Parte 1: Processamento de Dados Eleitorais para Geocodificação

Motivação e Objetivo

Esta primeira etapa visa integrar dados geográficos dos bairros com informações das seções eleitorais do Estado do Ceará, com foco específico na obtenção dos endereços dos locais de votação de Fortaleza. Desta maneira, os endereços formatados servirão como base de dados para uma aplicação de roteirização, simulando uma interface de aplicativo de viagens onde os locais de votação representarão pontos de entrega fictícios.

Visão Geral

Estes primeiros blocos de códigos realiza o pré-processamento de dados eleitorais do Tribunal Regional Eleitoral do Ceará (TRE-CE), com foco na preparação dos endereços dos locais de votação para posterior geocodificação (conversão de endereços textuais em coordenadas geográficas).

Processo de Pré-processamento

1. **Importação e Carregamento de Dados**
    - Carrega o arquivo CSV contendo informações sobre locais de votação do segundo turno das eleições de 2024
    - Utiliza codificação 'latin1' para suportar caracteres especiais do português
    - Ignora as 6 primeiras linhas do arquivo que contêm informações de cabeçalho
2. **Limpeza e Seleção de Dados**
    - Seleciona apenas as colunas relevantes: 'Bairro', 'Endereço', 'Município' e 'CEP'
    - Remove registros com valores ausentes
    - Padroniza o formato do CEP, convertendo para string e garantindo 8 dígitos
3. **Formatação de Endereços**
    - Cria uma nova coluna 'Endereço completo' concatenando as informações de endereço, bairro e município
    - Adiciona 'CE, Brasil' ao final para melhorar a precisão da geocodificação

```
import pandas as pd
from geopy.extra.rate_limiter import RateLimiter
from geopy.geocoders import Nominatim

# 1. Pré-processamento de informações
path = '/content/drive/MyDrive/CC0464/tre-ce-eleicoes-2024-locais-de-votacao-por-municip
```

```
sections_df = pd.read_csv(path, encoding='latin1', sep=';', quotechar='"', skiprows=6)
sections_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 670 entries, 0 to 669
Data columns (total 12 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   Zona                                  670 non-null    object 
 1   Cod. Município                        669 non-null    float64
 2   Município                             669 non-null    object 
 3   Cod. Local                            670 non-null    int64  
 4   Local de Votação                      669 non-null    object 
 5   Endereço                              669 non-null    object 
 6   Bairro                                669 non-null    object 
 7   CEP                                   669 non-null    float64
 8   Qtd. Total de Seções                  670 non-null    int64  
 9   Qtd. Seções Agregadas                 670 non-null    int64  
 10  Qtd. Seções Principais (com urna)     670 non-null    int64  
 11  Tot Eleitores Aptos Eleição           670 non-null    int64  
dtypes: float64(2), int64(5), object(5)
memory usage: 62.9+ KB
```

```
## Exclusão das colunas que não serão usadas no projeto
sections_df = sections_df[['Bairro', 'Endereço', 'Município', 'CEP']]
sections_df = sections_df.dropna()
sections_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 669 entries, 0 to 668
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Bairro     669 non-null    object 
 1   Endereço   669 non-null    object 
 2   Município  669 non-null    object 
 3   CEP        669 non-null    float64
dtypes: float64(1), object(3)
memory usage: 26.1+ KB
```

```
# Convertendo a coluna 'CEP' para a formatação do tipo string e formatada corretamente
sections_df['CEP'] = sections_df['CEP'].astype(str).str.zfill(8)

# Criação de uma nova coluna com o endereço completo
sections_df['Endereço completo'] = sections_df[['Endereço', 'Bairro', 'Município']].agg(
sections_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 669 entries, 0 to 668
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Bairro             669 non-null    object
 1   Endereço           669 non-null    object
 2   Município          669 non-null    object
 3   CEP                669 non-null    object
 4   Endereço completo  669 non-null    object
dtypes: object(5)
memory usage: 31.4+ KB
```

Parte 2: Geocodificação de Endereços

Visão Geral

Estes blocos de códigos abaixo, implementam um processo de geocodificação para converter endereços textuais em coordenadas geográficas (latitude e longitude). O processo utiliza a biblioteca Nominatim para realizar as consultas de geocodificação e implementa controles de taxa para evitar sobrecarregar o serviço.

