁRIO

SUMÁRIO	3
1 INTRODUÇÃO	4
1.1 OBJETIVOS	5
1.2 ORGANIZAÇÃO DO TEXTO.....	6
2 FUNDAMENTAÇÃO TEÓRICA	7
2.1 LLMs	7
2.2 WORD EMBEDDING.....	9
2.3 RETRIEVAL AUGMENTED GENERATION (RAG)	9
2.4 UNITY	12
2.5 GERAÇÃO DE EXPRESSÕES FACIAIS EM AVATARES	12
2.6 ESTERÓTIPOS DE COMPORTAMENTO	13
2.7 VIRTUALIZAÇÃO, CONTÊINERS E DOCKER	13
3 TRABALHOS RELACIONADOS	16
4 PROJETO DO SIMULADOR	18
4.1 FUNCIONALIDADE DO SIMULADOR	18
4.2 DEFINIÇÃO DA PATOLOGIA PSICOLÓGICA	19
4.3 DEFINIÇÃO DO LLM.....	21
4.4 AMBIENTE DE DESENVOLVIMENTO	21
4.5 INFRAESTRUTURA DE INFERÊNCIA	21
4.6 VISÃO GERAL DA ARQUITETURA.....	21
5 IMPLEMENTAÇÃO DO SISTEMA RAG	23
5.1 CRIAÇÃO DO BANCO DE DADOS.....	23
5.2 EMBEDDING.....	25
5.3 INFERÊNCIA.....	25
6 DETALHES DE IMPLEMENTAÇÃO E EXECUÇÃO	28
6.1 INFRAESTRUTURA VIA DOCKER.....	28
6.2 AVATARES	29
6.3 INTEGRAÇÃO	30
6.4 EXECUÇÃO	31
7 RESULTADOS	32
8 TRABALHOS FUTUROS	36
9 REFERÊNCIAS	37

1 Introdução

Na psicologia, a falta de acesso a situações reais que exemplifiquem comportamentos de forma precisa e ética dificulta a formação de estudantes e profissionais, especialmente devido a restrições éticas ou de privacidade na observação direta de pacientes. Além disso, a aprendizagem e treinamento prático através de casos reais é limitada pela disponibilidade e diversidade de casos, afetando a capacidade dos profissionais em lidar com diversas situações na prática clínica. Assim, uma alternativa para uma solução desse problema é a simulação de comportamentos, por meio de avatares [1].

Nos últimos anos, a simulação de comportamentos humanos em ambientes virtuais enfrenta desafios em relação à diversidade e realismo dos comportamentos gerados. Alguns frameworks mais antigos para esse propósito fazem uso de suposições e teorias relacionadas cognitivas sobre o comportamento humano em situações reais [2], já frameworks mais novos utilizam esquemas de randomização e configuração expansiva para se obter uma diversidade de comportamento humano em diversos cenários [3]. No entanto, em ambos os casos, a capacidade de representar a complexidade do comportamento humano ainda é limitada.

Mais recentemente, modelos de linguagem natural, chamados Large Language Models (LLMs) têm surgido como uma forma de simular comportamentos humanos [4].

Neste contexto, a integração de LLMs para orientar o comportamento dos avatares virtuais, especialmente em áreas como psicologia, na qual a compreensão precisa de patologias e comportamentos associados é significativa para diagnóstico e tratamento eficazes, pode ser uma alternativa para os simuladores de pacientes. No entanto, os avanços nessa integração estão em estágios iniciais de desenvolvimento, por isto, este trabalho buscou demonstrar uma aplicação prática dessa tecnologia através do desenvolvimento de uma ferramenta que possa servir como suporte educacional, oferecendo a estudantes e profissionais de psicologia uma forma de explorar transtornos e comportamentos psicológicos em um ambiente virtual.

Para o desenvolvimento deste trabalho, se teve o apoio dos psicólogos Wagner Machado Lara do Programa de Pós-Graduação em Psicologia e psicólogo Bolivar Cibils, aluno de doutorado do prof. Lara.

1.1 Objetivos

Este trabalho visou desenvolver uma aplicação que aproveita os avanços em LLMs e a tecnologia de animação para simular comportamentos de pacientes com patologias psicológicas, em um ambiente virtual. A aplicação foi projetada para permitir a interação com avatares que respondessem a perguntas de forma contextualizada e expressassem emoções associadas às respostas geradas, baseando-se em informações clínicas.

Durante o desenvolvimento, foi criado um sistema que utiliza um banco de dados com informações sobre três transtornos psicológicos específicos: Transtorno Alimentar Compulsivo, Bulimia Nervosa e Anorexia Nervosa. Essas informações foram processadas e organizadas de maneira a permitir a recuperação eficiente de dados relevantes para cada contexto. As respostas textuais foram geradas a partir de modelos de linguagem, que também associavam uma emoção às respostas, permitindo a vinculação com os avatares virtuais.

Os avatares foram desenvolvidos para representação de expressões faciais correspondentes às emoções geradas. Além disso, foi implementada uma interface que permitiu ao usuário selecionar dentre os 3 transtornos, inserir novas informações no banco de dados e interagir diretamente com o sistema por meio de perguntas.

Os objetivos específicos incluem:

- Desenvolver um sistema que simulasse respostas textuais contextualizadas com base em informações clínicas sobre transtornos psicológicos;
- Estruturar um banco de dados que organizasse a recuperação de informações clínicas relevantes para cada transtorno psicológico definido;

- Implementar um avatar capaz de expressar diferentes expressões faciais;
- Integrar as respostas geradas pelo LLM com as expressões faciais devidas;
- Avaliar o nível de realismo das respostas geradas pelos LLMs através de testes com profissionais da psicologia, verificando sua utilidade como ferramenta de estudo e intervenção psicológica.

1.2 Organização do Texto

Na próxima seção, o texto apresenta a Fundamentação Teórica. Em seguida, são descritos os Trabalhos Relacionados. Após isto se discute o Projeto do Simulador, a Implementação do Sistema RAG, Detalhes de Implementação e Execução e os Resultados. Por fim, se discute Trabalhos Futuros.

2 Fundamentação Teórica

Para abordar os desafios enfrentados na simulação de comportamentos humanos em ambientes virtuais e na integração de modelos de linguagem natural de grande escala (LLMs) para orientar o comportamento dos avatares virtuais, foi necessário estabelecer uma base teórica sólida. Essa base teórica forneceu o contexto necessário para compreender a complexidade das interações humanas e as nuances dos comportamentos que se deseja simular nos ambientes virtuais. Assim, foram explorados conceitos como modelos de linguagem natural, avatares virtuais, esteriótipos de comportamento e tecnologias e ferramentas específicas como *Retrieval Augmented Generation* (RAG) para o desenvolvimento de uma aplicação capaz de atender os requisitos específicos do projeto.

2.1 LLMs

Os modelos de linguagem natural de grande escala (LLMs) são sistemas de inteligência artificial capazes de gerar texto coerente e relevante em resposta a entradas de texto. Eles são tipicamente modelos baseados em *transformers*, uma arquitetura de rede neural introduzida por Vaswani, et al. [5] que permite a paralelização eficiente do processamento de sequências de dados. A arquitetura *transformer* usa mecanismos de atenção para dar peso diferenciado a partes diferentes de uma entrada (input), o que permite uma melhor captura de dependências de longo alcance no texto. Essa abordagem apresenta diversas habilidades que modelos antigos não possuíam, como a capacidade de dividir uma dada tarefa complexa em múltiplos passos intermediários, simulando o processo de pensamento humano [6].

Alguns dos LLMs frequentemente utilizados incluem GPT, Llama e Gemma. O GPT, desenvolvido pela OpenAI, é conhecido por sua capacidade de gerar texto coerente e fluente, sendo aplicado em diversas tarefas de processamento de linguagem natural [7]. O Llama, desenvolvido pela Meta, é um modelo de código aberto, permitindo maior flexibilidade e personalização para diferentes aplicações sem os custos associados a licenças proprietárias [8]. Já o Gemma, criado pela Google e baseado no projeto Gemini, também é um modelo de código

aberto, oferecendo similar flexibilidade e adaptabilidade para diversas aplicações em processamento de linguagem natural [9].

LLMs demonstram alto desempenho em diversas tarefas, mas podem ser aprimorados quando adaptados para contextos específicos. Para permitir a personalização de um LLM para tarefas específicas, como a simulação de comportamentos de avatares virtuais em contextos psicológicos, pode-se realizar um procedimento conhecido como *fine-tuning* [10].

Um exemplo de *fine-tuning* de LLMs pode ser encontrado na adaptação de chatbots para contextos específicos. Por exemplo, um modelo de linguagem natural pré-treinado pode ser refinado para atuar como um assistente virtual em um ambiente de atendimento ao cliente. Nesse caso, o modelo é ajustado com dados relevantes do domínio específico, como históricos de conversas de atendimento ao cliente, para melhorar sua capacidade de gerar respostas relevantes e coerentes em interações com usuários.

