Building RDF with LA Public Safety Data

By Lauro Cabral, Christopher Vargas, and Sotheanith Sok

Datasets' Criteria

Why we chose these datasets

The relationship between arrest dataset and crime dataset is complex enough to derive many conclusions and facts about crime in LA

Datasets have inherent structures such as person, location, and weapon and we can theorize hierarchical structures from such datasets

Datasets are large enough such that a complex RDF file can be deduced.

Dataset 1: Arrest Data from 2020 to Present

URL: https://data.lacity.org/resource/amvf-fr72

Arrest Data from 2020 to Present: Labels

- Report ID
- Report Type
- Arrest Date
- Time
- Area ID
- Area Name
- Reporting District
- Age
- Sex Code
- Descent Code
- Charge Group Code
- Charge Group Description
- Arrest Type Code
- Charge
- Charge Description
- Disposition Description
- Address
- Cross Street
- LAT
- LON
- Location
- Booking Date
- Booking Time
- Booking Location
- Booking Location Code

Dataset 2: Crime Data from 2020 to Present

URL: https://data.lacity.org/resource/2nrs-mtv8

Crime Data from 2020 to Present: Labels

- DR_NO
- Date Rptd
- DATE OCC
- TIME OCC
- AREA
- AREA NAME
- Rpt Dist No
- Part 1-2
- Crm Cd
- Crm Cd Desc
- Mocodes
- Vict Age
- Vict Sex
- Vict Descent
- Premis Cd
- Premis Desc
- Weapon Used Cd
- Weapon Desc
- Status
- Status Desc
- Crm Cd 1
- Crm Cd 2
- Crm Cd 3
- Crm Cd 4
- LOCATION
- Cross Street
- LAT
- LON

Queries

Does age affect the likelihood that a person will be involved in a crime?

What is the safest time to travel in LA?

Based on your gender, how likely are you to be involved with a crime in a given neighborhood?

RDF Schema

Classes

There are 11 classes:

- Report
- Person
- Location
- ArrestReport
- Charge
- Booking
- CrimeReport
- Crime
- Premise
- Weapon
- Status

Report Class

- It is an instance of RDFS: Class
- Properties:
  - "hasID" - XSD: integer
  - "hasPerson" - Person class
  - "hasID" - XSD: integer
  - "hasTime" - XSD: integer
  - "hasDate" - XSD: integer
  - "hasLocation" - Location class

- It is an instance of RDFS: Class
- Properties:
    - "hasAge" - XSD: integer
    - "hasSex" - XSD: string
    - "hasDescendent" - XSD: string

- It is an instance of RDFS: Class
- Properties:
  - "hasReportingDistrictNumber" - XSD: integer
  - "hasAreaID" - XSD: integer
  - "hasAreaName" - XSD: string
  - "hasAddress" - XSD: string
  - "hasCrossStreet" - XSD: string
  - "hasLatitude" - XSD: double
  - "hasLongitude" - XSD: double

ArrestReport Class

- It is a subclass of Report class
- Properties:
  - "hasDispositionDescription" - XSD: string
  - "hasReportType" - XSD: string
  - "hasArrestType" - XSD: string
  - "hasCharge" - Charge class
  - "hasBooking" - Booking class

Charge Class

- It is an instance of RDFS: Class
- Properties:
  - "hasChargeGroupCode" - XSD: integer
  - "hasChargeGroupDescription" - XSD: string
  - "hasChargeCode" - XSD: integer
  - "hasChargeDescription" - XSD: string

- It is an instance of RDFS: Class
- Properties:
  - "hasBookingDate" - XSD: date
  - "hasBookingTime" - XSD: time
  - "hasBookingLocation" - XSD: string
  - "hasBookingCode" - XSD: integer

CrimeReport Class

- It is a subclass of Report class
- Properties:
  - "hasDateReported" - XSD: date
  - "hasMocodes" - XSD: string
  - "hasCrime" - Crime class
  - "hasStatus" - Status class
  - "hasWeapon" - Weapon class
  - "hasPremise" - Premise class
  - "hasPart1-2 " - XSD: integer

Crime Class

- It is an instance of RDFS: Class
- Properties:
  - "hasCrimeCommitted" - XSD: string
  - "hasCrimeCommittedDescription" - XSD: string
  - "hasCrimeCommitted1" - XSD: string
  - "hasCrimeCommitted2" - XSD: string
  - "hasCrimeCommitted3" - XSD: string
  - "hasCrimeCommitted4" - XSD: string

