INFNET

Engenharia de Prompts para Ciência de Dados [24E4_4] - AT

Rodrigo Avila - 22/12/2024

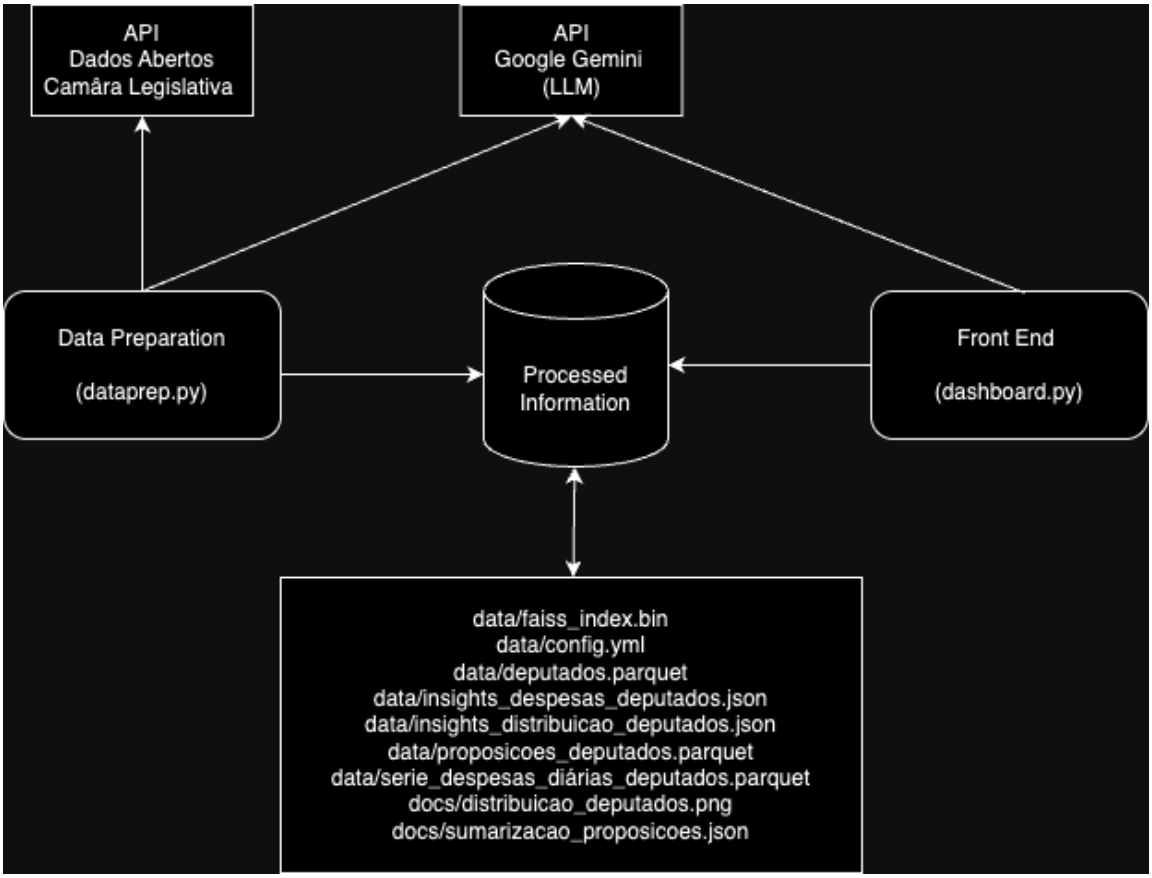
GITHUB: https://github.com/r-moreira/eng-prompt-at

Exercício 1) Arquitetura da Solução

Desenhe a arquitetura da solução com o programa da sua escolha. A arquitetura deve indicar os pontos de processamento de informação, LLMs utilizados, bases de dados (parquets, jsons e faiss), arquivos de configuração (yaml), abas do dashboard e suas funcionalidades.

a) Exporte a arquitetura para o arquivo pdf importado no sistema.

Verificar arquivo `rodrigo_avila_DR4_AT_aquitetura.pdf` na raiz do repositório.



b) Descreva a arquitetura, explicando seus pontos importantes.

Trata-se de um dashboard com informações preprocessadas da API de dados abertos da Câmara Legislativa.

A arquitetura é composta por:

- **API de Dados Abertos da Câmara Legislativa**: Fornecedora de dados brutos sobre atividades parlamentares.
- **Data Prepartion**: Realiza a extração, transformação e disponibilização dos dados brutos em bases de dados estruturadas (parquets, jsons, yaml e faiss).
- **Dashboard**: Interface web para visualização e interação com os dados preprocessados, utilizando streamlit. Possui abas para diferentes funcionalidades.
- **API Google Gemini (LLM)**: Utilizado para geração de código e sumarização de textos (dataprep.py) e para um chatbot de perguntas e respostas (dashboard.py).

c) Descreva o funcionamento de LLMs e como isso pode ser utilizado para atividades de sumarização.

Modelos de Linguagem de Grande Escala (LLMs) são redes neurais treinadas em grandes quantidades de texto para entender e gerar linguagem natural. Eles utilizam arquiteturas como Transformers, que permitem processar e gerar texto de forma eficiente e coerente.

Funcionamento de LLMs

- Pré-treinamento: O modelo é treinado em um grande corpus de texto para aprender padrões de linguagem, gramática, conhecimento factual e até mesmo alguns aspectos de raciocínio.
- Ajuste fino (Fine-tuning): O modelo pré-treinado pode ser ajustado em tarefas específicas, como tradução, resposta a perguntas ou sumarização, usando um conjunto de dados menor e mais específico.
- Inferência: Após o treinamento, o modelo pode gerar texto ou realizar tarefas de linguagem natural com base em novos dados de entrada.

Sumarização com LLMs

O modelo de LLM pode ser utilizado para sumarização, pois ele é capaz de entender e gerar texto de forma coerente e informativa. Inclusive, existem modelos treinados especificamente em um conjunto de dados de pares de texto original e resumo, para aprender a gerar resumos de forma eficiente e precisa.

Exemplo:

```
from transformers import pipeline

summarizer = pipeline("summarization",
model="username/my_awesome_billsum_model")
summarizer(text)
```

referência: https://huggingface.co/docs/transformers/en/tasks/summarization

Exercício 2: Criação de Textos com LLMs

Utilize a sua conta no “poe.com” para gerar um texto curto (2 parágrafos) que explique a Câmara dos Deputados. Execute o mesmo prompt com 3 LLMs diferentes (claude, gemini e chatgpt) e:

a) Explique as vantagens e desvantagens dos três LLMs escolhidos.

