

ATENDENTE ARTIFICIAL HUMANIZADA USANDO LANGCHAIN PARA MANIPULAÇÃO DE MODELOS DE LINGUAGEM EM LARGA ESCALA

Daniel Ferreira Lara¹ (EG), Matheus Siston Galdino¹ (EG), Ryan Felipe Ferreira Ribeiro¹ (EG), Egon Luiz Muller Junior (PQ)¹

¹Universidade Federal de Itajubá - Campus Itajubá.

Palavras-chave: Inteligência Artificial, LLM, Chatbot, Langchain.

Introdução

O uso e desenvolvimento de inteligências artificiais (IA) aumentou consideravelmente nos últimos anos e uma das principais aplicações da IA é o atendimento ao consumidor (HO, 2021). Dessa forma, toda a motivação do projeto se dá pela necessidade do mercado de disponibilizar *chatbots* de atendimento capazes de interagir com o usuário de maneira humanizada sem perder a sequência de interações e saber reagir a situações que fujam do seu escopo evitando respostas incoerentes, tudo com a finalidade de aumentar a eficiência do serviço de atendimento ao consumidor (ANDRADE; TUMELERO, 2022).

Ademais, o trabalho teve como objetivo a construção de um agente artificial de atendimento bancário capaz de entender a demanda do usuário e prosseguir com a resolução do respectivo problema, percorrendo a fala de maneira natural. A construção geral da aplicação tem como base o *framework* LangChain que é capaz de interagir e manipular uma *Large Language Model* (LLM), onde se utilizou o GPT-3.5 através da API disponibilizada pela OpenAI. Além disso, para identificação precisa da demanda do cliente foi utilizado o modelo de classificação pré-treinado BERT-Banking77 que usa como fonte o *dataset* banking77 (CASANUEVA et al., 2020), que contém mais de 10 mil requisições de clientes classificadas em 77 rótulos. Desse modo, a lógica principal da aplicação baseia-se em fazer com que o agente seja capaz de agir sendo orientado pelo histórico da conversa e pelas diretrizes que são definidas conforme o estado atual do atendimento.

Metodologia

A pesquisa teve como objetivo desenvolver um agente artificial de atendimento bancário baseado em linguagem natural para fazer o atendimento completo ao cliente, conduzindo-o à resolução da requisição do cliente de forma livre, desde a análise do contexto da conversa, classificação da solicitação de serviço, condução do cliente nos diversos passos do processo até

a finalização do serviço identificado.

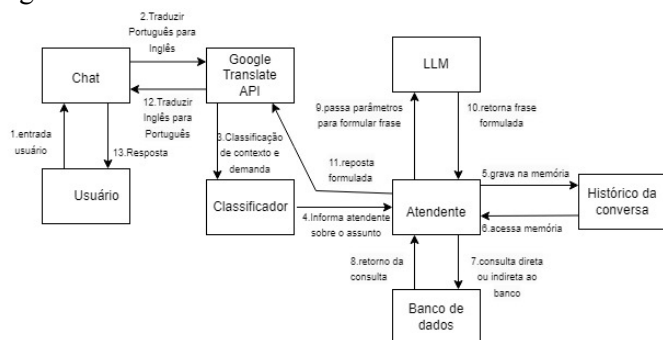
Para o desenvolvimento da solução foram usadas diversas ferramentas, de modo geral, utilizando-se o modelo de LLM GPT-3.5 através da OpenAI API para gerar todas as respostas da atendente ao usuário, analisar o contexto a partir de um *template*. O *framework* LangChain inseriu um contexto específico para que as respostas fossem mais assertivas baseadas no âmbito bancário cuja conversa é inserida. Ademais, foi empregado o modelo de classificação pré-treinado BERT-Banking77 para classificar qual o serviço requisitado pelo cliente dentro do contexto analisado. O armazenamento de informações foi feito a partir de um banco de dados utilizando bibliotecas da linguagem no qual o projeto foi implementado, o *python*.

