

Universidad de Córdoba

Máster en Inteligencia Computacional
e Internet de las Cosas

Procesamiento de Datos Localización y Telefonía Móvil

Análisis, Diseño y Procesamiento de Datos
Aplicados a las Ciencias y a las Tecnologías
(ADP Marzo 2025)



UNIVERSIDAD
DE CÓRDOBA

Autor:

Marcos Rivera Gavilán

Profesor:

Domigo Ortiz Boyer

Índice general

Índice general	1
1. Introducción	2
2. Descarga y Carga de Datos en MongoDB	3
2.1. Descarga de datos	3
2.2. Estructura de los ficheros	3
2.3. Inserción en MongoDB	4
3. Búsqueda de Países con Interacciones	7
3.1. Localizar todos los países con los que se interactúa	7
3.1.1. Shell de MongoDB	7
4. País con Mayor Interacción (excluyendo Italia)	8
4.1. Consulta en MongoDB	8
5. Celda con Mayor Comunicación con el Extranjero	9
5.1. Consulta	9
6. Celdas con Mayor Actividad por Tipo de Evento	10
6.1. Mayor <code>sms_in</code>	10
6.2. Mayor <code>sms_out</code>	10
6.3. Mayor <code>call_in</code>	11
6.4. Mayor <code>call_out</code>	11
6.5. Mayor <code>internet</code>	11
6.6. Mayor Total de Actividad	12
7. Creación de una Colección con un Documento por Celda	13
7.1. Operación de agregación	14
8. Colección con Datos Acumulados por Celda y Hora	15
8.1. Shell de MongoDB	16
9. Estudio de Celdas de Interés	17
9.1. Extracción de información para cada celda	17
9.2. Análisis y comentarios	20
Bibliografía	21

Capítulo 1

Introducción

El objetivo de este trabajo es profundizar en el uso de **MongoDB** para el tratamiento de datos procedentes de los registros de datos o CDR (Call Detail Record). Para ello, se dispone de datos de la ciudad de Milán (proyecto de Telecom Italia) **cdrlink** que recogen la actividad de telefonía móvil e internet en distintas celdas (o zonas) de la ciudad de Milán entre el 1 de noviembre de 2013 y el 1 de enero de 2014.

Cada alumno ha de seleccionar una semana completa de datos (de lunes a domingo). Sobre esa semana, se realizarán las operaciones y consultas que se detallan en los siguientes apartados.

Los pasos obligatorios incluyen:

- Descargar y cargar los datos en una colección de MongoDB.
- Localizar y analizar las interacciones con otros países.
- Determinar cuáles son las celdas con mayor actividad en distintos tipos de evento (SMS, llamadas, internet).
- Crear colecciones agregando información por celda y por franja horaria.
- Estudiar detenidamente el comportamiento de algunas celdas de interés (Bocconi, Navigli, Duomo, aeropuerto de Linate, etc.).

En los siguientes apartados se explica cada uno de los pasos, el código utilizado (Python/MongoDB) y los resultados obtenidos.

Capítulo 2

Descarga y Carga de Datos en MongoDB

2.1. Descarga de datos

Los datos originales se encuentran en el siguiente enlace de Harvard Dataverse **cdrlink**:

<https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/EGZHFV>

Dentro de esa página se han de localizar los ficheros de datos en formato **.txt** que correspondan a la semana completa que se desee analizar (en mi caso, del 02/12/2013 al 08/12/2013). Cada fichero contiene la información de las interacciones en bloques de 10 minutos durante el día seleccionado.

2.2. Estructura de los ficheros

La estructura de cada fichero suele ser similar a:

- **Square_ID**: Identificador de la celda.
- **Time_Interval**: Momento temporal o intervalo de tiempo.
- **Country_Code**: Código de país del que proviene la actividad.
- **SMS_in, SMS_out, Call_in, Call_out, Internet**: Actividad medida (normalmente agregada) de cada tipo en la celda.

