Desenvolvimento do Bot Enpower AI com o time Poli Júnior.

São Paulo

2024

Sumário

1. Resumo

O projeto Enpower AI foi dedicado ao desenvolvimento de um assistente virtual utilizando a API da OpenAI. Este chatbot foi especificamente projetado para auxiliar os atendentes de serviços de distribuição de energia elétrica, com a premissa de responder perguntas de forma assertiva, baseada nas leis estabelecidas pela resolução normativa 1000 de 2021. Além disso, o bot precisa ser capaz de responder a perguntas variadas, ser flexível e evitar alucinações.

Este documento descreve o processo conduzido pela equipe Poli Júnior, composta por Celso Tadaki Sinoka e Rafael Barsam Junqueira, em colaboração com Leandro Lind e Tiago Ferreira da equipe Enpower. O desenvolvimento foi realizado em um período de 8 semanas, durante o qual foram apresentados desenvolvimentos semanais do projeto, incluindo pesquisas, testes e o aprimoramento incremental do assistente com uma interface.

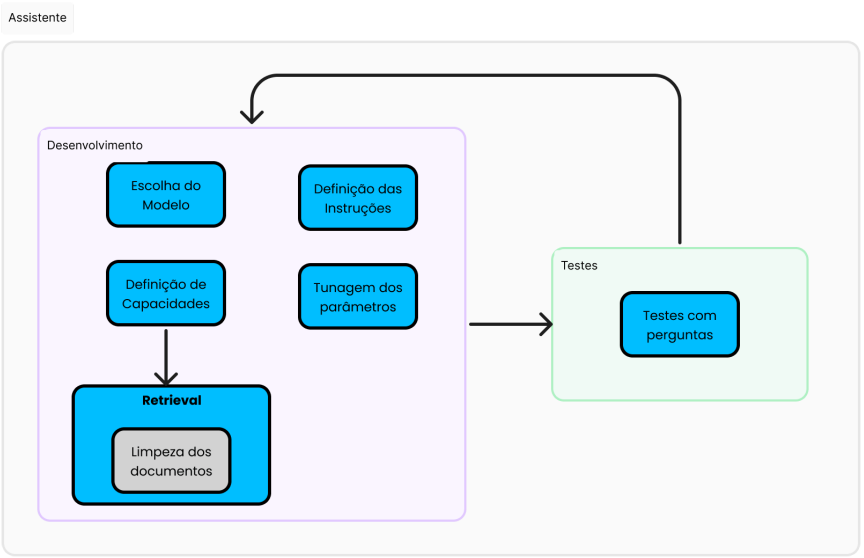
O documento detalha o caminho para o sucesso do projeto, abrangendo as iterações de testes da API e a formatação do conteúdo do modelo até alcançar o assistente final, que atendeu às expectativas como um assistente eficiente e com boa confiabilidade.

2. Primeira Etapa: RoadMap e Custos da API

Primeiramente, foram realizadas pesquisas sobre o uso da API da OpenAI para o desenvolvimento do assistente. Essas informações estão disponíveis na página da OpenAI sobre assistentes: Assistants overview - OpenAI API. Esse processo foi destacado em um roadmap apresentado logo em seguida.

O desenvolvimento do assistente é relativamente simples e envolve a definição do modelo utilizado, os parâmetros de ajuste e as capacidades adicionais desejadas. Uma capacidade importante a ser destacada é a de "retrieval", que envolve o armazenamento de documentos e o acesso a esses dados para gerar respostas.

É fundamental haver uma interação contínua entre o processo de desenvolvimento do assistente e a aplicação de testes, para aperfeiçoar seu desempenho. Isso inclui ajustes em cada um dos pontos, testando combinações de parâmetros, documentos tratados e instruções.



Fluxograma simples para o desenvolvimento do assistente

Além disso, é importante considerar os custos associados ao desenvolvimento e uso de um assistente da OpenAI. É útil explicitar como ocorre o cálculo dos custos, além de fornecer a calculadora de tokens para uma estimativa simples de gastos. De fato, para estimar os gastos que a API terá, leva-se em conta as capacidades e o modelo utilizado. No caso, o assistente selecionado foi o GPT-4o, que possui o melhor desempenho e um custo inferior ao do GPT-4, além da capacidade de File Search.