Claude:

- Vantagens:
  - Ética e segurança como foco central: Claude é projetado para minimizar respostas que possam ser prejudiciais ou tóxicas, priorizando interações seguras.
  - Foco em compliance: Útil para casos em que a conformidade com regulamentações e padrões éticos é essencial.
- Desvantagens:
  - Menor flexibilidade em tópicos técnicos: Pode não ser tão robusto quanto ChatGPT ou Gemini em tarefas altamente técnicas.
  - Menor capacidade multimodal

Gemini:

- Vantagens:
  - Capacidade multimodal: Gemini pode lidar com entradas de texto e imagem, o que pode ser útil para tarefas que envolvem múltiplos modos de entrada.
  - Flexibilidade em tópicos técnicos: Pode ser mais robusto em termos de conhecimento técnico e especializado.
  - Indicado em encontrar eventos mais recentes
- Desvantagens:
  - Menor foco em compliance: Pode gerar respostas que não são adequadas para ambientes regulamentados ou sensíveis.
  - Em geral tem menos profundidade nas respostas ao se comparar com ChatGPT

ChatGPT:

- Vantagens:
  - Profundidade nas respostas: ChatGPT pode fornecer respostas mais detalhadas e completas em comparação com outros modelos.
  - Versatilidade: É um dos modelos mais amplamente usados, com excelente desempenho em tarefas criativas, técnicas e educacionais.
  - Costuma ser melhor na parte criativa
- Desvantagens:

- Custos de uso: Em geral, pode ser mais caro de usar em comparação com outros modelos.
- Menor foco em compliance: Pode gerar respostas que não são adequadas para ambientes regulamentados ou sensíveis.

b) Argumente sobre a diferença entre a resposta dos 3 LLMs

Claude: Foi o que teve a resposta mais "rasa" e menos detalhada, focando mais em aspectos gerais, sem entrar em muitos detalhes específicos.

Gemini: Apresentou uma resposta mais detalhada e com mais informações sobre a Câmara dos Deputados, incluindo aspectos como a composição, funções e processos.

ChatGPT: Fornecendo uma resposta mais abrangente e detalhada, com uma explicação mais completa sobre a Câmara dos Deputados, acredito que abordou todos os aspectos necessários e importantes.

c) Justifique a escolha da resposta final

Para o "use case" em questão, a melhor resposta foi a do Chat GPT, pois foi uma pergunta bem genérica "Explique a Câmara dos Deputados", e nesse caso, acredito que a profundidade da resposta e o detalhamento é o mais importante.

d) Atualize o prompt do LLM final para gerar um arquivo data/config.yaml com a resposta final (chave: overview_summary).

Obs: verificar arquivo /data/config.yaml

Exercício 3: Processamento dos dados de deputados

a) Colete e salve os dados dos deputados atuais da câmara no arquivo data/deputados.parquet através da url: url_base+/deputados

Verificar src/dataprep.py

b) Executar prompt para criar o código que gere um gráfico de pizza com o total e o percentual de deputados de cada partido, salvo em 'docs/distribuicao_deputados.png

Verificar src/dataprep.py

c) Utilize os elementos de prompts dados, persona e exemplos para instruir o LLM. Explique o objetivo de cada elemento, avalie a resposta e salve-a em data/insights_distribuicao_deputados.json.

O objetivo de cada elemento é:

- Dados: Ao fornecer os dados sobre a distribuição de deputados por partido e estado, o LLM pode usar essas informações para gerar insights e análises mais precisas e relevantes.
- Persona: Ao utilizar um persona, a LLM trás uma resposta especializada, focada em um perfil específico, o que pode ser útil para obter insights mais específicos e detalhados, atendendo melhor a necessidade do usuário.
- Exemplos: Utilizado para informar a LLM como seriam os dados e também o formato que ela deveria responder a pergunta (json).

Com relação a resposta, acredito que foi bem precisa, dentro do padrão esperado, fornecendo informações relevantes sobre a distribuição de deputados por partido. Não notei nenhuma alucionação, a resposta foi um json válido e bem estruturado.

Exercício 4: Processamento dos dados de deputados

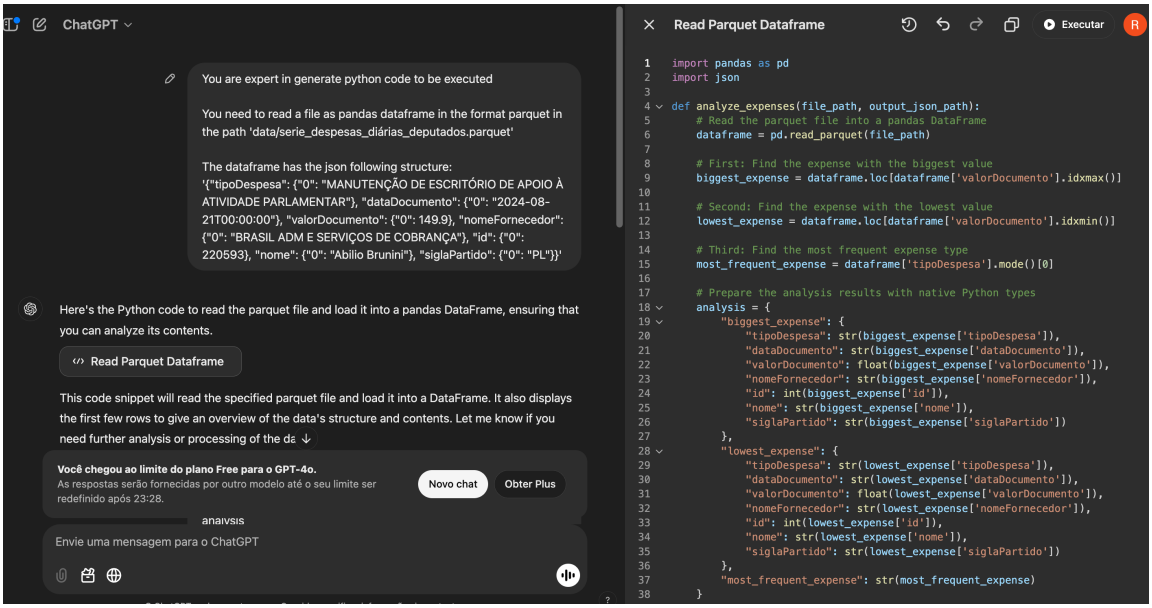
Implemente em dataprep.py uma função que colete as informações das despesas dos deputados atuais da câmara dos deputados no período de referência da solução (use a url: url_base+/deputados/{id}/despesas)

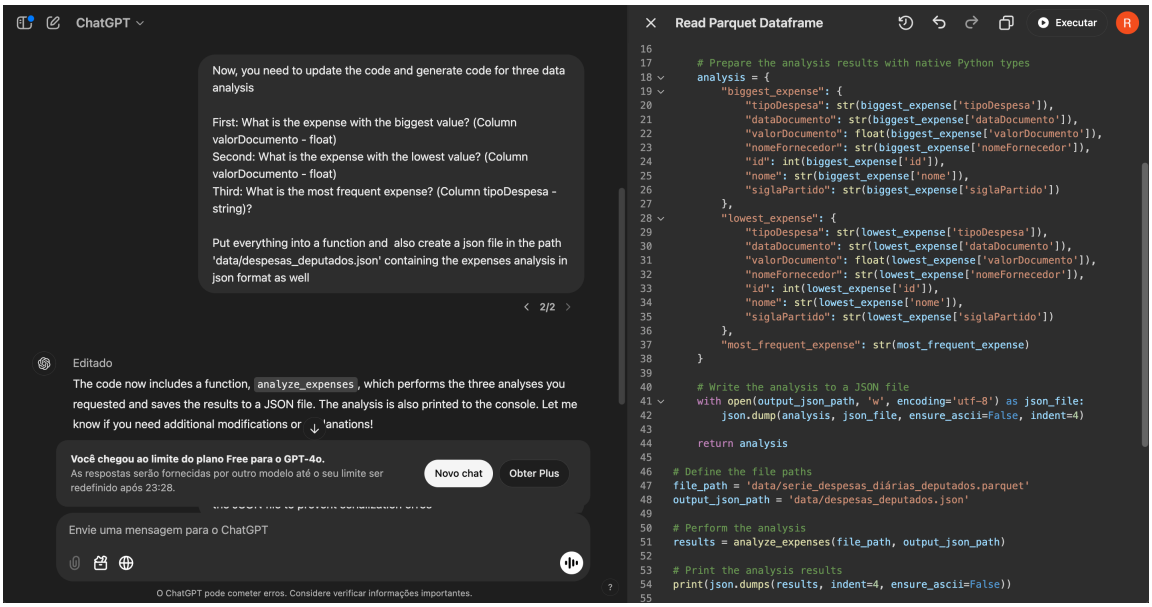
a) Agrupe os dados de despesas por dia, deputado e tipo de despesa e salve num arquivo parquet (data/serie_despesas_diárias_deputados.parquet).