Durante o início do atendimento, para compreender como o agente pode auxiliar o cliente é necessário analisar se o assunto é relacionado e a partir desse ponto se há algum problema ou solicitação pertinente. Assim, utilizou-se do LangChain para criar o escopo bancário como lastro para o LLM, quando o contexto é abordado é utilizado também o modelo Banking77 para classificação do serviço dentre 77 rotulações, para comunicar a entrada do usuário em português com esse modelo que trata informações em inglês foi usada a Google Translate API para traduzir a entrada e o rótulo identificado.

Quanto ao gerenciamento de dados, implementou-se um histórico de mensagens como fonte constante de informações para o sistema se basear, tal funcionalidade permite uma conversa mais contínua e fluida sem repetição de informações. Para ser possível realizar os serviços do agente foi necessário utilizar um banco de dados com os clientes da instituição financeira hipotética. Uma necessidade identificada foi o tratamento de dados sensíveis usados para consultas no banco de dados e realização do serviço pelo agente, os quais não podem ser direcionados ao LLM por segurança, ética e em respeito à legislação vigente. Como solução, o DbChain, ferramenta do LangChain, foi responsável por gerar consultas SQL ao banco de

dados de forma dinâmica para qualquer mensagem do cliente e sem que a resposta fosse acessada pelo servidor da openAI. Após as validações o atendimento segue para o estágio ideal. Segue abaixo o modelo de contexto construído para representar visualmente o sistema:

Figura 1 – Modelo de contexto



Para a construção da conversa, como o usuário tem liberdade durante a conversa em linguagem natural, o agente deve conduzir o atendimento tentando entender a solicitação e resolvê-la. Para que esse processo seja orgânico, as falas do atendente são geradas com base em orientações e no histórico, portanto são distintas a cada solicitação ao sistema. Os estágios da conversa modificam as ferramentas a serem utilizadas, existem algumas etapas bem definidas:

- Apresentação
- Análise de contexto
- Classificação de serviço
- Execução de roteiro
 - Consultas ao banco de dados
 - Solicitações de informações do usuário
 - Respostas às possíveis dúvidas
 - Informações de problemas identificados
 - Efetivação das transações requisitadas

Após os estágios iniciais de classificação, o agente passa a ter um *script* que fornece quais informações são necessárias para a realização do serviço, por meio do roteiro são geradas as orientações para o LLM responder ao cliente mudando o estágio da conversa. Além disso, a depender da etapa, são feitas consultas ao banco de dados para complementar a formulação da resposta.

Resultados e discussão

Devido às características da aplicação decidiu-se avaliar os resultados em casos de uso e de maneira qualitativa. A solução inicialmente se concentra em um agente

capaz de identificar a demanda do usuário e agir no contexto específico de “chegada de cartão” para dar seguimento ao caso. Por exemplo, exibindo uma caso em que tudo colaborou com o *script* :

```

atendente => Olá, meu nome é Alice e sou atendente do
banco Gringottes. Como posso ajudá-lo hoje?
usuário => Olá meu cartão ainda não chegou
atendente => Olá! Para que eu possa verificar o status do
seu cartão, preciso do número da sua conta. Poderia me
informar, por favor?
usuário => minha conta é 1
atendente => A conta de número 1 foi encontrada. Para
confirmar a titularidade, preciso do seu CPF, por favor.
usuário => meu cpf é 48142742039
atendente => Agradeço por verificar. Por favor, aguarde um
momento enquanto verifico se o cartão foi enviado.
atendente => Olá! Seu cartão já foi enviado e agora é só
aguardar a entrega. Caso queira acompanhar o status da
entrega, segue abaixo o código de rastreo: H7E9K2D4.
atendente => Posso ajudar em mais alguma coisa?
    
```

Nota-se que o agente lida bem com a situação, entendendo que a demanda trata-se de “chegada de cartão”, capturando e verificando o número da conta e CPF do usuário e por fim exibindo o código de rastreo correspondente com sucesso e questionando sobre algum outro problema no fim. Entretanto, no início, o agente diz “olá” duas vezes, o que não era esperado.

