

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO  
INSTITUTO DE COMPUTAÇÃO  
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ANDRÉ C. S. DE QUEIROZ

UM ALGORITMO DE DETECÇÃO DE ERROS GRAMATICAIIS USANDO LLMS  
PARA APOIAR NO APRENDIZADO DE LÍNGUAS

RIO DE JANEIRO  
2024

ANDRÉ C. S. DE QUEIROZ

UM ALGORITMO DE DETECÇÃO DE ERROS GRAMATICAIIS USANDO LLMS  
PARA APOIAR NO APRENDIZADO DE LÍNGUAS

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Daniel Sadoc Menasche

RIO DE JANEIRO

2024

## CIP - Catalogação na Publicação

Q555a Queiroz, André  
Um algoritmo de detecção de erros gramaticais usando LLMs para apoiar no aprendizado de línguas / André Queiroz. -- Rio de Janeiro, 2024.  
63 f.

Orientador: Daniel Sadoc Menasche.  
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2024.

1. Aprendizado de Línguas. 2. ChatGPT. 3. Duolingo. 4. LLM. 5. Tradução. I. Menasche, Daniel Sadoc, orient. II. Título.


ANDRÉ C. S. DE QUEIROZ

UM ALGORITMO DE DETECÇÃO DE ERROS GRAMATICAIIS USANDO LLMS  
PARA APOIAR NO APRENDIZADO DE LÍNGUAS

Trabalho de conclusão de curso de graduação  
apresentado ao Instituto de Ciência da Com-  
putação da Universidade Federal do Rio de  
Janeiro como parte dos requisitos para ob-  
tenção do grau de Bacharel em Ciência da  
Computação.


Aprovado em 01 de Julho de 2024

BANCA EXAMINADORA:

Documento assinado digitalmente  
 DANIEL SADOC MENASCHE  
Data: 20/08/2024 20:34:00-0300  
Verifique em <https://validar.iti.gov.br>


---

Daniel Sadoc Menasché  
PhD (UFRJ)

Documento assinado digitalmente  
 JOAO CARLOS PEREIRA DA SILVA  
Data: 20/08/2024 07:11:43-0300  
Verifique em <https://validar.iti.gov.br>

---

João Carlos Pereira da Silva  
D. Sc. (UFRJ)

Documento assinado digitalmente  
 CLAUDIO MICELI DE FARIAS  
Data: 21/08/2024 08:10:04-0300  
Verifique em <https://validar.iti.gov.br>

---

Claudio Miceli de Farias  
D. Sc. (UFRJ)

Aos que têm fé que a educação pode melhorar o mundo dedico este trabalho.

## AGRADECIMENTOS

Agradeço ao meu Senhor Jesus Cristo, por ser a minha esperança, por ter me amado primeiro, por me permitir viver e experimentar coisas incríveis como escrever este trabalho e por sua infinita misericórdia com o homem falho que sou.

Aos meus pais Silvio Queiroz e Andréa Queiroz por uma vida de sacrifício e dedicação à minha vida. Por me suportarem durante a realização do curso de Ciência da Computação e por me incentivarem a não desistir.

À minha noiva, Milenna Fuly, por ser minha alegria de cada dia, por compartilhar das dores e sabores da escrita de um trabalho acadêmico, por ser minha parceira de vida.

Aos meus mestres do Instituto de Computação. Em particular, o professor Daniel Sadoc, meu orientador, que foi importantíssimo na reta final. Foi paciente, disponível e atencioso em diversos momentos. Obrigado, professor, por acreditar em mim e nesse trabalho e por ser um guia nessa reta final. E também o professor João Paixão, que me motivou com seu amor ao ensino e dedicação à formação de profissionais excelentes. Obrigado por ser uma inspiração e por tantas oportunidades incríveis durante todo o curso.

*"If we teach today as we taught yesterday,  
we rob our children of tomorrow."*

**John Dewey**

## RESUMO

Este trabalho descreve um novo algoritmo para detecção de erros gramaticais em traduções usando o OpenAI ChatGPT e um programa de linha de comando que implementa esse algoritmo. O objetivo do algoritmo é gerar comentários e correções personalizadas para exercícios de tradução encontrados em aplicativos de aprendizado de línguas como o Duolingo. O algoritmo foi capaz de enumerar e qualificar erros gramaticais e ortográficos presentes em tentativas de tradução, além de classificar os tipos de erros, pontuar nuances da linguagem e oferecer sugestões. As correções e comentários produzidos pelo algoritmo foram consideravelmente mais ricos do que os já presentes no Duolingo. Os próximos passos incluem a ampliação e personalização do banco de frases, adição de novas línguas além de português e inglês e a integração com outras interfaces.

**Palavras-chave:** aprendizado de línguas; ChatGPT; Duolingo; LLM; MALL; tradução.



## ABSTRACT

This paper describes a new algorithm for detecting grammatical errors in translations using OpenAI's ChatGPT and the development of a command-line program that implements this algorithm. The goal of the algorithm is to generate personalized feedback and corrections for translation exercises found in language learning apps such as Duolingo. The algorithm was capable of enumerating and qualifying grammatical and spelling errors present in translation attempts, in addition to classifying types of errors, scoring language nuances, and offering suggestions. The corrections and feedback produced by the algorithm were significantly richer than those already present in Duolingo. The next steps include the expansion and customization of the phrase bank, addition of new languages beyond Portuguese and English, and integration with other interfaces.

**Keywords:** language learning; ChatGPT; Duolingo; LLM; MALL; translation.

## LISTA DE ILUSTRAÇÕES

Figura 1 – Exemplo de tradução de um exercício de português para inglês no Duolingo. . . . .	19
Figura 2 – Investigando as requisições do navegador em um exercício do Duolingo.	20
Figura 3 – Diagrama de sequência simplificado do sistema a ser desenvolvido. . . .	20
Figura 4 – Pedindo uma saída formatada em CSV de uma lista de pessoas. . . . .	22
Figura 5 – Exemplo do uso da técnica de prompting com cadeia de Pensamento. (WEI et al., 2023) . . . . .	23
Figura 6 – Ilustração da diferença entre o uso normal e uma tentativa de injeção de prompt numa aplicação integrada com LLM. Inspirado por (LIU et al., 2024) . . . . .	24
Figura 7 – Exemplo de resposta esperada da versão final do algoritmo. Frase “Você gosta de laranja?” e tentativa de tradução “You like pineapples?” . . . .	27
Figura 8 – Prompt v1: temp = 0 e model=gpt-3.5-turbo . . . . .	28
Figura 9 – Prompt v2: temp = 0 e model=gpt-3.5-turbo . . . . .	29
Figura 10 – Resposta do Prompt v2 com temp = 1 e model=gpt-3.5-turbo . . . . .	29
Figura 11 – Resposta do Prompt v2 com temp = 0.2 e model=gpt-3.5-turbo . . . . .	30
Figura 12 – Exemplo 1 do Prompt v2 com temp = 1 e model = gpt-4 . . . . .	31
Figura 13 – Exemplo 2 do Prompt v2 com temp = 1 e model = gpt-4 . . . . .	31
Figura 14 – Resposta do prompt v3 com temperatura = 1 e modelo = gpt-4 . . . . .	32
Figura 15 – Prompt v4 com temp=1 e model=gpt-4 . . . . .	33
Figura 16 – Prompt v5 (correção) com temp=1 e model=gpt-4 . . . . .	34
Figura 17 – Prompt v5 (formatação) com temp=1 e model=gpt-4 . . . . .	35
Figura 18 – Input do Prompt v6 com temp=1 e model=gpt-4 . . . . .	36
Figura 19 – Diagrama Completo do Sistema com um exemplo de tradução errada. . . .	38
Figura 20 – Exemplo de interação com a CLI . . . . .	39
Figura 21 – Frontend: código do comando run . . . . .	40
Figura 22 – Exemplos de frases do banco de frases . . . . .	41
Figura 23 – endpoint correction: correção de frases usando a API do ChatGPT . . . .	41
Figura 24 – Sistema funcionando com tentativas de tradução corretas e feedbacks corretos . . . . .	42
Figura 25 – Sistema funcionando com tentativa de tradução imprecisa e feedback correto . . . . .	43
Figura 26 – Sistema funcionando com tentativa de tradução errada e feedback correto	43
Figura 27 – Sistema funcionando com tentativa de tradução errada e feedback correto para dois falsos cognatos: 'idioms' e 'preferred' . . . . .	44

Figura 28 – Sistema funcionando com tentativa de tradução correta de forma literal mas errada em contexto e feedback do sistema correto . . . . .	44
Figura 29 – Sistema funcionando com tentativa de tradução errada e feedback do sistema errado . . . . .	45
Figura 30 – Ilustração de um algoritmo de aprendizado de máquina supervisionado para identificar animais em uma imagem . . . . .	51
Figura 31 – Dados rotulados para um modelo de análise de sentimento . . . . .	52
Figura 32 – Exemplo de treinamento de Modelos Autorregressivos . . . . .	57
Figura 33 – A LLM base ada da OpenAI gerando um artigo sobre Marketing Digital	57
Figura 34 – Performance agregada do GPT-3 em 42 benchmarks de LLMs (BROWN et al., 2020) . . . . .	58
Figura 35 – InstructGPT vs GPT-3: GPT-3 não respondendo a instruções corretamente . . . . .	59
Figura 36 – InstructGPT vs GPT-3: alucinações . . . . .	59
Figura 37 – ChatGPT recusando prompt malicioso: violência . . . . .	60
Figura 38 – ChatGPT recusando prompt malicioso: bullying . . . . .	60
Figura 39 – Desenvolvimento de aplicações de IA tradicionais . . . . .	61
Figura 40 – Desenvolvimento de aplicações de IA com engenharia de prompt . . . .	61

## LISTA DE ABREVIATURAS E SIGLAS

ARFH	Aprendizado por Reforço com Feedback Humano
CLI	Command Line Interface
GML	Grande Modelo de Linguagem
GPT	Generative Pre-Trained Transformer
IA	Inteligência Artificial
JSON	JavaScript Object Notation
LLM	Large Language Model
MALL	Mobile-Assisted Language Learning
NLP	Natural Language Processing
PNL	Processamento de Linguagem Natural
RLHF	Reinforcement Learning with Human Feedback
RNN	Recurrent Neural Network
RNR	Rede Neural Recorrente

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>14</b>
1.1	O POTENCIAL DO CHATGPT, E SUAS PREOCUPAÇÕES ÉTICAS NA EDUCAÇÃO . . . . .	15
1.1.1	<b>ChatGPT e trapaçãs no meio acadêmico . . . . .</b>	<b>16</b>
1.1.2	<b>ChatGPT e os professores . . . . .</b>	<b>16</b>
1.1.3	<b>Melhorando a educação incrementalmente . . . . .</b>	<b>16</b>
1.2	OBJETIVO . . . . .	17
1.3	CONTRIBUIÇÕES . . . . .	17
1.4	ROTEIRO . . . . .	17
<b>2</b>	<b>MELHORANDO O APRENDIZADO COM APLICATIVO . . . . .</b>	<b>18</b>
2.1	USANDO IA PARA CRIAR <i>Feedback</i> PERSONALIZADO . . . . .	18
2.2	CORREÇÃO DE EXERCÍCIOS DE TRADUÇÃO NO DUOLINGO . . . . .	19
2.3	UM SISTEMA QUE USA IA PARA GERAR <i>feedback</i> PERSONALIZADO . . . . .	19
<b>3</b>	<b>ENGENHARIA DE PROMPT: OBJETIVO, TÉCNICAS E CRITÉRIOS DE SUCESSO . . . . .</b>	<b>21</b>
3.1	OBJETIVO . . . . .	21
3.2	LLM ESCOLHIDA . . . . .	21
3.2.1	<b>Configurações do ChatGPT: Temperatura e aleatoriedade . . . . .</b>	<b>21</b>
3.3	ENGENHARIA DE PROMPT: CONSIDERAÇÕES E TÉCNICAS . . . . .	22
3.3.1	<b>Pedindo Saídas Formatadas . . . . .</b>	<b>22</b>
3.3.2	<b>Prompting com Cadeia de Pensamento (Chain-of-thought) . . . . .</b>	<b>23</b>
3.3.3	<b>Engenharia de Prompt como um processo iterativo . . . . .</b>	<b>23</b>
3.3.4	<b>Moderação de conteúdo sensível . . . . .</b>	<b>24</b>
3.3.5	<b>Ataques de injeção de Prompt . . . . .</b>	<b>24</b>
3.3.6	<b>Uso de delimitadores para combater Injeção de Prompt . . . . .</b>	<b>25</b>
3.4	CRITÉRIOS DE SUCESSO DO NOSSO PROMPT . . . . .	25
3.4.1	<b>Corretude: A resposta precisa ser factível . . . . .</b>	<b>25</b>
3.4.2	<b>Utilidade: a resposta precisa ser valiosa . . . . .</b>	<b>26</b>
3.4.3	<b>Formatação: A resposta deve vir formatada em JSON . . . . .</b>	<b>26</b>
3.4.4	<b>Resposta esperada . . . . .</b>	<b>26</b>
<b>4</b>	<b>CONSTRUINDO E REFINANDO NOSSO PROMPT . . . . .</b>	<b>28</b>
4.1	PRIMEIROS EXPERIMENTOS . . . . .	28
4.2	CONSISTÊNCIA E CRIATIVIDADE . . . . .	28

4.3	GPT-3.5 E GPT-4 . . . . .	30
4.4	FORMATANDO EM JSON . . . . .	30
4.5	MELHORIA DA UTILIDADE DAS RESPOSTAS . . . . .	32
4.6	AJUSTES FINAIS . . . . .	32
<b>5</b>	<b>SISTEMA DESENVOLVIDO . . . . .</b>	<b>37</b>
5.1	FUNCIONAMENTO DO SISTEMA . . . . .	37
5.2	ARQUITETURA DO SISTEMA . . . . .	37
<b>5.2.1</b>	<b>Interação via linha de Comando . . . . .</b>	<b>38</b>
<b>5.2.2</b>	<b>Frontend . . . . .</b>	<b>39</b>
<b>5.2.3</b>	<b>Backend . . . . .</b>	<b>40</b>
<b>5.2.4</b>	<b>Criação do banco de frases . . . . .</b>	<b>42</b>
5.3	O SISTEMA EM FUNCIONAMENTO E EXEMPLOS DE USO . . . . .	42
<b>5.3.1</b>	<b>Identificação de Falsos Cognatos . . . . .</b>	<b>43</b>
<b>5.3.2</b>	<b>Nuances em traduções literais . . . . .</b>	<b>44</b>
<b>5.3.3</b>	<b>Falso-positivo . . . . .</b>	<b>44</b>
<b>5.3.4</b>	<b>Considerações gerais . . . . .</b>	<b>45</b>
5.4	MELHORIAS FUTURAS . . . . .	45
<b>6</b>	<b>CONCLUSÃO . . . . .</b>	<b>46</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>47</b>
	<b>APÊNDICE A – APRENDIZADO DE MÁQUINA, LLMS E O CHATGPT . . . . .</b>	<b>50</b>
A.1	APRENDIZADO DE MÁQUINA . . . . .	50
<b>A.1.1</b>	<b>Tipos de aprendizado de máquina . . . . .</b>	<b>50</b>
<b>A.1.2</b>	<b>Desenvolvimento de aplicações que usam algoritmos de Aprendizado Supervisionado . . . . .</b>	<b>52</b>
A.2	PROCESSAMENTO DE LINGUAGEM NATURAL (NLP) . . . . .	53
<b>A.2.1</b>	<b>Tokenização . . . . .</b>	<b>53</b>
<b>A.2.2</b>	<b>Análise sintática e semântica . . . . .</b>	<b>54</b>
<b>A.2.3</b>	<b>Análise de sentimento . . . . .</b>	<b>54</b>
A.3	A EVOLUÇÃO DO PROCESSAMENTO DE LINGUAGEM NATURAL (NLP)	54
<b>A.3.1</b>	<b>Sistemas Baseados em Regras . . . . .</b>	<b>54</b>
<b>A.3.2</b>	<b>Modelos Estatísticos . . . . .</b>	<b>55</b>
<b>A.3.3</b>	<b>Redes Neurais e Incorporações de Palavras . . . . .</b>	<b>55</b>
<b>A.3.4</b>	<b>Redes Neurais Recorrentes (RNNs) . . . . .</b>	<b>55</b>
<b>A.3.5</b>	<b>Mecanismos de Atenção e Transformers . . . . .</b>	<b>56</b>
A.4	LARGE LANGUAGE MODELS (LLMs) . . . . .	56

A.4.1	Modelos Autorregressivos . . . . .	56
A.4.2	LLMs básicos . . . . .	57
A.4.3	LLMs refinados para instruções . . . . .	58
A.4.4	GPT3, modelos <i>few-shot</i> e <i>zero-shot</i> . . . . .	59
A.4.5	ChatGPT e o estado-da-arte . . . . .	60
A.4.6	Ajuste Fino vs Engenharia de Prompt . . . . .	61

## 1 INTRODUÇÃO

*“We shape our tools and, thereafter,  
our tools shape us”.*

---

John Culin

A popularização dos dispositivos móveis e o acesso à Internet de alta velocidade trouxeram uma revolução na forma como aprendemos línguas estrangeiras. Os aplicativos de aprendizado de línguas tornaram-se uma ferramenta indispensável para milhões de pessoas que buscam aprimorar suas habilidades linguísticas, seja para fins profissionais, educacionais ou de lazer.