2.3. Inserción en MongoDB

Para insertar estos datos en MongoDB, se ha creado la base de datos `Milan_CDR_db` y la colección `Milan_CDR_c`. A continuación se muestra el código en Python (usando la librería `pymongo`) para cargar los datos de varios ficheros `.txt`:

Listing 2.1: set-up-db.py

```
1 import os
2 import pymongo
3 import datetime
4 import concurrent.futures
5 from tqdm import tqdm
6
7 # Connect to MongoDB (pymongo connections are thread-safe)
8 client = pymongo.MongoClient("mongodb://admin:1234@localhost:27017/"
9                               ")
10 db = client["Milan_CDR_db"]
11 collection = db["Milan_CDR_c"]
12
13 # Directory where .txt files are located
14 data_directory = "P3/data/"
15 batch_size = 1000 # Adjust this number based on memory and
16                   # performance needs
17
18 def process_file(filename, position):
19     filepath = os.path.join(data_directory, filename)
20
21     # Count total lines for a progress bar
22     with open(filepath, 'r') as f:
23         total_lines = sum(1 for _ in f)
24
25     docs = []
26     missing_fields_count = 0
27     invalid_timestamp_count = 0
28     inserted_count = 0
29
30     # Process file with progress bar
31     with open(filepath, 'r') as f, tqdm(total=total_lines, desc=
32         filename, position=position, leave=True) as pbar:
33         for line in f:
34             line = line.strip()
35             if not line:
36                 pbar.update(1)
37                 continue
38
39             parts = line.split() # Splits on any whitespace
40             if len(parts) < 8:
41                 missing_fields_count += 1
42                 pbar.update(1)
43                 continue
44
45             try:
46                 square_id = int(parts[0])
47                 time_interval_str = parts[1] # Unix timestamp in
48                     milliseconds
49                 country_code = int(parts[2])
50                 sms_in = float(parts[3])
51                 sms_out = float(parts[4])
52                 call_in = float(parts[5])
53                 call_out = float(parts[6])
54                 internet = float(parts[7])
```

```

51
52         # Convert Unix timestamp to datetime
53         try:
54             timestamp_ms = int(time_interval_str)
55             time_interval = datetime.datetime.
56                 utcfromtimestamp(timestamp_ms / 1000)
57         except ValueError:
58             invalid_timestamp_count += 1
59             pbar.update(1)
60             continue
61
62     except ValueError:
63         missing_fields_count += 1
64         pbar.update(1)
65         continue
66
67     # Build document for MongoDB
68     doc = {
69         "square_id": square_id,
70         "time_interval": time_interval,
71         "country_code": country_code,
72         "sms_in": sms_in,
73         "sms_out": sms_out,
74         "call_in": call_in,
75         "call_out": call_out,
76         "internet": internet
77     }
78     docs.append(doc)
79
80     # Insert batch when batch size is reached
81     if len(docs) >= batch_size:
82         try:
83             collection.insert_many(docs, ordered=False)
84             inserted_count += len(docs)
85         except Exception as e:
86             print(f"Error inserting batch from {filename}: {e}")
87             docs = []
88
89     pbar.update(1)
90
91     # Insert any remaining documents
92     if docs:
93         try:
94             collection.insert_many(docs, ordered=False)
95             inserted_count += len(docs)
96         except Exception as e:
97             print(f"Error inserting final batch from {filename}: {e}")
98
99     # Print summary for this file
100     print(f"Finished {filename}: Inserted {inserted_count} docs,
101         Missing fields: {missing_fields_count}, Invalid timestamps:
102         {invalid_timestamp_count}")
103
104
105
106
107

