

EXPLORANDO A IA GENERATIVA EM UM PIPELINE DE ETL COM PYTHON

O processo será realizado seguindo os seguintes passos:

- Extração: Com Pandas e consumo de uma API REST utilizando o método GET
- Transformação: Fazer a integração com a API do Chat GPT
- Carregamento: Consumo da API REST com o método PUT

O que é uma IA Generativa?

A **Inteligência Artificial Generativa** (IA Generativa), também conhecida como Redes Generativas Adversariais (GANs, do inglês ***Generative Adversarial Networks***), é uma classe de algoritmos de aprendizado de máquina que são usados para gerar novos dados sintéticos que se assemelham a dados de treinamento existentes. As GANs foram introduzidas por Ian Goodfellow e seus colegas em 2014 e se tornaram uma técnica popular em campos como visão computacional, processamento de linguagem natural, e muito mais.

O principal conceito por trás das GANs é a competição entre duas redes neurais, conhecidas como o "gerador" e o "discriminador":

- **Gerador:** O gerador cria amostras de dados sintéticos, como imagens, texto ou áudio, a partir de um ruído inicial. O objetivo do gerador é aprender a criar dados que sejam indistinguíveis dos dados reais.
- **Discriminador:** O discriminador atua como um detector de falsificações, tentando distinguir entre dados reais e dados gerados pelo gerador. Seu objetivo é aprender a diferenciar com precisão entre os dois tipos de dados.

A competição entre o gerador e o discriminador leva a um processo iterativo de treinamento, onde o gerador melhora sua capacidade de criar dados realistas à medida que o discriminador melhora sua capacidade de detectar falsificações. Com o tempo, o gerador é capaz de criar dados sintéticos que são cada vez mais difíceis de serem distinguidos dos dados reais.

As GANs têm uma ampla variedade de aplicações, incluindo a geração de imagens realistas, a criação de música, a geração de texto, a tradução de idiomas e muito mais. Elas também são usadas em pesquisa e desenvolvimento de inteligência artificial, criando dados de treinamento sintéticos quando os dados reais são escassos ou difíceis de obter.

EXPLORANDO O PYTHON PARA CIÊNCIA DE DADOS

Contexto de criação do Pipeline

- Você é um cientista de dados num banco e recebeu a tarefa de envolver seus clientes de maneira mais personalizada. Seu objetivo é usar o poder da IA Generativa para criar mensagens de marketing personalizadas que serão entregues a cada cliente.

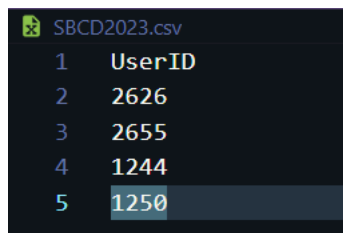
1. **Extract:** Extrair a lista de ID de usuário a partir do arquivo CSV. Para cada ID, fazer uma requisição GET para obter os dados do usuário correspondente.
2. **Transform:** Utilizar a API do Chat GPT para gerar uma mensagem de marketing personalizada para cada usuário.
3. **Load:** Atualizar a lista de *News* de cada usuário na API com a nova mensagem gerada.

CODANDO

Pegar na API os usuários necessários para o pipe.

[\(link\)](#)

Criar um arquivo de extensão csv para indicar a ID dos usuários presentes na API



| | User ID |
|---|---------|
| 1 | 2626 |
| 2 | 2655 |
| 3 | 1244 |
| 4 | 1250 |

EXTRAÇÃO

O primeiro passo da ETL é codar o código de requisição GET para obter todos os dados do usuário para seguir para a etapa de transformação.

```
# O pandas vai fazer a leitura do CSV e criar uma estrutura
import pandas as pd

df = pd.read_csv('SBCD2023.csv') # Indicando ao pandas qual
arquivo ele deve ler.
user_ids = df['UserID'].tolist() # Vai indicar a leitura de
todos os dados que tiver após esse cabeçalho
print(user_ids) # Print de teste de reconhecimento de leitura da
planilha.
```

EXTRAÇÃO: CHAMANDO GET DÁ API BASE

Vai chamar o requests (response) e o json para lidar com a estrutura json da API

```
import requests
import json

# URL BASE
bscd2023_api_url = 'https://sdw-2023-prd.up.railway.app'

def get_user(id):
    response = requests.get(f'{bscd2023_api_url}/users/{id}')
    return response.json() if response.status_code == 200 else
None
```

Em seguida:

Criar a função **get_user** que vai receber uma ID de base formatada que vai receber o link base, que é a URL da API + /users/ID de identificação de cada usuário.

O **return** pede um json se o status_code for = 200 que indica usuário cadastrado com sucesso. Caso contrário (**else**) retorne NONE.