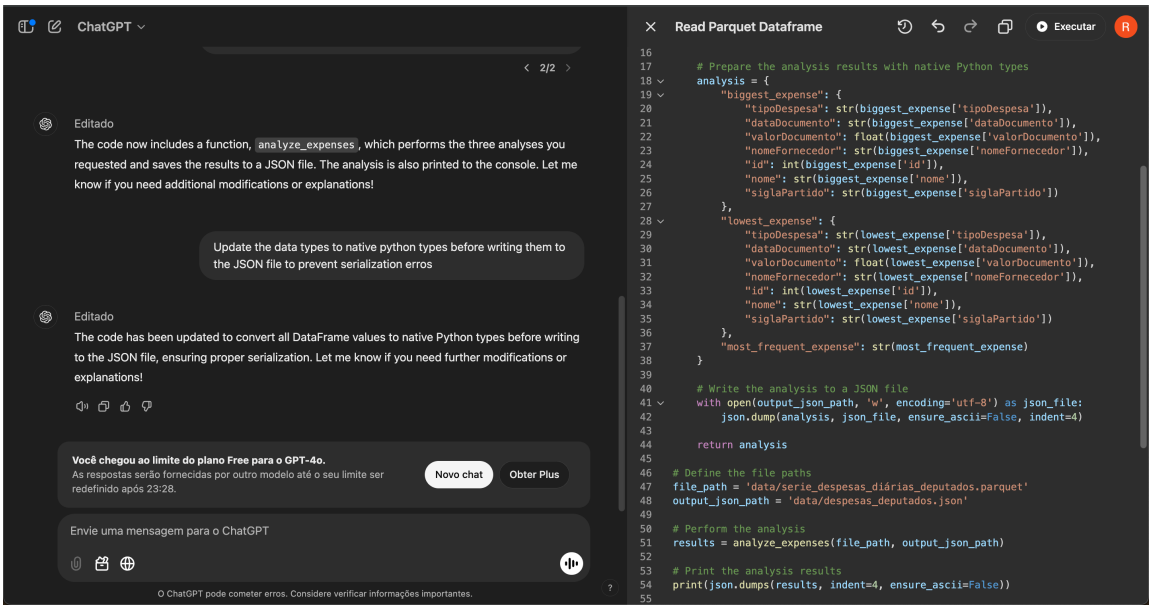
Verificar src/dataprep.py

b) Utilizando a técnica de prompt-chaining, crie um prompt que instrua o LLM a gerar um código python que analise os dados das despesas dos deputados. Peça para o LLM até 3 análises. Indique ao LLM quais dados estão disponíveis e o respectivo arquivo (salvo em a)) e execute as análises.

A ténica prompt-chaining é utilizada para guiar o LLM em uma sequência de prompts, onde cada prompt é baseado na resposta do prompt anterior, em chats onde se tem memória, o LLM consegue revisar o código anterior completamente e então atualizar, como se fosse o prompt-chaining, inclusive, o ChatGPT agora parece ter uma espécie de IDE integrada, bem interessante, cada prompt ele revisa o código anterior completamente e então atualiza, aplicando a técnica de prompt-chaining automaticamente.


ChatGPT
You are expert in generate python code to be executed
You need to read a file as pandas dataframe in the format parquet in the path 'data/serie_despesas_diárias_deputados.parquet'
The dataframe has the json following structure:
'{"tipoDespesa": {"0": "MANUTENÇÃO DE ESCRITÓRIO DE APOIO À ATIVIDADE PARLAMENTAR"}, "dataDocumento": {"0": "2024-08-21T00:00:00"}, "valorDocumento": {"0": 149.9}, "nomeFornecedor": {"0": "BRASIL ADM E SERVIÇOS DE COBRANÇA"}, "id": {"0": 220593}, "nome": {"0": "Abilio Brunini"}, "siglaPartido": {"0": "PL"}}'
Here's the Python code to read the parquet file and load it into a pandas DataFrame, ensuring that you can analyze its contents.
Read Parquet Dataframe
This code snippet will read the specified parquet file and load it into a DataFrame. It also displays the first few rows to give an overview of the data's structure and contents. Let me know if you need further analysis or processing of the da
Você chegou ao limite do plano Free para o GPT-4o.
As respostas serão fornecidas por outro modelo até o seu limite ser redefinido após 23:28.
Novo chat
Obter Plus
anaivsis
Envie uma mensagem para o ChatGPT
Read Parquet Dataframe
Executar
import pandas as pd
import json
def analyze_expenses(file_path, output_json_path):
# Read the parquet file into a pandas DataFrame
dataframe = pd.read_parquet(file_path)
# First: Find the expense with the biggest value
biggest_expense = dataframe.loc[dataframe['valorDocumento'].idxmax()]
# Second: Find the expense with the lowest value
lowest_expense = dataframe.loc[dataframe['valorDocumento'].idxmin()]
# Third: Find the most frequent expense type
most_frequent_expense = dataframe['tipoDespesa'].mode()[0]
# Prepare the analysis results with native Python types
analysis = {
"biggest_expense": {
"tipoDespesa": str(biggest_expense['tipoDespesa']),
"dataDocumento": str(biggest_expense['dataDocumento']),
"valorDocumento": float(biggest_expense['valorDocumento']),
"nomeFornecedor": str(biggest_expense['nomeFornecedor']),
"id": int(biggest_expense['id']),
"nome": str(biggest_expense['nome']),
"siglaPartido": str(biggest_expense['siglaPartido'])
},
"lowest_expense": {
"tipoDespesa": str(lowest_expense['tipoDespesa']),
"dataDocumento": str(lowest_expense['dataDocumento']),
"valorDocumento": float(lowest_expense['valorDocumento']),
"nomeFornecedor": str(lowest_expense['nomeFornecedor']),
"id": int(lowest_expense['id']),
"nome": str(lowest_expense['nome']),
"siglaPartido": str(lowest_expense['siglaPartido'])
},
"most_frequent_expense": str(most_frequent_expense)
}