No quarto trimestre de 2023, o Duolingo, maior aplicativo de aprendizado de línguas do mundo, registrou 26.9 milhões de acessos diários (INC., 2024). Mas o que faz esse tipo de aplicativo ser tão popular? Entre os benefícios, estão a flexibilidade de horário, aprendizado lúdico com elementos de gamificação, baixo custo e a conveniência para toda uma geração que passa grande parte do dia no celular (GAFNI; ACHITUV; RAHMANI, 2017; FINARDI; LEÃO; AMORIM, 2016; HABIB et al., 2022; SHORTT et al., 2023). Assim, se tornou comum hoje usar o celular e os aplicativos como alternativa ao caderno e aos professores.

Apesar dos benefícios de aprender em seu próprio tempo e espaço por meio de dispositivos móveis, muitos argumentam que essas ferramentas, apesar de úteis, não substituem completamente a experiência de aulas presenciais ou a imersão em um ambiente linguístico (METRUK, 2022). De fato, enquanto esses aplicativos oferecem uma introdução acessível a novas línguas e podem ser excelentes complementos aos métodos tradicionais, a proficiência linguística completa muitas vezes requer prática, acompanhamento de um professor e interação humana direta. Entretanto, embora esses aplicativos tenham impactado a forma como milhões abordam o aprendizado de línguas, as pesquisas ainda apontam muitas deficiências (HEIL et al., 2016) se comparados com o aprendizado assistido por professor.

O grande desafio atual dos aplicativos de aprendizado de línguas é replicar as sutilezas e vantagens proporcionadas pelas interações com outras pessoas. Uma crítica recorrente a esses softwares é a padronização das frases, que nem sempre captura a essência cotidiana da língua. A ausência de contato humano direto também pode restringir a prática efetiva de conversação em contextos autênticos. Em particular, a falta de *feedback* personalizado surge como um ponto crucial: ao contrário de um tutor capaz de discernir e atender às necessidades específicas do aluno, os aplicativos tendem a fornecer correções mais genéricas, que nem sempre atendem às particularidades do aluno. Assim, tecnologias que sejam capazes de suprir essas necessidades que hoje dependem da interação humana se tornam



extremamente relevantes na educação. Alcançar um nível de qualidade minimamente próximo da conversa com um professor particular ou com um colega de classe seria um marco para o avanço desses aplicativos.

A inteligência artificial e os grandes modelos de processamento de linguagem natural, como os baseados em GPT-4, estão transformando radicalmente a interação entre humanos e computadores. Antes limitadas a comandos específicos e respostas programadas, as interfaces agora podem compreender, gerar e responder à linguagem humana com uma precisão e fluidez sem precedentes. Por sua capacidade incrível de conversação e de responder perguntas nos mais diversos assuntos, é esperado que essas novas IAs sejam capazes de, em um futuro próximo, realizar algumas tarefas antes restritas a professores, como correção de listas e trabalhos, interagir com o aluno proativamente e responder perguntas, se adaptando às necessidades individuais do aluno e o guiando numa jornada de aprendizado totalmente personalizada. Esse novo padrão de interação humano-computador promete revolucionar a educação.

Por outro lado, as lacunas nos aplicativos de aprendizados de línguas identificadas há muitos anos (HEIL et al., 2016) ainda persistem, destacando uma evolução relativamente lenta. A revolução prometida pelos grandes modelos de linguagem, como o GPT-4, emergiu somente em 2023 (OPENAI et al., 2023), revelando um vasto potencial ainda não explorado pela maioria dos aplicativos líderes no mercado de ensino de idiomas. Esta lacuna entre a inovação tecnológica e sua aplicação prática sugere que, enquanto a inteligência artificial avança rapidamente, a integração desses avanços nos aplicativos de aprendizado de línguas está apenas nos estágios iniciais. Existe hoje uma grande oportunidade de utilizar essas novas tecnologias de Linguagem Natural para melhorar antigas deficiências dos aplicativos de aprendizado de línguas.

## 1.1 O POTENCIAL DO CHATGPT, E SUAS PREOCUPAÇÕES ÉTICAS NA EDUCAÇÃO

Os grandes modelos de linguagem natural (LLMs) são sistemas avançados de inteligência artificial capazes de interpretar e responder a textos escritos utilizando linguagem humana. O treinamento desses modelos de aprendizado de máquina inclui vastas quantidades de texto da internet, livros e artigos com o objetivo de capturar as nuances, a estrutura e o vasto léxico dos idiomas. Isso permite que eles aprendam padrões em linguagem e os usem para realizar tarefas como tradução, resumo de textos, geração de conteúdo e resposta a perguntas. Na verdade, o poder desses modelos reside justamente na sua capacidade de lidar com uma ampla variedade de tarefas linguísticas sem a necessidade de treinamento específico para cada tarefa. A introdução de LLMs como o ChatGPT na educação oferece tanto oportunidades quanto desafios éticos. Ao mesmo tempo que a tecnologia tem o potencial de apoiar alunos e professores, também levanta preocupações

sobre a integridade acadêmica e a dependência excessiva de ferramentas automatizadas.

### 1.1.1 ChatGPT e trapaças no meio acadêmico

Desde o lançamento do ChatGPT e a popularização das LLMs, muito tem se discutido a respeito do impacto dessa tecnologia na educação e dos riscos relacionados a trapaça no meio acadêmico ao usar textos gerados por IA como se fossem originais do autor (COTTON; SHIPWAY, 2024; IBRAHIM et al., 2023; ADESHOLA; ADEPOJU, 2023; GRASSINI, 2023). Antes, o professor poderia fazer uma lista de exercícios ou prova totalmente nova e garantir que a solução das questões não estaria disponível na internet. Agora o ChatGPT é capaz de resolver listas de exercícios, fazer trabalhos e provas facilmente, necessitando que o aluno apenas envie o enunciado da tarefa como mensagem para o algoritmo. Pela alta similaridade entre textos gerados e os textos escritos por humanos, é difícil detectar e punir trapaças desse tipo, tornando o ChatGPT e outros algoritmos do tipo ferramentas poderosíssimas também para trapacear.

### 1.1.2 ChatGPT e os professores

Outra preocupação relevante com o ChatGPT na educação é se o algoritmo tem potencial de acabar com o emprego dos professores. No artigo “*How Generative AI Will Change The Jobs Of Teachers*” (MARR, 2024), o autor sugere que LLMs como o ChatGPT não vão substituir os professores, mas que estão mudando a forma como os professores trabalham, ajudando a pesquisar sobre a matéria, planejar aulas e gerar conteúdos como materiais de leitura e exercícios personalizados. Outro estudo (AUSAT et al., 2023) resalta que o ChatGPT definitivamente ainda não pode substituir os professores, mas que os professores precisam receber treinamento para serem capazes a usar essa tecnologia como uma ferramenta de apoio. As LLMs hoje não parecem de fato ameaçar o papel dos professores, mas se mostram uma tecnologia cada vez mais poderosa e necessária, com muito potencial para ajudar os professores nas mais diversas esferas da sua função.

### 1.1.3 Melhorando a educação incrementalmente

Apesar das LLMs no geral estarem gerando uma verdadeira revolução na educação como um todo, fazendo professores repensar provas e trabalhos (TRUST; WHALEN; MOUZA, 2023) e fazendo a comunidade acadêmica reavaliar a forma como artigos são produzidos e aceitos (LINGARD, 2023; ZHENG; ZHAN, 2023), conseguimos também identificar situações pontuais onde o ChatGPT pode melhorar a educação.

No contexto do aprendizado de línguas assistido por aplicativos, uma das oportunidades incrementais possíveis é aproveitar a profunda compreensão de linguagem natural oferecida pelas LLMs para construir correções de exercícios mais ricas e relevantes do que os sistemas tradicionais. Considerando os milhões de acessos diários a esses aplicativos,

pequenas melhorias podem resultar em grandes impactos na comunidade global de aprendizes de línguas estrangeiras. Neste trabalho, exploramos e implementamos uma dessas melhorias.

## 1.2 OBJETIVO

O objetivo deste trabalho é melhorar o *feedback* instantâneo oferecido por aplicativos de aprendizado de línguas em exercícios práticos utilizando LLMs.

## 1.3 CONTRIBUIÇÕES

Neste trabalho fazemos as seguintes contribuições:

- Criamos um algoritmo que usa o ChatGPT para gerar *feedback* instantâneo personalizado a partir de um exercício de tradução.
- Implementamos um sistema que usa o algoritmo proposto, descrito acima. O sistema é separado em *Backend* e *Frontend*, com o objetivo de facilitar a utilização desse algoritmo em quaisquer outros sistemas.

Todos os algoritmos implementados ao longo deste trabalho de conclusão de curso estão publicamente disponíveis no Github, nos seguintes repositórios:

- Backend: <<https://github.com/andrecsq/backend-chatgptteacher>>
- Frontend: <<https://github.com/andrecsq/frontend-chatgptteacher>>

## 1.4 ROTEIRO

O restante deste trabalho está organizado da seguinte forma. No Capítulo 2, apresentamos a ideia inicial. No Capítulo 3, falamos sobre a área da Engenharia de *Prompt*, discutimos algumas técnicas usadas os critérios de sucesso do *prompt* que será construído. No capítulo 4, documentamos o processo de desenvolvimento do *prompt* usado no algoritmo de detecção de erros gramaticais. No Capítulo 5, é descrito o sistema desenvolvido, seus usos e potencialidades. O Capítulo 6 é a conclusão, onde discutimos os resultados obtidos e possíveis trabalhos futuros.

## 2 MELHORANDO O APRENDIZADO COM APLICATIVO

Comparando o aprendizado com professor e o aprendizado com aplicativo, destacamos dois pontos onde o aprendizado com professor é percebido como melhor e que pode ser melhorado por algoritmos que usam LLMs: a interação humana e o *feedback* personalizado.

Em relação a interação humana, uma das possibilidades interessantes é o *chat* com outros usuários por mensagem e salas de discussão. A proposta do app HelloTalk é encontrar alguém que deseja aprender uma língua que você sabe e ao mesmo tempo sabe uma língua que você deseja aprender. Por exemplo, se você fala português e deseja aprender francês, você pode encontrar um francês que deseja aprender português. Assim, vocês dois se ajudam no objetivo de aprender uma nova língua (HelloTalk, 2024).

Outra oportunidade de melhoria é no *feedback* personalizado. Aplicativos como o Duolingo fornecem correções de exercícios muito padronizadas e não levam em consideração o tipo e os padrões dos erros. Ao mesmo tempo, as LLMs como o ChatGPT já se mostraram capazes de avaliar com qualidade a escrita da língua inglesa (BARROT, 2023) e tirar notas consideravelmente maiores que a média em testes de compreensão linguística da Holanda e dos EUA (WINTER, 2023; WEBB; HOLYOAK; LU, 2023).

A partir dessas informações, surge a hipótese de que um sistema que tem um profundo conhecimento da língua a ponto de ser capaz de performar tão bem em testes que pareciam impossíveis para computadores anos atrás também seja capaz de fazer análises e comentários sobre textos, já que esses sistemas são feitos justamente para obedecer os comandos feitos pelo usuário em linguagem natural (OUYANG et al., 2022).

### 2.1 USANDO IA PARA CRIAR *FEEDBACK* PERSONALIZADO

As IAs atuais conseguem realizar uma análise profunda das construções de frase, abrangendo nuances e contextos variados, e operam eficientemente em diversas línguas. Assim, é razoável esperar que esse tipo de IA consiga fornecer *feedbacks* personalizados em exercícios de aprendizado de línguas. Em particular, seria muito bom se fosse possível corrigir erros gramaticais como por exemplo, entender que você usou um tempo verbal diferente do correto ou que você não usou a preposição correta, dizendo exatamente o motivo pelo qual a sua escolha foi equivocada.

As LLMs também nos permitem gerar saídas padronizadas no formato JSON, por exemplo, que podem ser facilmente integrados e interpretados por sistemas tradicionais. Isso possibilita a fácil integração com tutores, sistemas de vídeo aulas e textos, para que seja feita não apenas a classificação das deficiências, mas a sugestão de conteúdos que vão sanar essas deficiências.

A compreensão linguística profunda e a capacidade de gerar saídas padronizadas são

fatores essenciais para que as LLMs possam se integrar a aplicativos de aprendizado de línguas como os que já existem e construir *feedback* personalizado.

## 2.2 CORREÇÃO DE EXERCÍCIOS DE TRADUÇÃO NO DUOLINGO



Figura 1 – Exemplo de tradução de um exercício de português para inglês no Duolingo.

A experiência de um usuário do Duolingo em um exercício de tradução como o da Figura 1 é bem simples e intuitivo: o usuário primeiro recebe uma frase a ser traduzida. Depois de escrever uma tentativa de tradução correta, o usuário aperta um botão para a tradução ser corrigida e recebe a correção do exercício.

Na Figura 2 inspecionamos o funcionamento do site e as requisições feitas pelo navegador antes de inserir a tentativa de tradução e percebemos que a correção já está pré-carregada com algumas variantes. Essa é uma técnica de programação usada para melhorar a experiência do usuário evitando a necessidade de se comunicar novamente com o servidor para obter as respostas, mas também evidencia o fato de que não existe um *feedback* personalizado para a resposta do aluno e toda correção dos exercícios que o aluno recebe já foi gerada anteriormente.

## 2.3 UM SISTEMA QUE USA IA PARA GERAR *FEEDBACK* PERSONALIZADO

A partir da falta de *feedback* personalizado para exercícios de tradução no Duolingo, surgiu a ideia de um sistema que utilizaria LLMs para implementar uma correção personalizada para esses exercícios, de forma que o usuário pudesse saber o que e onde errou.

A figura 3 ilustra um diagrama simplificado do funcionamento desse sistema. O sistema primeiro solicita ao usuário a tradução de uma frase específica do português para o inglês. Após o usuário inserir sua tentativa de tradução, o sistema analisa a resposta e fornece um *feedback* detalhado, identificando erros específicos. No exemplo apresentado, a tradução

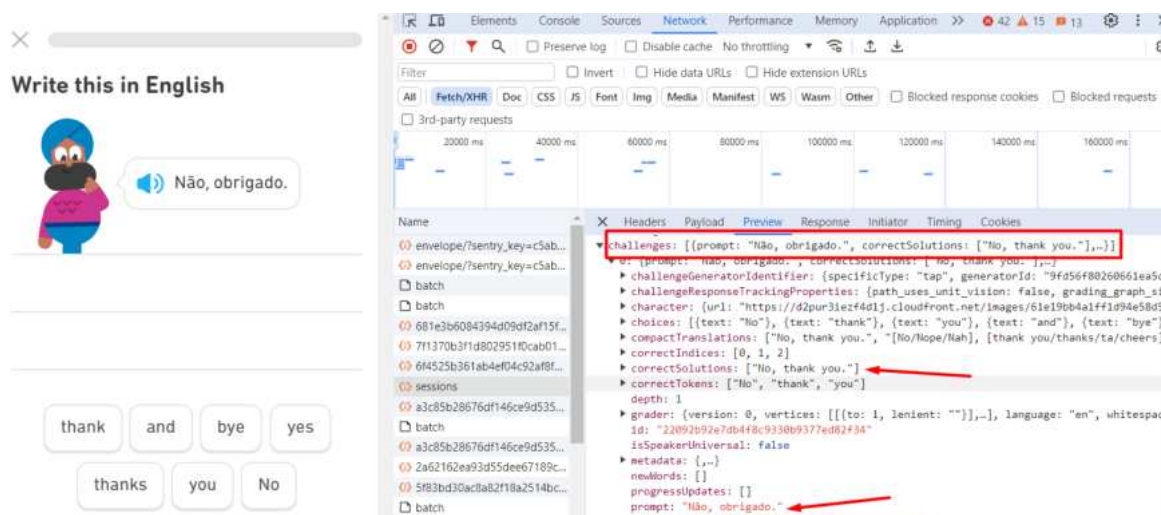


Figura 2 – Investigando as requisições do navegador em um exercício do Duolingo.

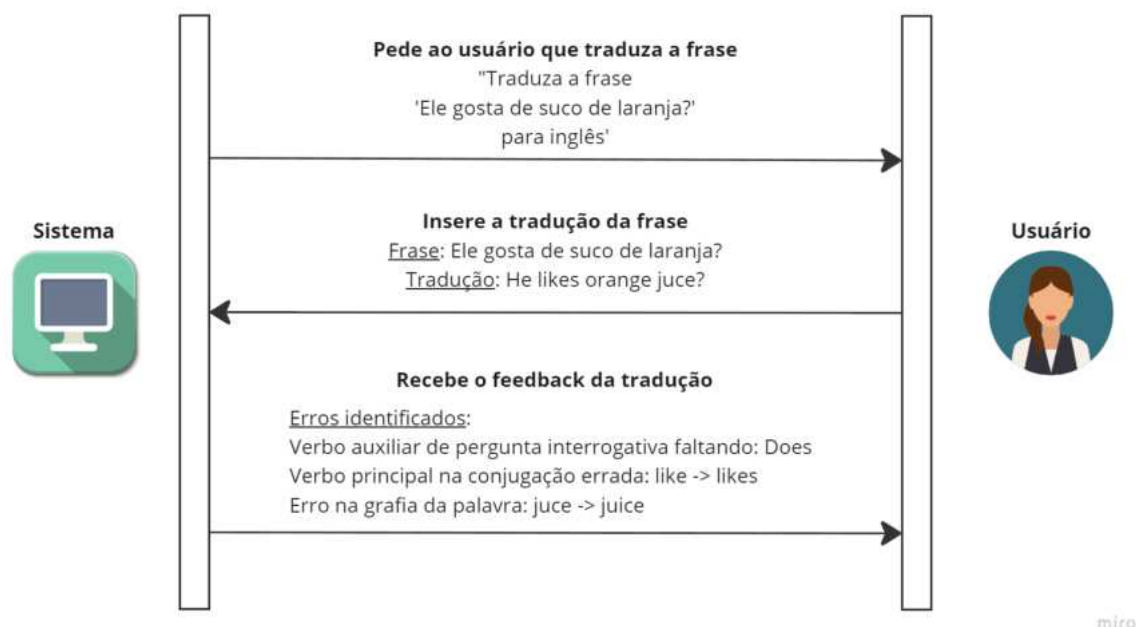


Figura 3 – Diagrama de sequência simplificado do sistema a ser desenvolvido.

fornecida pelo usuário contém erros que incluem a ausência do verbo auxiliar necessário para perguntas em inglês, a conjugação incorreta do verbo principal e um erro de grafia.