Em uma segunda situação, o agente foi colocado em cenários adversos, capturando a fuga de contexto do usuário na primeira fala, requisitando mais detalhes. Em uma nova tentativa, o usuário até se alinha ao contexto bancário, porém o atendente ainda não está programado para esse tipo de situação e reage corretamente. Logo em seguida o usuário acerta o contexto e o caso programado, porém testa algumas exceções com o número da conta inexistente, CPF inválido e o agente consegue contornar em todos os momentos. Contudo, no fim o agente não consegue retornar para um segundo atendimento simultâneo, o que é um problema a ser tratado.

```

atendente => Olá, meu nome é Alice e sou atendente do
banco Gringottes. Como posso ajudar você hoje?
usuário => eu sou um urubu
atendente => Eu entendo que você tem um assunto para
tratar, mas infelizmente, no momento, não tenho a
capacidade de solucionar o problema.
atendente => Desculpe, mas não entendi o que você está
procurando. Poderia me dar mais detalhes para que eu possa
ajudá-lo?
usuário => eu quero fazer uma transferência
atendente => Desculpe, mas ainda não tenho informações
sobre o assunto.
atendente => Desculpe, mas não entendi o que você está
    
```

procurando. Poderia me dar mais detalhes para que eu possa ajudá-Lo?

usuário => meu cartão está atrasado faz dias

atendente => Desculpe pelo transtorno. Para que eu possa verificar o status do seu cartão, preciso do número da sua conta. Poderia me informar, por favor?

usuário => eu não sei

atendente => Desculpe, mas não foi possível encontrar a conta "eu não sei". Por favor, poderia digitar o número da conta novamente?

usuário => 6

atendente => Desculpe, mas não foi possível encontrar a conta "6". Por favor, poderia digitar o número da conta novamente?

usuário => 3

atendente => A conta número 3 foi encontrada. Para confirmar a titularidade, preciso do seu CPF, por favor.

usuário => 222222222222

atendente => Desculpe, mas não foi possível verificar a titularidade da conta. Por favor, poderia digitar novamente o CPF?

usuário => 98884236061

atendente => Desculpe, mas não foi possível verificar a titularidade da conta. Por favor, poderia digitar novamente o CPF?

usuário => 20358962072

atendente => Agradeço pela verificação. Por favor, aguarde um momento enquanto verifico se o seu cartão foi enviado.

atendente => Desculpe pela demora, mas já verifiquei o status do seu cartão e ele já foi enviado. Agora é só aguardar a entrega. Se você quiser acompanhar a entrega, o código de rastreamento é P2R6T5Q8.

usuário => Posso ajudar em mais alguma coisa?

atendente => sim pode

Traceback (most recent call last): File
"/home/ryan/.pyenv/versions/3.10.2/Lib/python3.10/site-packages/urllib3/connectionpool.py", line 714, in urlopen
httplib_response = self._make_request(

porém só agir na classe “chegada de cartão”, contudo identifica de maneira errada relacionando “fazer depósito” com “saldo não atualizado após depósito” (3), o que torna necessário um refinamento da precisão de identificação do modelo, da tradução ou redução da faixa de tolerância de classificação. Em outra tentativa o usuário entra no assunto de uma forma diferente a requisição, deixando de maneira mais indireta (8), e o agente identifica com precisão da mesma forma. Além disso, houve a captura de algumas exceções e o uso de um dos módulos para retirar ‘.’ e ‘-’ do CPF para identificação (13). Por fim, novamente o mesmo problema com continuidade de atendimento.

atendente => Olá, meu nome é Alice e sou atendente do banco Gringottes. Como posso ajudar você hoje?

