Tutoria no Ensino Fundamental com Inteligência Artificial e Machine Learning: Implementação de Sistemas Inteligentes

Anelma Costa[1], Julia Santos[1], Juliane Santos[1], Natalia Melo[1], Felipe Fernandes[1]

[1] Department of Computer Engineering and Automation (DCA), Federal University of Rio Grande do Norte (UFRN), Natal, Rio Grande do Norte, Brazil

Resumo

O uso de IA na educação tem se mostrado transformador, sendo amplamente aplicado na personalização do aprendizado e no aprimoramento dos métodos de ensino. Dessa forma, este trabalho tem como objetivo implementar um sistema de tutoria para alunos do ensino fundamental, por meio de um *chatbot* (*TItiaMath) ,* no *Telegram* para auxiliar no aprendizado de matemática. A implementação do chatbot utiliza um modelo de *machine learning* (ML) baseado em *large language mode* (LLM), com a IA Gemini 1.5 Flash. O modelo foi ajustado para restringir o contexto, garantindo respostas precisas para conteúdos de matemática no ensino fundamental. Considerando que o modelo utilizado já é pré-treinado, não foi necessário realizar novo treinamento. Além disso, a aplicação foi testada para avaliar sua assertividade nas respostas, incluindo a verificação de respostas fora do contexto. Os resultados mostraram-se satisfatórios, com o *chatbot* respondendo de maneira assertiva. Dessa forma, a *TitiaMath* se revelou uma ferramenta funcional e acessível, contribuindo para o aprimoramento do aprendizado dos alunos na disciplina.

Abstract

The use of AI in education has proven to be transformative, being widely applied in the personalization of learning and the enhancement of teaching methods. Therefore, this work aims to implement a tutoring system for elementary school students through a chatbot (TitiaMath) on Telegram to assist in math learning. The implementation of the chatbot uses a machine learning (ML) model based on a large language model (LLM), with the AI Gemini 1.5 Flash. The model was fine-tuned to restrict the context, ensuring precise responses for elementary school math content. Considering that the model used is already pre-trained, no new training was required. Additionally, the application was tested to assess its accuracy in responses, including verifying off-context answers. The results were satisfactory, with the chatbot responding assertively. Thus, TitiaMath proved to be a functional and accessible tool, contributing to the improvement of students' learning in the subject.

Palavras-chave: Inteligência artificial, *Machine learning*, Tutoria educacional, *Large Language Models*, GPT.

1. Introdução

O conceito de Inteligência Artificial (IA) remonta à década de 1950 (SANTAELLA, 2023) e, ao longo do tempo, tem sido associado tanto a narrativas distópicas de ficção científica quanto a soluções inovadoras para problemas diversos. Atualmente, a IA está cada vez mais presente no cotidiano, desde a otimização de serviços bancários por meio de aplicativos de smartphones até dispositivos que interagem com os usuários utilizando linguagem natural, como a assistente virtual *Alexa*, da *Amazon* (RODRIGUES; RODRIGUES, 2023).

No contexto educacional, o uso da IA tem desempenhado um papel crescente e transformador, uma vez que essas tecnologias vêm sendo amplamente aplicadas para personalizar a aprendizagem, aprimorar o processo de ensino e enriquecer a experiência de alunos e professores.

No que se refere ao modo de operação, os sistemas de IA mais avançados atualmente são aqueles que empregam ML. O aprendizado de máquina refere-se à capacidade de um modelo adquirir conhecimento de forma autônoma, sem programação explícita, baseando-se na identificação de padrões e correlações em grandes volumes de dados. Destaca-se que, para conceitos mais restritos de IA, essa habilidade de aprendizado de máquina é o que diferencia a inteligência artificial da simples automação, que possui uma abrangência mais limitada (NUNES; ANDRADE, 2023, citado em BARROSO; MELLO, 2024).

Entre as diferentes abordagens de IA aplicadas à educação, destacam-se os sistemas de tutoria inteligentes, projetados para identificar e atender às necessidades individuais de cada estudante, fornecendo suporte adaptativo e eficaz para o desenvolvimento de habilidades e conhecimentos. Aplicações de IA utilizando LLMs em sistemas de tutoria têm se tornado cada vez mais avançadas e promissoras, especialmente no campo da educação personalizada, sendo empregadas de diversas formas para melhorar a experiência de aprendizado e o suporte ao estudante.

