# RELATÓRIO TÉCNICO: IMPLEMENTAÇÃO DE AGENTE DE IA PARA CONSULTA DE DADOS ESTRUTURADOS (CSV)

Nome do Grupo: **Grupo 294**

Data: 17/06/2025

Integrantes:

|  |  |  |
| --- | --- | --- |
| **Nome** | **E-mail** | **Telefone** |
| Larissa Garcia | larissagarcialp.lg@gmail.com | +5591981623492 |
| Filipe Mello | fanmello@gmail.com | +5531984204490 |
| Jair Milani | jairmilani21@gmail.com | +5561982745892 |
| Weslley Sbizera | mdisolucoestecnologia@gmail.com | +5513982296448 |
| Samuel | fisica141@gmail.com | +5524988790300 |
| Isis Coelho | isis.coelho@meta.com.br | +5555996718633 |
| Daniel | montanha78@gmail.com | +5511920007175 |
| Eduardo Bessa | eduardobessa10@gmail.com | +5511995091781 |

## Introdução

A crescente geração de dados em formatos estruturados, como arquivos CSV (Comma-Separated Values), apresenta um desafio significativo para usuários não técnicos que necessitam extrair informações valiosas sem o domínio de linguagens de consulta como SQL ou de programação como Python. O objetivo desta atividade foi desenvolver e avaliar um protótipo de software capaz de transpor essa barreira, permitindo a realização de consultas a um conjunto de dados de notas fiscais utilizando linguagem natural.

Para tal, foi empregado um agente de Inteligência Artificial (IA) construído sobre o framework Langchain. Este agente atua como um intérprete, traduzindo a intenção do usuário em código executável para manipulação dos dados e, subsequentemente, formulando uma resposta textual coesa e clara. O presente relatório detalha a arquitetura da solução, as ferramentas empregadas e os resultados obtidos em um ambiente de teste controlado.

## Ambiente e ferramentas utilizadas

A implementação e a execução do protótipo foram realizadas em um ambiente de desenvolvimento em nuvem, com o suporte de bibliotecas especializadas em manipulação de dados e IA.

* **Ambiente de Desenvolvimento:** **Google Colaboratory (Colab)**. Plataforma escolhida por sua facilidade de configuração, acesso a recursos computacionais e ambiente interativo baseado em notebooks, ideal para prototipagem e experimentação.
* **Serviço de LLM:** **OpenRouter**. Utilizado como um gateway para acessar diversos Modelos de Linguagem de Grande Porte (LLMs). Sua principal vantagem é a flexibilidade de alternar entre diferentes modelos (gratuitos e pagos) através de uma única API compatível com o padrão da OpenAI.  
  O modelo escolhido para o teste final foi o Gemma 3 de 27b de parâmetros, com o nome técnico: “google/gemma-3-27b-it:free”.  
  A API Key utilizada está protegida pelo recurso Secrets do Google Colab, para testar basta criar uma API Key e colocalo no ambiente através do Secrets usando o nome “OPENROUTER\_API\_KEY”.
* **Bibliotecas Principais (Python):**
  + **Pandas:** Biblioteca fundamental para a manipulação de dados em Python. Foi utilizada para carregar os arquivos CSV, realizar a conversão de tipos de dados (datas) e, crucialmente, para a fusão (merge) dos dados de cabeçalho e itens das notas fiscais em um único DataFrame.
  + **Langchain:** Framework central da solução. Utilizado para orquestrar a interação entre o LLM, os dados e as ferramentas disponíveis.
  + **langchain\_experimental:** Módulo do Langchain que contém implementações de agentes ainda em fase de avaliação, incluindo o create\_pandas\_dataframe\_agent.
  + **langchain\_openai:** Biblioteca de integração do Langchain que provê a classe ChatOpenAI, utilizada para se comunicar com a API do OpenRouter.

.

### 

## O framework escolhido: Langchain

Langchain é um framework de código aberto projetado para simplificar o desenvolvimento de aplicações que utilizam LLMs. Ele fornece um conjunto modular de abstrações e componentes que permitem construir cadeias (chains) e agentes complexos (LANGCHAIN, 2024).

Para este projeto, o conceito de **Agente** foi central. Diferentemente de uma simples cadeia que segue um fluxo predefinido, um agente utiliza um LLM como um "motor de raciocínio". O agente recebe uma entrada, avalia as ferramentas que possui à sua disposição e decide, passo a passo, qual ferramenta usar para alcançar o objetivo.