usuário => oi quero fazer um depósito

atendente => Eu entendo que o assunto é sobre o saldo não atualizado após depósito em cheque ou dinheiro, mas infelizmente ainda não posso solucionar o problema.

atendente => Olá, meu nome é Alice e sou atendente do banco Gringottes. Como posso ajudar você hoje?

usuário => hey Alice i want to make a transfer

atendente => Entendo que você tem um assunto a tratar, mas infelizmente ainda não posso solucionar o problema.

atendente => Desculpe, mas não entendi o que você quer dizer. Poderia me dar mais detalhes ou explicar melhor sua solicitação? Assim poderei tomar a ação adequada.

usuário => olha eu pedi um cartão ja faz 3 dias e ainda não veio

atendente => Peço desculpas pela demora na entrega do seu cartão. Para que eu possa verificar o status do seu pedido, poderia me informar o número da sua conta?

usuário => minha conta é catarina

atendente => Desculpe, mas não foi possível encontrar a conta "catarina". Por favor, poderia digitar o número da conta novamente?

usuário => número 2

atendente => A conta número 2 foi encontrada. Para confirmar a titularidade, preciso do seu CPF.

usuário => 988.842.360-61

atendente => Agradeço pela verificação. Por favor, aguarde um momento enquanto verifico se o seu cartão foi enviado.

atendente => Peço desculpas pela demora na entrega do seu cartão. Verifiquei que o cartão ainda não foi enviado, mas fique tranquilo, pois irei tomar as providências necessárias para que ele seja enviado o mais rápido possível.

atendente => Posso ajudar em mais alguma coisa?

usuário => pode

atendente => Desculpe, mas ainda não posso solucionar o problema.

atendente => Desculpe, mas não entendi o que você quer dizer. Poderia me dar mais detalhes ou explicar melhor sua solicitação? Assim poderei tomar a ação adequada.

usuário => pode, eu preciso de um empréstimo
Traceback (most recent call last): File

"/home/ryan/.pyenv/versions/3.10.2/Lib/python3.10/site-packages/urllib3/connectionpool.py", line 714, in urlopen
httplib_response = self._make_request(

No último teste quis-se testar uma outra requisição que

No terceiro caso analisado, temos um problema de verificação na lógica do código que busca constantemente casos em que o usuário deseja finalizar o atendimento, e como esse trabalho fica a cargo do próprio modelo de linguagem natural, ele pode ter interpretado a palavra “cancelar” como chave para a quebra do procedimento. Esse problema se enquadra em uma das necessidades de melhoria do projeto, que se refere a lógica *Hard Coded* usada na implementação, ou seja, uso de estruturas condicionais e de verificações puras da linguagem, com abuso de flags de confirmação, o que torna o código de difícil manutenção e expansão.

atendente => Olá, meu nome é Alice e sou atendente do banco Gringottes. Como posso ajudá-Lo hoje?
usuário => quero cancelar um pato
atendente => O Banco Gringottes agradece pelo contato.

Na quarta situação o agente no início faz o que deveria fazer, identificar os assuntos relacionados a banco,

levasse ao tema “chegada de cartão”, a primeira tentativa ocorreu sem sucesso porém a segunda foi identificada. Ademais, tentou-se variar um pouco mais a informação da conta do usuário, de maneira mais informal, e tudo ocorreu normalmente. A última característica testada foi a de sair do atendimento em qualquer momento, a primeira falhou e a segunda solicitação de fuga foi atendida.