TRANSFORMAÇÃO DOS DADOS EXTRAIDOS

Vai pegar o usuário na lista de ids, usando o conceito de compreensão de listas onde:

- Ele vai pegar o usuário somente se o (id != None) for diferente, ou **is not** None.

```
users = [user for id in user_ids if (user := get_user(id)) is
not None] # Compreensão de listas
```

Basicamente essa expressão vai percorrer cada lista de users e se o resultado for None, ele vai apenas ignorar e não atribuir a lista final de **users**.

Para testar se a compreensão der certo. Usar a função **dumps** do JSON:

```
print (json.dumps(users, indent= 2))
```

UTILIZANDO O CHAT GPT PARA CRIAR UMA MENSAGEM PERSONALIZADA

Instalando a biblioteca da OpenAI para utilizar o Chat GPT:

```
pip install openai
```

Vai instalar toda a gama de API necessárias para fazer a integração.

Passo a Passo:

1. Cria uma conta na OpenAI
2. Acesse a seção "API Keys"
3. Clique em "Create New Secret Key"

Link direto: <https://platform.openai.com/account/api-keys>

Criando a variável que vai comportar a key:

```
openai_api_key = 'substitua esse texto pela chave criada pela sua OpenAI'
```

- Documentação oficial da API OpenAI: <https://platform.openai.com/docs/api-reference/introduction>
- Informações sobre o período gratuito: <https://help.openai.com/en/articles/4936830>

OBS: Há três meses de período gratuito para usar API do chat GPT.

FUNÇÃO DE INTEGRAÇÃO COM A API

Na própria documentação da API existe um modelo em python para a integração:

<https://platform.openai.com/docs/api-reference/chat/create>

CODE:

```
import pandas as pd
import requests
import json
import openai

# URL BASE
bscd2023_api_url = 'https://sdw-2023-prd.up.railway.app'

# Indicando ao pandas qual arquivo ele deve ler.
df = pd.read_csv('SBCD2023.csv')
# Vai indicar a leitura de todos os dados que tiverem após esse cabeçalho
user_ids = df['UserID'].tolist()
print(user_ids) # Print de teste de reconhecimento de leitura da planilha.

def get_user(id):
    response = requests.get(f'{bscd2023_api_url}/users/{id}')
    if response.status_code == 200:
        return response.json()
    else:
        print(f"Erro ao obter usuário {id}: {response.status_code}")
        return None

users = [user for id in user_ids if (user := get_user(id)) is not None] #
# Compreensão de listas

print(json.dumps(users, indent=2))

# INTEGRAÇÃO COM O CHAT GPT
chave_api = 'sua_chave_de_api_aqui'

openai.api_key = chave_api

# FUNÇÃO DE INTEGRAÇÃO
def generate_ai_news(user):
    completion = openai.ChatCompletion.create(
        model='gpt-3.5-turbo-0613',
        messages=[
            {
                "role": "system",
                "content": "Você é um especialista em marketing bancário."
            },
            {
```

CARREGAMENTO DE DADOS OBTIDOS PELA API

Essa parte é responsável por atualizar a lista de 'news' de cada usuário com a nova mensagem gerada.

Basicamente é o mesmo método usado do **get** mas agora com o método **put**.

```
# FUNÇÃO DE ATUALIZAÇÃO
def update_user(user):
    response = requests.put(f"{bscd2023_api_url}/users/{user['id']}",
                             json=user)
    return True if response.status_code == 200 else False

for user in users: # Chamada da função de atualização
    success = update_user(user)
    print(f"{user['name']} updated? {success}")
```

A função **update_user(user)** tem como objetivo atualizar as informações de um usuário em uma API, especificamente na URL **https://sdw-2023-prd.up.railway.app/users/{user['id']}**, onde {user['id']} é o identificador do usuário.

- A função recebe um dicionário **user** como argumento. Esse dicionário deve conter informações sobre o usuário que você deseja atualizar, incluindo seu identificador (geralmente chamado de 'id') e outras informações relevantes.

Uma solicitação HTTP PUT é feita usando a biblioteca requests. O PUT é um método HTTP usado para atualizar informações existentes no servidor.

- A URL da API é construída concatenando a variável `bscd2023_api_url` com o identificador do usuário, o que resulta em algo como `"https://sdw-2023-prd.up.railway.app/users/{user['id']}"`.

A solicitação PUT envia os dados do usuário em formato JSON para a API, usando o dicionário `user` como dados. Isso significa que as informações do usuário serão atualizadas de acordo com os valores fornecidos no dicionário `user`.

- A função verifica a resposta da API. Se o status da resposta for 200 (indicando sucesso), a função retorna **True**. Caso contrário, ela retorna **false**.

O código fora da função itera sobre a lista de usuários (`users`) e chama a função `update_user(user)` para cada usuário. Em seguida, ele imprime uma mensagem indicando

se a atualização do usuário foi bem-sucedida ou não, com base no valor retornado pela função.

No geral, essa função é usada para enviar informações atualizadas de um usuário para uma API específica e verificar se a atualização foi realizada com sucesso, imprimindo um status de sucesso ou falha para cada usuário processado.