O agente implementado segue o paradigma **ReAct (Reasoning and Acting)**. O fluxo de execução de uma consulta no modelo ReAct é o seguinte:

1. **Pensamento (Thought):** O LLM analisa a pergunta do usuário e o estado atual, e raciocina sobre qual deve ser o próximo passo.
2. **Ação (Action):** Com base no pensamento, o LLM decide qual ferramenta utilizar (no nosso caso, a principal ferramenta é um REPL de Python) e qual o comando a ser executado.
3. **Observação (Observation):** O comando é executado pela ferramenta e o resultado (output) é retornado ao LLM como uma observação.
4. **Repetição:** O LLM analisa a observação e retorna ao passo 1, continuando o ciclo até que tenha informação suficiente para formular a resposta final.

Este método permite que o agente resolva problemas que exigem múltiplos passos e a interação com fontes de dados externas, como é o caso da consulta a um DataFrame Pandas.

## Estrutura da solução

A solução foi arquitetada em um fluxo sequencial de quatro etapas principais, desde o tratamento dos dados brutos até a entrega da resposta ao usuário.

1. **Carregamento e pré-processamento de dados**

A primeira etapa consistiu em preparar os dados para consulta. Os arquivos 202401\_NFs\_Cabecalho.csv e 202401\_NFs\_Itens.csv foram carregados em DataFrames Pandas. As colunas contendo datas (DATA EMISSAO, DATA/HORA EVENTO MAIS RECENTE) foram convertidas para o tipo datetime. O passo mais crítico foi a fusão dos dois DataFrames em uma única tabela, utilizando a coluna CHAVE DE ACESSO como chave primária. Essa fusão enriquece o contexto disponível para o agente, permitindo que ele correlacione informações do cabeçalho com os itens de uma mesma nota fiscal.

1. **Instanciação do modelo de linguagem (LLM)**

Foi utilizada a classe ChatOpenAI do Langchain, configurada para apontar para a API do OpenRouter. Nesta etapa, definiu-se o nome do modelo a ser utilizado (por exemplo, mistralai/mistral-7b-instruct ou google/gemma-7b-it), a chave de API e a temperatura do modelo (definida como 0 para respostas mais determinísticas).

1. **Criação do agente**

O agente foi criado utilizando a função create\_pandas\_dataframe\_agent do módulo langchain\_experimental. Os principais parâmetros configurados foram:

* **LLM:** O objeto LLM instanciado na etapa anterior.
* **DataFrame:** O DataFrame Pandas já mesclado e pré-processado.
* **agent\_type:** Definido como AgentType.ZERO\_SHOT\_REACT\_DESCRIPTION. Esta escolha foi motivada por um erro de compatibilidade, pois o endpoint do OpenRouter exige o formato "tools" em detrimento do formato "functions", e o agente ReAct é uma alternativa mais universal.
* **prefix\_prompt:** Um prompt customizado foi elaborado para fornecer ao LLM um contexto detalhado sobre os dados, incluindo os nomes exatos das colunas e o formato de resposta esperado (paradigma ReAct).
* **allow\_dangerous\_code=True:** Parâmetro de segurança que foi explicitamente ativado para permitir que o agente execute o código Python/Pandas gerado pelo LLM.

1. **Ciclo de interação e execução**

A interação com o usuário foi implementada através de um laço while que captura a entrada via input(). A pergunta do usuário é então passada ao método agent.invoke(), que inicia o ciclo ReAct descrito na seção 2. O agente executa os comandos Pandas necessários e, ao final, retorna um dicionário contendo a "output" com a resposta final em linguagem natural.

## Resultados experimentais

Foram submetidas quatro perguntas distintas ao agente para avaliar sua capacidade de interpretação semântica e de geração de código para extração de dados. As respostas abaixo são representações do resultado esperado pelo sistema.

Interface inicial:

--- Agente Pronto para Perguntas (Cabeçalho Ajustado) ---

Use os nomes de coluna fornecidos, como 'RAZÃO SOCIAL EMITENTE', 'VALOR NOTA FISCAL', 'DATA EMISSÃO'.

Exemplos de perguntas:

- Qual a 'RAZÃO SOCIAL EMITENTE' com a maior soma de 'VALOR NOTA FISCAL'?

- Liste as 5 notas com maior 'VALOR NOTA FISCAL'.

- Quantas notas fiscais foram emitidas na 'DATA EMISSÃO' '2024-01-15'?

- Qual o 'VALOR NOTA FISCAL' total para o 'CPF/CNPJ Emitente' '12345678000199'?

- (Se itens foram mesclados) Qual item (use a coluna de nome do produto do arquivo de itens) teve maior quantidade vendida (use a coluna de quantidade do arquivo de itens)?

Digite 'sair' para terminar.

Sua pergunta:



Pergunta 1:

Sua pergunta: Qual a 'RAZÃO SOCIAL EMITENTE' com a maior soma de 'VALOR NOTA FISCAL'

Processando sua pergunta...