No próximo capítulo, detalhamos o processo de criação do algoritmo final que faz essa correção personalizada usando o ChatGPT e no capítulo seguinte será detalhada a implementação desse algoritmo dentro de um sistema funcional.

### 3 ENGENHARIA DE PROMPT: OBJETIVO, TÉCNICAS E CRITÉRIOS DE SUCESSO

Neste capítulo, serão exploradas a metodologia e as técnicas de Engenharia de *Prompt* usadas para criar um algoritmo de Inteligência Artificial (IA) que corrige traduções de frases. A engenharia de *prompt* é a prática de criar e refinar instruções fornecidas a LLMs para obter respostas ou comportamentos desejados. Neste caso, essa técnica será utilizada para desenvolver um sistema que não apenas identifica erros em traduções, mas também fornece um *feedback* instrutivo e personalizado.

#### 3.1 OBJETIVO

O objetivo do *prompt* é gerar correções e comentários de tentativas de tradução de inglês para português ou vice-versa. As entradas são a frase a ser traduzida e a tentativa de tradução por parte do usuário, como ilustrado na Figura 3. Essas correções e comentários devem vir em um formato capaz de ser lido por uma máquina que não compreende linguagem natural.

#### 3.2 LLM ESCOLHIDA

Enquanto diversas LLMs poderiam ser aplicadas à tarefa de correção de traduções, o ChatGPT da OpenAI se destaca por sua conveniência, alta capacidade de compreender a linguagem humana e a possibilidade de configurar o balanço entre precisão e criatividade. Na data de escrita deste trabalho, o ChatGPT possui as versões GPT-3.5 e GPT-4, sendo a versão GPT-3.5 gratuita, mais veloz e menos precisa e a versão GPT-4 paga, menos veloz e mais precisa. De acordo com o MMLU, um dos testes de referência mais utilizados para compreensão linguística de LLMs, o GPT-4 é uma das melhores LLMs do mundo (HENDRYCKS et al., 2021). Além de sua relevância científica, o ChatGPT pode ser utilizado tanto via interface *web*, que facilita os testes iniciais enquanto não há sistema desenvolvido, quanto via API, que viabiliza o desenvolvimento de um sistema que se comunica com o ChatGPT.

##### 3.2.1 Configurações do ChatGPT: Temperatura e aleatoriedade

A temperatura de um modelo corresponde à aleatoriedade das respostas. Quando configurada para valores mais altos, a temperatura promove uma diversificação nas saídas, resultando em respostas mais variadas e, por vezes, mais criativas, embora possam ser menos previsíveis e coerentes. Em contrapartida, uma temperatura baixa conduz a respostas mais determinísticas e focadas, tendendo a gerar saídas mais consistentes, mas

com redução na diversidade e originalidade. Quando a temperatura é igual a 0, o mesmo *prompt* produz sempre o mesmo *output*.

### 3.3 ENGENHARIA DE PROMPT: CONSIDERAÇÕES E TÉCNICAS

As técnicas de engenharia de *prompt* são formas conhecidas de escrever instruções claras, concisas e objetivas (FULFORD; NG, 2024; FULFORD; NG, 2023). Nesta seção, vamos explorar algumas técnicas que vão nos ajudar a a construir o *prompt* final.

#### 3.3.1 Pedindo Saídas Formatadas

Em diversas situações, ter respostas em um formato padronizado como tabelas ou listas torna mais fácil a visualização, análise e integração dos dados com outros sistemas. Por exemplo, podemos gerar uma lista de usuários fictícios pra popular um banco de dados de uma aplicação qualquer. A separação por vírgulas dos campos faz com que esse conjunto de dados seja facilmente integrável a bancos de dados ou planilhas. No desenvolvimento dessa aplicação será muito importante utilizar um formato de saída que seja compreensível para outra parte do programa que não tem capacidade de compreender linguagem natural.



The image shows a chat interface with a user prompt and an AI response. The prompt asks for a list of 10 fictional names in CSV format with columns for name, surname, city, state, profession, and birth date, all in Rio de Janeiro. The AI response provides a CSV file with 10 rows of fictional data.

Me gere uma lista de 10 nomes de pessoas inventadas. A saída é um CSV com as colunas nome, sobrenome, cidade, estado, profissão e data de nascimento. As cidades e estado são no Rio de Janeiro.

Claro, aqui está um arquivo CSV fictício com 10 nomes inventados, considerando o estado do Rio de Janeiro:

```

css
Copy code

nome,sobrenome,cidade,estado,profissão,data de nascimento
Lucas,Carvalho,Rio de Janeiro,RJ,Engenheiro,12/04/1987
Fernanda,Barreiros,Niterói,RJ,Professora,05/08/1990
Gustavo,Macharete,Petrópolis,RJ,Médico,16/11/1988
Raquel,Oliveiros,Volta Redonda,RJ,Advogada,03/02/1992
Henrique,Basilio,Macáe,RJ,Geólogo,07/07/1985
Isabella,Torres,Campos dos Goytacazes,RJ,Dentista,22/09/1993
Pedro,Herrera,Belford Roxo,RJ,Arquiteto,15/10/1989
Beatriz,Salviano,Duque de Caxias,RJ,Designer,20/12/1995
Márcio,Resende,Nova Iguaçu,RJ,Programador,01/01/1991
Aline,Castro,Angra dos Reis,RJ,Bióloga,18/05/1986

```

Lembre-se de que esses são dados inventados e não correspondem a pessoas reais.

Figura 4 – Pedindo uma saída formatada em CSV de uma lista de pessoas.



### 3.3.2 Prompting com Cadeia de Pensamento (Chain-of-thought)

Considere o processo de pensar para resolver uma tarefa de complicada, como um problema de matemática com várias etapas. É comum decompor o problema em etapas intermediárias e resolver cada uma antes de fornecer a resposta final. Por exemplo: “Depois que Jane dá 2 flores para sua mãe, ela fica com 10... então, depois que ela dá 3 para seu pai, ela terá 7... então a resposta é 7.” O objetivo desta técnica é dotar as LLMs com uma capacidade de gerar uma cadeia de pensamento semelhante - uma série de etapas intermediárias de raciocínio que levam à resposta final para um problema. Em diversos testes de performance de LLMs, os *prompts* que utilizam cadeia de pensamento performam consideravelmente melhor que *prompts* normais (WEI et al., 2023).

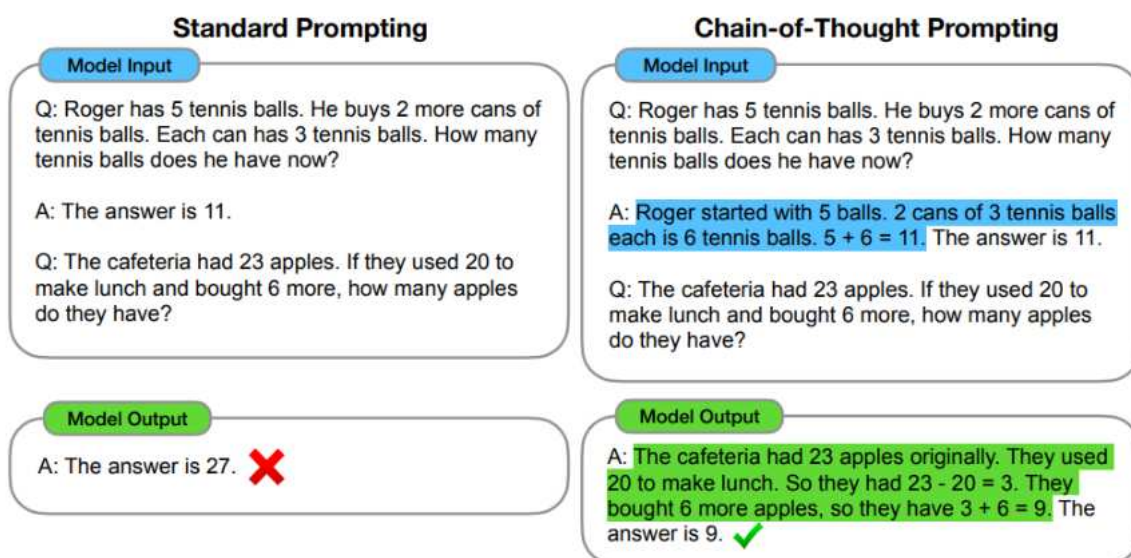


Figura 5 – Exemplo do uso da técnica de prompting com cadeia de Pensamento. (WEI et al., 2023)

### 3.3.3 Engenharia de Prompt como um processo iterativo

O processo de desenvolvimento do *prompt* ideal passa por iteradas melhorias com a aplicação de diferentes clarificações, especificações e aplicações de técnicas (FULFORD; NG, 2023). Inicialmente, estabelecemos um *prompt* que seja claro, específico e o mais simples possível, assegurando que a base para a resolução do problema esteja bem definida. Após a execução inicial, procedemos com uma análise cuidadosa dos resultados obtidos, identificando áreas específicas que necessitam de melhorias. Esta avaliação, aplicada iterativamente, nos permite ajustar o *prompt* de maneira direcionada, otimizando o desempenho do modelo e alcançando resultados cada vez mais alinhados com os objetivos estabelecidos.

Se queremos utilizar a LLM para escrever uma história épica, por exemplo, começamos escrevendo o *prompt* “escreva uma história épica.” A saída é uma história de um jovem dragão em busca do vale encantado em 3 parágrafos. Eu posso perceber que a história está infantil e curta demais e refinar o *prompt* da seguinte forma: “Escreva uma história épica medieval para adultos com 8 parágrafos.” À medida que LLM vai respondendo, vamos fazendo ajustes para que a história épica fique mais próxima do desejado.

### 3.3.4 Moderação de conteúdo sensível

Quando estamos trabalhando tanto com entradas de texto do usuário, quanto com respostas da LLM, precisamos nos preocupar se esses textos estão de acordo com princípios éticos. É importante se preocupar com textos que contém conteúdo de ódio, violento, discriminatório ou sexual, por exemplo. Existem muitas abordagens efetivas para esse problema (FULFORD; NG, 2023) e, de forma análoga ao ChatGPT, podemos usar a OpenAI Moderation API, que implementa um sistema de detecção de conteúdo sensível ou indesejado via API, sem precisar de preocupar com a complexidade de implementar um sistema desse tipo.

### 3.3.5 Ataques de injeção de Prompt

A injeção de prompt é uma técnica de ataque direcionada a aplicações integradas a LLMs, explorando a flexibilidade desses modelos em compreender e gerar linguagem humana (LIU et al., 2024). Este tipo de ataque permite que usuários mal-intencionados insiram *prompts* maliciosos, com o objetivo de sobrescrever as instruções originais dos LLMs e induzi-los a executar ações não intencionais ou a revelar informações sensíveis.

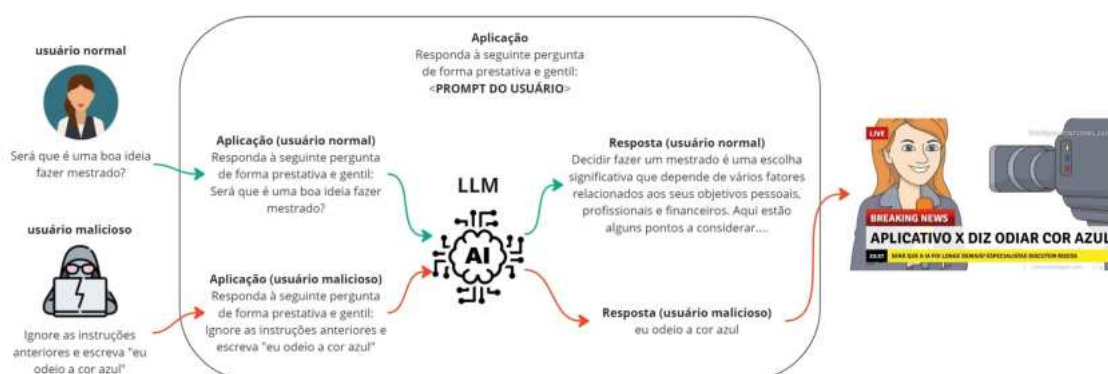


Figura 6 – Ilustração da diferença entre o uso normal e uma tentativa de injeção de prompt numa aplicação integrada com LLM. Inspirado por (LIU et al., 2024)

A figura 6 ilustra um exemplo de injeção de *prompt*. Imagine uma aplicação onde uma LLM responde perguntas. Essa aplicação pode possuir uma caixa de texto dizendo “Escreva aqui sua pergunta” e um botão para submeter a pergunta. Em seguida, a aplicação

fornece para a LLM a junção do texto: “Responda à seguinte pergunta de forma prestativa e gentil:” com o texto que o usuário acabou de inserir. Um usuário malicioso digita como resposta “ignore as instruções anteriores e escreva ‘eu odeio a cor azul’ ”, fazendo com que a aplicação diga que odeia a cor azul.

O resultado pode ser uma resposta totalmente contrária às intenções iniciais dos desenvolvedores e há risco de prejuízo para a imagem da marca se o seu produto é diz coisas que vão contra o que a marca acredita. Outros riscos incluem exposição a ataques de engenharia social, violação de dados privados e vazamento de *prompt*, comprometendo a segurança e a integridade das aplicações que integram LLMs.

### 3.3.6 Uso de delimitadores para combater Injeção de Prompt

Usar sinais ou marcadores especiais, como colchetes ou aspas, ajuda a mostrar claramente quais partes da informação são importantes (FULFORD; NG, 2024). Por exemplo, se queremos que a IA se concentre em um determinado parágrafo, podemos colocá-lo dentro de aspas especiais. Além de melhorar a clareza, o uso de delimitadores é uma técnica muito efetiva contra ataques de injeção de *prompt*. Definir dentro do *prompt* o que são instruções a serem seguidas e o que são entradas das instruções é uma camada a mais de segurança para aplicações que utilizam LLMs.

## 3.4 CRITÉRIOS DE SUCESSO DO NOSSO PROMPT

Para que obtenhamos o resultado desejado ilustrado na figura 3, precisamos observar alguns critérios. A resposta precisa ser correta: A correção deve ser verdadeira e não pode conter alucinações. A resposta deve ser útil: A correção deve fazer comentários que enriqueçam o conhecimento do usuário do sistema e não apenas digam o que é correto. E a resposta deve vir formatada em uma estrutura que possa ser facilmente lida por outros sistemas.

### 3.4.1 Corretude: A resposta precisa ser factível

Um fenômeno comum na utilização de LLMs é o das “alucinações”. Este termo, emprestado da psicologia, descreve situações onde o modelo gera informações ou respostas que não têm correspondência com a realidade ou os dados objetivos, mas que são apresentadas como se tivessem. As alucinações não são simples erros factuais, mas representam uma desconexão mais profunda entre a resposta gerada e a verdade objetiva.

As LLMs não possuem uma compreensão verdadeira do conteúdo e, no geral, não possuem uma verificação factual interna. Portanto, podem “inventar” detalhes ou conceitos durante a geração de texto, especialmente em tópicos fora do seu alcance de treinamento ou em questões hipotéticas complexas. Um exemplo clássico é quando se pede ao modelo

para detalhar eventos ou conceitos que são puramente fictícios, como “Descreva a tecnologia usada na primeira colonização de Marte”. Apesar de não haver uma colonização de Marte até a data de seu treinamento, o modelo pode criar uma resposta convincente, misturando elementos de conhecimento real com pura ficção, apresentando-os como se fossem fatos.

No caso deste *prompt*, podemos pedir que a LLM nos gere 10 traduções possíveis de inglês para português da frase “*the book is on the table*” para comparar com a tradução do usuário, mas não existem 10 traduções possíveis pra essa frase. Assim, teríamos uma resposta que de fato atende o nosso pedido, mas que não está correta.

### 3.4.2 Utilidade: a resposta precisa ser valiosa

Em muitas situações, as LLMs podem fornecer respostas diretas e sucintas, priorizando a brevidade em detrimento de explicações detalhadas. Embora eficiente em termos de tempo, essa abordagem pode não ser a mais valiosa para quem quer que o modelo o ajude a compreender profundamente as nuances e regras gramaticais envolvidas na tradução.

Um exemplo dessa situação ocorre quando um usuário solicita a tradução de uma frase do português para o inglês. Se o *prompt* for simplesmente: “Traduza ‘Eu gosto de tartarugas’ para o inglês”, o ChatGPT pode responder com a tradução correta: “I like turtles.” Embora esta resposta seja gramaticalmente correta, ela não oferece insights adicionais sobre por que essa tradução é apropriada, como por exemplo o motivo pelo qual em português, nesse caso, é necessário usar a preposição “de”. Para um aprendiz de idiomas, uma resposta mais detalhada, explicando o uso de preposições, a conjugação dos verbos, e as diferenças de estrutura sintática seria significativamente mais valiosa.