Status Class

- It is an instance of RDFS: Class
- Properties:
  - "hasStatusCode" - XSD: integer
  - "hasStatusDescription" - XSD: string
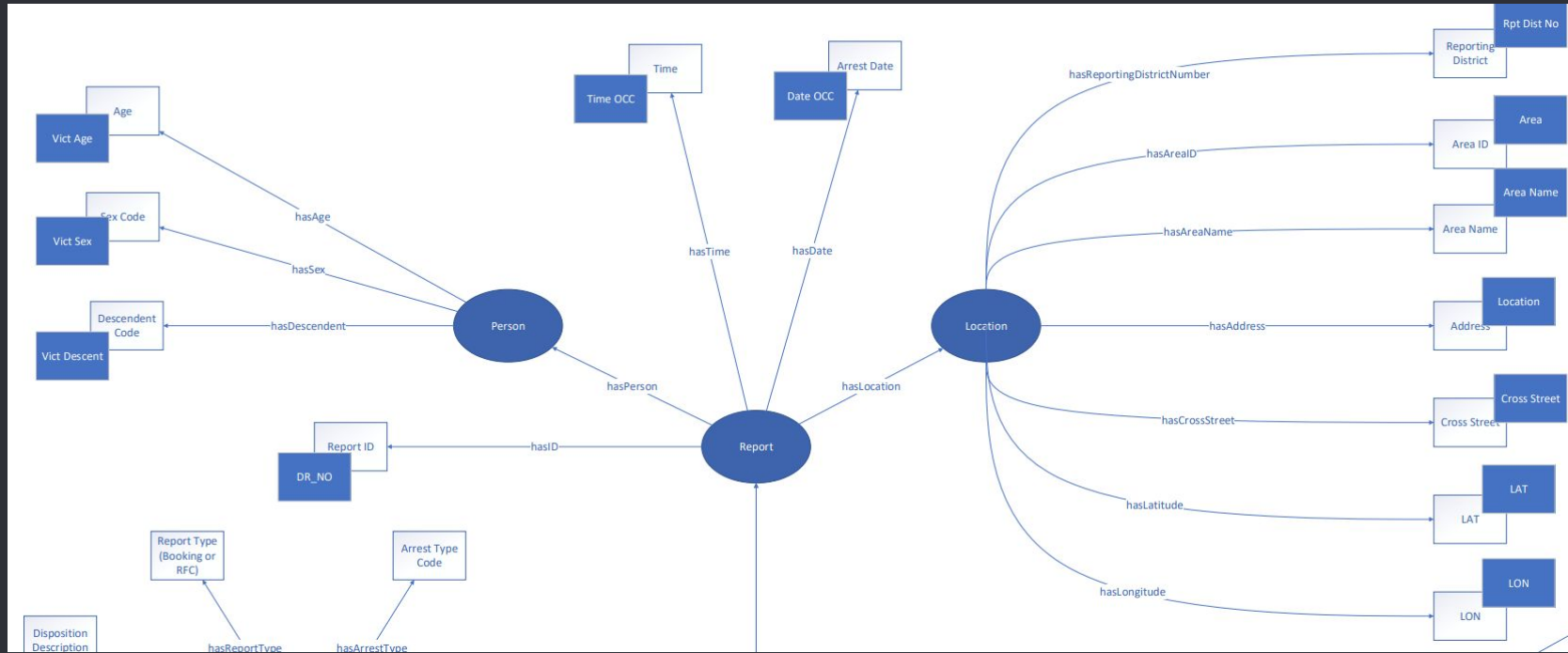
Weapon Class

