Proyecto Terminal 1, Trimestre: 25-I, 2025

Caracterización de datos de trayectorias individuales

Martínez Buenrostro Jorge Rafael.

Universidad Autónoma Metropolitana
Unidad Iztapalapa, México
*molap*96@*gmail.com*

Resumen: Este documento describe el proyecto terminal 1 cuyo objetivo es caracterizar datos de trayectorias individuales utilizando un conjunto de herramientas y técnicas de programación.

1. Introducción

1.1. Descripción general del proyecto

La simulación de una red de comunicaciones en donde intervienen dispositivos personales de comunucación requiere contar con modelos que representen fielmente los patrones de movimiento de las personas. De otra manera, la utilidad de las conclusiones que se puedan obtener de esa simulación es limitada.
Para avanzar hacia la definición de un modelo de trayectorias individuales, se propone la caracterización de los datos de una base de datos para poder modelar trayectorias eficazmente.

1.2. Objetivos y propósitos

El objetivo principal del proyecto es contar con una caracterización de las trayectorios. Identificando las características estadísticas que las componen.
Los propósitos del proyecto son:

- Caracterizar la base de datos para obtener las trayectorias contenidas.
- Usar un modelo de IA que permita identificar y caracterizar las trayectorias obtenitas.

1.3. Alcance del sistema

El sistema se enfoca en la identificaciión de trayectorias peatonales individuales y su análisis utilizando herramientas de IA. El alcance incluye:

- Caracterizar la base existente.

- Identificación de trayectorias peatonales individuales.
- Generación de reportes y visualizaciones de los resultados obtenidos.

No se considera dentro del alcance la implementación de modelos de IA desde cero; se utilizarán herramientas y modelos existentes.

2. Requisitos

2.1. Requisitos del sistema

- Versión mínima de Python 3.13.3
- Dependencias principales:
  - dask
  - numpy

2.2. Instrucciones de instalación

1. Clonar el repositorio del proyecto desde Github el proyecto se encuentra dentro de la carpeta `Implementación`.
2. Usar el entorno virtual que ya está en el proyecto:

   ```
   .venv/bin/activate  # Linux/Mac
   .\.venv\Scripts\activate.ps1  # Windows
   ```

   Al ejecutar el comando anterior en windows es posible que aparezca un error de permisos, para solucionarlo se tiene que ejecutar el siguiente comando en la terminal de PowerShell: `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope Process`
3. Instalar el proyecto del entorno virtual:

   ```
   pip install -e .
   ```

4. Configurar variable de entorno:

   ```
   nano .env  # Editar valores de acuerdo a tu configuracion
   ```

5. Verificar instalación:

   ```
   python tests/check_requirements/
   ```

3. Caracterización de datos de trayectorias individuales

El primer paso en el proceso de análisis de datos es la caracterización de los datos. Este proceso implica examinar y comprender la estructura, el contenido y las características de los datos antes de realizar cualquier análisis más profundo. A continuación, se describen las tareas realizadas para caracterizar los datos de trayectorias individuales:

- Cargar los datos de trayectorias individuales desde un archivo CSV.
- Explorar las primeras filas del conjunto de datos para obtener una visión general de su estructura.
- Verificar el número total de registros y columnas en el conjunto de datos.
- Identificar y eliminar columnas innecesarias que no aportan valor al análisis.
- Verificar que todos los campos sigan el mismo formato y contengan datos válidos.
- Identificar y manejar valores faltantes o nulos en el conjunto de datos.

A continuación, se detallan los pasos específicos realizados en el proceso de caracterización:

3.1. Carga de datos

Se cargaron los datos de trayectorias individuales desde un archivo CSV utilizando la biblioteca `dask`. El archivo contiene información sobre las trayectorias de diferentes individuos, incluyendo coordenadas geográficas y otros atributos relevantes.

3.2. Exploración inicial

Se exploraron las primeras 2 filas del conjunto de datos para obtener una visión general de su estructura. Esto incluye la identificación de las columnas presentes y un par de registros. Esto se logró gracias al siguiente código:

```
import dask.dataframe as dd

ruta_archivo = "Mobility_Data.csv"

ddf = dd.read_csv(
    ruta_archivo,
    encoding="utf-8",
    sep=",",
    dtype="object",     luego)
)

columnas = ddf.columns.tolist()

print("Columnas y 2 ejemplos por cada una:\n")
for col in columnas:
    ejemplos = ddf[col].head(2).values.tolist()
    print(f"- {col}: {ejemplos}")
```

Figura 1: csv_glance.py, exploracion inicial del conjunto de datos.