No Brasil, a educação básica apresenta deficiências significativas devido a diversos fatores econômicos e sociais. Esse cenário é evidenciado por dados do Programa

Internacional de Avaliação de Estudantes (PISA), segundo os quais apenas 27% dos alunos brasileiros alcançaram o nível 2 de proficiência em matemática, considerado o patamar mínimo de aprendizado. Em contrapartida, a média dos países da Organização para a Cooperação e Desenvolvimento Econômico (OCDE) é de 69%. Apenas 1% dos estudantes brasileiros atingiram os níveis 5 ou 6, que envolvem a resolução de problemas complexos e a comparação de estratégias, enquanto a média da OCDE é de 9%. Entre os 81 países e economias participantes do PISA 2022, apenas 16 apresentaram mais de 10% de seus alunos nos níveis mais altos de proficiência.

Nesse contexto, o presente trabalho tem como objetivo o desenvolvimento de um sistema de tutoria inteligente baseado em IA e ML, com foco em melhorar o desempenho dos alunos em matemática. A implementação do sistema ocorreu por meio de um modelo LLM integrado à plataforma *Gemini*, em conjunto com o *Telegram*, utilizando um *chatbot* denominado de *TItiaMath,* como interface de interação com os estudantes.

2. Métodos

O *chatbot* foi idealizado com o intuito de auxiliar estudantes do ensino fundamental, com foco, principalmente, na área de matemática básica. Dessa forma, como alternativa para a interação entre usuário e sistema, foi utilizado o serviço de mensagens instantâneas *Telegram*. O sistema de tutoria emprega um modelo de *Machine Learning* (ML) baseado em um modelo de *Large Language Mode* (LLM), utilizando como ferramenta para esse propósito a IA *Gemini 1.5 Flash*.

O primeiro passo para a implementação do *bot* foi a criação de chaves de acesso: uma para o *bot* criado no *Telegram* e outra para a integração com a IA. As chaves foram armazenadas em um arquivo .env. Foi necessário realizar o *download* e a instalação de duas bibliotecas para a integração dos dois sistemas, sendo elas: *telebot* e *google-generativeai*.

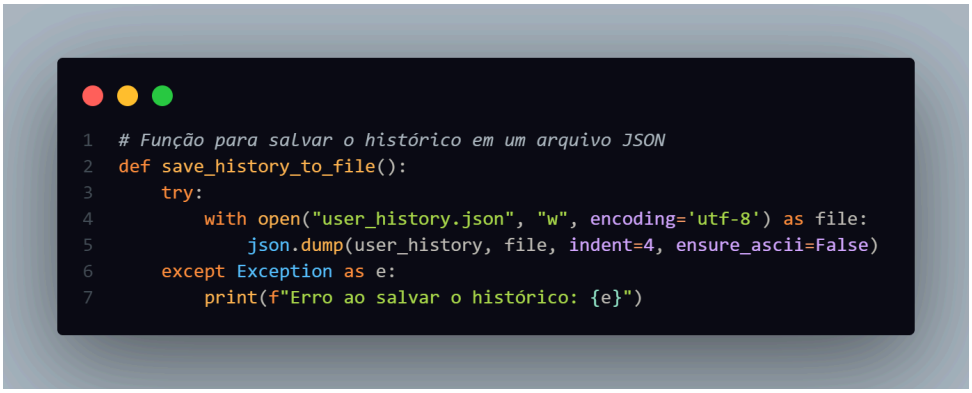
Na sequência, com base nas bibliotecas apresentadas, as linhas de código básicas para a integração são mostradas na Fig. 01.

```python
# Carrega as variáveis de ambiente do arquivo .env
load_dotenv()

# Obtém a variável de ambiente 'APIKEY' para o Google Generative AI
api_key = os.getenv('APIKEY')

# Configura o modelo com a chave da API
genai.configure(api_key=api_key)
model = genai.GenerativeModel('gemini-1.5-flash')

# Configuração do bot do Telegram
API_KEY_TELEGRAM = os.getenv('TELEGRAMKEY')  # A chave de API do Telegram

bot = telebot.TeleBot(API_KEY_TELEGRAM)
```

Fig. 01: Integração das APIs.