Fluxo de Operação

1. Inicializa o geocodificador Nominatim com um timeout de 10 segundos
2. Configura um limitador de taxa (RateLimiter) com atraso de 1,44 segundos entre requisições
3. Itera através de endereços armazenados em um DataFrame
4. Para cada endereço, tenta obter suas coordenadas geográficas
5. Armazena as coordenadas resultantes em um dicionário
6. Lida com casos onde a geocodificação falha, armazenando valores nulos
7. Exibe informações de progresso durante a execução

Observações

- O código inclui tratamento de exceções para garantir que o processo continue mesmo quando ocorrem erros
- O progresso é monitorado com contadores e mensagens informativas a cada 10 endereços processados
- O tempo total de processamento dependerá da quantidade de endereços devido ao atraso intencional entre requisições

```
def geocodificador(sections_df:pd.DataFrame):

    # Inicialização do geocodificador
    geolocator = Nominatim(user_agent="meu_app_geocoder", timeout=10)

    # Configuração do RateLimiter para adicionar um atraso entre as requisições.
    geocode = RateLimiter(geolocator.geocode, min_delay_seconds=1.44)

    # Dicionário com as listas de latitudes e de longitudes coletadas a partir da geocod
    coordenadas:dict = {
        'Lat': list(),
        'Lon': list()
    }

    # Tracking do progresso.
    total = len(sections_df)
    print(f"Iniciando geocodificação de {total} endereços...")
```

```
    # Aplicação da geocodificação
    for i, endereco in enumerate(sections_df['Endereço completo']):
        try:
            localizacao = geocode(endereco)
            if localizacao:
                coordenadas['Lat'].append(localizacao.latitude)
                coordenadas['Lon'].append(localizacao.longitude)
            else:
                coordenadas['Lat'].append(None)
                coordenadas['Lon'].append(None)
        except Exception as e:
            print(f"Erro ao geocodificar '{endereco}': {e}")
            coordenadas['Lat'].append(None)
            coordenadas['Lon'].append(None)

        # Print de progresso
        if (i+1) % 10 == 0 or (i+1) == total:
            print(f"Processado {i+1}/{total} endereços...")

    return coordenadas


if __name__=="__main__":
    coordenadas = geocodificador(sections_df)
    sections_df['Lat'] = coordenadas['Lat']
    sections_df['Lon'] = coordenadas['Lon']
    sections_df.to_csv('/content/drive/MyDrive/CCO464/coordenadas.csv', index=False)
    print("Geocodificação concluída!")
```

```
Processado 110/669 endereços...
Processado 120/669 endereços...
Processado 130/669 endereços...
Processado 140/669 endereços...
Processado 150/669 endereços...
Processado 160/669 endereços...
Processado 170/669 endereços...
Processado 180/669 endereços...
Processado 190/669 endereços...
Processado 200/669 endereços...
Processado 210/669 endereços...
Processado 220/669 endereços...
Processado 230/669 endereços...
Processado 240/669 endereços...
Processado 250/669 endereços...
Processado 260/669 endereços...
Processado 270/669 endereços...
Processado 280/669 endereços...
Processado 290/669 endereços...
Processado 300/669 endereços...
Processado 310/669 endereços...
```

```
Processado 430/669 endereços...
Processado 440/669 endereços...
Processado 450/669 endereços...
Processado 460/669 endereços...
Processado 470/669 endereços...
Processado 480/669 endereços...
Processado 490/669 endereços...
Processado 500/669 endereços...
Processado 510/669 endereços...
Processado 520/669 endereços...
Processado 530/669 endereços...
Processado 540/669 endereços...
Processado 550/669 endereços...
Processado 560/669 endereços...
Processado 570/669 endereços...
Processado 580/669 endereços...
Processado 590/669 endereços...
Processado 600/669 endereços...
Processado 610/669 endereços...
Processado 620/669 endereços...
Processado 630/669 endereços...
Processado 640/669 endereços...
Processado 650/669 endereços...
Processado 660/669 endereços...
Processado 669/669 endereços...
Geocodificação concluída!
```

Parte 3: Visualização de Dados Geográficos de Fortaleza

Visão Geral

Este código implementa uma visualização geográfica dos locais de votação de Fortaleza, utilizando as bibliotecas Folium e GeoPandas para criar um mapa interativo com os limites dos bairros e a localização dos pontos de votação.