- It is an instance of RDFS: Class
- Properties:
  - "hasWeaponUsedCode" - XSD: integer
  - "hasWeaponDescription" - XSD: string
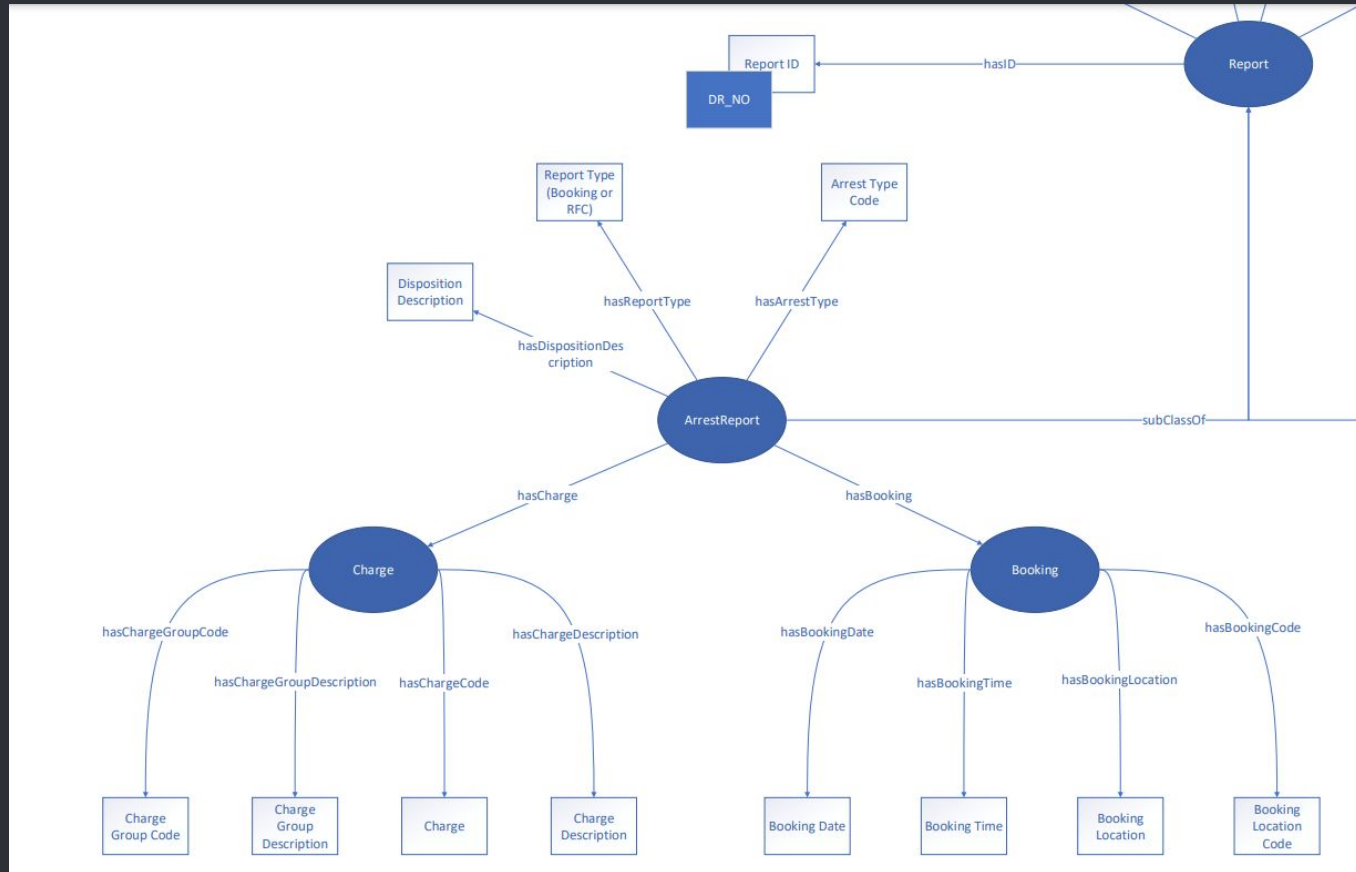
Premise Class