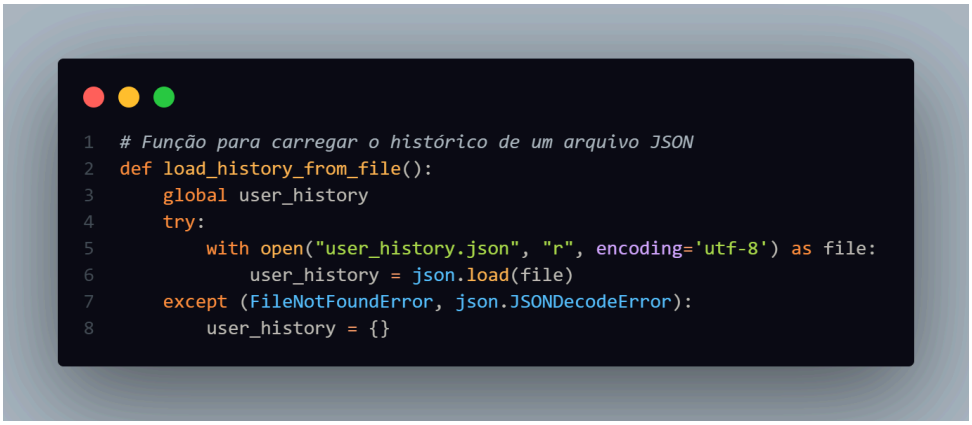
2.1. Funcionamento

O *chatbot* conta com um gerenciamento de usuários, que serve principalmente para controlar quais usuários utilizam o sistema e seus respectivos históricos. O sistema armazena essas informações em um arquivo .json, Fig. 02, que posteriormente é carregado em um dicionário denominado *user_history*. Esse dicionário pode ser acessado para verificar a existência ou inexistência da matrícula em questão, Fig. 03.



```python
# Função para salvar o histórico em um arquivo JSON
def save_history_to_file():
    try:
        with open("user_history.json", "w", encoding='utf-8') as file:
            json.dump(user_history, file, indent=4, ensure_ascii=False)
    except Exception as e:
        print(f"Erro ao salvar o histórico: {e}")
```

Fig. 02: Função para salvar o histórico em um arquivo .json.



```python
# Função para carregar o histórico de um arquivo JSON
def load_history_from_file():
    global user_history
    try:
        with open("user_history.json", "r", encoding='utf-8') as file:
            user_history = json.load(file)
    except (FileNotFoundError, json.JSONDecodeError):
        user_history = {}
```

Fig. 03: Função para carregar o histórico de matrículas registradas em um dicionário.

Ao iniciar o *bot*, o sistema verifica a existência da matrícula do aluno por meio da função *verify_matricula*, Fig. 4.. Após o envio da matrícula pelo usuário, o *chat* consultará o dicionário salvo, *user_history*, para analisar sua existência. Caso a matrícula esteja registrada, o *chat* dará continuidade à conversa normalmente. Caso contrário, o *chat* seguirá por outro caminho, acionando a função *process_name* (Figura 05).

```python
# Verifica se a matrícula existe no histórico
def verify_matricula(message):
    chat_id = message.chat.id
    matricula = message.text.strip()

     # 🚨 Verificação do tamanho da matrícula (exatamente 8 dígitos)
    if not matricula.isdigit() or len(matricula) != 8:
        msg = bot.reply_to(message, "Matrícula inválida! Ela deve conter exatamente 8 dígitos numéricos. Tente novamente:")
        bot.register_next_step_handler(msg, verify_matricula)  # Solicita novamente
        return

    if matricula in user_history:
        user_data = user_history[matricula]
        bot.reply_to(message, f"Bem-vindo de volta, {user_data['nome']}! Como posso ajudá-lo hoje?")

        # ⚡ Armazena a matrícula corretamente para futuras interações
        pending_registrations[chat_id] = matricula
    else:
        bot.reply_to(message, "Matrícula não encontrada. Vamos fazer seu registro.")
        pending_registrations[chat_id] = matricula  # Armazena a matrícula para continuar o registro
        msg = bot.reply_to(message, "Por favor, digite seu nome:")
        bot.register_next_step_handler(msg, process_name)
```

