Laboratório 16: Extração Sistêmica e JSON Mode

Objetivo do Laboratório

Demonstrar na prática que o LLM não serve apenas para "conversar", mas também para processar dados de forma assíncrona. Construiremos um script que lê uma série de reclamações ruidosas e informais de clientes, extraindo informações cruciais e padronizando-as estritamente em formato JSON.

Atividade Prática

1. Configuração do Código e Regras

Acesse o seu repositório Git e crie um arquivo chamado `extrator_json.py`. Você precisará utilizar os recursos de **System Instructions** e forçar o **MimeType** de saída para `application/json`.

2. O Código: Processamento Sistêmico

Crie o seguinte código no seu ambiente:

```
import os
import json
import google.generativeai as genai
from dotenv import load_dotenv

load_dotenv()
genai.configure(api_key=os.getenv("API_KEY"))

# 1. Definindo as Regras de Negócio (System Prompt)
instrucoes = """
Você é um extrator de dados corporativo.
A sua ÚNICA função é extrair Nome, Telefone e o Motivo da Reclamação do texto.
```

```
Você deve retornar ESTRITAMENTE um JSON válido com as chaves:
"nome", "telefone", "reclamacao".
NÃO adicione nenhum texto antes ou depois do JSON. Sem formatação Markdown.
"""

# 2. Configurando o Modelo com JSON Mode e Temperatura Baixa
modelo = genai.GenerativeModel(
    'gemini-1.5-flash',
    system_instruction=instrucoes,
    generation_config=genai.GenerationConfig(
        response_mime_type="application/json",
        temperature=0.1
    )
)

# 3. O Texto Caótico do Usuário
email_cliente = """
Pelo amor de deus, eu não aguento mais! A internet tá caindo toda hora.
Meu nome é João Carlos Silva e eu exijo que consertem isso logo.
Se quiserem falar comigo, me liguem no (11) 98765-4321 de tarde.
"""

# 4. Inferência e Conversão
print("Iniciando extração...")
resposta = modelo.generate_content(email_cliente)

try:
    # Como garantimos que é JSON, podemos converter direto para dicionário Python!
    dados_extraidos = json.loads(resposta.text)

    print("\n--- Objeto Python Resultante ---")
    print(f"Nome do Cliente: {dados_extraidos['nome']}")
    print(f"Contato: {dados_extraidos['telefone']}")
    print(f"Problema: {dados_extraidos['reclamacao']}")
except json.JSONDecodeError:
    print("ERRO: O modelo falhou em gerar um JSON válido.")
    print("Saída bruta:", resposta.text)
```

3. Entrega via Git

1 Execute o arquivo e valide que o retorno textual não conteve nenhuma palavra além do JSON formatado, e que as chaves foram mapeadas perfeitamente para o dicionário Python.

2 Experimente trocar a string `email_cliente` para mensagens ainda mais desorganizadas e comprove a robustez.

3 Adicione o arquivo: `git add extrator_json.py`.

4 Comite: `git commit -m "Implementa extração estruturada via JSON mode"`.

5 Submeta para o GitLab (`git push origin main`).

Critério de Avaliação

A funcionalidade deve realizar o "parse" limpo (`json.loads`) sem quebrar o sistema. Sua solução valida o poder da Engenharia de Prompt para converter linguística humana em tipagem rigorosa de dados.