```

```
108 def main():
109     files = [f for f in os.listdir(data_directory) if f.endswith(".
        txt")]
110     with concurrent.futures.ThreadPoolExecutor() as executor:
111         futures = []
112         # Assign a unique position for each progress bar
113         for pos, filename in enumerate(files):
114             futures.append(executor.submit(process_file, filename,
                pos))
115         concurrent.futures.wait(futures)
116
117 if __name__ == "__main__":
118     main()
```

Capítulo 3

Búsqueda de Países con Interacciones

3.1. Localizar todos los países con los que se interactúa

Para encontrar todos los códigos de país que aparecen en la colección `Milan_CDR_c` (excluyendo el propio país Italia, cuyo código es habitualmente 39), se puede ejecutar la siguiente consulta en la **shell de MongoDB** o con PyMongo en Python.

3.1.1. Shell de MongoDB

Listing 3.1: Localizar países distintos de Italia

```
1 Milan_CDR_db> db.Milan_CDR_c.distinct("country_code", { "  
   country_code": { $ne: 39 } })  
2 [  
3     0,     1,     7,    20,    30,    31,    32,  
4     33,    34,    36,    40,    41,    44,    46,  
5     47,    48,    49,    55,    86,    90,    91,  
6     258,   358,  371,  380,  385,  420,  421,  
7     505,  1647  
8 ]
```

Capítulo 4

País con Mayor Interacción (excluyendo Italia)

Una vez conocidos todos los países que aparecen, se desea saber con cuál de ellos existe mayor interacción (suma de SMS, llamadas e internet).

4.1. Consulta en MongoDB

Basta con agrupar por `country_code` y sumar los campos de interés (`sms_in`, `sms_out`, `call_in`, `call_out`, `internet`), excluyendo Italia (código 39). En la shell de MongoDB:

Listing 4.1: Consulta para encontrar el país con mayor interacción (excluye 39)

```
1 Milan_CDR_db> db.Milan_CDR_c.aggregate([
2 ...   { $match: { country_code: { $ne: 39 } } },
3 ...   { $group: {
4 ...       _id: "$country_code",
5 ...       totalInteraction: { $sum: {
6 ...           $add: ["$sms_in", "$sms_out", "$call_in", "
7 ...               $call_out", "$internet"]
8 ...       } } },
9 ...   { $sort: { totalInteraction: -1 } },
10 ...   { $limit: 1 }
11 ... ])
```

```
12 [ { _id: 44, totalInteraction: 8228.493137227808 } ]
```

El `country_code` que aparece como primer resultado tras la ordenación por `totalInteraction` es el país con mayor interacción, aparte de Italia. En este caso es el que se corresponde al `country_code` 44.

Capítulo 5

Celda con Mayor Comunicación con el Extranjero

Para encontrar la celda que más comunica con el extranjero (es decir, **todas las interacciones** con `country_code` distinto de 39), se puede realizar un agrupamiento adicional por `square_id`.

5.1. Consulta

Listing 5.1: Celda con mayor comunicación con el extranjero

```
1 Milan_CDR_db> db.Milan_CDR_c.aggregate([
2 ...     { $match: { country_code: { $ne: 39 } } },
3 ...     { $group: {
4 ...         _id: "$square_id",
5 ...         totalInteractionForeign: { $sum: {
6 ...             $add: ["$sms_in", "$sms_out", "$call_in", "
7 ...                 $call_out", "$internet"]
8 ...         } } },
9 ...     { $sort: { totalInteractionForeign: -1 } },
10 ...     { $limit: 1 }
11 ... ])
12 [ { _id: 5161, totalInteractionForeign: 1652.7010971616246 } ]
```

El `_id` del documento resultante corresponderá al **square_id** (celda) con más actividad proveniente del extranjero. En este caso la celda 5161.

Capítulo 6

Celdas con Mayor Actividad por Tipo de Evento

En este apartado se busca, por separado, la celda que tenga mayor **sms_in**, **sms_out**, **call_in**, **call_out**, **internet** y también la que tenga mayor **total de actividad** (suma de todos los campos).