O custo de uso do modelo depende diretamente do gasto em tokens, tanto para a entrada quanto para a saída, tais custos podem ser estipulados usando o tokenizer: Tokenizer - OpenAI Platform - Basta selecionar o modelo específico e adicionar o texto no espaço disponível e o cálculo será realizado. O custo do armazenamento será calculado com base na quantidade de vector storages mantidas por dia. O conteúdo até 1GB é gratuito; acima desse valor, será cobrado $0,10 por GB diariamente.

Abaixo, os principais preços, que também podem ser acessados em: https://openai.com/api/pricing/

Model	Input	Output
gpt-4o	US$ 5,00 / 1M tokens	US$ 15,00 / 1M tokens
gpt-4o-2024-05-13	US$ 5,00 / 1M tokens	US$ 15,00 / 1M tokens

Tool	Input
Code interpreter	$0.03 / session
File Search	$0.10 / GB of vector-storage per day (1 GB free)

3. Segunda Etapa: Limpeza de Dados

Para o desenvolvimento do modelo atual do assistente, os principais documentos utilizados foram a resolução nº 1000 de 2021, fornecida pela ANEEL, e um documento de exemplo de perguntas e respostas, obtido no documento “Word-Athena”. Todos os arquivos foram disponibilizados pela equipe Enpower.

A abordagem que mais apresentou resultados foi a reorganização dos arquivos em formato JSON, juntamente com a remoção de espaços em branco. Outra tentativa realizada foi a lematização, mas essa foi descartada por potencialmente prejudicar a interpretação dos nuances jurídicos presentes no documento da resolução.

Todo o código desenvolvido para a limpeza dos dados está disponível em: https://drive.google.com/drive/u/1/folders/1ntIEO2Hf8sxgLn3lrhvLxmESbpcZEPQ1.
O tratamento desenvolvido nesses códigos envolve a exploração do arquivo da resolução, segmentando-o em Títulos, Capítulos, Seções, Artigos e Parágrafos, nessa hierarquia, garantindo uma organização e fácil acesso para o modelo. Além disso, a parte contratual e de prazos também recebeu um tratamento à parte, para facilitar o acesso pelo modelo, uma vez que não seria possível deixar essa parte da resolução formatada exatamente da mesma forma que a parte anterior.

O tratamento dos documentos de perguntas e respostas também foi realizado neste código, fornecendo o formato JSON e também JSONL, que é o formato ideal para o fine-tuning.

4. Terceira Etapa: Versionamento do Assistente e Testes Realizados

O assistente passou, em seu desenvolvimento, por várias fases que refletem a maturidade do tratamento e da qualidade dos documentos e instruções fornecidas, bem como a do modelo utilizado. No processo de aprimoramento, foram primeiramente utilizados os modelos GPT-3.5, que não apresentaram bons resultados, sendo comum a menção a resoluções antigas, o que evidenciou a necessidade de evitá-los.

Em cada fase do desenvolvimento, o assistente foi testado pelos membros da equipe com uma lista de perguntas fornecidas pelo Leandro, que também desenvolveu um método iterativo para testar a qualidade das respostas das diferentes versões do assistente.

As primeiras tentativas de limpeza de dados envolveram a resolução de 2021 em formato TXT com uma limpeza básica de formatação, em conjunto com o modelo GPT-3.5 turbo. Esse foi o ponto de partida. O desenvolvimento prosseguiu com a limpeza e assimilação dos documentos de perguntas e respostas, além de tentativas de fine-tuning e mudança para o GPT-4 turbo.

Com o GPT-4 turbo, em conjunto com os arquivos organizados, filtrados e limpos em formato JSON, além de melhorias na geração do prompt da instrução, resultados animadores começaram a aparecer. A verdadeira virada de chave foi observada com a utilização do modelo GPT-4o, que conseguiu entregar resultados muito satisfatórios nos testes realizados utilizando o método iterativo fornecido pelo Leandro.