ChatGPT
Now, you need to update the code and generate code for three data analysis
First: What is the expense with the biggest value? (Column valorDocumento - float)
Second: What is the expense with the lowest value? (Column valorDocumento - float)
Third: What is the most frequent expense? (Column tipoDespesa - string)?
Put everything into a function and also create a json file in the path 'data/despesas_deputados.json' containing the expenses analysis in json format as well
2/2
Editado
The code now includes a function, analyze_expenses, which performs the three analyses you requested and saves the results to a JSON file. The analysis is also printed to the console. Let me know if you need additional modifications or
'anations!
Você chegou ao limite do plano Free para o GPT-4o.
As respostas serão fornecidas por outro modelo até o seu limite ser redefinido após 23:28.
Novo chat
Obter Plus
Envie uma mensagem para o ChatGPT
O ChatGPT pode cometer erros. Considere verificar informações importantes.
Read Parquet Dataframe
Executar
# Prepare the analysis results with native Python types
analysis = {
"biggest_expense": {
"tipoDespesa": str(biggest_expense['tipoDespesa']),
"dataDocumento": str(biggest_expense['dataDocumento']),
"valorDocumento": float(biggest_expense['valorDocumento']),
"nomeFornecedor": str(biggest_expense['nomeFornecedor']),
"id": int(biggest_expense['id']),
"nome": str(biggest_expense['nome']),
"siglaPartido": str(biggest_expense['siglaPartido'])
},
"lowest_expense": {
"tipoDespesa": str(lowest_expense['tipoDespesa']),
"dataDocumento": str(lowest_expense['dataDocumento']),
"valorDocumento": float(lowest_expense['valorDocumento']),
"nomeFornecedor": str(lowest_expense['nomeFornecedor']),
"id": int(lowest_expense['id']),
"nome": str(lowest_expense['nome']),
"siglaPartido": str(lowest_expense['siglaPartido'])
},
"most_frequent_expense": str(most_frequent_expense)
}
# Write the analysis to a JSON file
with open(output_json_path, 'w', encoding='utf-8') as json_file:
json.dump(analysis, json_file, ensure_ascii=False, indent=4)
return analysis
# Define the file paths
file_path = 'data/serie_despesas_diárias_deputados.parquet'
output_json_path = 'data/despesas_deputados.json'
# Perform the analysis
results = analyze_expenses(file_path, output_json_path)
# Print the analysis results
print(json.dumps(results, indent=4, ensure_ascii=False))


ChatGPT
2/2
Editado
The code now includes a function, analyze_expenses, which performs the three analyses you requested and saves the results to a JSON file. The analysis is also printed to the console. Let me know if you need additional modifications or explanations!
Update the data types to native python types before writing them to the JSON file to prevent serialization erros
Editado
The code has been updated to convert all DataFrame values to native Python types before writing to the JSON file, ensuring proper serialization. Let me know if you need further modifications or explanations!
Você chegou ao limite do plano Free para o GPT-4o.
As respostas serão fornecidas por outro modelo até o seu limite ser redefinido após 23:28.
Novo chat
Obter Plus
Envie uma mensagem para o ChatGPT
O ChatGPT pode cometer erros. Considere verificar informações importantes.
Read Parquet Dataframe
Executar
# Prepare the analysis results with native Python types
analysis = {
"biggest_expense": {
"tipoDespesa": str(biggest_expense['tipoDespesa']),
"dataDocumento": str(biggest_expense['dataDocumento']),
"valorDocumento": float(biggest_expense['valorDocumento']),
"nomeFornecedor": str(biggest_expense['nomeFornecedor']),
"id": int(biggest_expense['id']),
"nome": str(biggest_expense['nome']),
"siglaPartido": str(biggest_expense['siglaPartido'])
},
"lowest_expense": {
"tipoDespesa": str(lowest_expense['tipoDespesa']),
"dataDocumento": str(lowest_expense['dataDocumento']),
"valorDocumento": float(lowest_expense['valorDocumento']),
"nomeFornecedor": str(lowest_expense['nomeFornecedor']),
"id": int(lowest_expense['id']),
"nome": str(lowest_expense['nome']),
"siglaPartido": str(lowest_expense['siglaPartido'])
},
"most_frequent_expense": str(most_frequent_expense)
}
# Write the analysis to a JSON file
with open(output_json_path, 'w', encoding='utf-8') as json_file:
json.dump(analysis, json_file, ensure_ascii=False, indent=4)
return analysis
# Define the file paths
file_path = 'data/serie_despesas_diárias_deputados.parquet'
output_json_path = 'data/despesas_deputados.json'
# Perform the analysis
results = analyze_expenses(file_path, output_json_path)
# Print the analysis results
print(json.dumps(results, indent=4, ensure_ascii=False))

Abaixo, execução do código gerado pelo LLM para análise dos dados das despesas dos deputados:

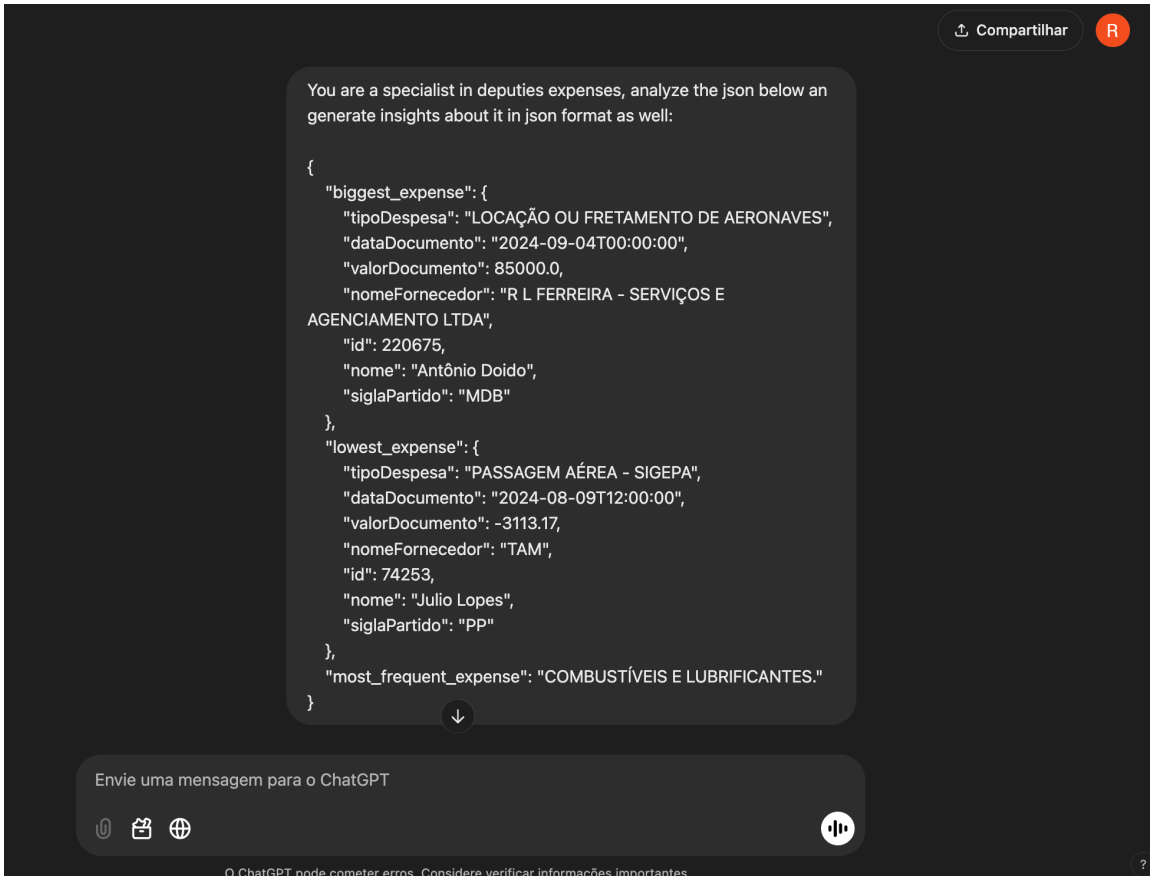
In [5]:
```python
import pandas as pd
import json

def analyze_expenses(file_path, output_json_path):
    # Read the parquet file into a pandas DataFrame
    dataframe = pd.read_parquet(file_path)

    # First: Find the expense with the biggest value
    biggest_expense = dataframe.loc[dataframe['valorDocumento'].idxmax()]

    # Second: Find the expense with the lowest value
    lowest_expense = dataframe.loc[dataframe['valorDocumento'].idxmin()]

    # Third: Find the most frequent expense type
    most_frequent_expense = dataframe['tipoDespesa'].mode()[0]

    # Prepare the analysis results with native Python types
    analysis = {
        "biggest_expense": {
            "tipoDespesa": str(biggest_expense['tipoDespesa']),
            "dataDocumento": str(biggest_expense['dataDocumento']),
            "valorDocumento": float(biggest_expense['valorDocumento']),
            "nomeFornecedor": str(biggest_expense['nomeFornecedor']),
            "id": int(biggest_expense['id']),
            "nome": str(biggest_expense['nome']),
            "siglaPartido": str(biggest_expense['siglaPartido'])
        },
        "lowest_expense": {
            "tipoDespesa": str(lowest_expense['tipoDespesa']),
            "dataDocumento": str(lowest_expense['dataDocumento']),
            "valorDocumento": float(lowest_expense['valorDocumento']),
            "nomeFornecedor": str(lowest_expense['nomeFornecedor']),
            "id": int(lowest_expense['id']),
            "nome": str(lowest_expense['nome']),
            "siglaPartido": str(lowest_expense['siglaPartido'])
        },
        "most_frequent_expense": str(most_frequent_expense)
    }

    # Write the analysis to a JSON file
    with open(output_json_path, 'w', encoding='utf-8') as json_file:
        json.dump(analysis, json_file, ensure_ascii=False, indent=4)

    return analysis

# Define the file paths
file_path = 'data/serie_despesas_diárias_deputados.parquet'
output_json_path = 'data/despesas_deputados.json'

# Perform the analysis
results = analyze_expenses(file_path, output_json_path)
```