6.1. Mayor sms_in

Listing 6.1: Consulta para la celda con mayor sms_{in}

```
1 Milan_CDR_db> db.Milan_CDR_c.aggregate([
2 ...   { $group: {
3 ...     _id: "$square_id",
4 ...     totalSmsIn: { $sum: "$sms_in" }
5 ...   }},
6 ...   { $sort: { totalSmsIn: -1 } },
7 ...   { $limit: 1 }
8 ... ])
9 [ { _id: 5059, totalSmsIn: 92171.53566678267 } ]
```

6.2. Mayor sms_out

Listing 6.2: Consulta para la celda con mayor sms_{out}

```
1 Milan_CDR_db> db.Milan_CDR_c.aggregate([
2 ...   { $group: {
3 ...     _id: "$square_id",
4 ...     totalSmsOut: { $sum: "$sms_out" }
5 ...   }},
6 ...   { $sort: { totalSmsOut: -1 } },
7 ...   { $limit: 1 }
8 ... ])
9 [ { _id: 5059, totalSmsOut: 58572.62761437156 } ]
```

6.3. Mayor call_in

Listing 6.3: Consulta para la celda con mayor call_{in}

```
1 Milan_CDR_db> db.Milan_CDR_c.aggregate([
2 ...     { $group: {
3 ...         _id: "$square_id",
4 ...         totalCallIn: { $sum: "$call_in" }
5 ...     }},
6 ...     { $sort: { totalCallIn: -1 } },
7 ...     { $limit: 1 }
8 ... ])
9 [ { _id: 5059, totalCallIn: 88107.4810220383 } ]
```

6.4. Mayor call_out

Listing 6.4: Consulta para la celda con mayor call_{out}

```
1 Milan_CDR_db> db.Milan_CDR_c.aggregate([
2 ...     { $group: {
3 ...         _id: "$square_id",
4 ...         totalCallOut: { $sum: "$call_out" }
5 ...     }},
6 ...     { $sort: { totalCallOut: -1 } },
7 ...     { $limit: 1 }
8 ... ])
9 [ { _id: 5059, totalCallOut: 108146.36961202238 } ]
```

6.5. Mayor internet

Listing 6.5: Consulta para la celda con mayor internet

```
1 Milan_CDR_db> db.Milan_CDR_c.aggregate([
2 ...     { $group: {
3 ...         _id: "$square_id",
4 ...         totalInternet: { $sum: "$internet" }
5 ...     }},
6 ...     { $sort: { totalInternet: -1 } },
7 ...     { $limit: 1 }
8 ... ])
9 [ { _id: 5161, totalInternet: 1559821.5028882942 } ]
```

6.6. Mayor Total de Actividad

Finalmente, para obtener la celda con la actividad total más alta:

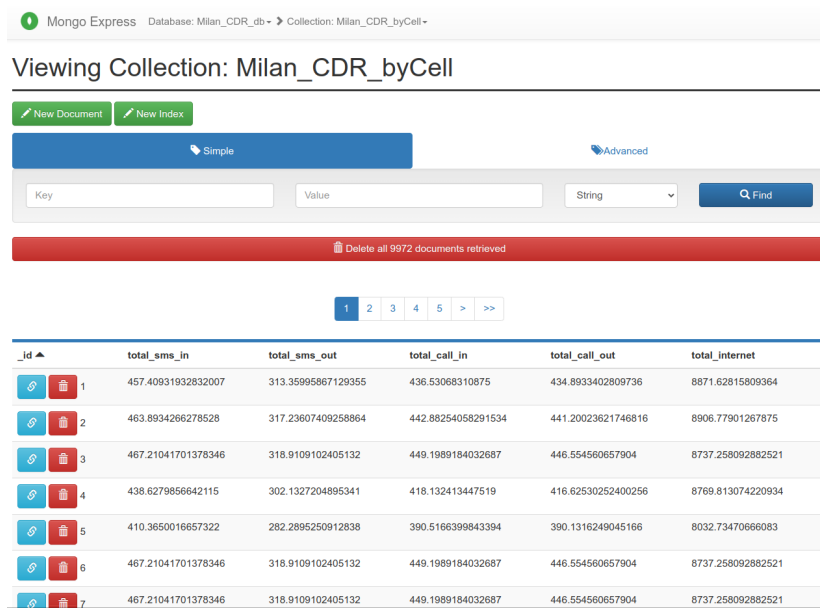
Listing 6.6: Celda con mayor actividad total

```
1 Milan_CDR_db> db.Milan_CDR_c.aggregate([
2 ...   { $group: {
3 ...     _id: "$square_id",
4 ...     totalActivity: {
5 ...       $sum: { $add: ["$sms_in", "$sms_out", "$call_in",
6 ...         "$call_out", "$internet"] }
7 ...     }},
8 ...   { $sort: { totalActivity: -1 } },
9 ...   { $limit: 1 }
10 ... ])
11 [ { _id: 5161, totalActivity: 1842674.4494343947 } ]
```