**> Entering new AgentExecutor chain...**

***Thought: A pergunta pede para encontrar a 'RAZÃO SOCIAL EMITENTE' com a maior soma de 'VALOR NOTA FISCAL'. Para responder a isso, preciso agrupar o DataFrame pela coluna 'RAZÃO SOCIAL EMITENTE\_cab' e calcular a soma da coluna 'VALOR NOTA FISCAL' para cada grupo. Em seguida, preciso encontrar o grupo com a maior soma.***

***Action: python\_repl\_ast***

***Action Input: ```python***

***df.groupby('RAZÃO SOCIAL EMITENTE\_cab')['VALOR NOTA FISCAL'].sum().idxmax()***

***```EDITORA FTD S.A.A resposta é EDITORA FTD S.A.***

***Final Answer: EDITORA FTD S.A.***

**> Finished chain.**

Resposta do Agente:

EDITORA FTD S.A.

Pergunta 2:

Sua pergunta: Qual é a nota de maior valor?

Processando sua pergunta...

**> Entering new AgentExecutor chain...**

***Thought: A pergunta pede o maior valor da nota fiscal. A coluna que contém o valor da nota fiscal é 'VALOR NOTA FISCAL'. Preciso encontrar o valor máximo nesta coluna.***

***Action: python\_repl\_ast***

***Action Input: `print(df['VALOR NOTA FISCAL'].max())`1292418.75***

***I now know the final answer.***

***Final Answer: 1292418.75***

**> Finished chain.**

Resposta do Agente:

1292418.75

Pergunta 3:

Sua pergunta: Qual item mais vendido?

Processando sua pergunta...

**> Entering new AgentExecutor chain...**

***Thought: A pergunta pede o item mais vendido. Para responder, preciso encontrar a coluna que representa a descrição do produto e a coluna que representa a quantidade vendida. Em seguida, preciso agrupar os dados pela descrição do produto e somar as quantidades vendidas para cada produto. Finalmente, preciso encontrar o produto com a maior quantidade total vendida. As colunas relevantes são 'DESCRIÇÃO DO PRODUTO/SERVIÇO' e 'QUANTIDADE'.***

***Action: python\_repl\_ast***

***Action Input: ```python***

***df.groupby('DESCRIÇÃO DO PRODUTO/SERVIÇO')['QUANTIDADE'].sum().sort\_values(ascending=False)***

***```DESCRIÇÃO DO PRODUTO/SERVIÇO***

***DIPIFARMA INJETAVEL(DIPIRONA MONOIDR 500MG/ML) 2ML 51000.00***

***HGC 4 Branco/Branco-300-CP 33000.00***

***SAPATILHA DESCARTAVEL- COR BRANCA-NAO ESTERIL 23000.00***

***SINVASTATINA 40 MG CP HOSP (G) SINVASTATINA 21600.00***

***(-) TERIFLUNOMIDA 14 MG COM REV CT BL AL AL X 30 (C1) - MARCA: NATCOFARMA (G) 11790.00***

***...***

***2023-AMARRA MEU CADARCO-2020 1.00***

***2023-AS PATAS DA VACA-2020 1.00***

***2023-AS GIRAFAS METIDAS DA MONTANHA-2020 1.00***

***0302030-000 Placa E-Button Ultra Lote: 24545S x1 Val: 22/11/2028 MS: 80083650080 1.00***

***PERSONALIZACAO NOVO PASSAPORTE BRASILEIRO PACOM DPF 32 PAG. 0.09***

***Name: QUANTIDADE, Length: 389, dtype: float64***

***O item mais vendido é "DIPIFARMA INJETAVEL(DIPIRONA MONOIDR 500MG/ML) 2ML", com uma quantidade total vendida de 51000.00.***

***Final Answer: DIPIFARMA INJETAVEL(DIPIRONA MONOIDR 500MG/ML) 2ML***

**> Finished chain.**

Resposta do Agente:

DIPIFARMA INJETAVEL(DIPIRONA MONOIDR 500MG/ML) 2ML

Pergunta 4:

Sua pergunta: Qual o ticket médio das notas fiscais?

Processando sua pergunta...

**> Entering new AgentExecutor chain...**

***Thought: O ticket médio das notas fiscais é calculado dividindo o valor total das notas fiscais pelo número de notas fiscais. A coluna 'VALOR NOTA FISCAL' contém o valor de cada nota fiscal. Para calcular o ticket médio, preciso somar todos os valores da coluna 'VALOR NOTA FISCAL' e dividir pelo número de linhas do DataFrame.***

***Action: python\_repl\_ast***

***Action Input: `df['VALOR NOTA FISCAL'].mean()`33544.27895575221O ticket médio das notas fiscais é 33544.28.***

***Final Answer: 33544.28***

**> Finished chain.**

Resposta do Agente:

33544.28

Pergunta 5:

Sua pergunta: Relacione os 3 produtos que menos venderam

Processando sua pergunta...