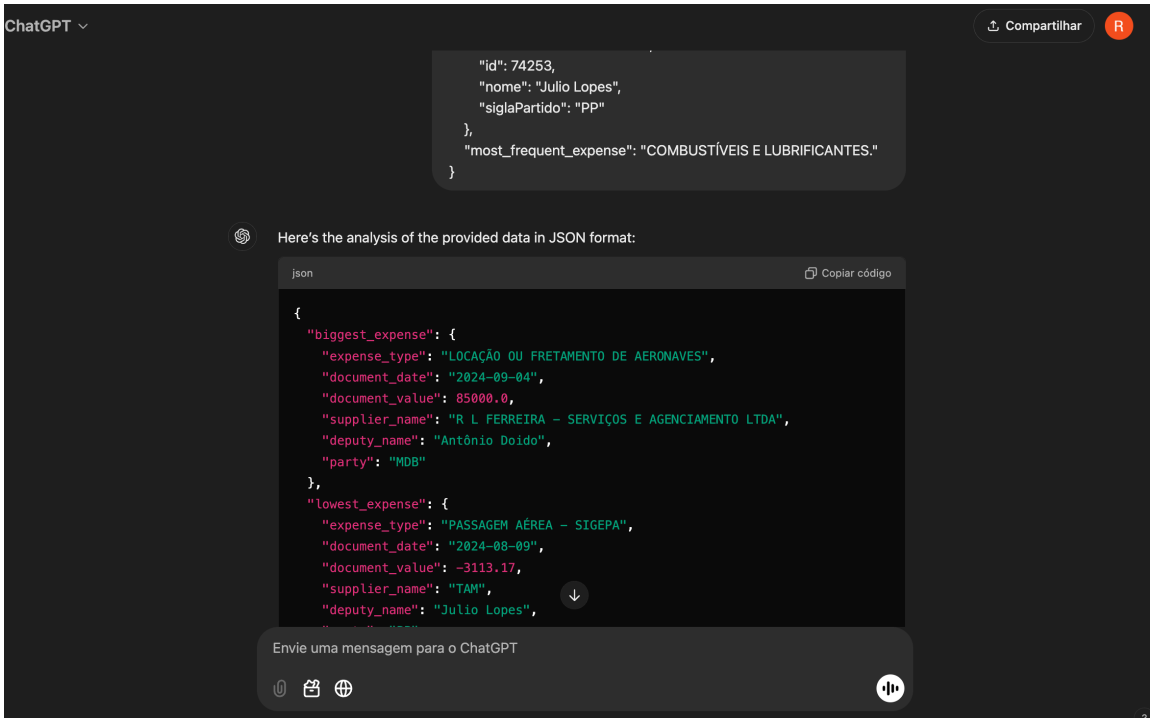
```python
# Print the analysis results
print(json.dumps(results, indent=4, ensure_ascii=False))
```


```
{
    "biggest_expense": {
        "tipoDespesa": "LOCAÇÃO OU FRETAMENTO DE AERONAVES",
        "dataDocumento": "2024-09-04T00:00:00",
        "valorDocumento": 85000.0,
        "nomeFornecedor": "R L FERREIRA - SERVIÇOS E AGENCIAMENTO LTDA",
        "id": 220675,
        "nome": "Antônio Doido",
        "siglaPartido": "MDB"
    },
    "lowest_expense": {
        "tipoDespesa": "PASSAGEM AÉREA - SIGEPA",
        "dataDocumento": "2024-08-09T12:00:00",
        "valorDocumento": -3113.17,
        "nomeFornecedor": "TAM",
        "id": 74253,
        "nome": "Julio Lopes",
        "siglaPartido": "PP"
    },
    "most_frequent_expense": "COMBUSTÍVEIS E LUBRIFICANTES."
}
```


c) Utilize os resultados das 3 análises para criar um prompt usando a técnica de Generated Knowledge para instruir o LLM a gerar insights. Salve o resultado como um JSON (data/insights_despesas_deputados.json).

Verificar a imagem ex4_generated_knowledge_1.png para visualizar o prompt.





Exercício 5) Processamento dos dados de proposições

a) Coletar um total de 10 proposiçoes por tema e salvar em data/proposicoes_deputados.parquet

Verificar src/dataprep.py

b) Utilize a sumarização por chunks para resumir as proposições tramitadas no período de referência. Avalie a resposta e salve-a em data/sumarizacao_proposicoes.json