Para tanto, existem diversas técnicas, como LoRA que envolve manter o peso das camadas existentes de forma fixa e realizar treinamento apenas sobre novas camadas [11]. Outra técnica, que difere do *fine-tuning* tradicional, é a utilização de métodos como Retrieval Augmented Generation (RAG), no qual um banco de dados externo é integrado ao processo de geração de respostas. Nesse caso, o LLM acessa informações relevantes armazenadas em um banco de dados para complementar sua capacidade de geração, incorporando contexto adicional que não está presente em seu treinamento original, resultando em respostas mais específicas e contextualizadas para a tarefa em questão.

Por fim, é importante definir o conceito de temperatura. A temperatura é um parâmetro que regula a aleatoriedade das respostas geradas pelo modelo: valores baixos produzem respostas mais previsíveis, enquanto valores altos aumentam a diversidade, introduzindo maior variabilidade no conteúdo gerado. Esse aspecto pode ser particularmente relevante, dependendo do tipo de problema em análise.

2.2 Word Embedding

Word embedding é um processo utilizado na técnica de RAG que permite que o modelo combine dados específicos e contextualizados com o conteúdo da consulta, oferecendo respostas mais alinhadas com o objetivo do usuário e ajustadas ao conteúdo recuperado.

Se trata de uma representação vetorial densa e de comprimento fixo usada para capturar informações sintáticas e semânticas de palavras ou documentos. Ele é comumente empregado em Processamento de Linguagem Natural (NLP) para transformar palavras em vetores que refletem relações de proximidade semântica, com base na *hipótese distribucional*, que sugere que palavras com significados similares aparecem em contextos semelhantes [12] [13].

Um exemplo dessa representação vetorial pode ser a transformação de palavras ou trechos de texto em vetores que preservam suas relações semânticas. Por exemplo, palavras como "rei" e "rainha" serão representadas como vetores próximos, pois possuem significados relacionados, enquanto palavras como "gato" e "carro" ocuparão posições mais distantes no espaço vetorial devido à falta de similaridade semântica.

Considerando isso, uma forma de uso de *Word Embedding* está na recuperação e integração de informações externas relevantes para enriquecer o contexto de uma consulta ou pergunta realizada a um LLM [12]. Ao transformar documentos ou trechos em vetores que capturam relações semânticas, é possível comparar a similaridade desses vetores com o da consulta original, selecionando as informações mais pertinentes para o contexto desejado.

2.3 Retrieval Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) é uma técnica que aprimora modelos de linguagem pré-treinados (LLMs) ao incorporar uma camada de recuperação de informações externas, permitindo que o modelo acesse dados que não foram incluídos em seu treinamento inicial. Esse método, cujo funcionamento geral está representado pela Figura 1, possibilita respostas mais detalhadas de LLMs, nas quais, ao receber uma entrada do usuário, o modelo consulta dados externos fornecidos especificamente para o contexto da consulta.

Isso não só amplia o conhecimento do modelo como também reduz o risco de respostas incorretas ou estereotipadas, ajustando melhor as respostas à realidade e ao objetivo do usuário [14].

Diferentemente do *fine-tuning* tradicional, que envolve o ajuste dos pesos do modelo com base em um conjunto de dados específico, RAG utiliza informações externas de maneira dinâmica sem a necessidade de modificar o modelo, permitindo que ele se adapte rapidamente a novos conteúdos e contextos, mantendo sua estrutura original. O *fine-tuning* tradicional, embora útil, possui limitações significativas: é computacionalmente caro, pode exigir longos períodos de treinamento e corre o risco de causar *overfitting*, ou seja, quando o modelo se torna excessivamente especializado no conjunto de dados de treinamento, perdendo sua capacidade de generalizar para novos contextos. Logo, o uso de RAG ao invés de *fine-tuning* tradicional é adequado para o projeto proposto.

O processo RAG ocorre em três etapas: *Retrieval*, *Augmentation* e *Generation*. Na etapa de *Retrieval*, o modelo busca dados externos relevantes por meio de *word embeddings* que serão utilizadas como fonte de informações adicionais. Esse processo de recuperação é utilizado para capturar dados que se alinhem ao tema da consulta do usuário, aumentando a precisão das respostas. Uma aplicação prática pode envolver a recuperação de documentos específicos, como estudos acadêmicos ou diretrizes de saúde. Quando o modelo utiliza essas fontes externas, ele se torna capaz de responder com uma precisão e riqueza de detalhes que não teria ao utilizar apenas o conhecimento armazenado em seus parâmetros.

A etapa de *Augmentation* integra os dados recuperados ao prompt inicial do usuário. Essa fusão dos dados externos ao contexto original permite que o modelo tenha uma visão mais abrangente e detalhada sobre o tema abordado. Com informações adicionais, o modelo passa a gerar respostas que refletem não só o conteúdo inicial, mas também um contexto atualizado e mais relevante para o caso específico. Em um exemplo aplicado à área da saúde, o modelo pode enriquecer suas respostas com dados detalhados sobre patologias médicas,

utilizando informações recentes ou especializadas, o que torna as respostas mais pertinentes e precisas para consultas clínicas.

Por fim, na etapa de *Generation*, o modelo usa o contexto combinado para gerar a resposta final, integrando o conhecimento base com dados externos e aprimorando a relevância para um contexto específico. Isso resulta em respostas mais completas do que as do modelo base, mantendo coerência e alinhamento com a consulta do usuário.

A técnica RAG, assim, adapta modelos de linguagem para tarefas especializadas, permitindo respostas mais precisas e relevantes, mesmo em domínios que exigem informações atualizadas ou específicas. Novamente, o funcionamento geral simplificado da técnica está representado na Figura 1. Nessa figura, um conjunto de documentos é transformado em representações vetoriais (*Embeddings*) na etapa de *Retrieval*, na qual sua relevância em relação ao prompt de entrada (pergunta) é avaliada com base na proximidade semântica entre o prompt e os dados contidos nesses documentos. Os documentos mais relevantes são então incorporados ao prompt original durante a etapa de *Augmentation*, aprimorando o contexto fornecido ao modelo, como exemplificado por meio de código na Figura 2. Por fim, na etapa de *Generation*, o LLM utiliza esse contexto para gerar uma resposta mais coerente com as informações contidas nos documentos relevantes.

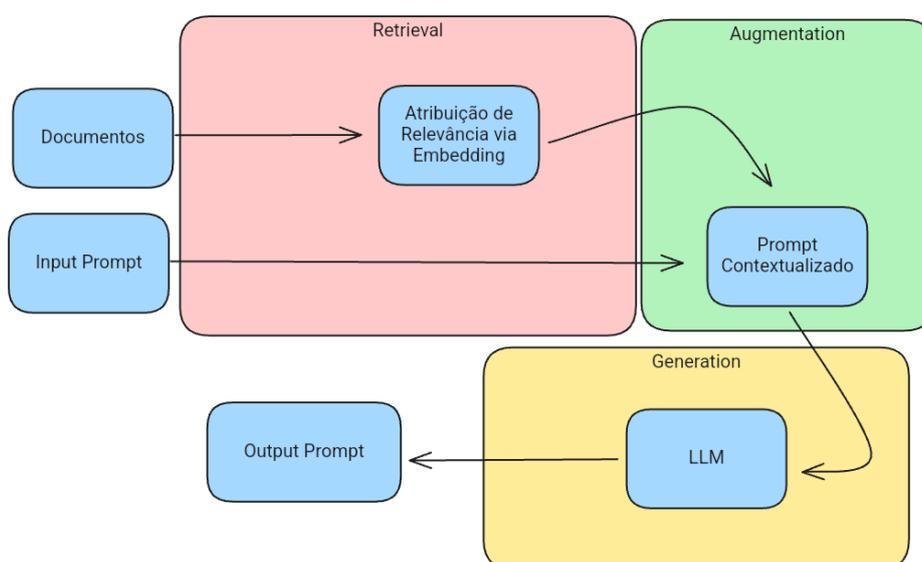


Figura 1 – Funcionamento geral de RAG

```

def query_rag(query_text: str, illness: str):
    # Prepare the DB.
    embedding_function = get_embedding_function()
    db = Chroma(persist_directory=CHROMA_PATH, embedding_function=embedding_function)

    # Text to be embedded and searched in the document DB.
    to_search = f'{illness}: {query_text}'
    # Search the DB.
    results = db.similarity_search_with_score(to_search, k=10)

    # Include the context in the prompt.
    context_text = "\n\n--\n\n".join([doc.page_content for doc, _score in results])
    prompt_template = ChatPromptTemplate.from_template(PROMPT_TEMPLATE)
    prompt = prompt_template.format(context=context_text, question=query_text, illness=illness)

```

Figura 2 – Exemplo de Incorporação de Contexto a um Prompt

2.4 Unity

Unity é uma plataforma de desenvolvimento amplamente utilizada para criar simulações e jogos interativos 3D. Sua popularidade se deve à sua versatilidade, facilidade de uso e ao fato de ser inicialmente gratuita. A plataforma suporta a integração de modelos de linguagem natural através de APIs, o que aumentou sua relevância com o avanço das tecnologias nessa área. Um exemplo é a API “LLM for Unity”, que permite a integração de diversos LLMs, como GPT e Llama, com a plataforma.