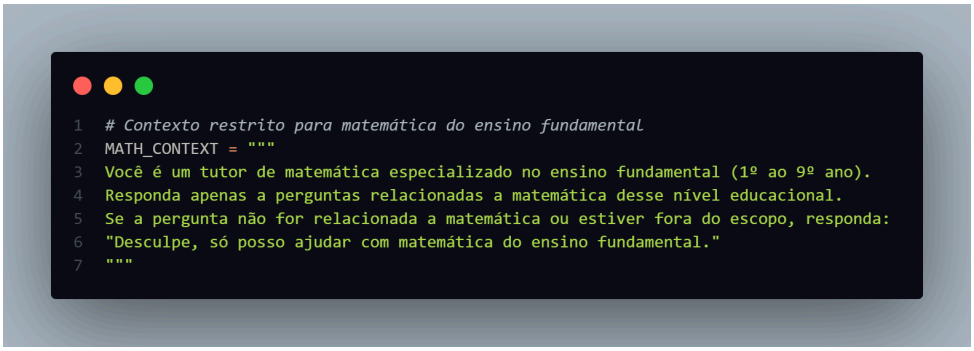
Fig. 4: Função para verificar o histórico de matrículas registradas em um dicionário.

A função *process_name* registrará o nome do usuário, armazenando-o no dicionário. Em seguida, ela chamará a função *process_class*, que será responsável por registrar a turma do aluno. Após isso, a função *process_class* chamará a função *process_teacher* para registrar o professor. As três funções terão como objetivo armazenar as informações de nome, turma e professor no dicionário.

```python
# Processo de registro de novo usuário
def process_name(message):
    chat_id = message.chat.id
    nome = message.text.strip()
    matricula = pending_registrations.get(chat_id)

    if not matricula:
        bot.reply_to(message, "Erro ao recuperar a matrícula. Tente novamente usando /start.")
        return

    user_history[matricula] = {"nome": nome, "turma": None, "professor": None, "interacoes": []}

    msg = bot.reply_to(message, f"Obrigado, {nome}! Agora, envie o nome da sua turma:")
    bot.register_next_step_handler(msg, process_class, matricula)

def process_class(message, matricula):
    turma = message.text.strip()
    user_history[matricula]["turma"] = turma

    msg = bot.reply_to(message, f"Entendido! Agora, envie o nome do(a) professor(a):")
    bot.register_next_step_handler(msg, process_teacher, matricula)

def process_teacher(message, matricula):
    professor = message.text.strip()
    user_history[matricula]["professor"] = professor

    bot.reply_to(message, f"Registro completo! Bem-vindo(a), {user_history[matricula]['nome']}!")

    # Salva o histórico após o registro
    save_history_to_file()
```

Fig. 5: Função para criação dos dados do usuário.

Para evitar que o *chatbot* responda ao usuário perguntas fora do contexto de matemática para o ensino fundamental, o modelo de IA foi ajustado para restringir o contexto. Para isso, foi adicionada uma constante *MATH_CONTEXT* que especifica ao bot que ele só deve responder a questões sobre matemática fundamental. Esse contexto é enviado para a IA com a pergunta do usuário sendo concatenada a ele. Caso a pergunta esteja fora do contexto restrito, o *bot* responderá que não pode responder a pergunta.



```python
# Contexto restrito para matemática do ensino fundamental
MATH_CONTEXT = """
Você é um tutor de matemática especializado no ensino fundamental (1º ao 9º ano).
Responda apenas a perguntas relacionadas a matemática desse nível educacional.
Se a pergunta não for relacionada a matemática ou estiver fora do escopo, responda:
"Desculpe, só posso ajudar com matemática do ensino fundamental."
"""
```

Fig. 6: Constante para restringir as respostas do *bot*.

Para tratar qualquer mensagem enviada ao *bot* após o *login*, foi definido um manipulador de mensagens através da função *respond_to_message*. Este manipulador organiza e registra as comunicações com os usuários. Quando o *bot* recebe uma mensagem, ele obtém o ID do *chat* e verifica se este ID está registrado no dicionário de matrículas pendentes. Caso o ID não esteja registrado, o *bot* solicita a matrícula do usuário e registra o próximo passo para validação. Se o ID estiver registrado, o *bot* recupera a matrícula associada e verifica se ela existe no histórico de usuários.