Verificar src/dataprep.py

Verificar data/sumarizacao_proposicoes.json

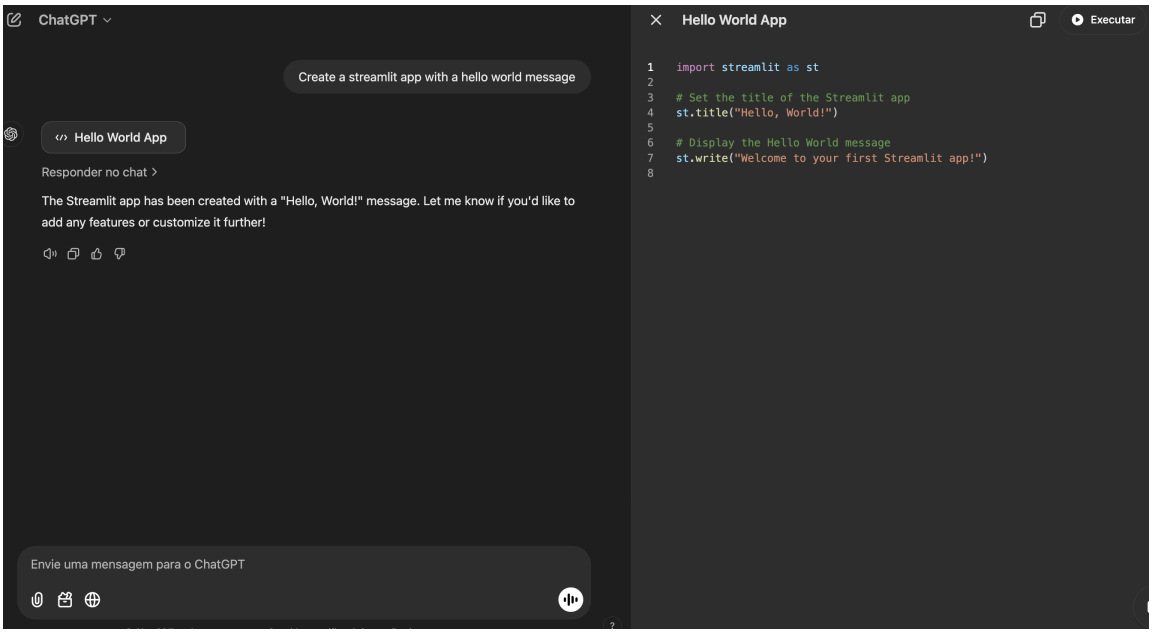
Exercício 6) Dashboards com Chain-of-thoughts

Utilize 3 etapas de Chain-of-Thought prompting para escrever o código inicial do dashboard, destacando as abas Overview, Despesas e Proposições. Explique o objetivo de cada prompt na evolução do código até o arquivo dashboard.py final:

- A aba Overview deve possuir um título e descrição da solução de sua escolha.
- O painel deve mostrar o texto sumarizado em config.yaml
- O painel deve mostrar o gráfico de barras em docs/distribuicao_deputados.png
- O painel deve mostrar os insights do LLM sobre a distribuição de deputados em data/insights_distribuicao_deputados.json

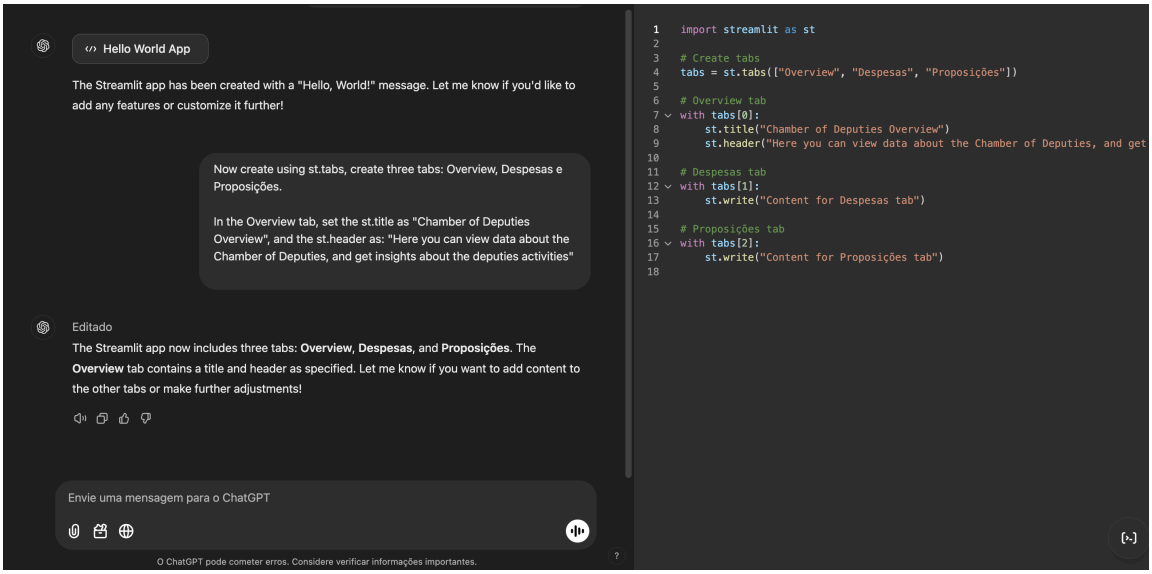
Prompt 1:

O objeto do primeiro prompt foi contextualizar a LLM pedindo para que ela fizesse um código simples e funcional para um app streamlit



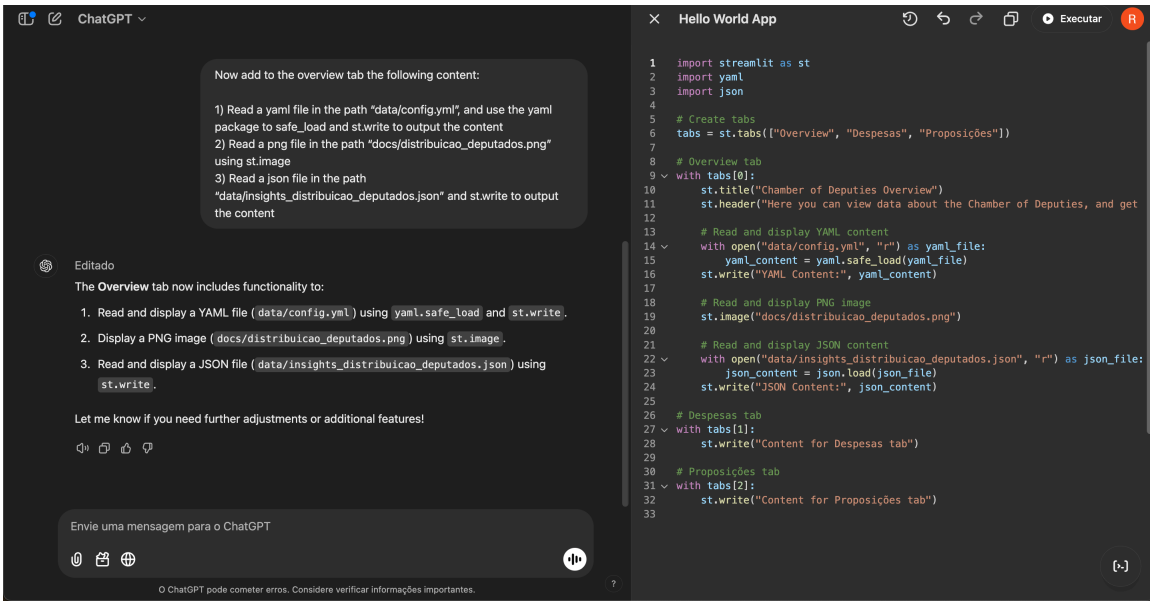
Prompt 2:

O objetivo do segundo prompt foi criar as aba Overview, Despesas e Proposições e adicionar o título e a descrição na Aba Overview



Prompt 3:

O objetivo do terceiro prompot foi adicionar conteúdo para a aba Overview, incluindo os arquivos generados nos exercícios anteriores