Finalmente, o modelo final do assistente atual é o X11.3, utilizando como documentos base aqueles contidos no vector storage Enpower_VSX_6:

- exemplos.json: Contendo exemplos de perguntas e respostas;
- Resolucao_2021.json: Autoexplicativo;
- saida_contrato.json: JSON contendo o contrato organizado;
- prazos.json: Arquivo que organiza a definição de prazos de pagamento em relação ao contrato em diversas situações definidas no documento.

No processo, também se destacam aqui algumas etapas em que se é indicado basear para a criação de bons prompts para o assistente:

Etapa	Descrição	Exemplo
Persona	Define o público-alvo ou a pessoa para quem a tarefa é direcionada ou idealizada.	Como gerente de produto,
Tarefa	Especifica a atividade ou tarefa a ser realizada.	Desenvolva cinco slogans para uma nova linha de sucos naturais
Etapas	Indica para quem ou onde a tarefa é direcionada.	Direcionado a consumidores de produtos saudáveis.
Contexto	Fornece o contexto ou a situação em que a tarefa será realizada.	O principal ponto de venda do produto é que ele é natural, saboroso e fornece energia de maneira saudável.
Restrições	Lista as limitações ou regras que devem ser	O slogan deve ter de três a sete palavras. Não use a palavra "fresco".

	seguidas ao realizar a tarefa.	
Objetivo	Descreve o objetivo final da tarefa.	O objetivo é transmitir a essência inovadora da nossa marca em um slogan breve e impactante.
Saída	Define o resultado esperado ou a entrega final da tarefa.	Gere pelo menos 5 slogans diferentes em uma lista de marcadores.

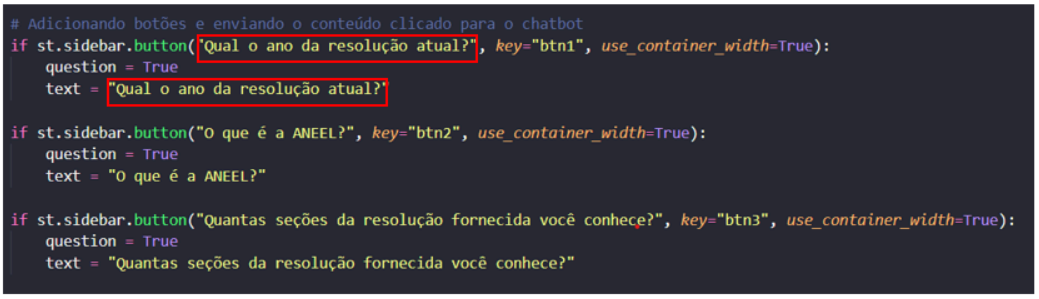
Engenharia De Prompt: 7 Passos Para O Prompt Perfeito | ChatGPT

5. Quarta Etapa: Interface Streamlit

Após a criação do chatbot personalizado, estruturamos uma interface que possibilitasse uma melhor interação entre o usuário e o modelo de chat. Para isso, optamos pela utilização do streamlit, uma biblioteca do python de código aberto que viabiliza a criação de aplicações web de forma mais simplificada e rápida. Por isso, foi necessário a transferência de nosso assistente virtual para o ambiente do streamlit, processo que foi realizado via criação de um código em python.

Desse modo, essa etapa visa explicar os elementos mais importantes do código criado, além de outros elementos no próprio site do streamlit, que viabilizem a manutenção, aprimoramento e personalização do chatbot.