El resultado de la ejecución de este código es el siguiente:

1. `id`: Identificador numérico único para cada registro ['34284565','34284566'].
2. `identifier`: Identificador único del dispositivo ['f2640430-7e39-41b7-80bb-3fddaa44779c', 'f2640430-7e39-41b7-80bb-3fddaa44779c'].
3. `identifier_type`: Tipo de identificador del dispositivo. En este caso, 'gaid' (Google Advertising ID para Android). Otros posibles: 'idfa' (Apple), 'imei' ['gaid', 'gaid'].
4. `timestamp`: Fecha y hora del registro de movilida ['2022-11-07 02:04:21', '2022-11-08 17:29:35']
5. `device_lat/device_lon`: Coordenadas geográficas (latitud y longitud) donde se detectó el dispositivo ['21.843149', '21.843149'], ['-102.196838', '-102.196838'].
6. `country_short/province_short`: Código del país (MX = México) y región (MX.01 = Aguascalientes, según estándar ISO) ['MX', 'MX'], ['MX.01', 'MX.01'].
7. `ip_address`: Dirección IP del dispositivo (en formato IPv6)['2806:103e:16::', '2806:103e:16::'].
8. `device_horizontal_accuracy`: Precisión del GPS en metro. Menor valor = mayor precisión ['8.0', '8.0'].
9. `source_id`: Hash único que identifica la fuente de los datos. Puede ser un identificador de la aplicación o del dispositivo ['449d086de6d9c3d192345c992dfac54319b9d550a92bcd20c37f8368cb428344', '449d086de6d9c3d192345c992dfac54319b9d550a92bcd20c37f8368cb428344'].

10. `record_id`: Identificador único del registro de movilidad (diferente al id) ['77d795df-6972-4f00-ac41-d10d1812bb2d', '8f8e1281-bc4d-4d2c-b00b-eb5c52d75bc1'].
11. `home_country_code`: País de residencia del usuario ['MX', 'MX'].
12. `home_geog_point/work_geog_point`: Coordenadas geográficas del hogar y del trabajo en formato WKT (Well-Known Text) ['POINT(-102.370380092263 22.2075340951743)', 'POINT(-102.370380092263 22.2075340951743)'].
13. `home_hex_id/work_hex_id`: Identificador hexadecimal del hogar y del trabajo, representando una ubicación geográfica en un sistema de cuadrícula hexagonal ['85498853fffffff', '85498853fffffff'].
14. `data_execute`: Fecha de procesamiento del registro de movilidad, no necesariamente la fecha de recolección ['2023-05-30', '2023-05-30'].
15. `time_zone_name`: Zona horaria del dispositivo ['America$/Mexico_City', 'America/Mexico_City'].

3.3. Número de registros y columnas

Se verificó el número total de registros y columnas en el conjunto de datos utilizando el siguiente código:

```
import dask.dataframe as dd

ruta_archivo = "Mobility_Data.csv"
columnas_usar = ["id"]

ddf = dd.read_csv(
    ruta_archivo,
    usecols=columnas_usar,
    sep=",",
    dtype={"id": "str"},
    blocksize="256MB",
)

print("Contando registros (paciencia para archivos
    grandes)...")
total_registros = ddf.shape[0].compute()

print(f"Total de registros: {total_registros:,}")
```

Figura 2: csv_count_registers.py, conteo de registros en el conjunto de datos.

El resultado de la ejecución de este código es que el conjunto de datos contiene un total de 69,980,000 registros y 19 campos. Esto indica que hay una cantidad significativa de datos disponibles para el análisis.

3.4. Identificar y eliminar campos innecesarios que no aportan valor al análisis

Ya que tenemmos un conjunto de datos con 19 campos, es importante identificar y eliminar aquellas que no aportan valor al análisis. Para ello, vamos a revisar los valores únicos de los siguiente campos, para determinar si son redundantes o no aportan información relevante. A continuación, se presentan los campos que se consideran innecesarios:

- `id`
- `identifier_type`
- `country_short`
- `province_short`
- `ip_address`

- source_id
- home_country_code
- home_geog_point
- work_geog_point
- home_hex_id
- work_hex_id
- data_execute

Para eliminar estos campos vamos a tomar solamente las columnas que sí vamos a conservar. A continuación, se muestra el código utilizado para realizar esta tarea:

```
import dask.dataframe as dd

# Columnas que si vamos a conservar
columnas_deseadas = [
    'identifier',
    'timestamp',
    'device_lat',
    'device_lon',
    'device_horizontal_accuracy',
    'record_id',
    'time_zone_name'
    ]

# Cargar solo las columnas necesarias
df = dd.read_csv('Mobility_Data.csv', usecols=
    columnas_deseadas)

# Guardar el resultado
df.to_csv('Mobility_Data_Slim.csv', index=False,
    single_file=True, encoding='utf-8-sig')
```

Figura 3: remove_columns.py, eliminación de campos innecesarios en el conjunto de datos.

El resultado de la ejecución de este código es un nuevo archivo CSV llamado `Mobility_Data_Slim.csv` que contiene únicamente las columnas seleccionadas, eliminando así las que no aportan valor al análisis.

3.5. Verificar que todos los campos sigan el mismo formato y contengan datos válidos

3.6. Identificar y manejar valores faltantes o nulos en el conjunto de datos

prueba de push