Capítulo 7

Creación de una Colección con un Documento por Celda

En este paso se requiere crear una nueva colección (`Milan_CDR_byCell`) donde se almacene **un documento por celda** con los acumulados de cada uno de los campos (`sms.in`, `sms.out`, `call.in`, `call.out`, `internet`).



Mongo Express Database: Milan_CDR_db -> Collection: Milan_CDR_byCell

Viewing Collection: Milan_CDR_byCell

Buttons: New Document, New Index

Simple Advanced

Key Value String Find

Delete all 9972 documents retrieved

_id	total_sms_in	total_sms_out	total_call_in	total_call_out	total_internet
1	457.40931932832007	313.35995867129355	436.53068310875	434.8933402809736	8871.62815809364
2	463.8934266278528	317.23607409258864	442.88254058291534	441.20023621746816	8906.77901267875
3	467.21041701378346	318.9109102405132	449.1989184032687	446.554560657904	8737.258092882521
4	438.6279856642115	302.1327204895341	418.132413447519	416.62530252400256	8769.813074220934
5	410.3650016657322	282.2895250912838	390.5166399843394	390.1316249045166	8032.73470666083
6	467.21041701378346	318.9109102405132	449.1989184032687	446.554560657904	8737.258092882521
7	467.21041701378346	318.9109102405132	449.1989184032687	446.554560657904	8737.258092882521

Figura 7.1: Colección `Milan_CDR_byCell` en la base de datos.

7.1. Operación de agregación

La idea es agrupar por `square_id` y sumar. A continuación, se inserta el resultado en la nueva colección.

Listing 7.1: Creación de la colección *Milan_CDR_{byCell}*

```
1 db.Milan_CDR_c.aggregate([
2   { $group: {
3     _id: "$square_id",
4     total_sms_in: { $sum: "$sms_in" },
5     total_sms_out: { $sum: "$sms_out" },
6     total_call_in: { $sum: "$call_in" },
7     total_call_out: { $sum: "$call_out" },
8     total_internet: { $sum: "$internet" }
9   }},
10  { $out: "Milan_CDR_byCell" }
11 ])
```

El comando `$out` crea la colección `Milan_CDR_byCell` con la salida del `aggregate`.

Capítulo 8

Colección con Datos Acumulados por Celda y Hora

Ahora se requiere crear una colección (`Milan_CDR_byCellHour`) con un documento por cada (**celda, hora**). Para ello, se debe **parsear** el campo `time_interval` para extraer la hora. Dependiendo de la estructura temporal (`YYYY-MM-DD HH:MM:SS` o similar), se puede usar un método de conversión a `Date` y después agrupar por la **celda** y la **hora**.