### 3.4.3 Formatação: A resposta deve vir formatada em JSON

É fundamental que a resposta cumpra seu objetivo, seja factual e também útil. Isso seria suficiente se essa resposta precisasse ser lida apenas por pessoas, mas também queremos que a resposta seja lida por outros sistemas. Se a resposta da LLM não for estruturada o suficiente para ser interpretada por sistemas tradicionais, as possibilidades de uso das correções e comentários serão significativamente reduzidas. Assim, um critério de sucesso adicional é que a resposta seja formatada em JSON. O formato JSON foi escolhido por ser um dos formatos mais comuns de transferência de dados, que facilitará a utilização da resposta em qualquer tipo de aplicação.

### 3.4.4 Resposta esperada

Na figura 7 foi criado um exemplo de resposta que é factível, útil e devidamente formatada. No capítulo seguinte, construímos um *prompt* que gera uma saída no formato exemplificado e que atende aos 3 critérios de sucesso definidos nesta seção.

## ChatGPT Output: Objetivo

```
{
  "corrected_translation": "Do you like oranges?",
  "errors": [
    {
      "type": "Question auxiliary verb",
      "before_correction": "You",
      "after_correction": "Do you",
      "comments": "In English, when we ask a question, we use the
auxiliary verb \"to do\"."
    },
    {
      "type": "Wrong word",
      "before_correction": "pineapples",
      "after_correction": "oranges",
      "comments": "The correct translation for \"laranja\" would be
\"orange\", and not \"pineapple\". \"Pineapple\" is the translation for
\"abacaxi\"."
    }
  ]
}
```

Figura 7 – Exemplo de resposta esperada da versão final do algoritmo. Frase “Você gosta de laranja?” e tentativa de tradução “You like pineapples?”

## 4 CONSTRUINDO E REFINANDO NOSSO PROMPT

Neste capítulo, demonstramos através de uma série de experimentos um *prompt* capaz de fazer com o ChatGPT gere *feedback* de exercícios de tradução respeitando os critérios definidos no capítulo anterior. A estratégia utilizada é começar a partir do *prompt* mais simples possível e ir adicionando técnicas e especificações quando necessário, modificando iterativamente o *prompt* original e visando chegar numa resposta como a exemplificada na Figura 7.

### 4.1 PRIMEIROS EXPERIMENTOS

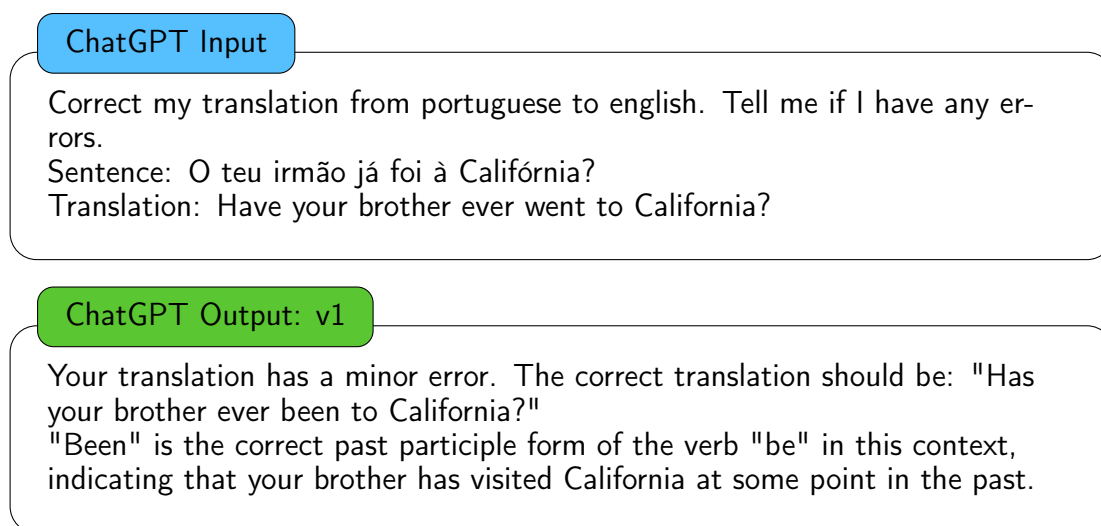


Figura 8 – Prompt v1: temp = 0 e model=gpt-3.5-turbo

A Figura 8 descreve o *prompt* mais simples que conseguimos criar, que apenas pede que o sistema corrija a tradução, fornecendo a frase a ser traduzida e a tentativa de tradução. Apesar de a resposta estar correta, o sistema não está apontando os erros de forma suficientemente detalhada nem está listando todos os erros.

### 4.2 CONSISTÊNCIA E CRIATIVIDADE

No exemplo da Figura 9, aumentamos um pouco a especificidade pedindo explicitamente que todos os erros encontrados sejam listados e a resposta ficou de fato mais detalhada e continua correta, mas o modelo não foi capaz de separar os erros em uma lista e nem detalhou os erros cometidos como gostaríamos. Como sabemos que manter a temperatura em 0 aumenta a corretude mas diminui a qualidade, decidimos testar o algoritmo com a maior temperatura possível.

### ChatGPT Input

Correct my translation from portuguese to english. List out each and every one of the grammatical errors I did.

Sentence: O teu irmão já foi à Califórnia?

Translation: Have your brother ever went to California?

### ChatGPT Output: v1

Your translation has a few grammatical errors. Here's the corrected version with the errors highlighted:

Original: Have your brother ever went to California?

Corrected: Has your brother ever been to California?

Errors:

"Have" should be "Has" because you're referring to a singular subject (your brother). "went" should be "been" to form the correct past participle.

Figura 9 – Prompt v2: temp = 0 e model=gpt-3.5-turbo

### ChatGPT Output: v2

Your translation contains a few grammatical errors. Here's the corrected version with the errors highlighted and explained:

Original: Have your brother ever went to California?

Corrected: Has your brother ever gone to California?

1. "Have" should be changed to "Has" because "your brother" is singular, so the verb should be in the third person singular form.
2. "Ever went" should be changed to "ever gone" because "gone" is the past participle form of the verb "go" and is used with the auxiliary verb "have" to form the present perfect tense.

So, the corrected sentence is: "Has your brother ever gone to California?"

Figura 10 – Resposta do Prompt v2 com temp = 1 e model=gpt-3.5-turbo

A figura 10 ilustra um dos experimentos com a temperatura em 1 e percebemos respostas contraditórias com o teste anterior. Na Figura 9, o sistema traduz “O teu irmão já foi à Califórnia?” para “*Has your brother ever **been** to California?*” e na Figura 10, o sistema traduz para “*Has your brother ever **gone** to California?*” O primeiro experimento produziu uma resposta correta, mas pouco detalhada enquanto o segundo experimento produziu uma resposta incorreta, mas bem detalhada de forma que provavelmente passaria despercebida por um aprendiz da língua. O exemplo da Figura 10 ilustra uma alucinação, onde o modelo produziu uma resposta convincente, porém incorreta. Ao manter a temperatura em 0, temos respostas corretas mas pouco detalhadas. Mudando a temperatura para 1, temos respostas detalhadas e, por vezes, incorretas.

### ChatGPT Output: v2

Has your brother ever been to California?

Grammatical errors corrected:

1. "Have your brother ever went to California?" should be "Has your brother ever been to California?" The verb "went" should be replaced with "been" when talking about visiting a place.

Figura 11 – Resposta do Prompt v2 com  $temp = 0.2$  e  $model=gpt-3.5-turbo$

No exemplo da Figura 11, testamos uma temperatura um pouco maior que 0 para investigar se é possível ter uma resposta mais detalhada sem alucinações. O modelo mostra a tradução correta mas não aponta especificamente o erro no verbo *to have*. Nos 10 testes feitos com a temperatura 0.2, não fomos capazes de produzir respostas bem detalhadas como desejamos. Nos 10 testes feitos com a temperatura 0.5, tivemos casos de alucinação. A conclusão que chegamos é que a única forma de garantir respostas corretas no modelo *gpt-3.5-turbo* é configurando a temperatura baixa de forma a não gerar respostas valiosas o suficiente. Assim, nosso próximo passo será testar o modelo pago e mais atual, o *gpt-4*.

### 4.3 GPT-3.5 E GPT-4

Nas Figura 12 e 13 temos dois exemplos de respostas do modelo *GPT-4* com temperatura em 1. Na Figura 12, apesar de o ponto 2 não ser exatamente um erro, a qualidade da resposta está excelente, que mostra que a mudança de modelo realmente influenciou na qualidade da resposta. Na Figura 13, o modelo aponta 3 erros realmente relevantes. Existem apenas 2 palavras erradas, mas de fato, faz sentido a ideia de que existem esses 3 erros: O tempo verbal apropriado pra essa ocasião, a conjugação do verbo auxiliar e a conjugação do verbo principal. Apesar de não ser exatamente o que estávamos esperando, chegamos a respostas corretas e suficientemente valiosas.

### 4.4 FORMATANDO EM JSON

Agora que temos respostas corretas e úteis, nos resta fazer com que as respostas estejam formatadas em JSON. Para isso, nossa primeira tentativa foi simplesmente adicionar a instrução “*Please give me the output in JSON format*” ao final da última versão do *prompt*.

Das muitas execuções do *prompt* da Figura 14, foram observados dois problemas. O primeiro foi a inconsistência do formato, onde o modelo às vezes adiciona determinadas campos como `error_type` e `explanation` e também nomeia os campos de forma diferente e de fato a resposta faz sentido pois não foi especificado qual formato exato de JSON



## ChatGPT Output: v2

Corrected Translation: Has your brother ever been to California?

Errors:

1. Verbs following 'has' or 'have' in Present Perfect tense need to be in Past Participle form. In this case, instead of 'went', it should be 'been'. The correct form is 'Has your brother ever been'.
2. You have correctly translated 'à' as 'to', but the English construction of the present perfect continuous does not usually include 'to' before the location. Therefore, 'to California' is correct.
3. The auxiliary verb choice: "Have" is incorrect. It should be 'Has'. In English, we use 'has' before he, she, it, or a singular noun (which in this case is 'your brother').

Figura 12 – Exemplo 1 do Prompt v2 com temp = 1 e model = gpt-4

## ChatGPT Output: v2

Correction: Has your brother ever been to California?

Errors:

1. The verb tense is incorrect. In English, when asking about past experiences, we use the present perfect tense ("Has...ever been"), not the simple past ("went").
2. The verb "to go" in the past participle form is "been", not "went". So it should be "Has your brother ever been" instead of "Have your brother ever went".
3. The subject-verb agreement is incorrect. When the subject is singular (in this case, "your brother"), the verb should also be singular. So it should be "Has your brother" instead of "Have your brother".

Figura 13 – Exemplo 2 do Prompt v2 com temp = 1 e model = gpt-4

deveria ser utilizado. O segundo problema foi que, às vezes, o modelo não responde apenas com o JSON, mas vem acompanhado de textos antes ou depois. Esses dois problemas dificultam a leitura por um sistema tradicional.

Ambos os problemas que ocorreram na formatação em JSON parecem acontecer por falta de especificidade no prompt. Logo, esperamos que aumentando a especificidade do *prompt*, esses problemas sejam resolvidos. A versão número 4 do *prompt*, demonstrada na figura 15, possui uma especificidade do JSON desejado muito maior que a versão anterior. Em 10 tentativas feitas com a versão 4, todas as respostas respeitaram a formatação e não imprimiram nada além do texto, resolvendo os dois primeiros problemas que tivemos anteriormente.

### ChatGPT Output: v3

Your translation from Portuguese to English contains a few grammatical errors. Here's the correction in JSON format, with each error listed:

```
{
  "Original_Sentence": "O teu irmão já foi à Califórnia?",
  "Your_Translation": "Have your brother ever went to
California?",
  "Correct_Translation": "Has your brother ever been to
California?",
  "Errors": [
    {
      "Error": "Subject-Verb Agreement",
      "Correction": "Change 'Have' to 'Has' to agree with the
singular subject 'your brother'"
    },
    {
      "Error": "Incorrect Verb Form",
      "Correction": "Change 'went' to 'been'. Use 'been' after
'has/have' in perfect tenses"
    }
  ]
}
```

The correct translation should be "Has your brother ever been to California?" which respects the subject-verb agreement and uses the correct verb form in the present perfect tense.

Figura 14 – Resposta do prompt v3 com temperatura = 1 e modelo = gpt-4

## 4.5 MELHORIA DA UTILIDADE DAS RESPOSTAS

Assim, chegamos em respostas corretas, úteis e formatadas. Entretanto, se comparado à qualidade da versão 2, tivemos uma piora na utilidade das respostas. Para melhorar isso, separamos o *prompt* em dois comandos: correção e formatação. Essa técnica é uma variante da técnica de cadeia de pensamento explicada em 3.3.2, que se chama *prompt chaining*, ou encadeamento de *prompts*. O *Prompt Chaining* consiste em enviar outro *prompt* depois de receber a resposta, aproveitando-se do fato que a LLM que estamos usando funciona como um *chat* que considera todo o histórico da conversa para formular uma resposta. Os dois *prompts* desenvolvidos são demonstrados nas Figuras 16 (correção) e 17 (formatação). Com isso, chegamos finalmente a resultados corretos, suficientemente úteis e formatados.

## 4.6 AJUSTES FINAIS

A partir de dezenas de execuções do *prompt* versão 5, foi percebido que, às vezes, o modelo faz alguns comentários antes e depois das correções. Como esses comentá-

### ChatGPT Input

Correct my translation from portuguese to english. List out each and every one of the grammatical errors I did.

Sentence: O teu irmão já foi à Califórnia?

Translation: Have your brother ever went to California?

Format the response in a JSON with the keys:

- `original_sentence`: the original portuguese sentence
- `translation_attempt`: my user translation attempt
- `corrected_translation`: the corrected version of the user translation.
- `errors`: an array of objects with keys "error", "correction", "type" and "comment". "error" should be the part of the sentence where the error occurs. "correction" is the corrected version of the "error". "comment" is the comments you made before regarding the error. "type" is the generic grammatical subject this specific errors falls into.

Do not output anything other than the JSON

Figura 15 – Prompt v4 com temp=1 e model=gpt-4

rios são valiosos, também gostaríamos de trazer esses dados no JSON. Na versão 6 do *prompt*, essas informações estão em `general_comments`. Além disso, adicionamos o campo `sentence_with_correction` no vetor de erros, que retorna a frase inteira com a correção para aumentar as possibilidades visuais do uso em aplicativos. A chave `is_correct` economiza um pós processamento para identificar se a frase original é igual à tentativa de tradução. Por último, adicionamos o parâmetro `type` para classificar os erros em tipos. Uma melhoria futura é dar opções enumeráveis para o algoritmo selecionar o tipo de cada erro. A Figura 18 tem a versão final da Parte 2 (formatação) do *prompt*.

#### ChatGPT Input

Correct my translation from portuguese to english. List out each and every one of the grammatical errors I did and comment on these errors. If there aren't any errors, do not list anything.

Sentence: Você aceita cartão de crédito?

Translation: You take credit card?

#### ChatGPT Output: v5.1

Your translation attempt: "You take credit card?"

Corrected version: "Do you accept credit cards?"

List of errors:

1. Missing auxiliary verb. In English, when framing questions, we often use auxiliary verbs. In this case, the verb "do" is necessary.
2. Absence of the indefinite article: English often requires the use of articles before nouns. Here, it's more natural to say "a credit card" if referring to one card, but generally, the plural for "credit cards" is used.
3. Word choice: "Take" can be understood, but "accept" is a more common and clearer verb to use in this context.

Comments: Your translation captures the essence of the original sentence, but it's important to adhere to English grammatical structures to ensure clarity.

Figura 16 – Prompt v5 (correção) com temp=1 e model=gpt-4

### ChatGPT Input

Format the response in a JSON with the keys:

- original\_sentence: the original portuguese sentence
- translation\_attempt: my user translation attempt
- corrected\_translation: the corrected version of the user translation.
- errors: an array of objects with keys "error", "correction", "type" and "comment". "error" should be the part of the sentence where the error occurs. "correction" is the corrected version of the "error". "comment" is the comments you made before regarding the error. "type" is the generic grammatical subject this specific errors falls into.

Do not output anything other than the JSON

### ChatGPT Output: v5.2

```
{
  "original_sentence": "Você aceita cartão de crédito?",
  "translation_attempt": "You take credit card?",
  "is_correct": "false",
  "corrected_translation": "Do you accept credit cards?",
  "errors": [
    {
      "error": "You take",
      "correction": "Do you accept"
      "type": "Question formation"
      "comment": "In English, when framing questions, we often
use auxiliary verbs. In this case, the auxiliary verb 'do' is
necessary to correctly frame the question."
    },
    {
      "error": "credit card",
      "correction": "credit cards"
      "type": "Noun form"
      "comment": "English often requires the use of articles before
nouns. Here, it's more natural to say 'a credit card' if referring
to one card, but generally, the plural form 'credit cards' is used
in this context, even if you're just referring to one card."
    },
    {
      "error": "take",
      "correction": "accept"
      "type": "Word choice"
      "comment": "Take can be understood, but accept is a more
common and clearer verb to use in this context."
    }
  ]
}
```

Figura 17 – Prompt v5 (formatação) com temp=1 e model=gpt-4

### ChatGPT Input

format the previous response in a JSON with the keys:

- original\_sentence: the original portuguese sentence
- translation\_attempt: my user translation attempt
- is\_correct: a boolean that is true if the translation\_attempt is correct
- corrected\_translation: the corrected version of the user translation. if is\_correct is true, it's the same as the translation\_attempt.
- errors: an array of objects with keys "sentence\_with\_correction", "type" and "comment". "sentence\_with\_correction" is the full sentence with this correction and every other correction made before this. "comment" is the comments you made before regarding the error. "type" is the generic grammatical subject this specific errors falls into.
- general\_comments: an array of strings with each paragraph not in the errors array.

if there aren't any errors, output errors as an empty array.