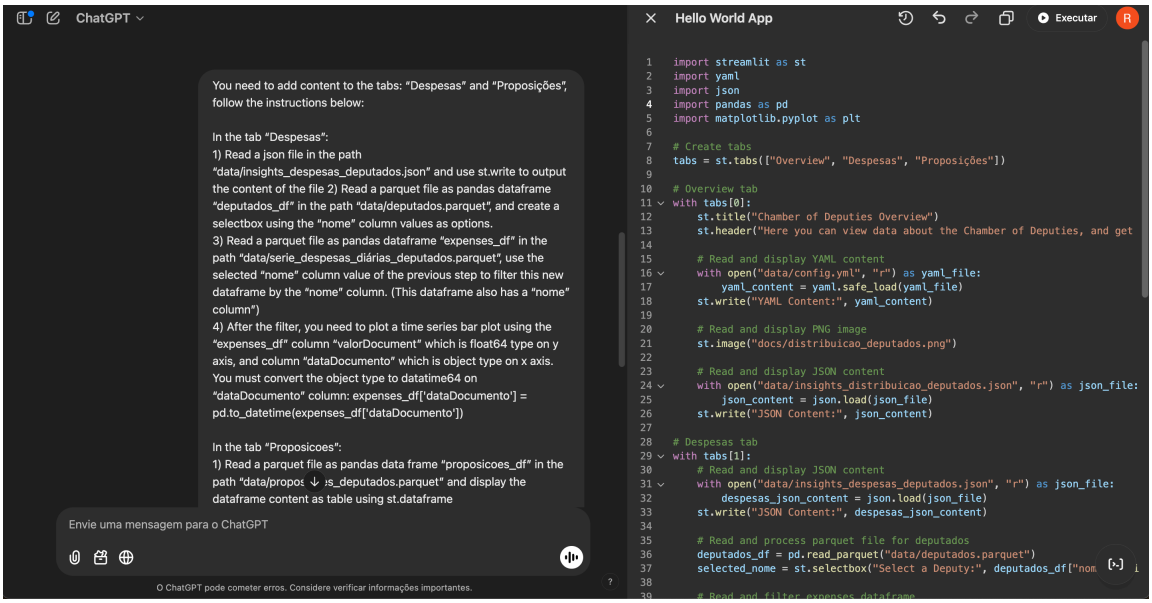
Dessa forma, de maneira incrimental, utilizando a técnica de Chain-of-Thoughts, foi possível criar um dashboard funcional e com informações relevantes para o usuário. O código ficou 100% funcional.

Exercício 7) Dashboards com Batch-prompting

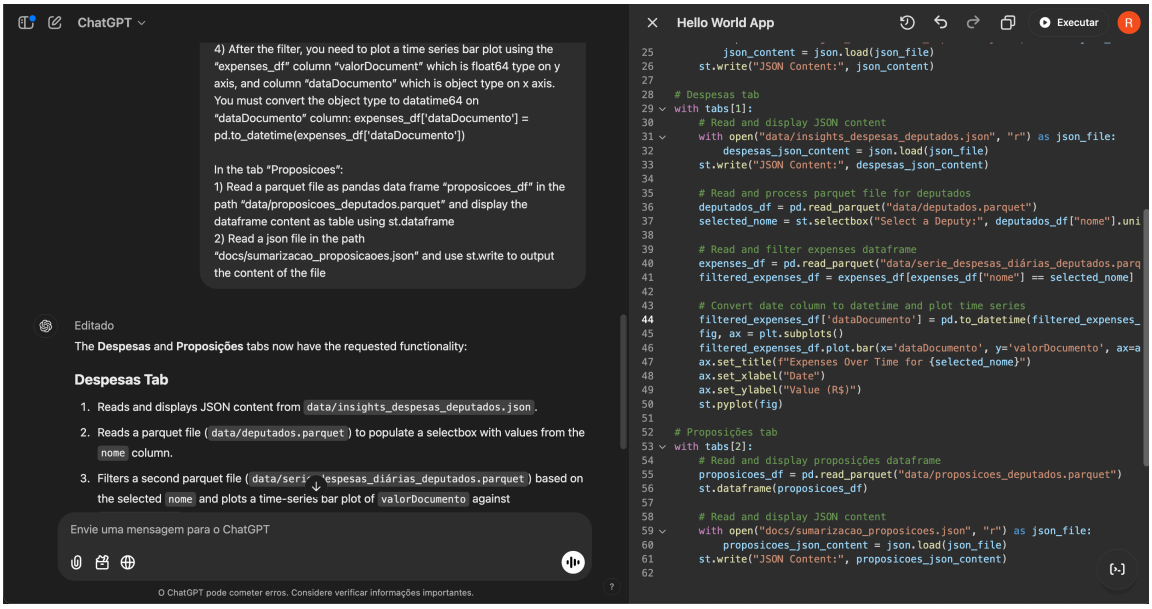
Utilize a técnica de Batch-prompting para escrever o código streamlit que preencha as abas Despesas e Proposições do código em dashboard.py. O prompt deve descrever com detalhes cada aba para geração de:

- Aba Despesas deve mostrar os insights sobre as despesas dos deputados (data/insights_despesas_deputados.json)
- Aba Despesas deve conter um st.selectbox para seleção do deputado.
- Aba Despesas deve mostrar gráfico de barras com a série temporal de despesas do deputado selecionado (data/serie_despesas_diárias_deputados.parquet).
- O painel deve mostrar uma tabela com os dados das proposições (data/proposicoes_deputados.parquet)
- O painel deve mostrar o resumo das proposições em (data/sumarizacao_proposicoes.json)

Prompt (parte 1):



Prompt (parte 2):



Compare o resultado dos códigos gerados pelas técnicas de Chain-of-Thoughts e Batch-prompting

O resultado da técnica Chain-of-Thougts foi 100% funcional, gerando um código que atendeu a todos os requisitos do exercício.

Surpreendentemente, o resultado da técnica Batch-prompting também funcionou perfeitamente, mesmo sendo mais complexo, enviando tudo de uma vez e com o requisito de carregamento de dataframes para gerar plots.

Ambas técnicas são muito eficientes e funcionais, o que muda é a forma de interação com o LLM, onde no Chain-of-Thoughts é de maneira incremental e no Batch-prompting é de maneira simultânea, sendo bem mais complexo, eu demorei bastante tempo para conseguir escrever o Batch-Prompting, acredito que o Chain-of-Thoughts é mais fácil de escrever e mais intuitivo.

Conclusão: Chain-of-Thougts é mais simples e intuitivo, porém depende de mais interações com a LLM. Batch-prompting é mais complexo, porém mais eficiente. Desde que o Batch-Prompting seja feito na medida certa, a diferença no resultado esperado é pequena.

Exercício 8 Assistente online com base vetorial

Adicione ao código da aba Proposições uma interface para chat com um assistente virtual especialista em câmara dos deputados. As informações coletadas dos deputados, despesas e proposições (e suas sumarizações) devem ser vetorizadas usando o modelo "neuralmind/bert-base-portuguese-cased" para armazenamento na base vetorial FAISS. O prompt do sistema para o assistente virtual deve ser feito com a técnica Self-Ask:

> Observação: st.chat_message() e st.chat_input() estava bugando dentro de uma tab (st.tab()), então eu tomei a liberdade de colocar o chat fora da tab, espero que não seja um problema!

a) Explique como a técnica de self-ask pode ser utilizada nesse contexto.

A técnica self-ask pode ser utilizada no contexto atual, pois é uma ténica que auxilia o LLM a responder perguntas complexas, fazendo com que ele mesmo se faça perguntas para obter informações adicionais.