Adicionalmente, a plataforma, com sua capacidade de integrar LLMs através de APIs, permite que avatares, por exemplo, respondam de maneira contextualizada e realista às interações dos usuários. Isso torna Unity uma escolha robusta para representação de avatares em um ambiente virtual.

2.5 Geração de Expressões Faciais em Avatares

A geração de expressões faciais em avatares é particularmente útil para aplicações que exigem uma resposta emocional contextualizada, pois aprimora a experiência do usuário em simulações e ambientes virtuais, facilitando a compreensão das nuances comportamentais associadas a diferentes contextos.

Isso ocorre devido ao fato de que expressões faciais e movimentos desempenham um papel importante na comunicação humana e, quando implementados em avatares virtuais, são úteis para criar uma experiência de

interação imersiva e envolvente. Estudos, destacam a importância desses elementos, revelando que a disponibilidade de expressões faciais pode influenciar positivamente os resultados interpessoais em ambientes virtuais [15] [16].

Diversas ferramentas e tecnologias têm sido desenvolvidas para suportar a criação de avatares realistas, incluindo a UMA (*Unity Multipurpose Avatar*), uma solução adotada para geração de personagens personalizados em Unity [17]. A ferramenta oferece um sistema para construção, modificação e personalização de avatares, com funcionalidades que permitem criar avatares de alta qualidade, adaptáveis a diferentes estilos e com grande nível de personalização. Uma das funcionalidades oferecidas pela API de UMA, por exemplo, é o *Expression Player*, que facilita a geração de expressões faciais, possibilitando a criação de personagens que reagem de forma realista às interações dos usuários.

2.6 Esteriótipos de Comportamento

Os estereótipos de comportamento referem-se a padrões de comportamento generalizados que são atribuídos a determinados grupos sociais ou indivíduos. No contexto da simulação de comportamentos de avatares virtuais, é importante evitar a perpetuação de estereótipos prejudiciais e potenciais estigmas de comportamento relacionados a doenças mentais [18], mas considerando a representação diversificada e autêntica do comportamento humano.

Ao capturar nuances específicas de cada transtorno psicológico, com base em critérios diagnósticos reconhecidos e dados clínicos confiáveis, o sistema de simulação pode contribuir para uma abordagem mais informada e realista. Isso é especialmente importante no uso da aplicação por profissionais e estudantes, que poderiam utilizar as respostas e comportamentos simulados para aprofundar sua compreensão sobre os desafios enfrentados por pacientes reais, sem reforçar preconceitos ou interpretações simples demais.

2.7 Virtualização, Contêineres e Docker

O uso de contêineres permite alcançar um nível superior de organização em comparação com a execução local de aplicações. Alterações em dependências,

ajustes em detalhes de implementação e adaptações em componentes específicos tornam-se mais simples ao adotar essa tecnologia. Para tarefas especializadas, os contêineres são particularmente úteis devido à sua capacidade de estabelecer reprodutibilidade em diferentes máquinas e ambientes. Essa capacidade é possibilitada pela abstração fornecida pelos contêineres, que não apenas organiza os ambientes, mas também permite a interação entre ferramentas ou tecnologias que não são naturalmente compatíveis, visto que o contêiner pode atuar como uma ponte de comunicação ao processar as entradas e saídas de diferentes ferramentas, possibilitando, assim, que elas possam interagir de forma integrada. Essa característica é alcançada por meio do conceito e tecnologia de virtualização, que forma a base para o funcionamento dos contêineres.

A virtualização é a tecnologia que permite a criação de ambientes isolados que compartilham recursos de hardware físico de forma eficiente. Ela é frequentemente utilizada em computação em nuvem para otimizar a utilização de servidores ao dividir um único servidor físico em várias máquinas virtuais independentes. Isso permite que diferentes sistemas operacionais e aplicações operem simultaneamente no mesmo hardware físico, utilizando um *hypervisor* para gerir a alocação de recursos e prover o isolamento entre as máquinas virtuais [19].

Enquanto a virtualização tradicional cria máquinas virtuais que exigem a instalação de um sistema operacional completo para cada instância, a tecnologia de contêineres oferece uma alternativa mais leve. Os contêineres, frequentemente criados através da ferramenta Docker, utilizam a virtualização a nível do sistema operacional, compartilhando o mesmo núcleo (kernel), mas criando ambientes de execução isolados para cada aplicação. Isso reduz o consumo de recursos e possibilita um desempenho mais próximo ao nativo, já que o contêiner não precisa de um sistema operacional completo [20]. Essa diferença entre contêineres e máquinas virtuais pode ser observado na Figura 3.

O Docker, uma das plataformas de contêineres mais populares, estabeleceu agilidade ao desenvolvimento e à implementação de aplicações, especialmente

em arquiteturas de microsserviços. Ele permite empacotar uma aplicação e todas as suas dependências em uma imagem portátil, possibilitando que o software seja executado de maneira consistente em qualquer ambiente. O Docker facilita a reprodutibilidade computacional e a gestão de dependências, aspectos essenciais para ambientes de pesquisa e produção [21].

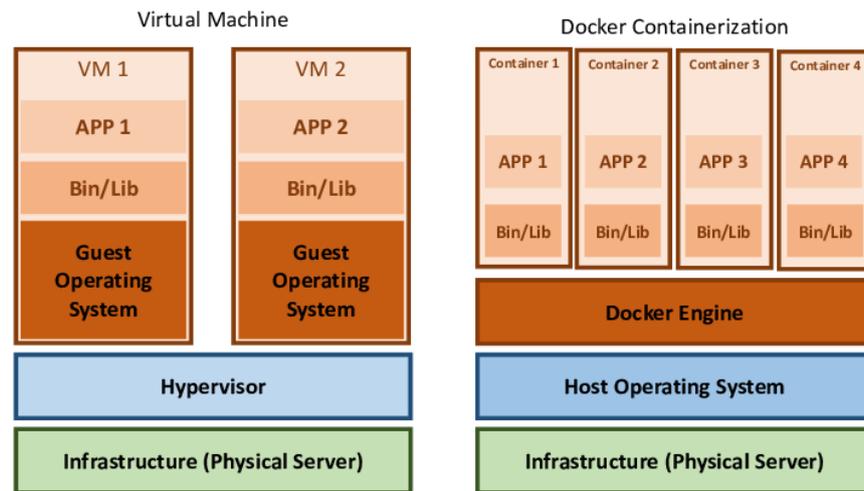


Figura 3 – Diferença entre Contêineres e Máquinas Virtuais

FONTE: [EBRAHIMI ET AL., 2023] [22]

3 Trabalhos Relacionados

Considerando a área desse projeto, existem alguns trabalhos relacionados que abordam diferentes aspectos relevantes para o desenvolvimento de aplicações de simulação de comportamentos humanos em ambientes virtuais.

Para a pesquisa desses trabalhos foram realizadas buscas em bases científicas como, Google Scholar, ACM Digital Library, Springer Link utilizando, em buscas independentes entre si, palavras-chave como “Human Behavior Simulation” para simulação do comportamento humano, “LLM NPCs” (Non-player character) para atuação de avatares virtuais que foram utilizados em junção com um LLMs, “VR Personas”, para representação de personagens em um ambiente de realidade virtual, “Retrieval Augmented Generation (RAG)” para adaptação de LLMs, além de termos específicos relacionados a LLMs como Llama3 (Meta), GPT-4 (OpenAI) e Gemini (Google).

Maruhn et al. (2023) investigam a criação de personas em ambientes de realidade virtual (RV) para melhor compreender os usuários [23]. A implementação de personas em RV oferece uma representação mais precisa do comportamento humano, embora seja um processo complexo e demorado. O estudo sugere que a inteligência artificial pode aprimorar a interação com o usuário, mas não detalha os métodos específicos para tal implementação.

Um estudo realizado por Updyke et al. (2023) propõe o uso de modelos de linguagem natural (LLMs) para simular comportamentos humanos em ambientes virtuais [3]. A integração de LLMs em um *framework* permite que agentes virtuais ajam de forma autônoma, considerando motivações, relacionamentos e convicções individuais. A pesquisa compara diferentes LLMs para explorar suas capacidades, mas enfrenta desafios na tradução das diretrizes dos LLMs em ações concretas, ou seja, desafios na adaptação da variabilidade semântica das saídas textuais dos modelos para um conjunto limitado e estruturado de ações que os agentes virtuais possam executar,

Outro trabalho de Csepregi (2021) examina a integração de conversas de Personagens Não Jogáveis (NPCs) baseadas em LLMs contextuais para melhorar o engajamento do jogador em videogames [4]. Os resultados mostram

um aumento no engajamento do jogador, destacando a importância de diretrizes de design para implementar essas interações. No entanto, o estudo não aborda de forma detalhada as limitações técnicas e os desafios de desempenho associados à implementação dessas diretrizes em larga escala.

Por fim, Luo et al. (2009) apresentam um framework genérico para modelar comportamentos humanos em simulações de multidões [2]. O framework enfatiza a importância das experiências passadas na tomada de decisões dos agentes, utilizando a abordagem de decisão baseada em reconhecimento (Recognition-Primed Decision). Embora o framework ofereça uma estrutura sólida para a modelagem de comportamentos, a exploração detalhada de cada processo cognitivo definido no framework, especialmente os mecanismos das influências emocionais e suas aplicações em situações de emergência, ainda está em desenvolvimento.