Se a matrícula for válida, o histórico de interações do usuário é atualizado com a mensagem recebida. Em seguida, o *bot* gera uma resposta da IA usando um contexto pré-definido, adiciona a resposta gerada ao histórico do usuário e a envia de volta para o *chat*. Caso a matrícula não seja reconhecida, o *bot* informa o usuário e solicita que ele reinicie o processo de autenticação. Todas as interações realizadas são armazenadas em um arquivo .json para garantir a persistência dos dados.

```python
# Manipulador de mensagens para interação normal após o login
@bot.message_handler(func=lambda message: True)
def respond_to_message(message):
    chat_id = message.chat.id

    # Verifica se temos a matrícula salva para esse chat_id
    if chat_id not in pending_registrations:
        msg = bot.reply_to(message, "Antes de prosseguirmos, por favor, informe sua matrícula:")
        bot.register_next_step_handler(msg, verify_matricula)
        return

    # Recupera a matrícula salva para esse usuário
    matricula = pending_registrations[chat_id]

    # ⚡ Verifica corretamente se a matrícula existe no histórico
    if matricula in user_history:
        user_data = user_history[matricula]

        # Adiciona a interação ao histórico
        user_data["interacoes"].append({"role": "user", "text": message.text})

        try:
            # Envia o prompt com contexto para a IA
            response = model.generate_content(MATH_CONTEXT + "\nUsuário: " + message.text + "\nIA:")

            # Adiciona a resposta ao histórico
            user_data["interacoes"].append({"role": "bot", "text": response.text})

            # Envia a resposta ao usuário
            bot.reply_to(message, response.text)

        except Exception as e:
            bot.reply_to(message, "Houve um problema ao processar sua solicitação. Tente novamente mais tarde.")

        # Salva o histórico após a interação
        save_history_to_file()
    else:
        bot.reply_to(message, "Matrícula não reconhecida. Por favor, inicie novamente com /start.")
```

Fig. 7: Função para lidar com a forma de resposta do bot.

3. Resultados

Considerando que o modelo LLM utilizado já é pré-treinado, não foi necessário realizar um novo treinamento ou aplicar métricas tradicionais de avaliação. No entanto, foram realizados testes com o *chatbot* para verificar a acurácia e assertividade das respostas geradas, bem como avaliar as restrições implementadas para garantir que ele forneça exclusivamente conteúdos de matemática voltados ao ensino fundamental.

A Fig. 8 exibe o registro de um usuário, tendo em vista que a matrícula informada ainda estava cadastrada, solicitando assim informações como matrícula, nome, turma e professor. Por sua vez, a Fig. 9 demonstra o retorno ao *chatbot*, apresentando a matrícula do usuário, o que evidencia a capacidade do sistema de armazenar e recuperar dados fornecidos durante as interações.

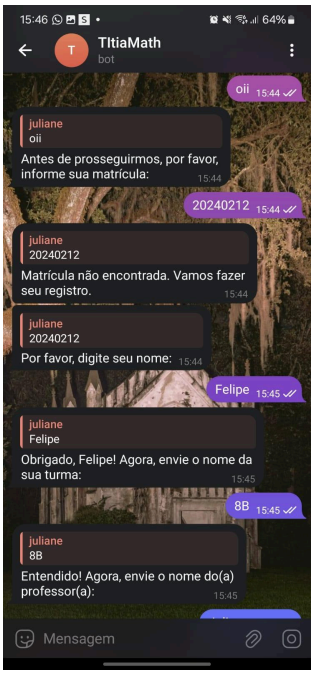


Fig. 8: Registro de usuário.

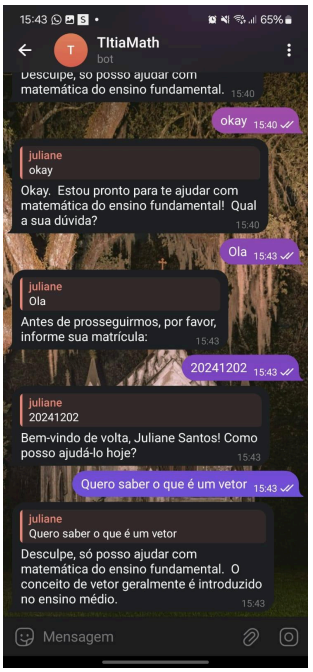


Fig. 9: Retorno do usuário ao chat.