5.1 Personalização dos botões de perguntas frequentes



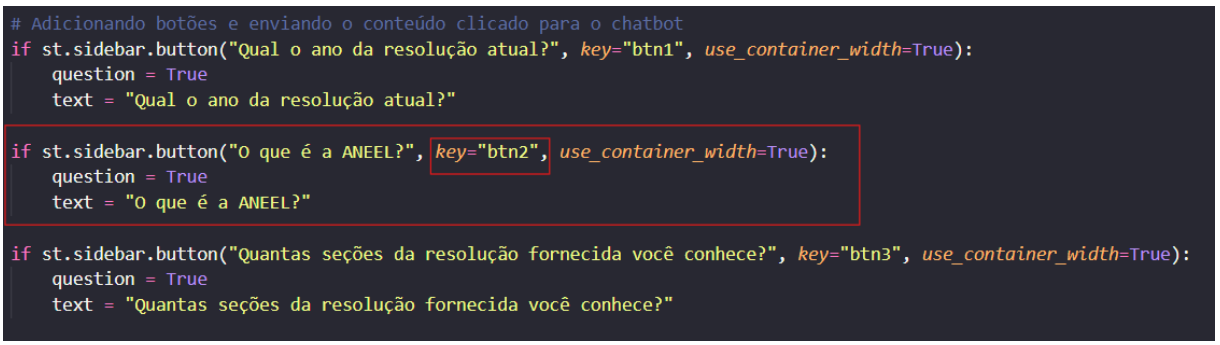
```python
# Adicionando botões e enviando o conteúdo clicado para o chatbot
if st.sidebar.button("Qual o ano da resolução atual?", key="btn1", use_container_width=True):
    question = True
    text = "Qual o ano da resolução atual?"

if st.sidebar.button("O que é a ANEEL?", key="btn2", use_container_width=True):
    question = True
    text = "O que é a ANEEL?"

if st.sidebar.button("Quantas seções da resolução fornecida você conhece?", key="btn3", use_container_width=True):
    question = True
    text = "Quantas seções da resolução fornecida você conhece?"
```

Nessa figura, temos a parte do código que faz referência aos botões de perguntas frequentes. Eles são compostos, basicamente, por dois elementos de texto (string), como pode ser visto destacado em vermelho. A primeira string, faz referência ao conteúdo que será exibido no botão de perguntas frequentes, já o segundo, faz referência ao conteúdo que será encaminhado ao chatbot. Caso queira mudá-los, **basta alterar ambas as strings destacadas em vermelho.**

Por último, caso seja necessário adicionar ou retirar um botão, basta adicionar ou retirar um bloco de código, sendo seu conteúdo tudo o que está dentro do retângulo vermelho maior. O importante é garantir que o valor do parâmetro “key” seja diferente dos já existentes. Exemplo: key=”btn3”.



```python
# Adicionando botões e enviando o conteúdo clicado para o chatbot
if st.sidebar.button("Qual o ano da resolução atual?", key="btn1", use_container_width=True):
    question = True
    text = "Qual o ano da resolução atual?"

if st.sidebar.button("O que é a ANEEL?", key="btn2", use_container_width=True):
    question = True
    text = "O que é a ANEEL?"

if st.sidebar.button("Quantas seções da resolução fornecida você conhece?", key="btn3", use_container_width=True):
    question = True
    text = "Quantas seções da resolução fornecida você conhece?"
```

5.2 Personalização das Instruções e Temperatura

```
run = client.beta.threads.runs.create(
    thread_id=st.session_state.thread_id,
    assistant_id=assistant_id,
    instructions="Você atua como um representante da Agência Nacional de Energia Elétrica. Você responde
    temperature=0.3
)
```

O parâmetro de temperatura (0 < temperature < 1) é fundamental na criação de um chatbot personalizado, pois ele ajusta a aleatoriedade e a criatividade das respostas geradas pelo modelo. Temperaturas mais baixas (por exemplo: 0.2) irão entregar chatbots mais previsíveis e menos criativos, o que é mais adequado para atendimentos aos clientes. Temperaturas mais elevadas já elevam a imprevisibilidade e a criatividade do modelo, o que é indicado para chats de entretenimento, por exemplo. Já as temperaturas médias (por exemplo: 0.5), irão equilibrar essas relações, entregando um modelo mais balanceado nos aspectos de criatividade e imprevisibilidade das respostas.

O outro parâmetro em discussão é o de instruções, que é crucial para garantir que o chat opere de maneira eficiente, forneça respostas precisas e consistentes, e esteja alinhado com os objetivos e a missão da aplicação. Instruções detalhadas ajudam a melhorar a experiência do usuário, aumentar a precisão e confiabilidade do chatbot, e assegurar que ele opere dentro dos limites de segurança e conformidade necessários.