Mongo Express Database: Milan_CDR_db Collection: Milan_CDR_byCellHour

Viewing Collection: Milan_CDR_byCellHour

Buttons: New Document, New Index

Simple (selected) Advanced

Key Value String Find

Delete all 224866 documents retrieved

1 2 3 4 5 > >>

_id	total_sms_in	total_sms_out	total_call_in	total_call_out	total_internet
	0.6117080084175437	0.11401239379403556	0.08385255705676568	0.22370463951262828	35.01659983911114
	74.09161065723124	38.8328464840773	44.16128466936735	52.41060997247578	1084.751959048266

Document 1: { "square_id": 201, "hour": 4 }

Document 2: { "square_id": 1982, "hour": 7 }

Figura 8.1: Colección `Milan_CDR_byCellHour` en la base de datos.

8.1. Shell de MongoDB

Como el campo `time_interval` lo hemos insertado en formato `ISODate`, podemos extraer la hora correctamente y proceder con la agregación.

Listing 8.1: Creación de la colección `Milan_CDR_byCellHour`

```
1 Milan_CDR_db> db.Milan_CDR_c.aggregate([
2 ...   {
3 ...     $project: {
4 ...       square_id: 1,
5 ...       sms_in: 1,
6 ...       sms_out: 1,
7 ...       call_in: 1,
8 ...       call_out: 1,
9 ...       internet: 1,
10 ...      hour: { $hour: "$time_interval" }
11 ...    },
12 ...   },
13 ...   {
14 ...     $group: {
15 ...       _id: { square_id: "$square_id", hour: "$hour" },
16 ...       total_sms_in: { $sum: "$sms_in" },
17 ...       total_sms_out: { $sum: "$sms_out" },
18 ...       total_call_in: { $sum: "$call_in" },
19 ...       total_call_out: { $sum: "$call_out" },
20 ...       total_internet: { $sum: "$internet" }
21 ...     },
22 ...   },
23 ...   {
24 ...     $out: "Milan_CDR_byCellHour"
25 ...   }
26 ... ])
```

Este procedimiento asegurará que la colección `Milan_CDR_byCellHour` almacene datos agregados por cada combinación de `square_id` y `hora`. Esto facilita análisis de tendencias de uso y comportamiento en distintos momentos del día.

Capítulo 9

Estudio de Celdas de Interés

En este capítulo se examinan en detalle los valores totales de SMS (`in` y `out`), llamadas (`in` y `out`) e `internet` en las celdas listadas a continuación:

- 4259 (Bocconi)
- 4456 (Navigli)
- 5060 (Duomo)
- 1419 (Terreno agrícola)
- 2436 (Área industrial)
- 4990 (Aeropuerto de Linate)
- 945 (Residencial aislado)
- 5048 (Residencial céntrico)

9.1. Extracción de información para cada celda

Para obtener los totales de la semana seleccionada, se empleó la siguiente consulta de agregación, filtrando únicamente aquellas celdas incluidas en la lista anterior:

Listing 9.1: Consulta para extraer los totales de las celdas de interés

```
1 db.Milan_CDR_c.aggregate([
2   { $match: { square_id: { $in: [4259, 4456, 5060, 1419, 2436,
3     4990, 945, 5048] } } },
4   { $group: {
5     _id: "$square_id",
6     total_sms_in: { $sum: "$sms_in" },
7     total_sms_out: { $sum: "$sms_out" },
8     total_call_in: { $sum: "$call_in" },
9     total_call_out: { $sum: "$call_out" },
10    total_internet: { $sum: "$internet" }
11  }}
12 ])
```

La salida devuelve, para cada `square_id`, los valores totales en `sms_in`, `sms_out`, `call_in`, `call_out` e `internet`. A continuación, se listan los resultados resumidos:

■ **2436 (Área industrial):**

- `sms_in`: 1906.995
- `sms_out`: 1705.821
- `call_in`: 2014.024
- `call_out`: 2671.531
- `internet`: 33021.997