Etapas do Processo

1. Configuração do Mapa Base

- Define os limites geográficos de Fortaleza usando coordenadas de latitude e longitude
- Calcula o centro do mapa para posicionamento inicial
- Cria um mapa Folium usando os tiles do CARTO para uma visualização clara
- Configura parâmetros de visualização como zoom inicial e limites de navegação

2. Restrição de Limites

- Aplica restrições para que o usuário não navegue para muito além dos limites da cidade
- Utiliza JavaScript para garantir que os limites definidos sejam estritamente respeitados
- Configura a propriedade `maxBoundsViscosity` para criar um efeito de "parede" nos limites

3. Integração dos Dados Geográficos dos Bairros

- Carrega o arquivo GeoJSON contendo os polígonos dos bairros de Fortaleza
- Adiciona os contornos dos bairros ao mapa com estilo personalizado
- Usa linhas finas em cinza para demarcar os bairros sem preencher suas áreas

4. Adição dos Pontos de Votação

- Carrega o arquivo CSV contendo as coordenadas dos locais de votação (gerado no processo anterior)
- Filtra os dados para remover registros sem coordenadas geográficas
- Adiciona marcadores para os primeiros 30 locais de votação
- Configura cada marcador com informações detalhadas do endereço no popup e tooltip

Resultados

O resultado é um mapa interativo que apresenta:

- Os limites geográficos de Fortaleza
- A divisão dos bairros da cidade
- A localização dos locais de votação como pontos interativos
- Informações detalhadas sobre cada local ao passar o mouse ou clicar nos marcadores

Esta visualização serve como parte da interface de um sistema de roteirização, permitindo visualizar espacialmente os pontos de entrega (representados pelos locais de votação).

```
import geopandas as gpd
import pandas as ps
import folium
from branca.element import Element


# 1. Definindo limites geográficos de Fortaleza
min_lat, max_lat = -3.8890, -3.6880
min_lon, max_lon = -38.636, -38.401

# 2. Caixa de limites para fit_bounds
bounds = [[min_lat, min_lon], [max_lat, max_lon]]

# 3. Calculando o centro inicial do mapa
center_lat = (min_lat + max_lat) / 2
center_lon = (min_lon + max_lon) / 2

# 4. Criando o mapa com Folium, usando tiles do CARTO e limitando os bounds
m = folium.Map(
    location=[center_lat, center_lon],
    zoom_start=12,
    min_zoom=10.8,
    tiles='https://{s}.basemaps.cartocdn.com/rastertiles/voyager/{z}/{x}/{y}{r}.png',
    attr='© CARTO contributors',
    max_bounds_viscosity=1.0,
)

# 5. Ajustando o mapa para encaixar exatamente nos limites definidos
m.fit_bounds(bounds)

# 2) Gera script para tornar os limites sólidos
```

```
map_var = m.get_name()  # ex: "Map_1234567890abcdef"
js = f"<script>{map_var}.options.maxBoundsViscosity = 1.0;</script>"
m.get_root().html.add_child(Element(js))

# 6. Salvando em HTML (ou exiba diretamente em um notebook Jupyter)
# m.save('mapa_fortaleza.html')
# no Jupyter, basta: m
# m
```

```
<branca.element.Element at 0x7e90a55fe790>
```

```
# Abrindo GeoJson dos bairros de Fortaleza
arquivo_geojson_path = "/content/drive/MyDrive/CC0464/Bairros_de_Fortaleza.geojson"
bairros = gpd.read_file(arquivo_geojson_path)
bairros.info()
```