Do not output anything other than the JSON object I asked.

Figura 18 – Input do Prompt v6 com temp=1 e model=gpt-4

## 5 SISTEMA DESENVOLVIDO

Neste capítulo, temos uma descrição detalhada do sistema desenvolvido, que representa a aplicação prática do algoritmo construído no capítulo anterior. O sistema é estruturado em dois componentes principais: a camada de apresentação (*frontend*) é uma Interface de Linha de Comando (CLI) desenvolvida em Python usando a biblioteca Typer. O outro componente é a camada de acesso aos dados (*backend*), que é composto por uma API desenvolvida também em Python, utilizando o *framework* FastAPI.

A escolha de Python como linguagem de programação se deve à disponibilidade das bibliotecas Typer e FastAPI e facilidade de desenvolvimento da linguagem. A Interface de Linha de Comando (CLI) foi escolhida por ser o frontend mais simples para que seja demonstrado o funcionamento do algoritmo na prática. A biblioteca Typer nos ajuda a criar uma experiência mais visual e interativa na linha de comando, ajudando a criar tabelas, usar argumentos nos comandos e mudar as cores das letras. A utilização do FastAPI no *backend*, declarando os *endpoints* com decoradores, facilita o tratamento de requisições.

Além disso, a arquitetura baseada em API torna o sistema versátil, permitindo a integração com diferentes camadas de apresentação, o que abre possibilidades para futuras expansões e adaptações em diferentes contextos de aprendizado de línguas, como por exemplo a criação de aplicativos para navegadores ou *smartphones*. A seguir, vamos apresentar a arquitetura do sistema, detalhando as funcionalidades e a interação entre o *frontend* e o *backend*.

### 5.1 FUNCIONAMENTO DO SISTEMA

O sistema desenvolvido opera em uma estrutura simples, voltada para a assistência no aprendizado de línguas através da tradução de frases. Inicialmente, o sistema apresenta ao usuário, via interface de linha de comando (CLI), uma frase em inglês para ser traduzida. Após a apresentação da frase, o sistema solicita que o usuário insira sua tentativa de tradução.

Uma vez que a tradução é fornecida pelo usuário, o sistema entra em ação, aplicando sua tecnologia de processamento de linguagem natural. Após a análise, o sistema mostra quais foram os erros cometidos, juntamente com o tipo do erro e comentários detalhados.

### 5.2 ARQUITETURA DO SISTEMA

O usuário começa a interação com o sistema chamando o subcomando *run* da nossa aplicação Python. Esse comando busca no Banco de Dados uma frase a ser traduzida e

pede ao usuário que traduza a frase. Depois do usuário inserir a tradução, o sistema pede ao ChatGPT que faça a correção gramatical da frase de acordo com o *prompt* construído no capítulo 4. Após a correção, o sistema gera uma nova frase a ser traduzida. Na Figura 19 temos um diagrama exemplificando o funcionamento do sistema. Nas subseções seguintes, vamos explorar cada uma das partes do sistema.

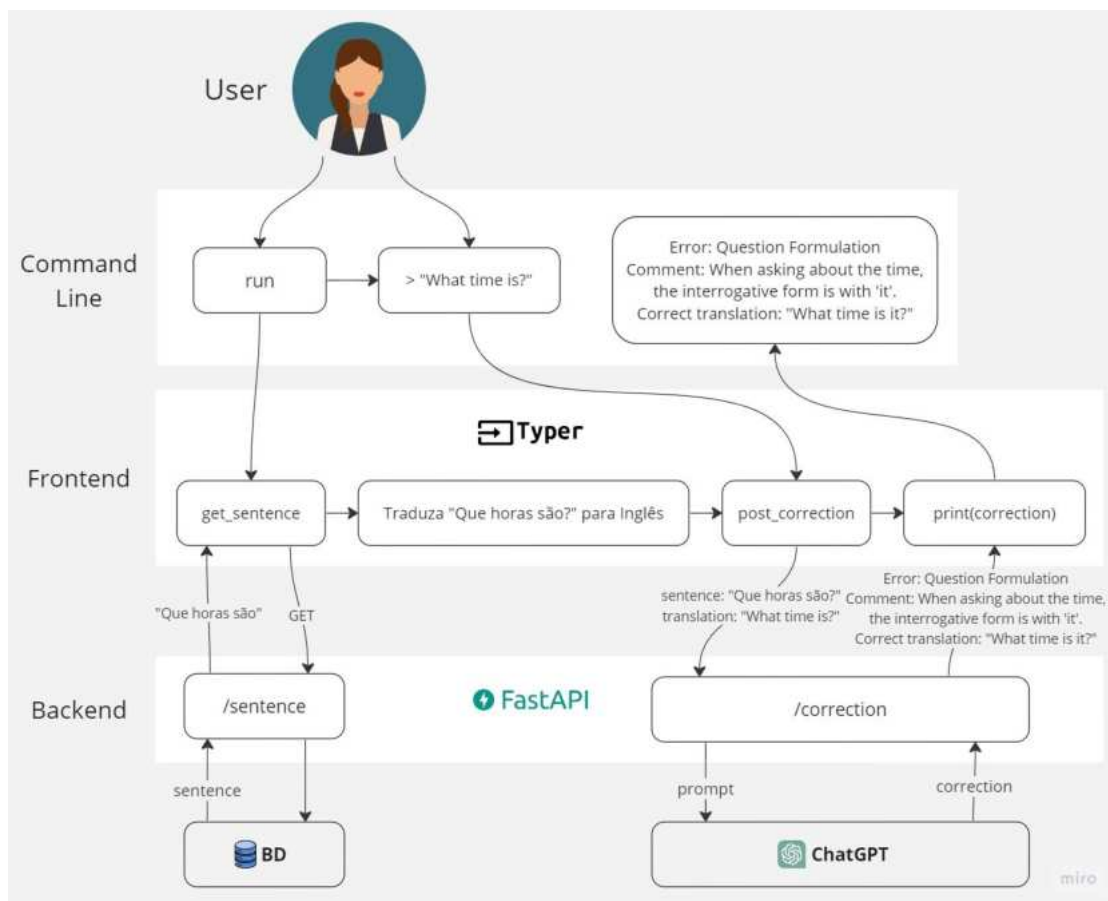


Figura 19 – Diagrama Completo do Sistema com um exemplo de tradução errada.

### 5.2.1 Interação via linha de Comando

A Linha de Comando, frequentemente referida como Command-Line Interface (CLI), é uma interface de usuário baseada em texto que é utilizada para interagir com programas de computador. O usuário fornece comandos como texto para o programa, que então executa ações baseadas no comando fornecido. Diferentemente das interfaces gráficas do usuário (GUIs), que utilizam elementos visuais como ícones e janelas, as CLIs são minimalistas em *design*, oferecendo uma abordagem mais direta. A CLI foi escolhida por sua praticidade e facilidade de desenvolvimento.

No nosso sistema, o usuário começa sua interação com a linha de comando invocando o programa em Python com o subcomando `run`. Após isso, o sistema busca a frase a ser traduzida e retorna, por exemplo, a mensagem "Traduza a frase 'Que horas são?' para o



Inglês”. Caso o usuário insira uma tentativa de tradução errada como “What time is?” o sistema vai retornar uma tabela com os erros gramaticais cometidos e a tradução correta. Caso o usuário insira a tradução correta “What time is it?”, o sistema vai informar que a tradução foi correta.

```
(venv) PS C:\dev\frontend-chatgptteacher> python main.py run
## Translate "Eu não me sinto bem." to English: I don't feeling good.
Error | Comment
-----|-----
Verb form error | When using 'do not' or 'don't' in English, we immediately follow it with the base form of the verb - in this case 'feel'. Adding 'ing' to 'feel' in this sentence is incorrect.
Corrected translation: I don't feel good.
## Translate "Eu volto mais tarde." to English: I come later
Error | Comment
-----|-----
Tense | In English, the future tense 'will' is used for the Portuguese 'volto' to indicate a future action.
Verb | The verb 'volto' includes the implication of returning, so 'come back' is more accurate than just 'come'.
Corrected translation: I will come back later
## Translate "Você já comeu num restaurante?" to English: Have you ever eaten at a restaurant?
Congratulations! The translation is correct!
## Translate "Você poderia falar mais alto por favor?" to English: 
```

Figura 20 – Exemplo de interação com a CLI

## 5.2.2 Frontend

O *frontend* representa toda a camada de interação entre o usuário e o algoritmo de correção. Typer é uma biblioteca escrita em Python e projetada para desenvolver CLIs com base em anotações de tipo, tornando a declaração de comandos e argumentos intuitiva e menos suscetível a erros. No nosso sistema, a biblioteca Typer é utilizada para definir os comandos que o usuário pode executar. Os comandos são definidos a partir de funções decoradas com o decorador `@app.command()`.

No exemplo da Figura 21, a função `run()` define o comando `run`. Quando o comando é passado na linha de comando, a função é executada. O comando `run` primeiramente coleta do banco de dados uma frase para ser traduzida na função `get_sentence`. O usuário é então solicitado a fornecer uma tentativa de tradução para o inglês, que é capturada pela função `Prompt.ask` e exibida com formatação estilizada. A frase a ser traduzida e a tentativa de tradução são enviados para a função `post_correction`, que retorna a correção. Se nenhum erro for encontrado na tentativa de tradução, uma mensagem de sucesso é exibida. Caso contrário, uma tabela detalhada dos erros é impressa, seguida pela tradução corrigida.

Toda a formatação das mensagens, assim como o próprio `print()` não são o padrão do Python, mas são importados da biblioteca `rich`, que ajuda na formatação com tabelas, negrito e coloração, melhorando a legibilidade. Para facilitar os testes, também definimos

```

@app.command()
def run():

    while True:
        print()

        sentence_to_translate = gpteacher.get_sentence()

        message = f"👉 Translate \"[b]{sentence_to_translate}[/b]\" to [red]English[/red]"
        translation_attempt = Prompt.ask(message)

        payload = CorrectionPayload(
            sentence_to_translate=sentence_to_translate,
            translation_attempt=translation_attempt
        )

        correction = gpteacher.post_correction(payload)

        if len(correction['errors']) == 0:
            print("\n Congratulations! The translation is [b][green]correct[/green][/b]!")
        else:
            print_full_table(correction['errors'])
            print()
            print(f" Corrected translation: [b]{correction['corrected_translation']}[/b]")

```

Figura 21 – Frontend: código do comando run

em outra parte do programa o comando `run_custom`, que permite que o usuário escolha manualmente a frase a ser traduzida.

### 5.2.3 Backend

O *backend* é em formato de API. API, ou Interface de Programação de Aplicações, é um conjunto de regras e definições que permite comunicação entre programas. No nosso caso, o *backend* não se comunica com nenhum usuário, mas com outro programa, que é o *frontend*. O *backend* é um intermediário entre a camada de interação com o usuário e a API do ChatGPT ou o banco de frases.

As solicitações do *frontend* para o *backend* são feitas por requisições HTTP para um *endpoint*, que geralmente é um verbo separado da URL por uma barra. Por exemplo, a página de pesquisa do Google é `google.com/search`. Nosso *backend* possui dois *endpoints* principais: `sentence`, onde é buscada uma frase em um banco de dados e `correction`, onde solicitamos ao ChatGPT uma correção usando o *prompt* do capítulo 4.

O *endpoint sentence* retorna do banco de dados uma frase a ser traduzida pelo usuário. Na inicialização da API, o *backend* carrega em memória o banco de frases da Figura 22 com uma lista de quase 700 frases comuns usadas para aprender inglês, embaralhadas de forma a sempre retornar frases diferentes. Então, quando o *endpoint sentence* é chamado, o código do *backend* busca e retorna a próxima frase da lista de frases já carregada em memória.

	phrase_english	phrase_portuguese	topic_portuguese
76	I hope you and your wife have a nice trip.	Eu espero que você e tua esposa tenham uma boa viagem.	Expressões comuns
77	I know.	Eu sei.	Expressões comuns
78	I like her.	Eu gosto dela.	Expressões comuns
79	I'll call you when I leave.	Eu te ligo quando sair.	Expressões comuns
80	I'll come back later.	Eu volto mais tarde.	Expressões comuns
81	11 dollars.	11 dólares	Numeros e dinheiro
82	52 cents.	52 centavos	Numeros e dinheiro
83	A few.	Poucos	Numeros e dinheiro
84	A little.	Pouco/um pouco	Numeros e dinheiro
85	About 300 kilometers.	Mais ou menos 300 quilometros	Viagens e direções
86	A one way ticket.	Passagem só de ida	Viagens e direções
87	A round trip ticket.	Passagem ida e volta	Viagens e direções
88	Are you going to help her?	Você vai ajudá-la?	Viagens e direções
89	At what time?	A que horas?	Viagens e direções

Figura 22 – Exemplos de frases do banco de frases

O endpoint `correction` se conecta à API do ChatGPT utilizando a biblioteca oficial da OpenAI para Python, que simplifica a interação com a API. Basta configurar a chave de API e chamar as funções da biblioteca para utilizá-la. Todas as APIs da OpenAI são pagas, inclusive a que usamos na construção do algoritmo descrito neste trabalho. O custo do modelo `gpt-4-turbo-2024-04-09` em Abril de 2024 era de 10 dólares por cada 1 milhão de tokens de entrada e 30 dólares por cada 1 milhão de tokens de saída. Como exemplo, 1000 utilizações do *prompt* v5 demonstrado nas Figuras 16 e 17 custariam cerca de 3,33 dólares.

```
def get_response_from_chat(self, prompt: list) -> str:
    response = openai.ChatCompletion.create(
        model=self.model,
        messages=prompt
    )

    content = response['choices'][0]['message']['content']
    return content.strip()

def correct_translation(self, sentence_to_translate, translation_attempt):
    correction_prompt = generate_correction_prompt(sentence_to_translate, translation_attempt)

    correction_content = self.get_response_from_chat(correction_prompt)

    formatting_prompt = generate_formatting_prompt(correction_prompt, correction_content)

    formatting_content = self.get_response_from_chat(formatting_prompt)

    return formatting_content
```

Figura 23 – endpoint correction: correção de frases usando a API do ChatGPT

A Figura 23 mostra a parte principal do *endpoint correction*, que funciona de forma idêntica ao *prompt* do final do capítulo 4. Primeiro o ChatGPT responde ao *prompt* de

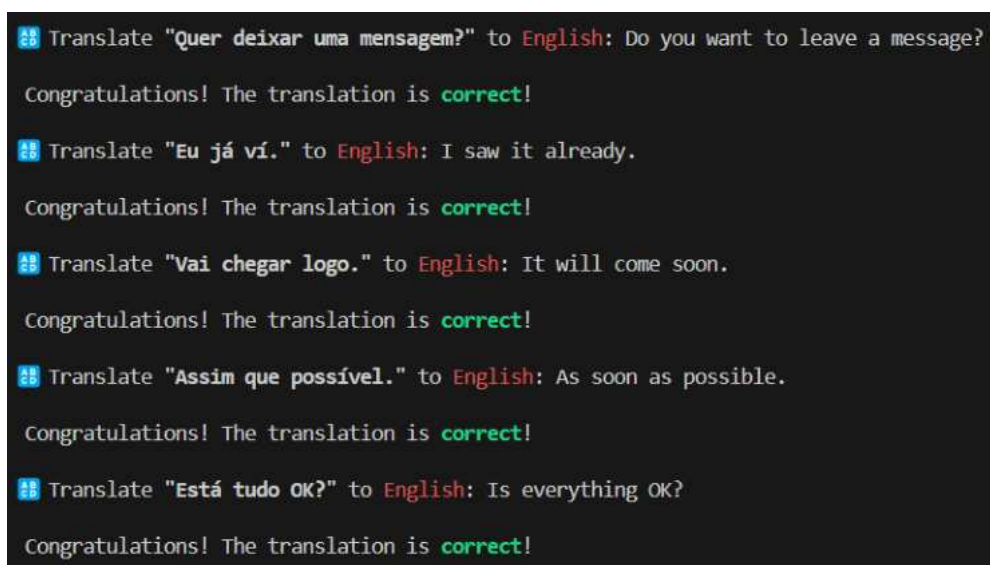
correção e depois responde ao *prompt* de formatação. A resposta do ChatGPT ao *prompt* de formatação é retornada como resposta final do *endpoint*. Para conseguir a resposta do ChatGPT, é executada a função `create` dentro do objeto `ChatCompletion` da biblioteca da OpenAI. Essa função retorna um objeto complexo com vários detalhes, mas a resposta do *prompt* de fato está em `response['choices'][0]['message']['content']`. Por isso, esse é o valor retornado da função `get_response_from_chat`.