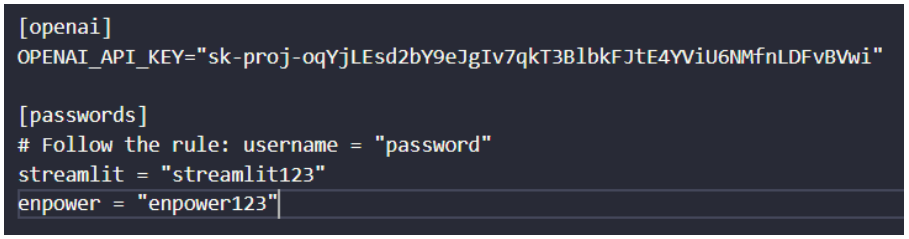
No site da OpenAi, definimos ambos os parâmetros manualmente, que são representados por “instructions” e “temperature” . Assim, para caso seja necessário alterá-los, basta alterar o seu conteúdo.

5.3 OpenAi API Key e Usuários e Senhas

Para modificar esses parâmetros, temos que realizar duas etapas: A primeira consiste em acessar a pasta que contém o arquivo **secrets.toml.** Já a segunda, consiste em mudar efetivamente o valor dos parâmetros.

1) Navegue até pasta .streamlit e acesse o arquivo secrets.toml

2) Agora, iremos modificar efetivamente os parâmetros. Para mudar a OpenAi API Key, basta mudar o valor de seu parâmetro e, para criar um novo usuário e senha, basta seguir a regra especificada na figura abaixo.



5.4 Deployment e outras funcionalidades

A fim de explicar melhor essa etapa de deployment e outros aspectos envolvendo o site do streamlit, elaboramos um vídeo que será entregue juntamente com esse documento e os arquivos do chatbot.

6. Extra: Fine Tuning

O Fine Tuning é um processo de treinamento do modelo LLM (Large Language Model) que serve para aprimorar a qualidade das respostas em um determinado contexto. No projeto, sua principal função é garantir que as respostas fornecidas pelo assistente mencionem sempre os artigos, parágrafos e incisos da legislação, seguindo um modelo semelhante ao fornecido no documento de exemplos de perguntas e respostas (Word-Athena.docx).

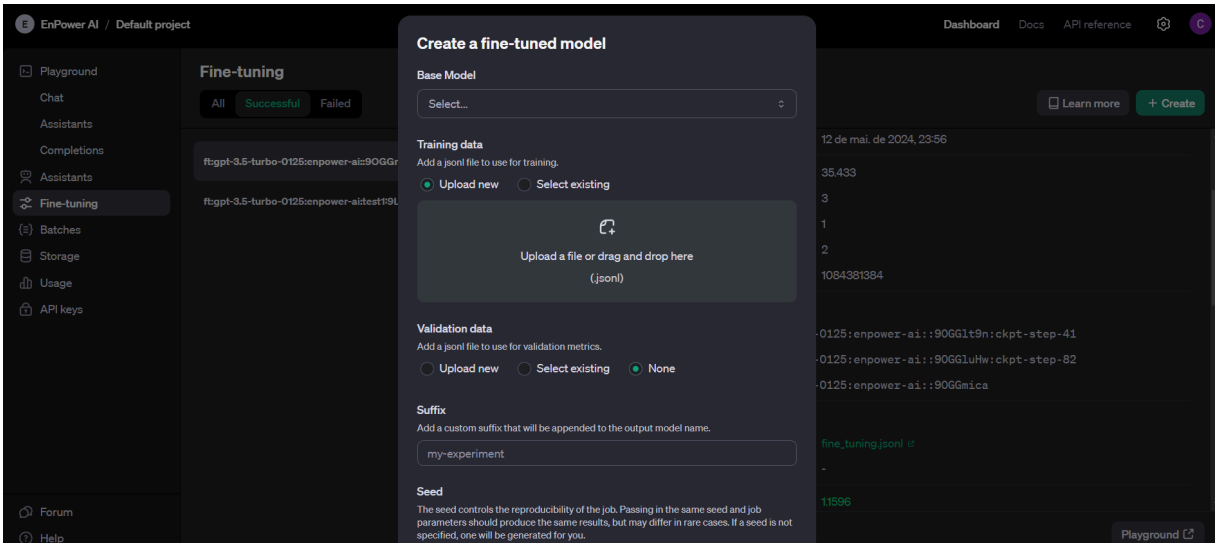
No projeto, o fine tuning foi realizado utilizando dados de treino e de teste formatados em arquivos JSON Lines (jsonl), onde cada linha contém o conteúdo referente à persona do sistema, a pergunta feita pelo usuário e a resposta ideal fornecida pelo assistente. Durante o processo de testes de geração de modelos de fine tuning, o mais promissor foi aquele realizado em 12 de maio de 2024 - ft 3.5-turbo-0125:enpower-ai:test1:9LIJOVzY. Este modelo foi treinado com os dados do arquivo de perguntas e respostas completo, focado totalmente no treino e não na validação.