Na Fig. 10 é apresentado o retorno do chatbot quando o usuário insere uma matrícula inválida, ou seja, com mais ou menos de oito dígitos.

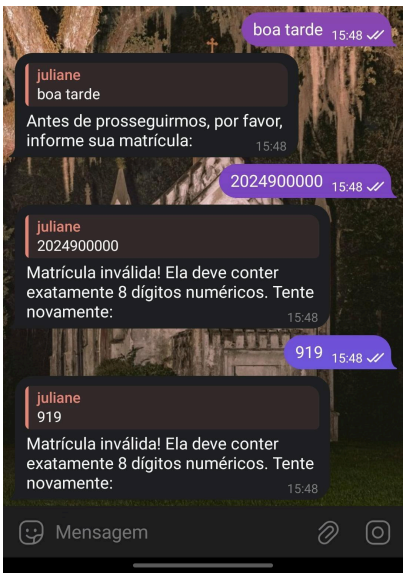
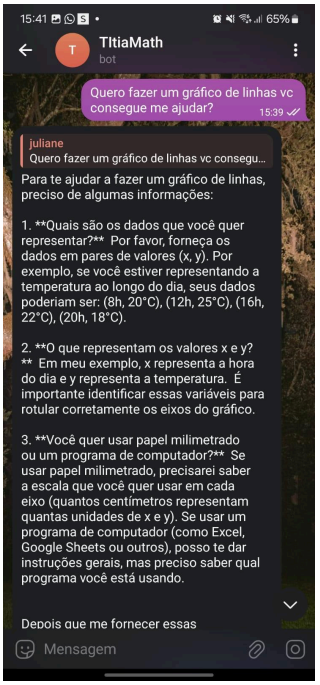


Fig. 10: Retorno do *chatbot* para matrículas inválidas.

Além disso, as Figs. 11 a 15 ilustram interações com o *chatbot*, destacando sua capacidade de gerar respostas precisas e alinhadas ao contexto definido. Nas Figs. 11 a 14, observa-se que o *chatbot* responde de forma coerente sempre que o usuário solicita informações dentro do escopo estabelecido. No entanto, conforme apresentado na Fig. 15, quando são feitas solicitações fora do contexto, o *chatbot* gera uma resposta padrão, explicando que as interações estão restritas ao tema de matemática voltado ao ensino fundamental.



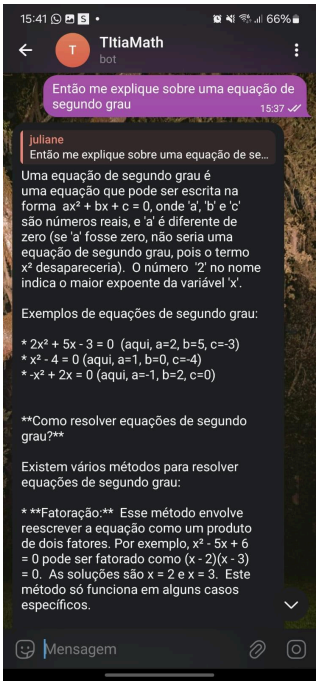


Fig. 11: Interação com o *chatbot*. **Fig. 12:** Interação com o *chatbot*

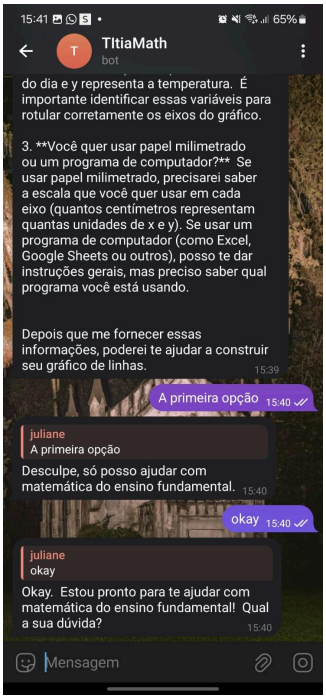


Fig. 13: Interação com o *chatbot*.