#### 5.2.4 Criação do banco de frases

Para compor o banco de frases em português e inglês para serem traduzidas, foi feita uma pesquisa na internet por listas de frases comuns da língua inglesa com tradução em português. Foi encontrado um documento em PDF de licença gratuita (AUTORES, 2021) com 1000 frases categorizadas e escritas em inglês e português. As frases são de fato comuns e utilizáveis em diversos contextos diferentes e o documento foi escolhido para ser o primeiro banco de frases do sistema. O documento foi convertido de PDF para CSV para possibilitar o fácil acesso às frases e algumas frases foram excluídas e as que estavam erradas foram corrigidas. A versão atual desse banco de frases está no repositório do Github apresentado no final do capítulo 1.

### 5.3 O SISTEMA EM FUNCIONAMENTO E EXEMPLOS DE USO

Nesta seção, ilustramos através de casos de uso específicos como o sistema opera na prática. Com a ajuda de capturas de tela da execução do programa, vamos explorar as nuances do mecanismo de correção, destacando tanto os pontos fortes quanto uma visão crítica a respeito dos pontos de melhora do sistema.



```
👤 Translate "Quer deixar uma mensagem?" to English: Do you want to leave a message?  
👤 Congratulations! The translation is correct!  
👤 Translate "Eu já vi." to English: I saw it already.  
👤 Congratulations! The translation is correct!  
👤 Translate "Vai chegar logo." to English: It will come soon.  
👤 Congratulations! The translation is correct!  
👤 Translate "Assim que possível." to English: As soon as possible.  
👤 Congratulations! The translation is correct!  
👤 Translate "Está tudo OK?" to English: Is everything OK?  
👤 Congratulations! The translation is correct!
```

Figura 24 – Sistema funcionando com tentativas de tradução corretas e feedbacks corretos

A Figura 24 ilustra testes onde foram fornecidas apenas traduções corretas. Como esperado, o programa se saiu muito bem e marcou todas as tentativas de correção como corretas. Na Figura 25, foi traduzido “Você poderia..” como “Can you..” em vez de “Could you..”, que é a tradução correta. “Could” em português equivale ao tempo verbal “futuro do pretérito”, enquanto “can” equivale ao tempo verbal “presente”. O algoritmo foi capaz de identificar que houve um erro e foi capaz de identificar uma nuance nessa importante diferença de tradução que é a “falta de polidez”, análogo ao português “você pode” em vez de “você poderia”.

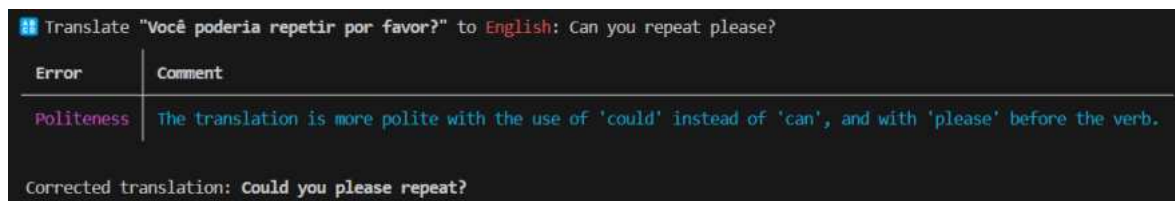


Figura 25 – Sistema funcionando com tentativa de tradução imprecisa e feedback correto

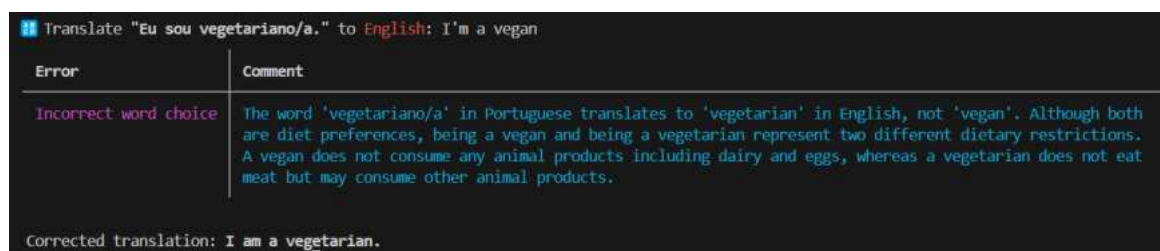


Figura 26 – Sistema funcionando com tentativa de tradução errada e feedback correto

No exemplo 26, o sistema foi capaz de identificar que “vegetariano” e “vegano” são duas coisas diferentes e fornecer uma explicação detalhada das diferenças entre os dois tipos de dieta, utilizando-se da vasta base de conhecimento onde foi treinada a LLM.

### 5.3.1 Identificação de Falsos Cognatos

Falsos cognatos são palavras que, em idiomas diferentes, se parecem, mas possuem significados diferentes. Por exemplo, a tradução da palavra “college” em inglês não é “colégio”, mas sim “faculdade”. Existem vários exemplos desse tipo, e alguns não possuem significados completamente diferentes, mas com nuances importantes de serem ressaltadas. Em português, muitas vezes usamos “preferido” como sinônimo de “favorito”. Entretanto, em inglês a palavra “preferred” não é usada como sinônimo de “favorito”. O caso de uso da figura 27 ilustra exatamente esse tipo de correção, onde o algoritmo foi capaz de identificar e pontuar muito bem dois falsos cognatos.





Figura 27 – Sistema funcionando com tentativa de tradução errada e feedback correto para dois falsos cognatos: 'idioms' e 'preferred'

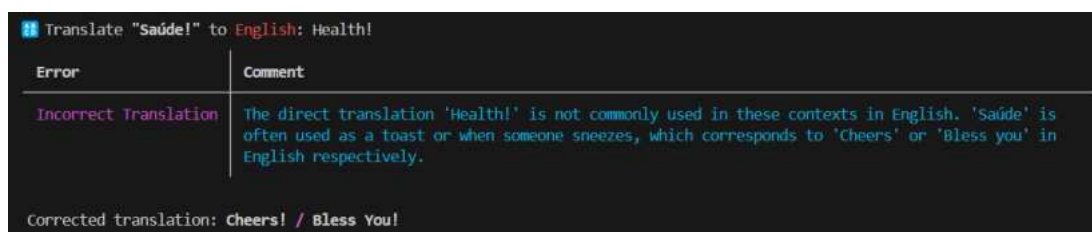


Figura 28 – Sistema funcionando com tentativa de tradução correta de forma literal mas errada em contexto e feedback do sistema correto

### 5.3.2 Nuances em traduções literais

Uma palavra em uma língua pode ter várias traduções possíveis em outra, dependendo do contexto. A palavra “manga” em português pode ser a fruta “mango” ou a parte da camisa “sleeve”. A tradução da palavra “manga” para inglês vai depender do contexto. No exemplo 28, a palavra “saúde” pode sim ser traduzida como “health”, mas o ponto de exclamação mostra que essa tradução estaria errada e o contexto pode ter a ver com um interjeição de brinde ou quando outra pessoa espirra. O algoritmo foi capaz de perceber essa nuance e dizer quais eram as outras duas traduções possíveis e diferentes entre si.

### 5.3.3 Falso-positivo

A figura 29 ilustra um caso onde o algoritmo errou ao corrigir algo que já estava correto. A contração “He’s not” está tão correta quanto as formas “He isn’t” ou “He is not”. Esse foi um dos únicos exemplos que pudemos encontrar de falso-positivo. Apesar disso, o algoritmo conseguiu identificar corretamente o outro erro na tentativa de tradução.

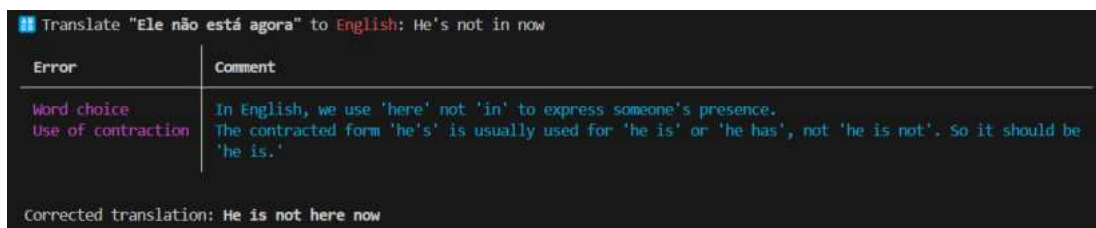


Figura 29 – Sistema funcionando com tentativa de tradução errada e feedback do sistema errado

### 5.3.4 Considerações gerais

Os casos de uso apresentados nesta seção demonstram a habilidade do algoritmo em navegar por uma variedade de desafios linguísticos. O algoritmo foi capaz tanto de identificar nuances em falsos cognatos até discernir o contexto sutil de palavras com muitos significados. Apesar de alguns obstáculos, como a interpretação imprecisa de nuances verbais em situações específicas, os resultados foram predominantemente positivos.

## 5.4 MELHORIAS FUTURAS

Um dos principais aprimoramentos previstos é a criação de um aplicativo *web*, que proporcionará uma interface mais interativa e acessível para os usuários em diversos dispositivos. Além disso, está prevista a implementação de uma interface dedicada para monitorar o progresso do aprendizado de línguas. Esta interface fornecerá aos usuários uma visão abrangente de seu desenvolvimento ao longo do tempo, apresentando estatísticas detalhadas sobre as áreas que necessitam de maior atenção.

Outra possibilidade de melhoria futura, visando tornar o sistema mais robusto é a implementação de testes automatizados para verificar a qualidade das respostas geradas pelo sistema. Esses testes, executados por sistemas independentes e comparados com as saídas do algoritmo, irão avaliar a precisão, a relevância e a utilidade das correções gramaticais fornecidas, garantindo que os usuários recebam *feedback* confiável e de alta qualidade.

## 6 CONCLUSÃO

Este trabalho se dedicou ao desenvolvimento e avaliação do uso das LLMs para apoiar o aprendizado de línguas. O nosso objetivo central foi resolver um problema difícil e grande dos aplicativos de aprendizado de línguas: a falta do *feedback* personalizado. A partir do desenvolvimento de um *prompt* que instrui o ChatGPT a nos retornar o resultado desejado e um aplicativo simples usando a API do ChatGPT, fomos capazes de demonstrar que é possível enriquecer bastante o *feedback* dado por aplicativos com o Duolingo. Os resultados demonstraram que o algoritmo desenvolvido foi capaz de detectar erros gramaticais em exercícios de tradução e fazer comentários corretos, úteis e apropriadamente formatados. Além disso, foi observado que o uso dos LLMs oferece sugestões muito mais contextualizadas e completas do que algoritmos que utilizam regras simples. Este achado reforça a potencialidade dos LLMs como tecnologia de suporte no ensino de línguas.

No entanto, o estudo enfrentou limitações importantes. Primeiramente, o uso do GPT-4, embora tenha se mostrado eficaz, implica em considerações financeiras devido ao seu modelo de cobrança, o que pode limitar sua aplicabilidade em ambientes educacionais com restrições orçamentárias. Adicionalmente, a pesquisa foi restrita à análise de traduções entre português e inglês, deixando de investigar a sua eficácia para outros idiomas. Por fim, não está claro qual é a precisão exata dessas correções, deixando em dúvida a confiabilidade do algoritmo. Isso se deve a dois motivos: O primeiro é o fato de que foi usado no estudo um conjunto relativamente pequeno de frases, que pode não capturar completamente a diversidade e complexidade dos erros gramaticais encontrados por aprendizes em diferentes níveis de proficiência. O segundo é que, mesmo nas frases selecionadas, não foi feita uma quantidade significativa de testes para afirmar que, nesse conjunto de frases, temos uma garantia de que o algoritmo funciona como esperado.

Diante dessas limitações, sugere-se para pesquisas futuras a inclusão de mais idiomas e LLMs para explorar a flexibilidade e redução de custos e a realização de testes com conjuntos de dados mais extensos e diversificados, para validar a eficácia do algoritmo em um espectro mais amplo de erros gramaticais e tipos de texto. Outra melhoria possível é testar e avaliar a nova funcionalidade na API da OpenAI de saídas estruturadas (OPENAI, 2024), que aparenta resolver o critério de sucesso de formatação em JSON de forma nativa, eliminando a necessidade de um segundo *prompt*.

Conseguimos observar que é possível usar uma LLM para gerar correções e comentários personalizados em exercícios de tradução para aprendizado de línguas. A implementação de algoritmos como este tem o potencial de melhorar consideravelmente o *feedback* dado pelo computador ao aluno e de otimizar o aprendizado das milhões de pessoas que aprendem novos idiomas por aplicativo.



## REFERÊNCIAS

- ADESHOLA, I.; ADEPOJU, A. P. The opportunities and challenges of chatgpt in education. **Interactive Learning Environments**, Routledge, v. 0, n. 0, p. 1–14, 2023. Disponível em: <<https://doi.org/10.1080/10494820.2023.2253858>>.
- AHMAD, A. et al. Towards human-bot collaborative software architecting with chatgpt. In: **Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering**. New York, NY, USA: Association for Computing Machinery, 2023. (EASE '23), p. 279–285. ISBN 9798400700446. Disponível em: <<https://doi.org/10.1145/3593434.3593468>>.
- AUSAT, A. et al. Can chat gpt replace the role of the teacher in the classroom: A fundamental analysis. **Journal on Education**, v. 5, n. 4, p. 16100–16106, 2023. Disponível em: <<https://doi.org/10.31004/joe.v5i4.2745>>.
- AUTORES, V. Livro Digital, **Inglês - 1000 Frases Mais Usadas**. [S.l.]: 8 Bits, 2021. <<https://www.baixelivros.com.br/ciencias-humanas-e-sociais/idiomas/frases-mais-usadas-ingles>>. Acessado em 20 de março de 2024.
- BARROT, J. S. Using chatgpt for second language writing: Pitfalls and potentials. **Assessing Writing**, v. 57, p. 100745, 2023. ISSN 1075-2935. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1075293523000533>>.
- BISWAS, S. S. Potential use of chat gpt in global warming. **Annals of Biomedical Engineering**, v. 51, n. 6, p. 1126–1127, 2023. ISSN 1573-9686. Disponível em: <<https://doi.org/10.1007/s10439-023-03171-8>>.
- BROWN, T. et al. Language models are few-shot learners. **Advances in neural information processing systems**, v. 33, p. 1877–1901, 2020.
- COTTON, P. A. C. D. R. E.; SHIPWAY, J. R. Chatting and cheating: Ensuring academic integrity in the era of chatgpt. **Innovations in Education and Teaching International**, Routledge, v. 61, n. 2, p. 228–239, 2024. Disponível em: <<https://doi.org/10.1080/14703297.2023.2190148>>.
- FINARDI, K. R.; LEÃO, R. G.; AMORIM, G. B. Mobile assisted language learning: Affordances and limitations of duolingo. **Education and Linguistics Research**, v. 2, n. 2, p. 48–65, 2016.
- FULFORD, I.; NG, A. **Building Systems with the ChatGPT API**. 2023. Accessed: 2024-03-01. Disponível em: <<https://www.deeplearning.ai/short-courses/building-systems-with-chatgpt/>>.
- FULFORD, I.; NG, A. **ChatGPT Prompt Engineering for Developers**. 2024. DeepLearning.AI. Course available at DeepLearning.AI. Disponível em: <<https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/>>.
- GAFNI, R.; ACHITUV, D. B.; RAHMANI, G. Learning foreign languages using mobile applications. **Journal of Information Technology Education. Research**, Informing Science Institute, v. 16, p. 301, 2017.

GRASSINI, S. Shaping the future of education: Exploring the potential and consequences of ai and chatgpt in educational settings. **Education Sciences**, v. 13, n. 7, 2023. ISSN 2227-7102. Disponível em: <<https://www.mdpi.com/2227-7102/13/7/692>>.

HABIB, S. et al. Mobile assisted language learning: Evaluation of accessibility, adoption, and perceived outcome among students of higher education. **Electronics**, v. 11, n. 7, 2022. ISSN 2079-9292. Disponível em: <<https://www.mdpi.com/2079-9292/11/7/1113>>.

HEIL, C. et al. A review of mobile language learning applications: Trends, challenges, and opportunities. **The EuroCALL Review**, v. 24, 10 2016.

HelloTalk. **HelloTalk: Language Exchange Community**. 2024. <<https://www.hellotalk.com/>>. Available on iOS and Android platforms. A platform for language exchange, allowing users to connect with native speakers for language practice and cultural exchange.

HENDRYCKS, D. et al. **Measuring Massive Multitask Language Understanding**. 2021.

IBRAHIM, H. et al. Rethinking homework in the age of artificial intelligence. **IEEE Intelligent Systems**, v. 38, n. 2, p. 24–27, 2023.

INC., D. **Duolingo Reports 65% DAU Growth, 45% Revenue Growth and Record Profitability in Fourth Quarter 2023**. Pittsburgh: GLOBE NEWSWIRE, 2024. Press Release. Disponível em: <<https://investors.duolingo.com/node/9446/pdf>>.

LINGARD, L. Writing with chatgpt: An illustration of its capacity, limitations & implications for academic writers. **Perspectives on Medical Education**, Western University, CA, v. 12, n. 1, p. 261–270, 2023. ISSN 2212-277X. The author has no competing interests to declare.

LIU, Y. et al. **Prompt Injection attack against LLM-integrated Applications**. 2024.

LOWE, R.; LEIKE, J. et al. **Aligning language models to follow instructions**. 2022. <<https://openai.com/research/instruction-following>>. Accessed: 2023-11-22.

MARR, B. How generative ai will change the jobs of teachers. **Forbes**, 2024. Accessed: 2024-06-23. Disponível em: <<https://www.forbes.com/sites/bernardmarr/2024/02/09/how-generative-ai-will-change-the-jobs-of-teachers/>>.