atendente => Olá, meu nome é Alice e sou atendente do banco Gringottes. Como posso ajudá-Lo hoje?
usuário => você pode checar o estatus do meu cartão?
atendente => Entendo que você tem um assunto a tratar, mas infelizmente ainda não posso solucionar o problema.
atendente => Desculpe, não entendi o que você quer dizer. Poderia me dar mais detalhes ou informações para que eu possa te ajudar melhor?
usuário => Um cartão foi enviado para mim, como faço para rastreá-lo?
atendente => Para rastrear o seu cartão, preciso do número da sua conta bancária. Você poderia me fornecer o número da sua conta?
usuário => número 7
atendente => Desculpe, não foi possível encontrar a conta número 7. Por favor, poderia digitar o número da conta novamente?
usuário => umm entendi o número da conta em questão é 4
atendente => A conta número 4 foi encontrada. Para confirmar a titularidade, preciso do seu CPF. Por favor, poderia me fornecer o seu CPF?
usuário => eu quero sair
atendente => Desculpe, não foi possível verificar a titularidade da conta. Por favor, poderia digitar novamente o seu CPF?
usuário => eu gostaria de sair do atendimento
atendente => O Banco Gringottes agradece pelo contato.

A partir dos casos de uso analisados é possível verificar que a aplicação possui potencial e boa funcionalidade nas situações programadas dentro do assunto de “chegada de cartão”, porém ainda apresenta falhas e imprecisões em continuidade de atendimento, identificação de assuntos dentro do contexto bancário que ainda não fazem parte da bagagem da atuação da agente, saída instantânea do atendimento e lógica *Hard Coded*. Contudo, pode-se apontar sucesso na funcionalidade de manipulação principal da agente mediante orientações por estado de atendimento, uso do histórico de conversa para ações futuras, pré e pós-processamento das entradas de usuário para uso de recursos estrangeiros e desenvolvimento da fala de maneira natural com variabilidade pela agente.

Conclusões

Pode-se concluir com base na análise dos casos que, apesar de conseguir interagir bem em um caso de uso

ideal, ainda não é possível validar o uso do agente para ações diversificadas, pois dependendo do manejo das frases ela acaba perdendo a linha de interação ou não concluindo de maneira esperada. Dessa forma, seria necessário a realização de mais testes com usuários diferentes para identificação concisa do padrão das fragilidades a serem trabalhadas. Além disso, a implementação atual apresenta desafios de manutenção devido à lógica *Hard Coded* e a complexidade das estruturas condicionais, bem como alguns *bugs* identificados e apresentados. Para trabalhos futuros, pretende-se abstrair e padronizar a estrutura da lógica programada, tornando-a de mais fácil manutenção. Ademais, após a realização de mais testes, é necessário corrigir os *bugs* e fragilidades entre as interações no caso “chegada de cartão” para que seja possível ampliar os casos de uso.

Agradecimentos

Agradecemos a todos os colegas e professores do PET-TEC por cultivarem um ambiente de inovação tecnológica no qual esse trabalho cresceu. Além disso, a UNIFEI e ao FNDE pelo apoio ao projeto.

Referências

- ANDRADE, Ivan Martins De; TUMELERO, Cleonir. **Increasing customer service efficiency through artificial intelligence chatbot**. Revista de Gestão, v. 29, n. 3, p. 238-251, jul. 2021.
- CASANUEVA, Iñigo; TEMCINAS, Temcinas; GERZ, Daniela; HENDERSON, Matthew; VULIC, Ivan. **Efficient intent detection with dual sentence encoders**. In Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI, Association for Computational Linguistics, p. 38-45, Online, jul. 2020.
- HO, Ree. **Chatbot for Online Customer Service: Customer Engagement in the Era of Artificial Intelligence**. Advances in E-Business Research, p. 16-31, jan. 2021.
- SCHMID, Philipp. **BERT-Banking77**. Huggingface. Disponível em: <https://huggingface.co/philschmid/BERT-Banking77> Acesso em: ago.2023.
- OpenAI. **GPT-3.5 API**. Disponível em: <https://platform.openai.com/docs/>. Acesso em: ago.2023.
- Langchain. **Documentation**. Disponível em: <https://python.langchain.com/docs/>. Acesso em: ago.2023.