Relatório do Challenge HP - Sprint 1: Estruturação de Dados

Integrantes do grupo:
Danilo Ramalho Silva | RM: 555183
Israel Dalcin Alves Diniz | RM: 554668
João Vitor Pires da Silva | RM: 556213
Matheus Hungaro | RM: 555677
Pablo Menezes Barreto | RM: 556389
Tiago Toshio Kumagai Gibo | 556984

Link google colab:
https://colab.research.google.com/drive/1Yth6t7Q6CrSCoQfF4aPqnjvsRhLtGWpS?usp=sharing

Estratégia de Extração Estruturada

A estratégia adotada utiliza **Large Language Models (LLMs)**, especificamente o modelo GPT-4o-mini da OpenAI, para transformar HTML bruto de anúncios de cartuchos HP em dados estruturados claramente definidos. A abordagem inclui:

1. **Definição clara dos dados esperados**: Utilização de um modelo tipado ProductInfo com a biblioteca Pydantic que garante a consistência dos dados extraídos.
2. **Prompt estruturado**: Um prompt detalhado de sistema orienta o modelo a extrair informações específicas, tais como título, marca, modelo, preço, cor, avaliações qualitativas e quantitativas, vendedor e classificação de confiança (Original, Pirata ou Suspeito).
3. **Extração Automatizada**: Uso da API OpenAI com a funcionalidade de extração tipada, facilitando a conversão direta da resposta JSON em objetos Python.

Código Desenvolvido

O código em Python realiza as seguintes etapas:

- Define um modelo de dados (ProductInfo) com campos específicos.
- Monta um prompt de sistema detalhado.
- Inicializa o cliente OpenAI com autenticação via chave segura.
- Implementa uma função para extrair dados estruturados automaticamente do HTML fornecido.

Este método permite uma padronização rápida e escalável da extração de informações a partir de diferentes anúncios.

Aplicabilidade e Argumentos para Próximas Etapas

A estruturação eficaz dos dados coletados fornece uma base robusta para as próximas fases do projeto:

- **Classificação de Originalidade**: Com informações estruturadas sobre preço, avaliações qualitativas, quantitativas e confiança, pode-se facilmente treinar modelos de aprendizado supervisionado para classificar produtos como Originais ou Piratas.
- **Análise de Sentimento e Qualidade**: Avaliações qualitativas sobre a descrição e comentários permitem análises adicionais para correlacionar padrões linguísticos com a autenticidade do produto.
- **Automatização e Escalabilidade**: Com esta abordagem, novos anúncios podem ser rapidamente processados, permitindo um fluxo contínuo de dados atualizados, essenciais para alimentar modelos preditivos.

A estratégia adotada garante consistência, eficiência e uma estrutura sólida, fundamental para análises avançadas e para a tomada de decisão informada no projeto HP.

Código

```
import os
from pydantic import BaseModel
from openai import OpenAI
from google.colab import userdata

# 1) Defina o modelo de dados esperado
class ProductInfo(BaseModel):
    titulo: str
    marca: str
    modelo: str
    preco: float
    cor: str
    qualidade_descricao: str
    qualidade_comentarios: str
    avaliacao_revisao: float
    valor_revisao: int
    vendedor: str
    quantidade_reviews: int
    quantidade_fotos: int
    classificacao_confianca: str

# 2) Monte o prompt de sistema
SYSTEM_PROMPT = """
Você recebe o HTML bruto de uma página de anúncio de cartucho HP.
Retorne **somente** um JSON válido com os campos:
- titulo: string
- marca: string (ex: HP)
- modelo: string (ex: 65)
```

```
- preco: número decimal (ex: 120.00)
- cor: string (ex: preto, colorido)
- qualidade_descricao: breve avaliação da clareza/ortografia do texto
(ex: "boa", "média", "ruim")
- qualidade_comentarios: breve avaliação de sentimento dos clientes
(ex: "Satisfeitos", "Indiferentes", "Insatisfeitos")
- avaliacao_revisao: decimal
- valor_revisao: inteiro
- vendedor: string (nome da loja ou responsável)
- quantidade_reviews: inteiro
- quantidade_fotos: inteiro
- classificacao_confianca: string (Original, Pirata ou Suspeito)
"""

# 3) Inicialize o cliente
client = OpenAI(api_key=userdata.get('OPENAI_API_KEY'))

# 4) Função de extração tipada
def extract_structured(html: str) -> ProductInfo:
    completion = client.beta.chat.completions.parse(
        model="gpt-4o-mini",
        messages=[
            {"role": "system", "content": SYSTEM_PROMPT},
            {"role": "user",   "content": html},
        ],
        response_format=ProductInfo
    )
    # .parsed já é uma instância de ProductInfo
    return completion.choices[0].message.parsed
```

```
from typing import List

# 5 modelos de anúncios de cartucho HP
html_examples: List[str] = [raw_html1, raw_html2, raw_html3, raw_html4,
raw_html5 ]

for idx, raw_html in enumerate(html_examples, start=1):
    produto: ProductInfo = extract_structured(raw_html)
    print(f"--- Produto {idx} ---")
    print(produto)
```