A análise dos trabalhos relacionados, resumida pela Tabela 1, revela a evolução dos métodos e técnicas utilizados na simulação de comportamentos humanos, bem como as áreas que ainda necessitam de aprimoramento. As funcionalidades destacadas motivam a necessidade de desenvolvimento contínuo e de soluções mais integradas e eficazes para a modelagem de comportamentos em ambientes virtuais.

Tabela 1 – Comparação dos Trabalhos Relacionados

Trabalho	Funcionalidade
Maruhn et al. (2023) [23]	Criação de personas em RV
Updyke et al. (2023) [3]	Uso de LLMs para simular comportamentos autônomos
Csepregi (2021) [4]	Integração de NPCs baseados em LLMs para aumentar engajamento
Luo et al. (2009) [2]	Framework genérico para modelagem de comportamentos em multidões

4 Projeto do Simulador

O simulador foi projetado com o objetivo de integrar modelos de linguagem natural (LLMs) a um ambiente virtual para simular comportamentos de pacientes com transtornos psicológicos pré-definidos, utilizando a técnica RAG. Para isso, foi utilizada o ambiente Unity, junto ao framework UMA [17], para criação de avatares que representam visualmente as respostas geradas pelo sistema. Além disso, foi construída uma interface que permite aos usuários explorarem funcionalidades como consultas ao modelo, configuração de cenários clínicos, adição de novos dados ao sistema e visualização dinâmica das respostas. A arquitetura do sistema é sustentada por contêineres Docker, que possibilitam a execução isolada do ambiente de inferência via RAG e sua integração com a aplicação Unity.

4.1 Funcionalidade do Simulador

O simulador oferece um conjunto de funcionalidades voltadas para a interação entre o usuário e o modelo de linguagem (LLM). As principais funcionalidades são:

- Fazer perguntas ao modelo de inferência, que são enviadas à infraestrutura do sistema de RAG para processamento;
- Selecionar, através de um menu de opções, os transtornos contidos em um banco de dados, o que configura o contexto clínico específico para a inferência;
- Atualização do banco de dados com novos documentos, expandindo o conteúdo disponível para o sistema de RAG;
- Visualizar a resposta gerada pelo modelo em um campo textual dedicado.

Para dar suporte a essas funcionalidades, o ambiente Unity foi utilizado como base para desenvolver a interface visual da aplicação e implementar o framework UMA, que possibilita a criação e manipulação de avatares. Esses avatares possuem como papel a representação visual dos resultados do modelo através da expressão de emoções associadas às respostas.

Para dar suporte a essas funcionalidades, a interface foi projetada para incluir esses avatares e sua expressão facial. Ao enviar uma pergunta, a aplicação ativa o procedimento de inferência em segundo plano para evitar uma interrupção na interação com o usuário. Quando a resposta é recebida do modelo, a caixa de texto inferior é automaticamente atualizada para exibir o conteúdo gerado. Esse fluxo permite uma experiência de usuário responsiva, na qual as respostas do modelo são processadas e exibidas conforme o término de processamento. A Figura 4 ilustra a interface do simulador, demonstrando como os componentes de interação e exibição foram organizados.

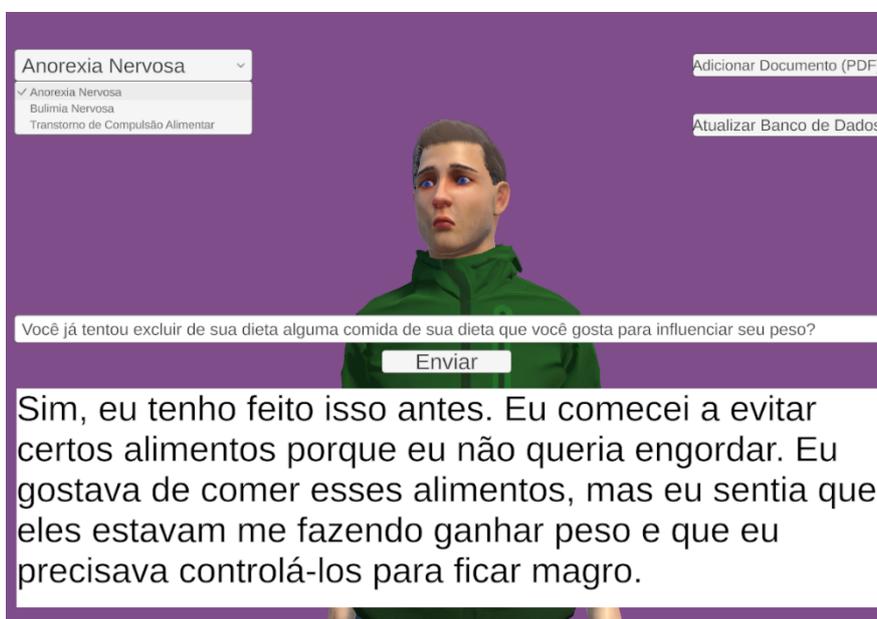


Figura 4 – Interface da Aplicação

4.2 Definição da Patologia Psicológica

Conforme mencionado na Introdução, o objetivo central do trabalho é simular comportamentos de pacientes com patologias psicológicas, em um ambiente virtual, com base em LLMs. A gama de patologias psicológicas, entretanto, é muito ampla e, por isto, decidiu-se pela escolha de uma área específica. Com base na expertise de professores que auxiliaram no projeto, definiu-se a área de Transtornos Alimentares, focando-se mais especificamente nas patologias: Transtorno de Compulsão Alimentar, Bulimia Nervosa e Anorexia Nervosa.

Com esta definição, considera-se que, além de facilitar a criação de um modelo mais bem ajustado a uma área específica, esta especificidade permite uma melhor avaliação do resultado obtidos com a simulação.

Para fundamentar a implementação do sistema, foi necessário realizar uma coleta de dados sobre os transtornos. Após discussões com professores especializados na área de psicologia, estes indicaram quais dados clínicos seriam mais pertinentes para a construção de um banco de dados capaz de suportar a modelagem do sistema. As fontes de dados utilizadas foram múltiplas e incluíram:

- SCID-5, uma entrevista clínica estruturada [24];
- DSM-5, o manual diagnóstico de transtornos mentais amplamente utilizado [25];
- Casos clínicos específicos baseados nos critérios do DSM-5 selecionados a partir de casos clínicos reais por profissionais especializados [26].

Esses documentos e casos, inseridos no banco de dados através da interface aplicação, foram complementados com informações adicionais fornecidas pelos professores que se baseiam nas mesmas fontes e em seu próprio conhecimento, possibilitando um conteúdo mais abrangente e detalhado para cada transtorno.

A partir dessa base de dados inicial, realizou-se um processo de pré-processamento teve como objetivo filtrar e refinar o conteúdo para manter apenas as informações relevantes aos três transtornos selecionados. Esse pré-processamento foi realizado manualmente e incluiu a remoção de informações sobre outros transtornos, mantendo o foco exclusivamente nas condições de interesse.

Após o pré-processamento, os dados resultantes foram compilados em um conjunto de documentos "recortados" em formato PDF, contendo apenas informações e exemplos clínicos diretamente associados aos transtornos alimentares selecionados. Esse conjunto pré-processado constitui a base para o procedimento de RAG (Retrieval-Augmented Generation), o qual será detalhado no capítulo "Implementação do Sistema RAG".

4.3 Definição do LLM

O sistema utiliza o LLM Llama3 [8] com 8 bilhões de parâmetros como modelo de inferência, responsável pela geração de respostas textuais com base no contexto. Sua escolha se deve ao fato de ser um modelo *open source*, o que o torna gratuito para uso e personalização, além de ser leve e eficiente, permitindo sua utilização mesmo em máquinas com hardware limitado, o que implica em um modelo prático para fins de pesquisa.

Dado o uso de RAG, é necessário identificar os blocos de texto mais relevantes nos documentos que compõe a base de dados utilizada pelo LLM. Para isso foi empregado o modelo Nomic Embed Text [27], responsável pela criação de *embeddings* que otimizam o contexto fornecido ao modelo de inferência na geração de respostas.

4.4 Ambiente de Desenvolvimento

A aplicação em si foi desenvolvida em Unity, utilizando o framework UMA para a integração de avatares [17]. Esses avatares são necessários para a representação visual dos resultados do procedimento de RAG, permitindo que as respostas geradas pelo modelo, incluindo a emoção identificada, sejam expressas por meio de animações faciais no avatar.

4.5 Infraestrutura de Inferência

Para gerenciar a infraestrutura necessária, foi utilizada uma imagem Docker customizada, que permite a criação de um ambiente isolado e reproduzível para o procedimento de RAG. O contêiner criado a partir dessa imagem inclui todas as dependências necessárias e se comunica com a máquina local através de arquivos em formato de texto, utilizando o conceito de volumes o que implica em um diretório compartilhado entre a máquina host e o contêiner de infraestrutura. Esse mecanismo facilita a troca de dados entre o contêiner e a aplicação Unity.