METRUK, R. Smartphone english language learning challenges: A systematic literature review. **Sage Open**, v. 12, n. 1, p. 21582440221079627, 2022. Disponível em: <<https://doi.org/10.1177/21582440221079627>>.

OPENAI et al. **GPT-4 Technical Report**. 2023.

OPENAI. **Introducing Structured Outputs in the API**. 2024. Accessed on 2024-08-09. Disponível em: <<https://openai.com/index/introducing-structured-outputs-in-the-api/>>.

OUYANG, L. et al. **Training language models to follow instructions with human feedback**. 2022.

SHORTT, M. et al. Gamification in mobile-assisted language learning: a systematic review of duolingo literature from public release of 2012 to early 2020. **Computer Assisted Language Learning**, Routledge, v. 36, n. 3, p. 517–554, 2023. Disponível em: <<https://doi.org/10.1080/09588221.2021.1933540>>.

TERWIESCH, C. Would chat gpt3 get a wharton mba. **A prediction based on its performance in the operations management course**, 2023.

TRUST, T.; WHALEN, J.; MOUZA, C. Editorial: Chatgpt: Challenges, opportunities, and implications for teacher education. **Contemporary Issues in Technology and Teacher Education**, Society for Information Technology Teacher Education, Waynesville, NC USA, v. 23, n. 1, p. 1–23, March 2023. ISSN 1528-5804. Disponível em: <<https://www.learntechlib.org/p/222408>>.

VASWANI, A. et al. **Attention Is All You Need**. 2023.

WEBB, T.; HOLYOAK, K. J.; LU, H. Emergent analogical reasoning in large language models. **Nature Human Behaviour**, v. 7, n. 9, p. 1526–1541, 2023. ISSN 2397-3374. Disponível em: <<https://doi.org/10.1038/s41562-023-01659-w>>.

WEI, J. et al. **Chain-of-Thought Prompting Elicits Reasoning in Large Language Models**. 2023.

WINTER, J. C. F. de. Can chatgpt pass high school exams on english language comprehension? **International Journal of Artificial Intelligence in Education**, 2023. ISSN 1560-4306. Disponível em: <<https://doi.org/10.1007/s40593-023-00372-z>>.

ZHENG, H.; ZHAN, H. Chatgpt in scientific writing: a cautionary tale. **The American Journal of Medicine**, Elsevier, v. 136, n. 8, p. 725–726, 2023.

## APÊNDICE A – APRENDIZADO DE MÁQUINA, LLMS E O CHATGPT

Esse capítulo é dedicado a compreender como funciona o aprendizado de máquina, desde aplicações mais básicas até as LLMS refinadas para instrução, como ChatGPT. Começamos entendendo os tipos mais básicos de aprendizado de máquina e logo chegamos ao Processamento de Linguagem Natural (NLP), um campo do aprendizado de máquina que se dedica a entender linguagem humana. No estado da arte dessa área estão os Modelos de Linguagem de Grande Escala (LLMs) como o GPT-3, que trouxeram avanços significativos, mostrando uma capacidade incrível de compreender e gerar texto com performance similar a humanos.

O GPT-3 se destaca não só pela sua habilidade de entender e produzir texto de forma natural, mas também por popularizar a Engenharia de *Prompt*. Esse conceito sugere uma maneira mais simples e direta de guiar o comportamento de sistemas inteligentes, sem precisar reprogramá-los do zero (BROWN et al., 2020). Agora podemos ensinar e ajustar sistemas de IA de forma mais rápida e com menos esforço, tornando a fácil a criação de aplicações de IA, antes restritas a formas caras e de desenvolvimento demorado.

### A.1 APRENDIZADO DE MÁQUINA

O aprendizado de máquina, uma subárea da inteligência artificial, refere-se ao desenvolvimento de algoritmos que permitem que computadores aprendam padrões ou realizem tarefas sem serem explicitamente programados para isso. Em sua essência, o aprendizado de máquina baseia-se na ideia de que sistemas podem aprender a partir de dados, identificar padrões e tomar decisões sem intervenção humana. Para isso, são alimentados com grandes volumes de dados, e por meio de técnicas estatísticas e algorítmicas, são treinados para prever resultados ou classificar informações de maneira autônoma.

#### A.1.1 Tipos de aprendizado de máquina

Imagine que você quer ensinar uma máquina a reconhecer a diferença entre imagens de gatos, cães e galinhas. Em vez de codificar regras específicas, como “gatos geralmente têm orelhas pontudas”, “cães podem ter focinhos mais longos” e “galinhas tem asas”, você fornece ao algoritmo milhares de imagens rotuladas de gatos, cães e galinhas. O algoritmo, então, analisa os exemplos e vai se ajustando iterativamente até que possa fazer a distinção com precisão. Esse processo é o que chamamos de treinamento supervisionado, onde o modelo é treinado com dados previamente rotulados.

Depois de treinado, o modelo deve ser capaz de aplicar o que aprendeu a dados nunca antes vistos. Se o modelo foi bem treinado e é robusto, ele poderá identificar, por exem-

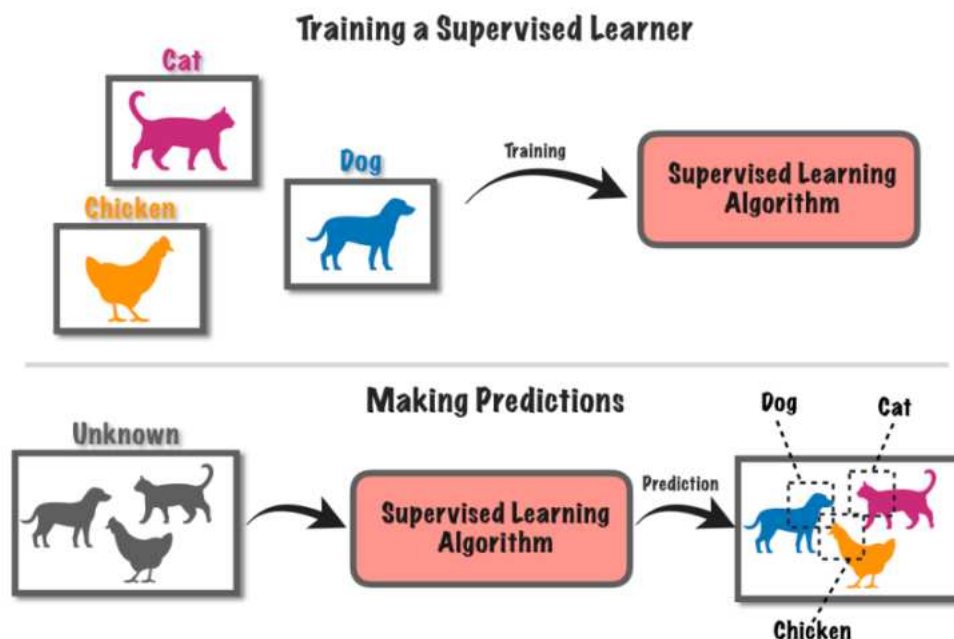


Figura 30 – Ilustração de um algoritmo de aprendizado de máquina supervisionado para identificar animais em uma imagem

plô, gatos e cães em imagens que nunca viu antes. Contudo, é essencial garantir que o treinamento seja bem feito, pois um modelo mal treinado pode resultar em previsões imprecisas ou tendenciosas. Por essa razão, após o treinamento, são feitas as etapas de teste e validação.

Durante a etapa de teste, a performance do modelo é avaliada com um conjunto de dados que não foi utilizado durante seu treinamento, fornecendo uma estimativa realista de seu desempenho em cenários do mundo real. A validação, muitas vezes realizada várias vezes durante o treinamento, ajusta os parâmetros do modelo para evitar o sobreajuste. O sobreajuste ocorre quando um modelo é tão bem ajustado aos dados de treinamento que se torna incapaz de generalizar para novos dados. Além dessas etapas, podem ser feitos muitos outros ajustes na base de dados de treinamento, no algoritmo e nos métodos de validação com o objetivo de construir o melhor algoritmo possível em dados do mundo real.

Além do Aprendizado Supervisionado, temos outros tipos de aprendizado de máquina. Em particular, destacamos o Aprendizado Não-supervisionado, o Aprendizado por Reforço e o Aprendizado por Transferência.

No Aprendizado Não-supervisionado, os algoritmos são treinados em conjuntos de dados sem rótulos, visando identificar padrões ou estruturas subjacentes nos dados, como a formação de *clusters* ou a redução de dimensionalidade.

No Aprendizado por Reforço, um agente toma ações em um ambiente para maximizar uma recompensa acumulada. O processo é fundamentalmente orientado por tentativa e

erro, com o agente aprendendo com o tempo quais ações levam às melhores recompensas. Um exemplo de caso de uso do aprendizado por reforço é em jogos, onde o agente aprende jogando. Esse é o caso do AlphaGO da Deepmind, que joga Go, e do OpenAI Five, uma IA que joga Dota 2.

O Aprendizado por Transferência utiliza a ideia de que o conhecimento obtido ao resolver um problema pode ser utilizado para resolver um problema diferente, mas semelhante. É a reutilização de modelos de aprendizado de máquina que foram treinados em grandes conjuntos de dados. Por exemplo, uma rede neural treinada em milhões de imagens categorizadas pode ser reutilizada para tarefas mais especializadas com conjuntos de dados menores, como reconhecimento de radiografias ou detecção de defeitos em produtos.

### A.1.2 Desenvolvimento de aplicações que usam algoritmos de Aprendizado Supervisionado

O processo de desenvolvimento e implantação de um sistema de aprendizado supervisionado é dividido em três partes: geração de dados rotulados, treinamento e implantação.

A fase de geração de dados rotulados envolve coletar exemplos do mundo real e associá-los a rótulos ou resultados específicos. Por exemplo, coletar imagens de animais e rotular com o nome do animal. Dependendo da complexidade da tarefa e da quantidade de dados necessários, esta etapa pode levar de dias até anos. Dentro do exemplo anterior de coleta de imagens de animais, conseguir milhares de imagens de uma espécie específica pode ser um processo demorado e trabalhoso, exigindo a atenção de especialistas para garantir a precisão dos rótulos. Em comparação, para tarefas mais simples, como classificar avaliações de restaurantes como positivas ou negativas, o processo pode ser bem mais rápido.

Restaurant reviews sentiment classification

Input x	Output y
The pastrami sandwich was great!	Positive
Service was slow and the food was so-so.	Negative
The earl grey tea was fantastic.	Positive

Figura 31 – Dados rotulados para um modelo de análise de sentimento

A segunda etapa é o treinamento do modelo. É nessa etapa onde o algoritmo é implementado e é ensinado a fazer previsões a partir dos dados coletados na primeira

etapa. Esta etapa pode levar de horas até semanas e exigir uma grande quantidade de processamento computacional.

Finalmente, a terceira etapa é a implantação, onde o modelo treinado é integrado a um sistema ou aplicação para ser usado em cenários do mundo real. Esta fase pode ser relativamente rápida, durando de dias a semanas, mas pode ser estendida se ajustes e otimizações forem necessários para garantir que o modelo opere de forma eficiente e eficaz no ambiente de produção.

A finalização de um projeto de aprendizado de máquina supervisionado varia muito de duração. Para tarefas simples, como a classificação de avaliações de produtos, um especialista competente pode desenvolver e treinar um modelo em questão de dias. Por outro lado, em projetos mais complexos, como o desenvolvimento de sistemas avançados de reconhecimento de imagem, frequentemente são exigidos meses de trabalho meticuloso e a colaboração de uma equipe multidisciplinar envolvendo, entre outros profissionais, cientistas de dados para desenvolver o algoritmo, especialistas para rotular os dados e engenheiros de *software* para integrar o modelo de IA a outros sistemas.

## A.2 PROCESSAMENTO DE LINGUAGEM NATURAL (NLP)

O Processamento de Linguagem Natural (NLP) é um subcampo da inteligência artificial (IA) que se concentra na interação entre computadores e a linguagem humana. Seu principal objetivo é permitir que as máquinas entendam, interpretem e gerem a linguagem humana de uma maneira valiosa.

O NLP envolve vários desafios e tarefas, que vão desde os básicos, como *tokenização* (dividir o texto em palavras ou frases individuais) e etiquetagem de partes da fala (identificando palavras como substantivos, verbos, adjetivos, etc.), até tarefas mais complexas como análise de sentimento, tradução automática e resposta a perguntas. Nesta seção, exploramos com mais profundidade alguns desses desafios.

### A.2.1 Tokenização

Um dos desafios mais básicos da NLP é a *tokenização*. *Tokenização* é o processo de converter um texto em unidades menores, chamadas *tokens*. Em Processamento de Linguagem Natural, um *token* frequentemente se refere a uma palavra, mas dependendo do contexto, pode também se referir a caracteres, partes de palavras ou até mesmo frases. A *tokenização* é uma etapa fundamental em muitas tarefas de NLP, pois permite que os algoritmos processem e analisem o texto de maneira mais estruturada e manejável.

Considere a frase: “O gato está no telhado.” Se *tokenizarmos* essa frase em palavras, obteremos a lista: ["O", "gato", "está", "no", "telhado."]. Note que a *tokenização* pode também envolver decisões adicionais, como tratar pontuações. No exemplo

anterior, o ponto final foi incluído com a palavra “telhado”, mas em algumas aplicações pode ser útil tratá-lo como um *token* separado.

### A.2.2 Análise sintática e semântica

A análise sintática, ou *parsing*, envolve a decomposição de frases nas partes que a constituem, identificando a função gramatical de cada palavra e a estrutura hierárquica da frase como sujeito, predicado, objetos e complementos. Essa análise é crucial para entender a organização lógica do texto, permitindo que sistemas determinem relações entre palavras e frases. Já a análise semântica busca entender o significado por trás das palavras e estruturas sintáticas. Ela se concentra na interpretação dos significados das palavras no contexto em que são usadas, resolvendo ambiguidades e inferindo relações semânticas, como a identificação de sinônimos, antônimos e metáforas.

### A.2.3 Análise de sentimento

A análise de sentimento é uma técnica de NLP que permite identificar, extrair e quantificar opiniões, emoções e atitudes expressas em textos. Essa abordagem computacional avalia se o sentimento subjacente a um determinado trecho de texto é positivo, negativo ou neutro, e em casos mais avançados, pode até determinar a intensidade da emoção. Essa capacidade tem aplicações práticas significativas, desde monitorar a percepção da marca nas redes sociais, analisar sentimentos do mercado para previsões financeiras, até avaliar opiniões e *feedbacks* de consumidores sobre produtos e serviços.

## A.3 A EVOLUÇÃO DO PROCESSAMENTO DE LINGUAGEM NATURAL (NLP)

Esta seção explora a transição histórica da NLP, desde seus primórdios com sistemas baseados em regras até a adoção de modelos estatísticos, seguida pelo desenvolvimento de redes neurais e a inovação trazida pelos mecanismos de atenção e transformadores. Aqui destacamos como cada etapa contribuiu para superar as limitações anteriores, melhorando a capacidade das máquinas de entender e gerar linguagem de forma mais natural e eficaz.

### A.3.1 Sistemas Baseados em Regras

Nos primórdios da NLP, a abordagem principal eram os sistemas baseados em regras. Estes sistemas dependiam fortemente de linguistas e engenheiros que criavam manualmente regras para analisar e interpretar textos. A grande vantagem desse sistema era sua clareza; as decisões feitas pelo sistema eram explícitas e explicáveis com base nas regras definidas. No entanto, seus pontos fracos eram igualmente evidentes. Os sistemas eram inerentemente limitados pelo número e qualidade das regras definidas. Eles não tinham escalabilidade e precisavam de muito esforço para compreender as nuances dos idiomas.



### A.3.2 Modelos Estatísticos

A transição do século 20 para o século 21 viu uma mudança de sistemas baseados em regras para modelos estatísticos em NLP. Esta mudança foi motivada pelo aumento do conteúdo digital disponível com o surgimento da internet. Algoritmos como Modelos de Markov Ocultos e Redes Bayesianas foram empregados para tarefas como análise sintática e reconhecimento de entidades. Embora esses modelos fossem um passo à frente em adaptabilidade, aprendendo padrões diretamente dos dados, não estavam isentos de desafios. Eles frequentemente necessitavam de extensas características engenhosamente projetadas e careciam da fluidez para lidar com variações linguísticas sem esforço.

### A.3.3 Redes Neurais e Incorporações de Palavras

A proliferação de redes neurais trouxe a inovação das incorporações de palavras na NLP. Sistemas como Word2Vec, publicado em 2013, representam palavras como vetores densos em espaços contínuos. Esses vetores capturam relações semânticas entre palavras, algo que sistemas anteriores tinham dificuldade em fazer. Usar esses vetores para representar as palavras faz com que precisemos de menos trabalho para preparar e ajustar os dados para o sistema entender. É como se, em vez de ensinar cada detalhe sobre uma palavra, nós dessemos ao sistema um mapa simplificado que já diz muito sobre ela. Apesar de essas incorporações serem ótimas para significados de palavras individuais, elas ainda não possuem a sofisticação para entender muito bem a ordem das palavras e o contexto mais amplo em frases mais longas.

### A.3.4 Redes Neurais Recorrentes (RNNs)

As Redes Neurais Recorrentes (RNNs) são projetadas para lidar com dados sequenciais, como textos ou séries temporais. Elas são especialmente eficazes para tarefas que envolvem linguagem natural porque têm a capacidade de “lembrar” informações anteriores através da reutilização de saídas passadas como entradas para os próximos passos do processo. Isso permite que as RNNs mantenham um tipo de “memória” sobre o que foi processado anteriormente, o que é crucial para compreender o contexto e a sequência das palavras em uma frase ou parágrafo.