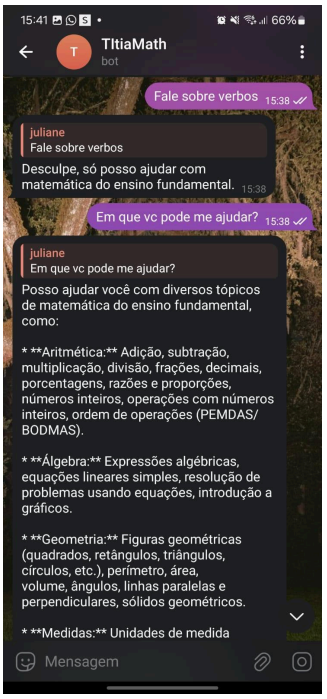


Fig. 14: Interação com o *chatbot*.

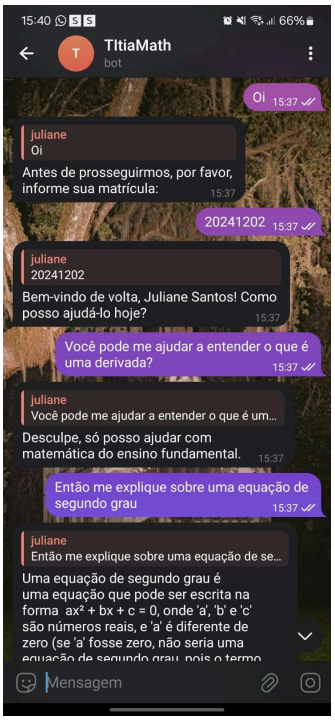


Fig. 15: Resposta padrão para solicitações que não estão no contexto.

Dessa forma, os usuários da aplicação podem utilizá-la para aprimorar seu desempenho em matemática, já que o *chatbot* demonstrou ser capaz de gerar respostas precisas e alinhadas ao contexto estabelecido, oferecendo uma ferramenta prática e eficiente para esclarecer dúvidas relacionadas exclusivamente ao conteúdo de matemática do ensino fundamental.

4. Conclusão

Portanto, a IA, amplamente aplicada no processo de aprendizagem, destaca-se por oferecer aos seus usuários uma vasta rede de conteúdos de forma rápida e eficaz. Nesse sentido, aplicações como a desenvolvida neste trabalho, voltadas para sistemas de tutoria inteligente, ampliam ainda mais as possibilidades de melhoria no aprendizado, uma vez que, atualmente, a grande maioria do público-alvo do projeto possui smartphones com fácil acesso a essa tecnologia. Dessa forma, a *TitiaMath* se mostrou uma ferramenta funcional e acessível aos alunos, permitindo que os alunos esclareçam suas dúvidas de forma eficiente, contribuindo significativamente para o aprimoramento do aprendizado na disciplina de matemática.

6. Referências

AGÊNCIA BRASIL. Menos de 50% dos alunos sabem o básico em matemática e ciências. Disponível em: https://agenciabrasil.ebc.com.br/educacao/noticia/2023-12/menos-de%2050%25-dos-alunos-sabem-o-b%C3%A1sico-em-matem%C3%A1tica-e-ci%C3%AAncias. Acesso em: 15 jan. 2025.

BARROSO, Luís Roberto; MELLO, Patrícia Perrone Campos. Inteligência artificial: promessas, riscos e regulação. algo de novo debaixo do sol. Revista Direito e Práxis, [S.L.], v. 15, n. 4, p. 1-45, jun. 2024. FapUNIFESP (SciELO). http://dx.doi.org/10.1590/2179-8966/2024/84479.

RODRIGUES, Olira Saraiva; RODRIGUES, Karoline Santos. **A inteligência artificial na educação: os desafios do chatgpt.** Texto Livre, [S.L.], v. 16, p. 1-12, jul. 2023. FapUNIFESP (SciELO). http://dx.doi.org/10.1590/1983-3652.2023.45997.

SANTAELLA, Lucia. A inteligência artificial é inteligente? São Paulo: Almedina, 2023.

7. Material Suplementar

Link da apresentação: https://youtu.be/3l1suFm9Sts.

Link do git: https://github.com/julianessantos/TitiaMathBot.