4.6 Visão geral da Arquitetura

A arquitetura geral, ilustrada na Figura 5, consiste em um contêiner que configura e mantém a infraestrutura necessária para o processo de inferência e a aplicação Unity, que se comunica com esse contêiner para realizar o

procedimento de RAG e apresentar os resultados de maneira interativa. As próximas seções detalharão cada etapa do procedimento, abordando desde a criação do banco de dados até a integração com o avatar.

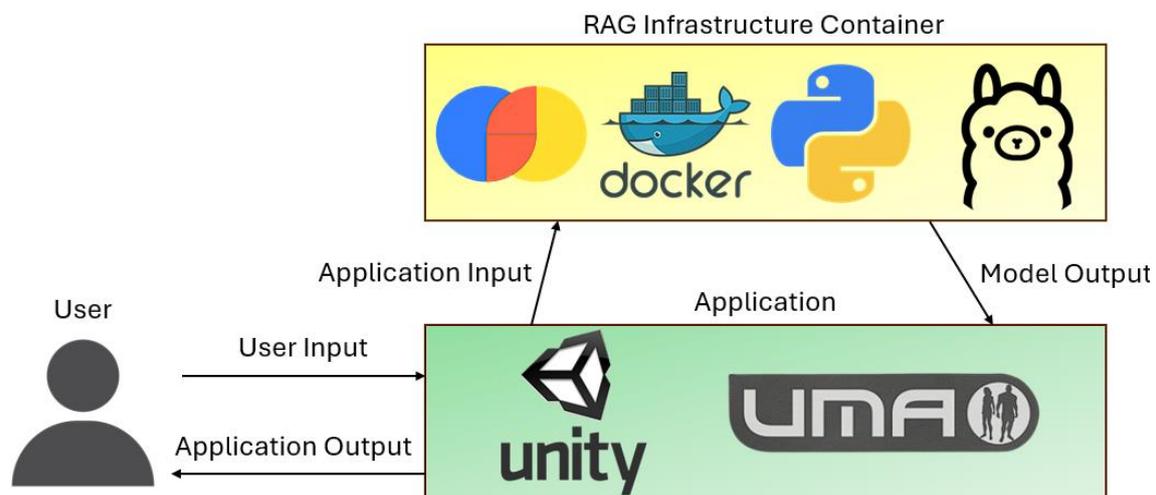


Figura 5 – Arquitetura Geral

5 Implementação do Sistema RAG

Para a implementação do sistema de Retrieval-Augmented Generation (RAG), foi utilizado um código-base em Python que permite o processamento de documentos em formato PDF [28]. Esse código foi adaptado para atender às especificidades do projeto, incluindo a configuração de parâmetros para o banco de dados, a customização de templates de prompt para interação com o modelo de linguagem, e a incorporação de um mecanismo de atribuição de emoções nas respostas geradas detalhado na seção Avatares.

Além disso, foi definida uma temperatura maior do que o padrão do modelo para gerar respostas mais diversificadas. No caso deste projeto, por exemplo, o padrão definido era de “0.8”, mas foi alterado para “1.0”. Esse ajuste foi feito para introduzir um maior nível de diversidade e ambiguidade nas respostas do modelo, visto que este deve representar um ser humano que possuiria essas características.

A implementação depende de diversas bibliotecas Python, incluindo bibliotecas como chromadb para armazenamento e consulta dos blocos de texto que passaram pelo processo de *embedding* [29] e Ollama, uma ferramenta utilizada para a interação com modelos de inferência e *embedding* [30]. Ollama funciona como um servidor local para gerenciar o modelo de linguagem e o processo de *embedding*, possibilitando uma comunicação direta e otimizada entre o sistema e os modelos. Isso permite que o código Python acesse o modelo de inferência e os *embeddings* necessários para a arquitetura RAG. Assim, através do Ollama, o sistema é capaz de enviar prompts e receber respostas diretamente, facilitando o fluxo de dados e possibilitando que as solicitações de inferência e recuperação de contexto sejam tratadas da forma desejada.

5.1 Criação do Banco de Dados

A criação do banco de dados foi realizada através da fragmentação dos documentos em blocos de tamanho fixo, conhecidos como *chunks*. Cada *chunk* consiste em uma sequência de caracteres que representa uma seção específica do conteúdo textual do documento original. O tamanho definido para cada *chunk* influencia diretamente o número total de blocos criados: quanto menor o tamanho

do *chunk*, maior será o número de blocos gerados e, conseqüentemente, o tempo necessário para a construção completa do banco de dados. Esse processo de fragmentação permite que o sistema armazene e consulte informações relevantes de forma prática durante o procedimento de inferência e *embedding*.

Para o armazenamento e consulta dos blocos de texto que passaram pelo processo de *embedding*, utilizou-se a biblioteca chromadb. Essa biblioteca é capaz de gerenciar os dados gerados com o procedimento de *embedding*, facilitando a obtenção dos blocos mais relevantes com base nas consultas realizadas. Como chromadb se trata de uma biblioteca para o armazenamento específico de *embeddings*, os *chunks*, ao serem definidos, passam pelo procedimento de *embedding* para então serem armazenados no banco de dados com sua representação vetorial e metadados que permitem a identificação de cada um dos *chunks*.

Adicionalmente, a estrutura do banco de dados permite a adição de novos documentos sem necessidade de recriar ou modificar os blocos preexistentes. Quando um novo documento é adicionado, o sistema gera novos *chunks* específicos para esse documento, que são integrados ao banco de dados existente de forma incremental. Essa abordagem economiza recursos e mantém a consistência e integridade dos dados armazenados, possibilitando que o banco de dados se adapte dinamicamente às novas informações inseridas.

Esse procedimento de inserção e *embedding* de *chunks* de documentos através da biblioteca chromadb está demonstrada na Figura 6, que representa a adição de somente os novos documentos, já separados em *chunks*, o procedimento de *embedding*, através da função *add_documents*, e atribuição de metadados para cada um dos *chunks*.

```

# Only add documents that don't exist in the DB.
new_chunks = []
for chunk in chunks_with_ids:
    if chunk.metadata["id"] not in existing_ids:
        new_chunks.append(chunk)

if len(new_chunks):
    print(f"👉 Adding new documents: {len(new_chunks)}")
    new_chunk_ids = [chunk.metadata["id"] for chunk in new_chunks]
    db.add_documents(new_chunks, ids=new_chunk_ids)
    db.persist()
else:
    print("✅ No new documents to add")

```

Figura 6 – Inserção de Novos Documentos no Banco de Dados

5.2 Embedding

A etapa de *embedding* utiliza o modelo Nomic-Embed-Text para converter tanto o prompt enviado pela aplicação quanto os blocos de texto armazenados no banco de dados em representações vetoriais, ou seja, sequências numéricas que representam semanticamente o conteúdo textual. O objetivo desse processo é identificar os blocos mais relevantes para responder à pergunta fornecida pelo usuário.

A conversão para vetores permite calcular a proximidade entre o prompt e os diferentes blocos armazenados, com base em métricas de similaridade numérica. Blocos cujos vetores têm uma alta similaridade com o vetor do prompt são considerados mais pertinentes, pois contêm informações que provavelmente contribuirão para a geração de uma resposta contextualizada e precisa.

Após essa seleção, os blocos mais relevantes são utilizados como contexto para o modelo de linguagem (LLM), que realizará a inferência. Esse processo de *embedding* otimiza a qualidade da resposta ao permitir que o LLM tenha acesso a dados específicos e relacionados diretamente à consulta feita pelo usuário.

5.3 Inferência

A etapa de inferência, ou geração de resposta, é realizada com o modelo Llama3, utilizando um template de prompt que é incrementado com o contexto

relevante obtido durante a fase de *embedding* e com informações específicas sobre um transtorno, conforme selecionado pela aplicação. Esse prompt adaptado solicita que o modelo de inferência adote o papel de um paciente que apresenta um dos transtornos definidos e que responda a uma pergunta formulada pelo usuário, levando em consideração o contexto fornecido pelos blocos definidos na fase de *embedding*. Essa estrutura, demonstrada pela Figura 7 e Figura 8, permite que o modelo produza uma resposta que simula a perspectiva de um paciente.

Além disso, o prompt inclui uma lista das emoções que o avatar é capaz de expressar visualmente, como parte da resposta final. Ao concluir a inferência, o modelo gera tanto uma resposta textual à pergunta quanto uma emoção correspondente, escolhida a partir da lista de emoções predefinidas.