A seguir, são apresentados alguns exemplos comparativos de respostas geradas pelo chat 3.5 Turbo comum e pelo modelo treinado. Apesar de o modelo treinado gerar respostas com formato semelhante aos exemplos de perguntas e

respostas, seu conteúdo ainda está aquém do desejado por mencionar resoluções inapropriadas.

Pergunta: O que é autoconsumo remoto?	
Gpt 3.5 - Turbo 0125	Fine Tuning
Autoconsumo remoto é um modelo de geração de energia solar em que a energia produzida por um sistema fotovoltaico é consumida em outro local, diferente do local onde a instalação está localizada. Isso é possível graças à regulamentação do setor elétrico que permite que a energia excedente produzida seja injetada na rede elétrica e posteriormente consumida em outro local, beneficiando, por exemplo, condomínios, empresas ou consumidores que possuem mais de uma unidade consumidora. Essa prática contribui para a redução dos custos com energia elétrica e para a sustentabilidade do sistema energético.	O autoconsumo remoto é uma configuração da microgeração ou minigeração distribuída na qual a energia excedente atende a outra unidade consumidora do mesmo titular. Resolução Normativa da ANEEL que define esta configuração: Resolução Normativa nº 482/2012, art. 6º.

Para realizar o fine tuning, basta selecionar a opção correspondente no painel da OpenAI, clicar em “create”, selecionar o modelo base e adicionar o conteúdo do jsonl para treinar o modelo:



O arquivo JsonL deve seguir o modelo seguinte, também disponível no arquivo jsonl utilizado: https://drive.google.com/file/d/1EunB4gTddlSV-SaO4IKZGuv06yuN2UbN/view?usp=drive_link.

{"messages": [{"role": "system", "content": "Enpower GPT é um assistente cordial que responde dúvidas de legislação do fornecimento de energia da ANEEL com base na resolução nº1000/2021."}

Seguindo o modelo acima, dividido nessas seções, o fine tuning pode ser realizado sem problemas.

7. Conclusões e Próximos Passos

Durante o processo de desenvolvimento do assistente, destaca-se a importância da interatividade e dos testes de ferramentas para avaliar sua eficácia. É crucial estabelecer métodos de teste para verificar o desempenho do assistente. Como próximos passos, sugere-se a testagem contínua com perguntas e respostas, preferencialmente envolvendo o usuário final ou um especialista na área, para garantir a qualidade e precisão das respostas fornecidas.

Outro ponto relevante é a capacidade de interpretação de documentos e imagens, que o modelo GPT-4o possui. Esse recurso oferece um grande potencial, especialmente se a segurança do acesso à informação dos usuários e clientes for priorizada.

8. Referências

Para obter mais informações sobre o desenvolvimento e uso de assistentes virtuais, os seguintes recursos podem ser úteis:

1. Overview dos Assistentes - OpenAI API
2. Preços da API da OpenAI
3. Tokenizador - OpenAI Platform
4. Engenharia de Prompt: 7 Passos para o Prompt Perfeito | ChatGPT
5. Como Treinar o ChatGPT com Dados Customizados e Criar seu Próprio Chatbot Usando MacOS
6. Como Treinar o ChatGPT com Seus Próprios Dados
7. Streamlit: Construindo Aplicativos de Conversação
8. Video Tutorial: Como Treinar o ChatGPT (Fine Tuning)