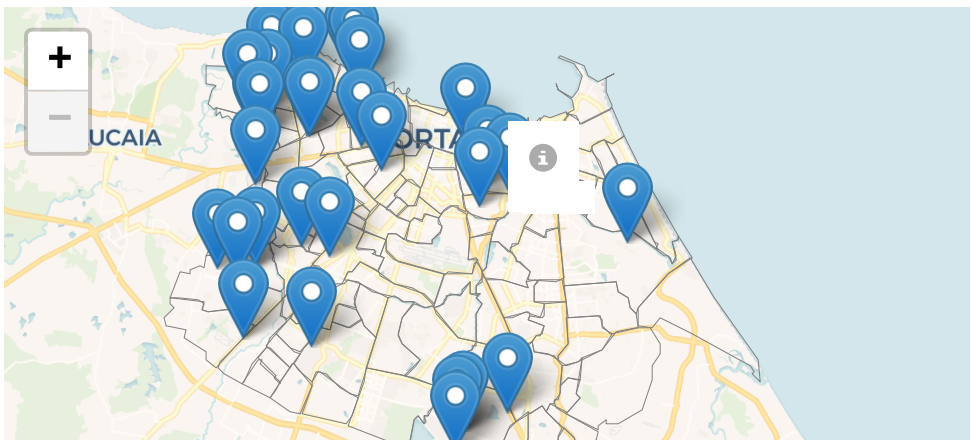
```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 121 entries, 0 to 120
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 121 non-null    int32
 1   Nome               121 non-null    object
 2   Código do  IBGE    121 non-null    float64
 3   Código do  Bairro  121 non-null    int32
 4   Área (ha)          121 non-null    float64
 5   Regional Antiga    121 non-null    object
 6   Regional Atual     121 non-null    object
 7   Código da  Região  121 non-null    object
 8   Território         121 non-null    int32
 9   Fonte              121 non-null    object
 10  ano_ref            121 non-null    int32
 11  data               121 non-null    datetime64[ms]
 12  epsg_codif         121 non-null    object
 13  geometry           121 non-null    geometry
dtypes: datetime64[ms](1), float64(2), geometry(1), int32(4), object(6)
memory usage: 11.5+ KB
```

```
# Tentando vizualizar bairros de Fortaleza
folium.GeoJson(
    data=bairros['geometry'],
    name='Bairros',
    style_function=lambda feature: {
        'fillOpacity': 0,
        'color': 'gray',
        'weight': 0.45,
    }
).add_to(m)
m
```


```
# Abrindo o arquivo gerado na etapa anterior
coordenadas = ps.read_csv('/content/drive/MyDrive/CC0464/coordenadas.csv')
# Eliminando as linhas sem informações da latitude e longitude
coordenadas = coordenadas.dropna(subset=['Lat', 'Lon']).reset_index(drop=True)


# Testando a adição de marcações no mapa (Distribuidora)
folium.Marker(
    location=(-3.757541, -38.488726),
    popup='Magalu',
    tooltip='Magalu',
    icon=folium.Icon(color='red', icon='info-sign')
).add_to(m)

# Testando a adição de marcações no mapa (Entregas)
for i in range(50):
    folium.Marker(
        location=(coordenadas['Lat'][i], coordenadas['Lon'][i]),
        popup=coordenadas.loc[i, 'Endereço completo'],
        tooltip=coordenadas.loc[i, 'Endereço completo']
    ).add_to(m)
m
```



Referências

- **Limites dos bairros de Fortaleza**: GeoJSON contendo a delimitação dos bairros da cidade de Fortaleza, disponível no portal oficial de mapas da Prefeitura: https://mapas.fortaleza.ce.gov.br/?view=21
- **Seções eleitorais de Fortaleza (2024)**: Dados atualizados sobre os locais de votação e respectivas seções eleitorais de Fortaleza para o ano de 2024, disponibilizados pelo

Tribunal Regional Eleitoral do Ceará (TRE-CE): https://www.tre-ce.jus.br/eleicao/eleicoes-2024/consultas/locais-de-votacao-por-municipio-2t

- **Localização da cidade de Fortaleza**: Limites para geração do mapa de Fortaleza: https://data.maptiler.com/downloads/south-america/brazil/fortaleza/