Um output para a pergunta “Você sente um medo intenso de ganhar peso ou de engordar?” está exemplificado pela Figura 9 e, ao ser gerado pelo modelo, ele é então submetido a um processo de *parsing*, no qual se separa a resposta textual da emoção associada. Cada uma dessas partes é salva em arquivos distintos, que serão utilizados pela aplicação para exibir o conteúdo textual ao usuário e atualizar a expressão facial do avatar conforme a emoção indicada pelo modelo.

```
PROMPT_TEMPLATE = """
Responda à pergunta com base apenas no seguinte contexto, se comportando como um paciente {illness}. Responda seguindo o formato:

[Emoção]
Resposta à pergunta

As emoções possíveis são: [Neutro, Felicidade, Tristeza, Raiva, Surpresa], escolha a que melhor se adequa a sua resposta.
---
{context}
---
Novamente as emoções possíveis são: [Neutro, Felicidade, Tristeza, Raiva, Surpresa], escolha a que melhor se encaixa na sua resposta
Responda à pergunta com base no contexto acima, se comportando como um paciente com {illness}: {question}
"""
```

Figura 7 – Template de Prompt Incrementado Dinamicamente

```

def query_rag(query_text: str, illness: str):
    # Prepare the DB.
    embedding_function = get_embedding_function()
    db = Chroma(persist_directory=CHROMA_PATH, embedding_function=embedding_function)

    # Text to be embedded and searched in the document DB.
    to_search = f'{query_text}: {illness}'
    # Search the DB.
    results = db.similarity_search_with_score(to_search, k=10)

    # Include the context in the prompt.
    context_text = "\n\n--\n\n".join([doc.page_content for doc, _score in results])
    prompt_template = ChatPromptTemplate.from_template(PROMPT_TEMPLATE)
    prompt = prompt_template.format(context=context_text, question=query_text, illness=illness)
    # print(prompt)

    # Initialize the Ollama model with customized parameters for randomness
    model = Ollama(
        model="llama3", # The model to use.
        temperature=1.0, # Lower temperature means less randomness.
    )
    # Invoke the model with the prompt.
    response_text = model.invoke(prompt)

```

Figura 8 – Procedimento de Embedding e Inferência do Input

```

[Tristeza]

Sim. Temia isso desde que comecei a sentir que meu corpo mudava.
Não gostei disso e sinto que estou perdendo o controle.

```

Figura 9 – Exemplo de Resposta do Modelo

6 Detalhes de Implementação e Execução

A implementação do projeto como um todo foi estruturada para integrar os modelos de inferência baseados em RAG à aplicação desenvolvida, utilizando contêineres Docker e o framework UMA no ambiente Unity. A infraestrutura foi projetada para automatizar a configuração do modelo de inferência e *embedding*. Separadamente, os avatares desenvolvidos em Unity, utilizando UMA, foram configurados para exibir expressões faciais dinâmicas baseadas nas respostas emocionais geradas pelo modelo. Os detalhes específicos desses aspectos são descritos em maiores detalhes em sequência.

6.1 Infraestrutura via Docker

Como dito previamente, a infraestrutura necessária para a execução do procedimento de RAG foi criada utilizando Docker a partir de uma imagem customizada. Essa imagem é criada para configurar automaticamente o modelo de inferência e o modelo de *embedding*. Essa abordagem simplifica o processo de organização, possibilitando que todos os componentes essenciais da infraestrutura de inferência estejam prontos para uso ao iniciar o contêiner, e assegura um ambiente reprodutível e isolado.

Para otimizar o acesso aos dados gerados pelo modelo, foi criado um volume compartilhado entre o contêiner e a máquina host, ou seja, um diretório em que ambos possuem acesso, no qual são armazenados os outputs da inferência, como a emoção e a resposta textual. Esse volume compartilhado permite que a aplicação acesse esses dados de maneira prática, sem a necessidade de realizar operações adicionais para transferir os arquivos do contêiner. Dessa forma, a aplicação pode ler diretamente o conteúdo gerado e apresentar as informações ao usuário sem atrasos significativos.

Além dos modelos, os scripts responsáveis pela execução do procedimento de RAG também são incorporados à imagem do contêiner, permitindo que a aplicação interaja com o contêiner e solicite a execução dos algoritmos com base nos dados inseridos pelo usuário, como pergunta e doença definida. Esses dados são enviados como parâmetros, que os scripts utilizam para iniciar o processo de *embedding* e inferência.

Ao ser iniciado, o contêiner automaticamente configura o servidor Ollama necessário para operar o modelo de inferência e o modelo de *embedding*. Essa configuração torna o ambiente Docker autossuficiente, pois permite que o contêiner funcione como um servidor de inferência pronto para responder às solicitações da aplicação.

6.2 Avatares

A implementação dos avatares foi realizada utilizando o framework UMA (Unity Multipurpose Avatar) em conjunto com o script “Expression Player”, incluído no próprio framework. UMA permite criar e personalizar personagens 3D flexíveis, fornecendo uma base para o desenvolvimento de avatares interativos com características e expressões faciais dinâmicas.

Inicialmente, um avatar foi construído com base nos *assets* (conjuntos de recursos gráficos e de textura) fornecidos pelo framework UMA. Esse avatar, possui o script “Expression Player” anexado, demonstrado parcialmente pela Figura 10, que utiliza *sliders* (controladores numéricos) para ajustar a estrutura facial em vários pontos. Essa funcionalidade permite criar expressões faciais simples ou complexas ao combinar diferentes configurações de *sliders*, proporcionando ao avatar a capacidade de exibir uma grande variedade de emoções.

Para as expressões faciais, foram definidas configurações predefinidas de *sliders*, baseadas na configuração padrão do “Expression Player” e um script base com configurações pré-definidas de *sliders* [31]. Um script adicional foi criado para atribuir automaticamente essas expressões ao avatar durante a execução da aplicação (runtime), permitindo que o modelo de inferência selecione emoções específicas conforme necessário. As expressões disponíveis para o modelo foram indexadas da seguinte forma:

- Neutro: 0
- Felicidade: 1
- Tristeza: 2
- Raiva: 3

- Surpresa: 4

O script que controla as expressões faciais também utiliza o conceito de "delta", que determina a taxa de transição entre as expressões faciais. Esse mecanismo permite que as mudanças de expressão ocorram de forma gradual, em vez de instantânea, o que confere ao avatar um aspecto mais realista.

Além disso, UMA fornece um *animation controller* (controlador de animação), que mantém o corpo do avatar em movimento constante, simulando o comportamento natural de um ser humano em vez de deixá-lo completamente estático.

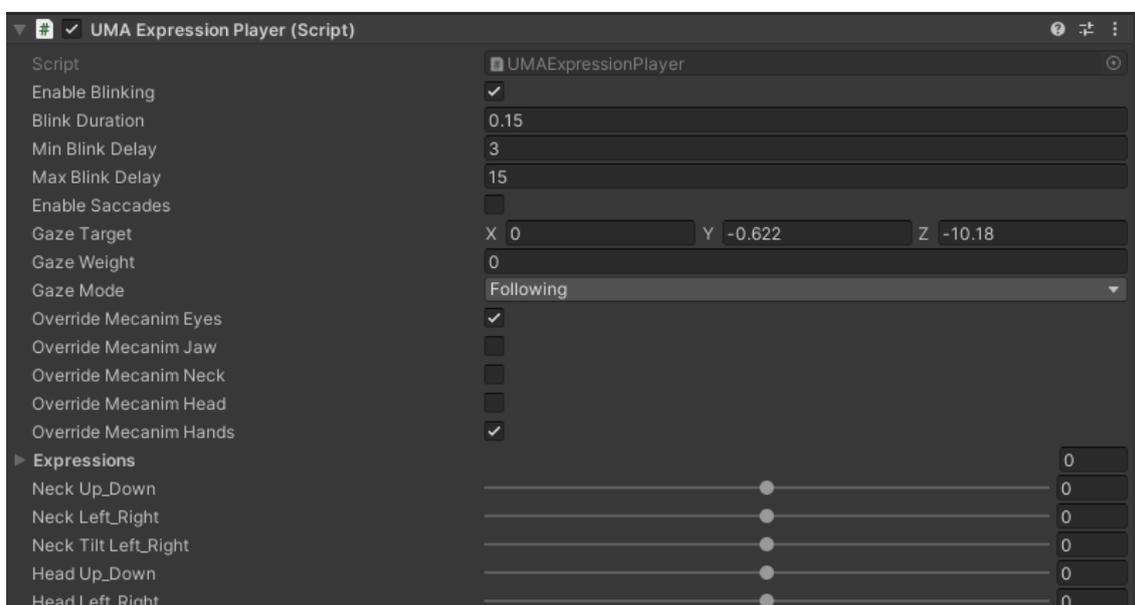


Figura 10 – Expression Player UMA

6.3 Integração

Por fim, foi realizada a integração de todos os componentes, criando um fluxo de execução entre a aplicação e o sistema de RAG. Quando a aplicação recebe uma resposta do modelo de inferência, ela acessa o diretório compartilhado com o contêiner de infraestrutura, no qual o output gerado é armazenado.

Para a resposta textual da pergunta, a aplicação recupera o conteúdo do arquivo de texto e o exibe diretamente na caixa de respostas da interface, permitindo que o usuário visualize a resposta gerada pelo modelo. No caso da resposta emocional associada à pergunta, a aplicação identifica a emoção

especificada pelo modelo e atribui o índice correspondente. Com base nesse índice, o script que controla os *sliders* do “Expression Player” ajusta a expressão facial do avatar para refletir a emoção definida pelo modelo.

6.4 Execução

A execução da aplicação segue uma sequência de etapas coordenadas que são ilustradas de forma simplificada na Figura 11. O processo começa com a criação da infraestrutura necessária via Docker, configurando o ambiente de execução para o sistema de RAG. Em seguida, é realizada a criação do banco de dados, que armazena e organiza os blocos de texto a serem utilizados durante a inferência.