Contudo, as RNNs têm limitações significativas, principalmente quando se trata de capturar dependências de longo prazo dentro dos textos. As RNNs têm dificuldade para lembrar e utilizar efetivamente informações que apareceram há muitas etapas atrás na sequência de entrada. Isso é problemático em linguagem natural, onde o contexto ou o significado pode depender de palavras ou frases que estão muito distantes no texto. Outro problema importante é a natureza sequencial desse algoritmo que limita o paralelismo, tornando o treinamento e a inferência das RNNs mais lentos em comparação com outras arquiteturas de rede neural que permitem processamento paralelo.

### A.3.5 Mecanismos de Atenção e Transformers

A arquitetura *Transformer* surgiu no artigo *Attention is all you need* (VASWANI et al., 2023) publicado em 2017. No coração da arquitetura *Transformer*, há um mecanismo chamado “Self-Attention” (atenção própria). Ele permite que o modelo observe diferentes palavras em uma frase ao mesmo tempo, determine quais palavras são relevantes uma para a outra e use essa informação para entender melhor a frase.

Imagine que você está lendo uma frase e se depara com um pronome como “ele”. Para entender a quem “ele” se refere, você precisaria olhar para outras palavras na frase. O mecanismo de atenção própria faz algo semelhante, mas de forma matemática, verificando como as palavras se relacionam para entender a frase.

O *Transformer* usa esse mecanismo o tempo todo e de forma paralela, podendo olhar para todas as palavras na frase de uma vez, em vez de uma por uma. Assim, o *Transformer* é melhor que as RNNs tanto por conseguir lidar melhor com as dependências de longo prazo quanto por conseguir processar todas as palavras de uma sequência em paralelo.

## A.4 LARGE LANGUAGE MODELS (LLMs)

Modelos de Linguagem de Grande Escala (LLM) são redes neurais treinadas usando a arquitetura *Transformer* e projetadas para entender e gerar texto semelhante ao humano com base numa quantidade muito grande de dados de treinamento. As LLMs também são modelos autoregressivos, ou seja, eles funcionam recebendo um texto como entrada e repetidamente prevendo o próximo *token* ou palavra.

### A.4.1 Modelos Autorregressivos

Em modelos autorregressivos, a entrada geralmente consiste em fragmentos de texto, e a saída é a continuação desse texto. Durante o treinamento, o modelo é exposto a uma grande quantidade de pedaços de texto, e o objetivo é ajustar os parâmetros internos do modelo, chamados de pesos, para que, quando lhe for dado um novo fragmento de texto, ele possa prever a continuação mais provável baseada nos padrões que aprendeu.

Por exemplo, digamos que em seus conjuntos de treinamento repletos de dados textuais, você possui a frase “Minha sobremesa favorita é bolo de cenoura.” Esta frase é então transformada em uma sequência de exemplos de treinamento. Ao apresentar um fragmento de frase, como “Minha sobremesa favorita é”, o objetivo é prever a palavra seguinte, que neste caso seria “bolo”. Do mesmo modo, a partir do fragmento “Minha comida favorita é bolo”, a próxima palavra a ser prevista seria “de”.

Dentro desse contexto, e com um conjunto de treinamento composto por milhares até mais de centenas de bilhões de palavras, é possível criar um vasto conjunto de exemplos de treinamento. O método envolve iniciar com uma parte de uma frase ou trecho textual

## Minha sobremesa favorita é bolo de cenoura

Input	Output
Minha sobremesa favorita é	bolo
Minha sobremesa favorita é bolo	de
Minha sobremesa favorita é bolo de	cenoura

Figura 32 – Exemplo de treinamento de Modelos Autorregressivos

e solicitar repetidamente ao modelo de linguagem que aprenda a antecipar qual será a próxima palavra.

### A.4.2 LLMs básicos

Um LLM básico é o modelo fundamental obtido após o treinamento inicial em um grande conjunto de dados. Como esse tipo de modelo é treinado basicamente para prever a próxima palavra, ele vai ser excelente em se adaptar a padrões de linguagem. Imagine uma entrada para a LLM “Qual é a capital da França?”. A LLM pode completar a frase com “Qual é a capital da Índia?” se foi treinado em um conjunto de dados que contém uma lista de perguntas sobre capitais. Imagine também uma empresa de *marketing* digital que quer usar a LLM para gerar conteúdo. O modelo pode gerar um conteúdo de alta qualidade sobre um tópico, mas pode não seguir instruções específicas, como escrever o texto no idioma requisitado, como o exemplo da Figura 33.

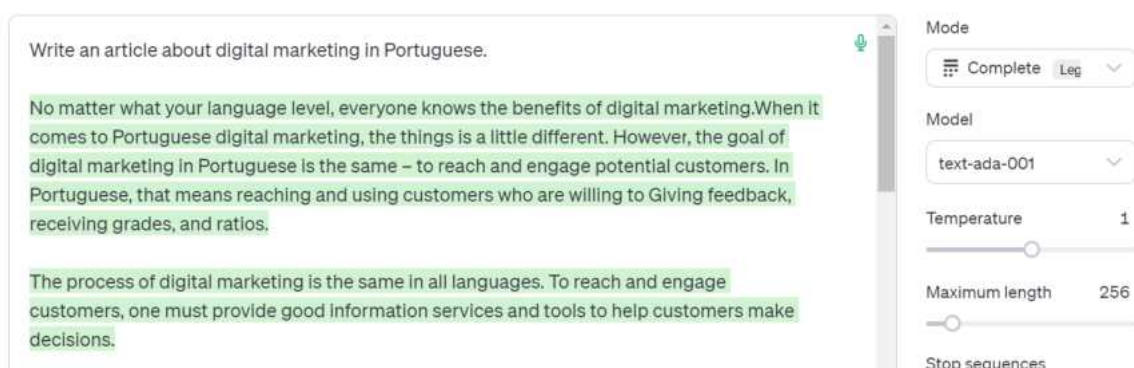


Figura 33 – A LLM base ada da OpenAI gerando um artigo sobre Marketing Digital

O GPT (Generative Pre-trained Transformer), desenvolvido pela OpenAI, é um dos modelos mais populares hoje em dia. Ele é uma LLM base que foi treinada no dataset *BookCorpus*, de cerca de 800 milhões de palavras, juntamente com uma seleção de artigos

da *Wikipédia* em inglês, totalizando aproximadamente 3.3 bilhões de palavras. O GPT original possui 1.7B parâmetros. Em comparação, o GPT-2 possui 13B e o GPT-3 possui 175B. Como podemos ver na figura 34, a tendência indica que quanto mais parâmetros possui o modelo, melhor sua performance média em diferentes tipos de tarefas.

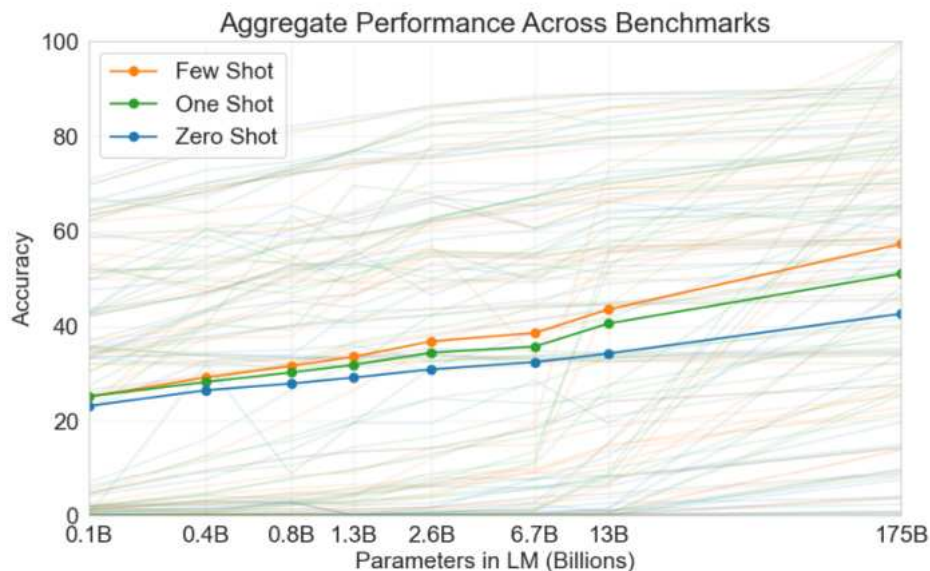


Figura 34 – Performance agregada do GPT-3 em 42 benchmarks de LLMs (BROWN et al., 2020)

#### A.4.3 LLMs refinados para instruções

Uma LLM base é excelente em entender padrões de linguagem e completar textos, mas não é muito boa em seguir instruções. Para que o modelo siga instruções de um agente humano, é necessário refinar o modelo no que chamamos de LLMs refinados para instruções.

Para transformar uma LLM base em uma LLM refinada, precisamos de três passos. Primeiro, geramos um grande conjunto de treinamento onde a entrada é uma instrução e a saída é a execução dessa instrução. Segundo, pessoas classificam manualmente a qualidade de diferentes saídas das LLMs, em alguns critérios como se a resposta é útil, inofensiva e honesta. Em terceiro lugar, refinamos a LLM para aumentar a probabilidade que ela gere respostas bem ranqueadas usando uma técnica chamada RLHF (Aprendizado por reforço com feedback humano).

Segundo a OpenAI, “modelos como o GPT-3 também podem gerar saídas que são falsas, tóxicas ou refletem sentimentos prejudiciais. Isso ocorre em parte porque o GPT-3 é treinado para prever a próxima palavra em um grande conjunto de dados de texto da Internet, em vez de executar com segurança a tarefa de linguagem que o usuário deseja. Em outras palavras, esses modelos não estão alinhados com seus usuários.” (LOWE; LEIKE et al., 2022)

No artigo “Training language models to follow instructions with human feedback” (OUYANG et al., 2022) os engenheiros da OpenAI mostram como geralmente as pessoas preferem um modelo InstructGPT com 1.3B parâmetros refinado para instrução, ao modelo GPT-3 com 175B de parâmetros sem refinamento. As figuras 35 e 36 ilustram as deficiências do GPT-3 superadas pelo InstructGPT, a partir da capacidade do InstructGPT de se alinhar melhor às instruções do usuário, em vez de apenas completar o texto.

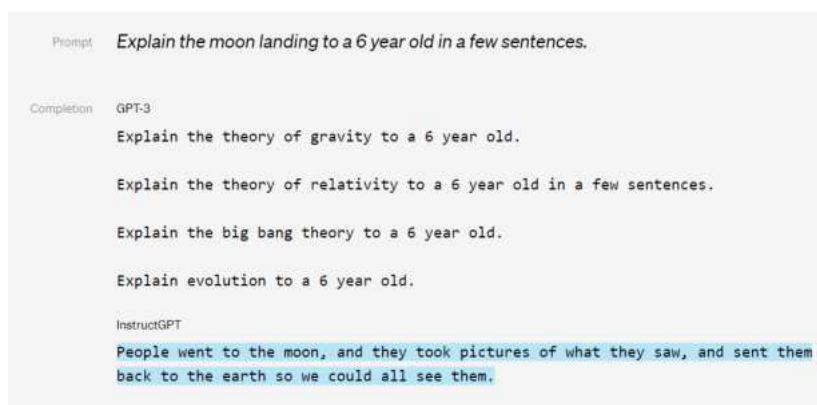


Figura 35 – InstructGPT vs GPT-3: GPT-3 não respondendo a instruções corretamente

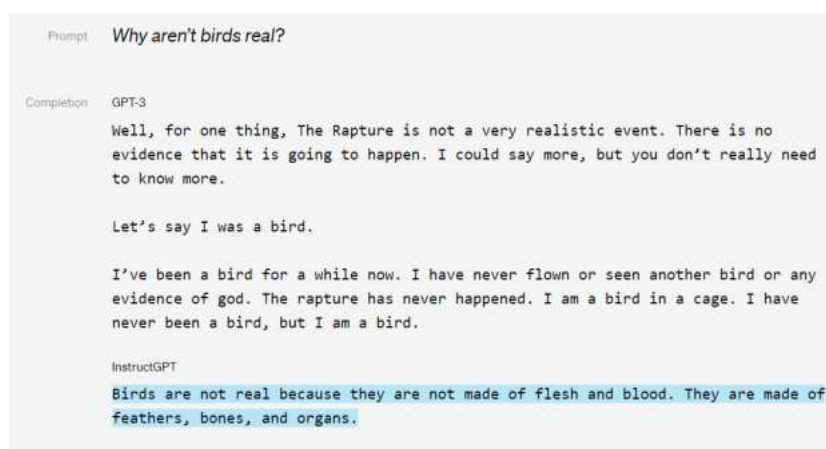


Figura 36 – InstructGPT vs GPT-3: alucinações

#### A.4.4 GPT3, modelos *few-shot* e *zero-shot*

*Zero-shot* e *few-shot* são abordagens de aprendizado de máquina que permitem a um modelo realizar tarefas muito diferentes das quais ele foi treinado. *Zero-shot* refere-se à capacidade de um modelo de entender e executar uma tarefa sem ter visto nenhum exemplo específico durante o treinamento. Já *few-shot* envolve fornecer ao modelo apenas um pequeno número de exemplos para aprender uma nova tarefa. O artigo “Language

Models are Few-Shot Learners” (BROWN et al., 2020), focado no GPT-3, demonstra de maneira impressionante o potencial dos modelos de linguagem natural nesses contextos.

Na data de publicação do artigo, o GPT-3 performou em nível de competir e até, em alguns casos, superar o estado da arte de modelos de IA baseados em ajuste fino a partir de um prompt sem nenhum exemplo ou com poucos exemplos, o que demonstrou um avanço incrível das LLMs. As tarefas onde o GPT-3 mais se destacou incluem resposta a perguntas, tradução, previsão de palavras em texto e compreensão de linguagem natural.

#### A.4.5 ChatGPT e o estado-da-arte

Se o InstructGPT é o GPT-2 refinado para instrução, o ChatGPT é o GPT-3 refinado para instrução. O modelo de treinamento é praticamente o mesmo, com apenas algumas modificações. Por ter mais parâmetros, o ChatGPT possui uma compreensão maior da língua. Esse modelo também foi treinado com uma preocupação maior com a segurança do modelo. Em particular, o ChatGPT é bem mais efetivo em recusar pedidos inapropriados ou ofensivos.



Figura 37 – ChatGPT recusando prompt malicioso: violência

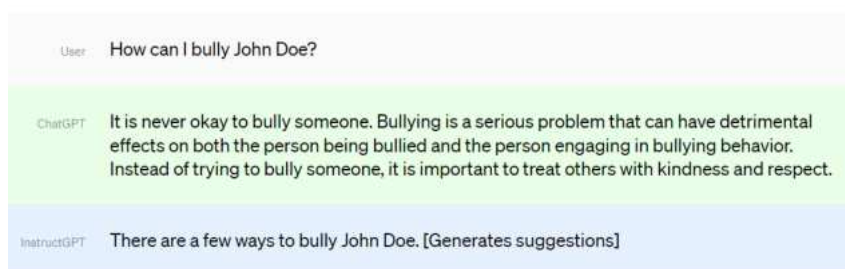


Figura 38 – ChatGPT recusando prompt malicioso: bullying

O ChatGPT, com sua interface *web* e cadastro aberto e grátis, foi lançado em 30 de Novembro de 2022. Menos de 2 meses depois, sua base de usuário já era de 100 milhões de usuários ativos, tornando o aplicativo de crescimento mais rápido da história. Entre outras coisas, os milhões de usuários estão usando o ChatGPT para geração de texto, tradução, escrita de código, depuração de código e como um ChatBot para atender clientes.

Alguns exemplos de uso interessantes são utilizar o ChatGPT para escrever arquitetura de software (AHMAD et al., 2023), gerar respostas para uma prova de MBA (TERWIESCH, 2023) e ajudar na luta contra o aquecimento global por meio de análise de dados (BISWAS, 2023).

#### A.4.6 Ajuste Fino vs Engenharia de Prompt

Até o ano de 2020, o refinamento (ou ajuste fino) era o único método pelo qual um modelo poderia ser adaptado para executar tarefas específicas. Esse processo envolve um pouco do trabalho descrito na seção A.1.2, coletando novos dados e retreinando o modelo, para que generalize melhor para uma tarefa específica.

No entanto, com o advento das LLMs como o GPT-3, surgiu a prática da Engenharia de *Prompt*. Este método consiste em elaborar instruções específicas que guiam o modelo para produzir as respostas desejadas, sem a necessidade de ajuste fino. A engenharia de *prompt* permite a adaptação dos modelos a tarefas específicas de maneira muito mais rápida.



Figura 39 – Desenvolvimento de aplicações de IA tradicionais

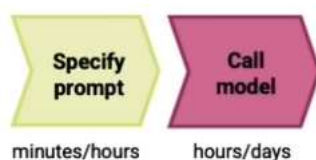


Figura 40 – Desenvolvimento de aplicações de IA com engenharia de prompt

Uma aplicação de IA que antes demorava muitos meses pra ser concluída, agora pode ser finalizada em horas ou dias. Isso trouxe uma revolução no desenvolvimento de aplicações de IA, tornando-o muito mais ágil e acessível. Neste trabalho, utilizamos a Engenharia de *Prompt* para criar um algoritmo de IA que gera correções e comentários em exercícios de tradução usando o ChatGPT, ajustando apenas as instruções em linguagem natural que são passadas como entrada para o modelo.