■ **4456 (Navigli):**

- `sms_in`: 23771.626
- `sms_out`: 15781.219
- `call_in`: 22978.403
- `call_out`: 21744.797
- `internet`: 668555.880

■ **1419 (Terreno agrícola):**

- `sms_in`: 359.401
- `sms_out`: 257.146
- `call_in`: 389.817
- `call_out`: 408.500
- `internet`: 4742.226

■ **4259 (Bocconi):**

- `sms_in`: 11997.288
- `sms_out`: 8784.143
- `call_in`: 9395.408
- `call_out`: 10624.558
- `internet`: 305456.481

■ **945 (Residencial aislado):**

- `sms_in`: 1044.900
- `sms_out`: 598.420
- `call_in`: 1093.799
- `call_out`: 1216.099
- `internet`: 20721.814

■ **5048 (Residencial céntrico):**

- sms_in: 20340.778
- sms_out: 21027.604
- call_in: 18927.497
- call_out: 18641.310
- internet: 365378.234

■ **4990 (Aeropuerto de Linate):**

- sms_in: 8034.901
- sms_out: 5230.980
- call_in: 7603.828
- call_out: 10225.593
- internet: 125189.060

■ **5060 (Duomo):**

- sms_in: 39882.930
- sms_out: 23755.940
- call_in: 37555.920
- call_out: 45409.888
- internet: 530960.865

9.2. Análisis y comentarios

A partir de estos totales, se pueden inferir diversas particularidades de cada zona:

- **Bocconi (4259)**: Presenta valores altos tanto en llamadas como en internet, lo cual coincide con el perfil de una zona universitaria con estudiantes y personal docente realizando múltiples comunicaciones y empleando la red de datos.
- **Navigli (4456)**: Supera notablemente a la mayoría en **internet** (668.5k) —más que cualquier otra celda—, lo que sugiere una intensa actividad de datos, posiblemente asociada al turismo, la vida nocturna y el ocio propio del barrio.
- **Duomo (5060)**: Registra la mayor cantidad de llamadas (**call_out** y **call_in** en torno a 45k y 37.5k respectivamente) y también altos valores de SMS. Esto coincide con su papel de centro histórico y comercial, muy concurrido por turistas y residentes.
- **Terreno agrícola (1419)**: Exhibe los niveles más bajos en casi todas las categorías, lo cual es esperable dada la menor densidad de población y de actividad en este tipo de entorno.
- **Área industrial (2436)**: Sus cifras (por ejemplo, 2k-2.7k en llamadas y 33k en internet) son moderadas, reflejando una actividad significativa en horario laboral, pero claramente inferior a las zonas más urbanas.
- **Aeropuerto de Linate (4990)**: Muestra valores reseñables en llamadas y **internet** (125k), reforzando la hipótesis de un gran número de viajeros y conexiones, muchas de ellas con el extranjero.
- **Residencial aislado (945)**: Con un tráfico relativamente bajo (p. ej., 1044 **sms_in** y 20721 **internet**), probablemente refleja la menor densidad de población y un estilo de uso más limitado.
- **Residencial céntrico (5048)**: Ofrece cifras altas en llamadas y datos (365k en **internet**), coherentes con un área más densamente poblada y, por tanto, con mayor interacción en comunicaciones.

En conclusión, los resultados cuantitativos concuerdan de manera razonable con la naturaleza de cada ubicación. De acuerdo con estos totales:

- **Navigli** sobresale en uso de internet.
- **Duomo** concentra la mayor actividad en llamadas y un volumen muy alto de SMS.
- **Bocconi, Linate y las zonas residenciales** siguen patrones intermedios o altos, según la densidad de población o el flujo de personas en tránsito.
- **Terreno agrícola** confirma la actividad mínima.

Bibliografía

- [1] HARVARD DATAVERSE - TELECOM ITALIA CDR DATA SET.
<https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/EGZHFV>