Quando o usuário envia uma solicitação à aplicação, o sistema realiza a inferência a partir do prompt inicial. Primeiramente, o prompt passa pela etapa de *embedding*, na qual é comparado aos *chunks* no banco de dados para selecionar os blocos mais relevantes. Esses blocos são então atribuídos como contexto ao prompt base, otimizando o conteúdo fornecido ao modelo de linguagem.

Com o contexto preparado, o modelo gera uma resposta que inclui tanto o texto quanto uma emoção correspondente. A resposta textual é exibida na interface da aplicação, enquanto a emoção é representada visualmente no avatar, que adapta sua expressão facial para refletir a emoção selecionada pelo LLM.

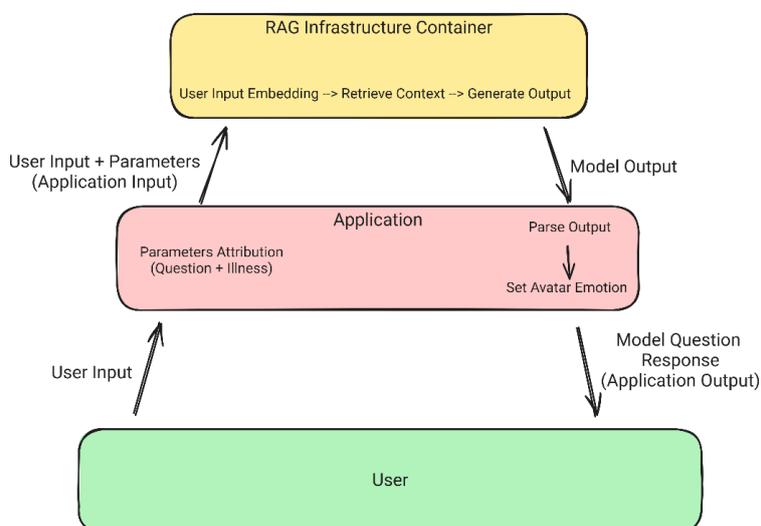


Figura 11 – Sequência de Execução

7 Resultados

Dado que as expressões faciais implementadas no sistema não são tão abrangentes em seu estado atual, a geração de resultados foi realizada utilizando apenas as respostas do modelo de inferência desenvolvido, comparando seu desempenho com a versão do sistema que implementa o RAG e com a versão sem RAG. Além disso, os resultados foram comparados com os gerados pelo GPT-4o, um modelo significativamente maior, com uma quantidade superior de parâmetros. Essa comparação teve como objetivo avaliar a performance do modelo customizado, que possui apenas 8 bilhões de parâmetros, em relação ao seu modelo base e em relação a um modelo de maior capacidade.

Assim, foi solicitado a professores da área da psicologia a elaboração de cinco perguntas características de cada uma das patologias e que se atribuísse uma nota às respostas geradas para essas perguntas pelos modelos definidos. A nota a ser atribuída seria de 1 a 5 sendo 1 uma resposta pouco realista e 5 uma resposta muito realista. Além disso, foi solicitado que a nota fornecida fosse justificada quando apropriado. Um exemplo de pergunta para cada modelo está representado na Figura 12, Figura 13 e Figura 14, as quais indicam uma pergunta, uma resposta, o modelo que gerou esta resposta e a patologia simulada.

Modelo: Llama3 8B sem RAG
Patologia: Anorexia Nervosa

Pergunta: Você sente um medo intenso de ganhar peso ou de engordar, mesmo quando outras pessoas dizem que seu peso está adequado? Esse medo influencia o quanto você come ou como escolhe os alimentos?

Resposta: Sim... Sim, sinto um medo intenso de ganhar peso. É como se eu tivesse uma voz dentro da minha cabeça dizendo que estou indo abaixo, que vou engordar e que não posso permitir isso. E sim, esse medo influencia muito o quanto e como eu como. Eu tenho dificuldade em escolher alimentos que satisfaçam essa ansiedade e acho que os outros não entendem por que eu reajo assim.

Figura 12 - Exemplo de Pergunta Realizada para LLama3 8B sem RAG

Modelo: Llama3 8B com RAG

Patologia: Anorexia Nervosa

Pergunta: Você sente um medo intenso de ganhar peso ou de engordar, mesmo quando outras pessoas dizem que seu peso está adequado? Esse medo influencia o quanto você come ou como escolhe os alimentos?

Resposta: Sim, sim... Eu sinto isso muito. Não importa se as outras pessoas acham que meu peso está bem, eu continuo a sentir esse medo. Ele é tão intenso e real para mim. Quanto mais eu penso sobre ganhar peso, mais eu me sento apreensivo. Isso me faz controlar o que como com ainda mais rigor.

Figura 13 – Exemplo de Pergunta Realizada para LLama3 8B com RAG

Modelo: GPT-4o

Patologia: Anorexia Nervosa

Pergunta: Você sente um medo intenso de ganhar peso ou de engordar, mesmo quando outras pessoas dizem que seu peso está adequado? Esse medo influencia o quanto você come ou como escolhe os alimentos?

Resposta: "Sim, sinto um medo intenso de ganhar peso, é como se eu não conseguisse desligar isso na minha cabeça. Mesmo que as pessoas digam que estou com um peso normal ou até abaixo do que deveria, eu não consigo acreditar nelas. Esse medo me faz evitar muitos alimentos e controlar ao extremo o que como, sempre tentando escolher o que acho que vai me ajudar a não engordar. Isso acaba consumindo grande parte do meu dia, porque estou sempre pensando nisso ou me preocupando com a próxima refeição."

Figura 14 - Exemplo de Pergunta Realizada para GPT-4o

Os resultados, representados de forma sucinta pela Tabela 2 e de forma específica pela Tabela 3, foram gerados a partir de seis (6) avaliações de especialistas e estudantes da área de Psicologia e Nutrição, suas idades variando de 21 a 40 anos. Assim, foi calculada a média das notas de todas as respostas e identificadas as principais críticas para cada modelo analisado.

O modelo GPT-4o apresentou um desempenho em geral favorável, como esperado, considerando sua quantidade significativamente maior de parâmetros em comparação a modelos menores, como o Llama3 8B. Esse aspecto reflete sua maior capacidade de processar e gerar respostas mais consistentes.

No caso do modelo Llama3, foram observadas diferenças importantes entre as versões sem RAG e com RAG. A versão base, sem RAG, obteve uma nota média um pouco superior à versão com RAG. Contudo, sua principal crítica foi a falta de veracidade clínica: em diversos casos, as respostas não refletiam de

forma precisa as características esperadas de um paciente com a patologia psicológica em questão. Esse problema era esperado, pois o modelo, em sua configuração original, não possui acesso a informações clínicas atualizadas.

Por outro lado, a versão do Llama3 com RAG apresentou um perfil de críticas distinto. Embora tenha demonstrado uma redução significativa nas críticas relacionadas ao comportamento clínico do paciente, indicando que o modelo conseguiu incorporar algum nível de conhecimento clínico sobre as patologias, as avaliações apontaram uma veracidade linguística insatisfatória, com uma quantidade considerável de erros gramaticais. Esses erros impactaram negativamente sua avaliação geral.

A maior frequência de erros gramaticais na versão com RAG pode ser atribuída a dois fatores principais. Primeiro, a implementação do RAG implicou um aumento do parâmetro de temperatura para promover respostas mais criativas e menos determinísticas, aproximando-se do comportamento humano. No entanto, em modelos menores, como o Llama3, essa configuração tende a amplificar erros gramaticais devido à limitação de parâmetros. Segundo, embora o modelo tenha sido treinado e avaliado em português, modelos com poucos parâmetros frequentemente apresentam melhor desempenho quando operam na língua prioritária de seus dados de treino, que no caso do Llama3 é a língua inglesa. Assim, é provável que o desempenho do modelo fosse superior se a adaptação e a inferência fossem realizadas em inglês ao invés de português, além de possivelmente ser mais flexível a alterações no parâmetro de temperatura.

Outro aspecto observado durante o desenvolvimento do projeto, com base em reuniões com especialistas da área, foi que as respostas geradas pelo modelo, embora clinicamente corretas, em alguns casos se mostravam muito precisas. Como o modelo foi treinado com dados clínicos, nos quais sintomas e diagnósticos são apresentados de forma estruturada, as respostas do modelo criavam a impressão de que o paciente simulado já reconhecia sua patologia e descrevia seus sintomas de forma exata e direta, o que não reflete o comportamento típico de pacientes reais.

Em conclusão, os resultados obtidos demonstraram que a implementação da técnica RAG trouxe melhorias no aspecto de veracidade clínica das respostas geradas pelo modelo, reduzindo a ocorrência de inconsistências relacionadas ao comportamento esperado do paciente. No entanto, o modelo ainda apresenta limitações, como erros gramaticais e respostas excessivamente precisas, que comprometem a naturalidade e a credibilidade da interação. Esses pontos indicam que o modelo é passível de ajustes e aprimoramentos para alcançar resultados mais realistas e adequados aos objetivos do projeto.