No caso do assistente virtual especialista em câmara dos deputados, a técnica self-ask pode ser utilizada para guiar o LLM a fornecer informações mais detalhadas e relevantes sobre os deputados, despesas e proposições, ajudando a fornecer respostas mais precisas e completas aos usuários.

b) Avalie o resultado do modelo para as seguintes perguntas:

Infezlimente eu costumo fazer meus prompts sempre em inglês, o que acabou gerando os arquivos em inglês e acabou prejudicando na hora da busca vetorial, só percebi esse detalhe agora. De qualquer forma a implementação foi feita.

1) Qual é o partido político com mais deputados na câmara?

O LLM não conseguiu responder essa pergunta, acredito que devido ao fato dos dados em json estarem com as chaves em português

2) Qual é o deputado com mais despesas na câmara?

Apesar de ter essa informação, acredito que a LLM não conseguiu responder porque o json que continha essa informação estava todo em inglês, o que acabou prejudicando na busca vetorial.

3) Qual é o tipo de despesa mais declarada pelos deputados da câmara?

Apesar de ter essa informação, acredito que a LLM não conseguiu responder porque o json que continha essa informação estava todo em inglês, o que acabou prejudicando na busca vetorial.

4) Quais são as informações mais relevantes sobre as proposições que falam de Economia?

A resposta foi bem satisfatória, de todas as proposições que estavam na minha base vetorial, ele citou as duas que tinha haver com economia, acredito que a busca vetorial foi bem sucedida.

Parte da resposta da LLM:

"PL 139/2024

- Ementa: Estabelece normas gerais em contratos de seguro privado e revoga dispositivos do Código Civil, do Código Comercial Brasileiro e do Decreto-Lei nº 73 de 1966.
- Relevância Econômica: Trata de regulamentações no setor de seguros privados, um segmento importante para a economia, por envolver proteção financeira, segurança de investimentos e estabilidade no mercado.

PLP 140/2007

- Ementa: Altera a Lei Complementar nº 101, de 4 de maio de 2000 - Lei de Responsabilidade Fiscal, para suspender temporariamente o pagamento das dívidas assumidas com a União dos Municípios que se encontrem em situação de emergência ou calamidade pública.
- Relevância Econômica: Impacta diretamente a gestão fiscal e financeira dos municípios, influenciando a capacidade de investimento e recuperação econômica

em situações críticas."

O LLM encontrou proposições, porém não havia nenhuma relacionada a economia.

5) Quais são as informações mais relevantes sobre as proposições que falam de 'Ciência, Tecnologia e Inovação'?

Novamente, a resposta foi bem satisfatória, de todas as proposições da minha base vetorial, tinha apenas uma relacionada a Ciência, Tecnologia e Inovação, e a LLM conseguiu encontrar.

Parte da resposta da LLM:

"PL 139/2007

- Ementa: Altera a Lei nº 9.998, de 17 de agosto de 2000, que institui o Fundo de Universalização dos Serviços de Telecomunicações (FUST) para determinar a aplicação de recursos em educação e em ciência e tecnologia.
- Relevância:
  - Envolve a destinação de recursos financeiros do FUST para promover o desenvolvimento em educação e nas áreas de ciência e tecnologia.
  - Contribui para o fortalecimento da infraestrutura tecnológica e científica no país.
  - Estimula a inovação e a formação de profissionais qualificados em áreas estratégicas para o desenvolvimento nacional."

Exercício 9) Geração de Imagens com Prompts

Utilizando as informações sumarizadas das proposições dos deputados, vamos gerar prompts que possam fazer alusão aos temas e o que está sendo proposto. Use o google Colab para gerar imagens com o modelo "CompVis/stable-diffusion-v1-4" para duas proposições de sua escolha. Com essas informações, responda:

a) Descreva o funcionamento dos modelo de imagem, segundo suas arquiteturas, limitações e vantagens:

- Stable Diffusion
- DALL-e
- MidJourney

Stable Diffusion:

É um modelo desenvolvido pela Stability AI, e usa uma abordagem chamada de "difusão estável" para criar imagens à partir de descrições textuais. É conhecido por oferecer um level de controle e especificidade para usuários que desejam fazer uma customização

detalhada das imagens, normalmente resultando em imagens mais coerentes e com mais detalhes, sendo uma das suas principais vantagens.

Inspirado em processos físicos de difusão de particulas e aplicado no contexto de geração de imagens, o modelo inicia com uma imagem ruidosa, e, por meio de várias etapas, remove o ruído de maneira controlada até formar uma imagem final. Diferente de outros modelos, o Stable Diffusion opera em um espaço latente, diferente de outros modelos que operam em um espaço de pixel.

Acredito que o modelo seja adequado para terafas onde é necessário um controle mais detalhado sobre a geração de imagem ou onde o fine-tunning dos elementos é crucial

DALL-e

É um modelo desenvolvido pela Open AI, e também cria imagens à partir de descrições textuais. DALL-e ganhou atenção por sua capacidade de gerar imagens vívidas e com alta qualidade. Sua principal vantagem é permitir que os usuários gerem imagens de alta qualidade a partir de simples prompts narrativos.

Seu destaque é a capacidade de transformar conceitos imaginativos em realidades visuais, sendo o favorito de artístas que buscam dar vida às suas ideias mais extravagantes.

O DALL-E usa o transformer para modelar tokens de texto e imagem como um único fluxo de dados, o que acaba gerando dois problemas:

- Pixels como tokens de imagem ocupam muita memória
- Objetivos de probabilidade priorizam dependências de curto alcance entre pixels

Por tanto, a solução do DALL-E para esses problemas foi criar um processo de treinamento em dois estágios:

1. Aprendizado do Visual Codebook (Utilizando Autoencoders Discretos Variacionais)
2. Aprendizado de Distribuição Prévia (Prior Distribution Learning, que utiliza a architetura Transformer)

O modelo é capaz de gerar imagens de alta qualidade e com grande fidelidade ao prompt, sendo uma excelente opção para tarefas criativas e artísticas.

MidJourney

É desenvolvido por uma equipe de pesquisadores independentes, o MidJourney se destaca por sua interpretações artísticas e a habilidade de misturar estilos e conceitos de maneira única. O modelo é conhecido por sua capacidade de criar imagens altamente estilizadas e com uma estética única, uma das vantagens do modelo é que ele também

pode ser utilizado para identificar objetos em uma imagem e oferecer soluções em diferentes áreas, como publicidade e análise de dados em empresas.

A especificação do MidJourney é proprietária, mas acredita-se que ele seja baseado em uma arquitetura de rede neural profunda, como o GAN, que é capaz de gerar imagens realistas e de alta qualidade.

Ele possui uma particularidade que é funcionar apenas no Discord, o que pode ser uma limitação para alguns usuários, mas para quem utiliza a plataforma, é uma excelente opção para tarefas criativas e artísticas.

b) Utilize diferentes técnicas de “Estilo Visual” e “Composição”, além de exemplos com negative prompting, para gerar 3 versões de imagem para cada proposição e avalie as diferenças entre os resultados (as imagens) e os prompts (as proposições).

Verificar arquivo rodrigo_avila_DR4_AT_geracao_imagens.ipynb

In [1]:
```
!jupyter nbconvert --to webpdf rodrigo_avila_DR4_AT.ipynb
```

```
[NbConvertApp] Converting notebook rodrigo_avila_DR4_AT.ipynb to webpdf
[NbConvertApp] Building PDF
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 3701129 bytes to rodrigo_avila_DR4_AT.pdf
```