**> Entering new AgentExecutor chain...**

***Thought: A pergunta pede para identificar os 3 produtos que menos venderam. Para isso, preciso encontrar a coluna que representa a quantidade vendida de cada produto e ordenar os dados em ordem crescente. A coluna 'QUANTIDADE' parece ser a mais adequada para representar a quantidade vendida. Como o DataFrame contém dados mesclados de cabeçalho e itens, preciso garantir que estou usando a coluna 'QUANTIDADE' dos itens e não do cabeçalho.***

***Action: python\_repl\_ast***

***Action Input: ```python***

***df\_sorted = df.sort\_values(by='QUANTIDADE', ascending=True)***

***print(df\_sorted[['NÚMERO PRODUTO', 'DESCRIÇÃO DO PRODUTO/SERVIÇO', 'QUANTIDADE']].head(3))***

***``` NÚMERO PRODUTO DESCRIÇÃO DO PRODUTO/SERVIÇO \***

***283 1 PERSONALIZACAO NOVO PASSAPORTE BRASILEIRO PACO...***

***552 17 EUP2012X -Cateter Balao Semi-Complacente EUPHO...***

***551 16 EUP2515X -Cateter Balao Semi-Complacente EUPHO...***

***QUANTIDADE***

***283 0.09***

***552 1.00***

***551 1.00***

***Os 3 produtos que menos venderam são:***

***1. PERSONALIZACAO NOVO PASSAPORTE BRASILEIRO PACOTE COM 10 UNIDADES com 0.09 unidades vendidas.***

***2. EUP2012X -Cateter Balao Semi-Complacente EUPHORIA com 1.00 unidade vendida.***

***3. EUP2515X -Cateter Balao Semi-Complacente EUPHORIA com 1.00 unidade vendida.***

***Final Answer: Os 3 produtos que menos venderam são: PERSONALIZACAO NOVO PASSAPORTE BRASILEIRO PACOTE COM 10 UNIDADES (0.09 unidades), EUP2012X -Cateter Balao Semi-Complacente EUPHORIA (1.00 unidade) e EUP2515X -Cateter Balao Semi-Complacente EUPHORIA (1.00 unidade).***

**> Finished chain.**

Resposta do Agente:

Os 3 produtos que menos venderam são: PERSONALIZACAO NOVO PASSAPORTE BRASILEIRO PACOTE COM 10 UNIDADES (0.09 unidades), EUP2012X -Cateter Balao Semi-Complacente EUPHORIA (1.00 unidade) e EUP2515X -Cateter Balao Semi-Complacente EUPHORIA (1.00 unidade).

## Link solução (Colab)

Visando facilitar acesso e utilizar dos excelentes recursos oferecidos pelo Google Colab, nossa solução foi montada neste ambiente e pode ser acessada pelo link que segue:

<https://drive.google.com/file/d/1DI0id5VATbrWb-c8HJsAiA9Ogm2kCKdR/view?usp=sharing>

É importante ter API Key do Openrouter devidamente configurada em Secrets para rodar a solução.

## Conclusão

O protótipo demonstrou com sucesso a viabilidade da utilização de um agente Langchain para realizar consultas em linguagem natural a dados estruturados. A solução abstraiu com eficácia a complexidade da manipulação de dados, fornecendo uma interface intuitiva para o usuário final.

As principais dificuldades encontradas foram relacionadas à compatibilidade de APIs entre Langchain e OpenRouter, especificamente a depreciação de "functions" em favor de "tools", o que exigiu a mudança do tipo de agente para um modelo ReAct mais genérico.

Como trabalhos futuros, sugere-se:

1. **Desenvolvimento de uma Interface Gráfica:** Substituir a interação via input() por uma interface web mais atraente.
2. **Otimização do Agente:** Testar diferentes modelos LLM disponíveis no OpenRouter para avaliar a relação custo/performance/precisão.
3. **Robustez:** Implementar um tratamento de erros mais sofisticado e um mecanismo de cache para consultas repetidas.
4. **Segurança:** Para ambientes de produção, implementar um ambiente de execução de código (sandbox) devidamente isolado para mitigar os riscos associados ao parâmetro allow\_dangerous\_code=True.

## Referências

LANGCHAIN, Inc. **LangChain Documentation**. 2024. Disponível em: <https://python.langchain.com>. Acesso em: 17 jun. 2025.

OPENROUTER. **OpenRouter Documentation**. 2024. Disponível em: <https://openrouter.ai/docs>. Acesso em: 17 jun. 2025.

THE PANDAS DEVELOPMENT TEAM. **pandas-dev/pandas: Pandas**. 2024. Disponível em: <https://pandas.pydata.org/docs/>. Acesso em: 17 jun. 2025.