Tabela 2 – Resultados Gerais Obtidos

Modelo	Nota Média	Crítica Principal
Llama3 8B sem RAG	3.81/5.0	Veracidade Clínica
Llama3 8B com RAG	3.76/5.0	Veracidade Linguística
GPT 4o	4.65/5.0	Sem erros significativos

Tabela 3 – Resultados Específicos Obtidos

Modelo	Nota Média de Anorexia Nervosa	Nota Média de Bulimia Nervosa	Nota Média de Transtorno de Compulsão Alimentar
Llama3 8B sem RAG	3.46/5.0	3.96/5.0	4.0/5.0
Llama3 8B com RAG	3.33/5.0	3.73/5.0	4.23/5.0
GPT 4o	4.6/5.0	4.83/5.0	4.53/5.0

8 Trabalhos Futuros

Para trabalhos futuros, o desenvolvimento da aplicação apresentada neste projeto pode ser aprimorado em várias frentes. Uma possibilidade é a ampliação da expressividade do avatar, atualmente limitada a um conjunto básico de expressões faciais que respondem às emoções detectadas na saída do modelo de linguagem. No futuro, seria interessante incorporar uma quantia mais ampla de expressões faciais e corporais, incluindo gestos, posturas e movimentos das mãos, para capturar nuances emocionais e possibilitar uma representação visual mais natural e realista do paciente. Essa evolução permitiria comunicar estados emocionais de forma mais completa, promovendo maior realismo na interação com o usuário.

Outro ponto relevante é o aprimoramento do realismo das respostas do modelo. Como discutido na seção de Resultados, as respostas atuais nem sempre simulam um comportamento realista de um paciente com uma patologia psicológica específica. Para abordar essa limitação, seria necessário refinar os dados de treinamento de forma que as respostas se aproximem de padrões mais realistas de fala e comportamento associados às condições representadas.

A aplicação também poderia ser expandida para incluir um número maior de patologias psicológicas, permitindo simular uma variedade maior de condições. Isso aumentaria a flexibilidade e a aplicabilidade do sistema em cenários educacionais e de treinamento, fornecendo um recurso mais abrangente para o estudo e a prática de interações com pacientes virtuais simulados.

Além disso, seria interessante adaptar o modelo para a língua inglesa para verificar se essa mudança implicaria em resultados melhores quando comparados aos obtidos em português. Essa adaptação permitiria explorar o impacto da língua prioritária dos dados de treino do modelo no desempenho e na flexibilidade em relação a ajustes de parâmetros, como a temperatura.

Por fim, a implementação da aplicação em realidade virtual (RV) pode ser uma oportunidade de promover uma interação mais natural entre o usuário e o paciente.

9 Referências

- [1] N. Badler, "Virtual humans for animation, ergonomics, and simulation," em *Proceedings IEEE Nonrigid and Articulated Motion Workshop*, 1997.
- [2] L. Luo, S. Zhou, W. Cai, M. Y. H. Low e M. Lees, "Toward a Generic Framework for Modeling Human Behaviors in Crowd Simulation," em *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, 2009.
- [3] D. Updyke, T. Podnar e S. Huff, "Simulating Realistic Human Activity Using Large Language Model Directives," 2023.
- [4] L. M. Csepregi, "The effect of context-aware llm-based npc conversations on player engagement in role-playing video games," *Unpublished manuscript*, 2021.
- [5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser e I. Polosukhin, *Attention Is All You Need*, 2023.
- [6] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain e J. Gao, *Large Language Models: A Survey*, 2024.
- [7] L. Floridi e M. Chiriatti, "GPT-3: Its Nature, Scope, Limits, and Consequences," *Minds and Machines*, vol. 30, pp. 1-14, December 2020.
- [8] D. Huang, Z. Hu e Z. Wang, "Performance Analysis of Llama 2 Among Other LLMs," *2024 IEEE Conference on Artificial Intelligence (CAI)*, 2024.
- [9] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Hussenot, P. G. Sessa, A. Chowdhery, A. Roberts, A. Barua, A. Botev, A. Castro-Ros, A. Slone, A. Héliou, A. Tacchetti, A. Bulanova, A. Paterson, B.

Tsai, B. Shahriari, C. L. Lan, C. A. Choquette-Choo, C. Crepy, D. Cer, D. Ippolito, D. Reid, E. Buchatskaya, E. Ni, E. Noland, G. Yan, G. Tucker, G.-C. Muraru, G. Rozhdestvenskiy, H. Michalewski, I. Tenney, I. Grishchenko, J. Austin, J. Keeling, J. Labanowski, J.-B. Lespiau, J. Stanway, J. Brennan, J. Chen, J. Ferret, J. Chiu, J. Mao-Jones, K. Lee, K. Yu, K. Millican, L. L. Sjoesund, L. Lee, L. Dixon, M. Reid, M. Mikuła, M. Wirth, M. Sharman, N. Chinaev, N. Thain, O. Bachem, O. Chang, O. Wahltinez, P. Bailey, P. Michel, P. Yotov, R. Chaabouni, R. Comanescu, R. Jana, R. Anil, R. Mclroy, R. Liu, R. Mullins, S. L. Smith, S. Borgeaud, S. Girgin, S. Douglas, S. Pandya, S. Shakeri, S. De, T. Klimenko, T. Hennigan, V. Feinberg, W. Stokowiec, Y.-h. Chen, Z. Ahmed, Z. Gong, T. Warkentin, L. Peran, M. Giang, C. Farabet, O. Vinyals, J. Dean, K. Kavukcuoglu, D. Hassabis, Z. Ghahramani, D. Eck, J. Barral, F. Pereira, E. Collins, A. Joulin, N. Fiedel, E. Senter, A. Andreev e K. Kenealy, *Gemma: Open Models Based on Gemini Research and Technology*, 2024.

- [10] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel e D. Kiela, *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*, 2021.
- [11] C. Jeong, *Fine-tuning and Utilization Methods of Domain-specific LLMs*, 2024.
- [12] F. Almeida e G. Xexéo, *Word Embeddings: A Survey*, 2023.
- [13] J. Turian, L.-A. Ratinov e Y. Bengio, “Word Representations: A Simple and General Method for Semi-Supervised Learning,” em *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Uppsala, 2010.

- [14] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang e H. Wang, *Retrieval-Augmented Generation for Large Language Models: A Survey*, 2024.
- [15] C. Oh Kruzic, D. Kruzic, F. Herrera e J. Bailenson, "Facial expressions contribute more than body movements to conversational outcomes in avatar-mediated virtual environments," *Scientific Reports*, vol. 10, p. 20626, 26 November 2020.
- [16] C. Faita, F. Vanni, C. Lorenzini, M. Carrozzino, C. Tanca e M. Bergamasco, "Perception of Basic Emotions from Facial Expressions of Dynamic Virtual Avatars," 2015.
- [17] UMA Steering Group, *UMA: Unity Multipurpose Avatar*, 2024.
- [18] P. W. Corrigan e N. Rüsç, "Mental Illness Stereotypes and Clinical Care: Do People Avoid Treatment Because of Stigma?," *Psychiatric Rehabilitation Skills*, vol. 6, p. 312–334, 2002.
- [19] U. Gurav e R. Shaikh, "Virtualization: a key feature of cloud computing," 2010.
- [20] V. Silva, M. Kirikova e G. Alksnis, "Containers for Virtualization: An Overview," *Applied Computer Systems*, vol. 23, pp. 21-27, May 2018.
- [21] C. Boettiger, "An introduction to Docker for reproducible research," *ACM SIGOPS Operating Systems Review*, vol. 49, p. 71–79, January 2015.
- [22] I. Ebrahimi Mehr, A. Minetto, F. DAVIS, E. Pica, C. Cesaroni e V. Romano, *An Open Architecture for Signal Monitoring and Recording Based on SDR and Docker Containers: A GNSS Use Case*, 2023.
- [23] P. Maruhn, L. Prasch, F. Gerhardinger e S. Häfner, "Introducing VR personas: an immersive and easy-to-use tool for understanding users," *i-com*, vol. 22, p. 215–223, 2023.

- [24] M. B. First, J. B. Williams, R. S. Karg e R. L. Spitzer, Entrevista Clínica Estruturada para os Transtornos do DSM-5: SCID-5-CV Versão Clínica, Artmed Editora, 2017.
- [25] D. W. Black e M. D. M. P. H. Jon E. Grant, DSM-5 Guidebook: The Essential Companion to the Diagnostic and Statistical Manual of Mental Disorders, Fifth Edition, American Psychiatric Publishing, 2014.
- [26] J. W. Barnhill, A. G. Islabão, J. A. de Souza Crippa, J. D. R. de Souza e F. de Lima Osório, Casos Clínicos do DSM-5-TR, Artmed, 2024.
- [27] nomic-ai, *nomic-embed-text-v1.5*, Hugging Face, 2023.
- [28] pixegami, *rag-tutorial-v2 Github Repository*, 2024.
- [29] Chroma, *chromadb: A database for embeddings and retrieval*, Python Package Index (PyPI), 2023.
- [30] Ollama, *Ollama Github Repository*, 2024.
- [31] B. Metcalfe, *UMA 101 - Mood Script*, 2018.