- It is an instance of RDFS: Class
- Properties:
  - "hasPremiseCode" - XSD: integer
  - "hasPremiseDescription" - XSD: string
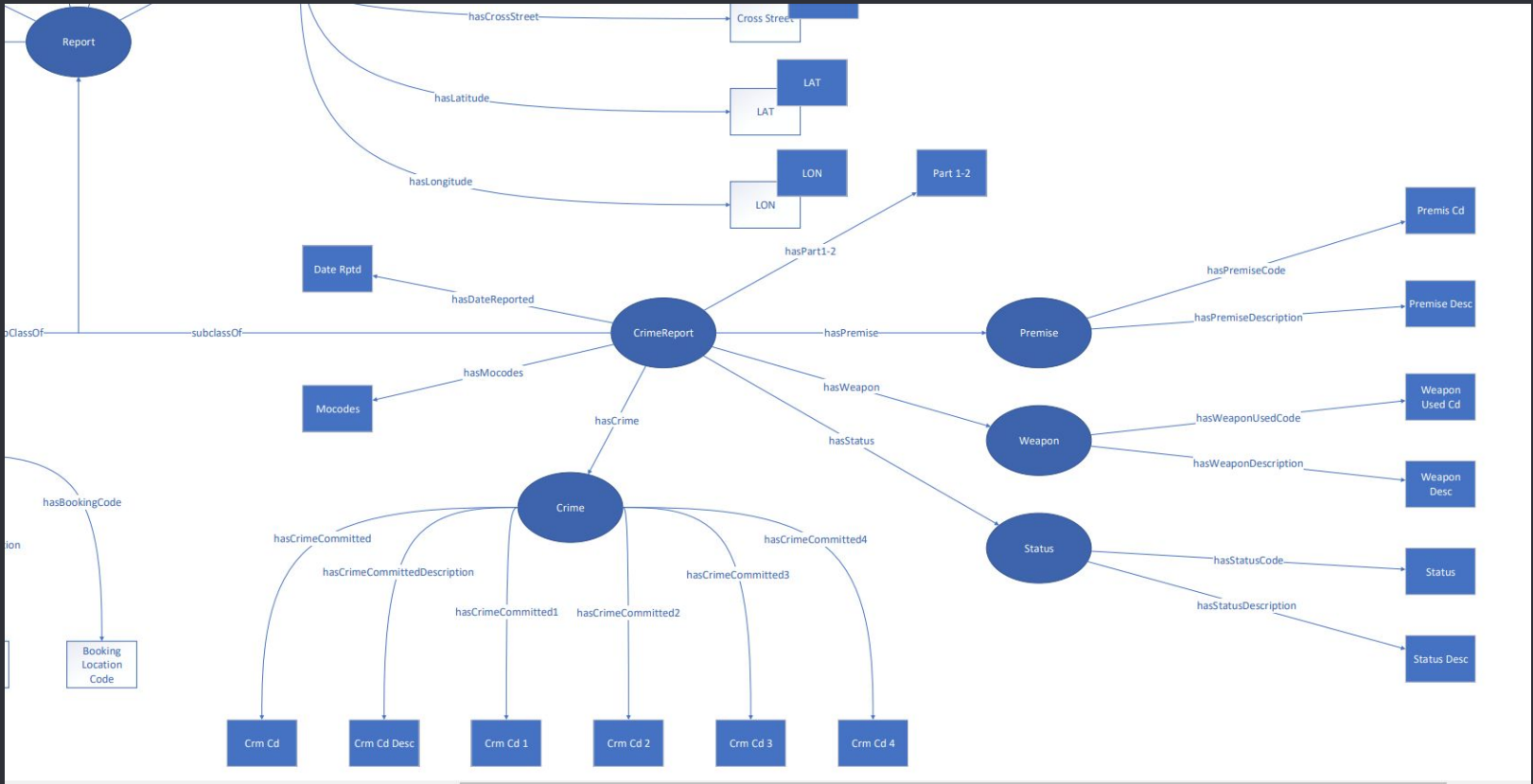
Source Code

Requirements

- Python
- Request
- RdfLib
- Pandas
- Auto-Py-to-Exe

Code Snippet 1: Downloading datasets

```python
def _get_dataset(self, url):
    """Downnalod dataset and decode them as csv

    Args:
        url (string): URL to download resources

    Returns:
        [string]: dataset formatted as csv
    """

    isAvailable = self._validate_url(url)

    if isAvailable:
        print("INFO: Downloading dataset from \""+url+"\"...")
        with closing(requests.get(url+".csv?$limit=99999999",stream=True)) as response:
            decoded_dataset = [line.decode('utf-8') for line in response.iter_lines()]
            dataset = csv.reader(decoded_dataset, delimiter=',')
            return list(dataset)
```

Code Snippet 2.0: Adding Arrest Reports data to the RDF graph

```python
def _add_arrest_reports_dataset_to_graph(self, arrest_reports_dataset, graph, namespace):
    """Add arrest reports dataset to the RDF graph

    Args:
        arrest_reports_dataset (string): a CSV contains arrest reports with well-formatted datapoints
        graph (Graph): an RDF graph
        namespace (string): base namespace for all resources

    Returns:
        [Graph]: an RDF graph contains data from the arrest report dataset
    """
    print("INFO: Add arrests dataset to graph...")

    for i in range(1, len(arrest_reports_dataset)):
        number_report = len(list(graph.subject_objects(predicate=namespace["hasID"])))
        # add base arrest report that inherits from the report class
        graph.add((namespace["report" + str(number_report)], RDF.type, namespace["ArrestReport"]))
        graph.add((namespace["report" + str(number_report)], namespace["hasID"], Literal(arrest_reports_dataset[i][0], datatype=XSD.integer)))
        graph.add((namespace["report" + str(number_report)], namespace["hasDate"], Literal(arrest_reports_dataset[i][2], datatype=XSD.date)))
        graph.add((namespace["report" + str(number_report)], namespace["hasTime"], Literal(arrest_reports_dataset[i][3], datatype=XSD.time)))
        graph.add((namespace["report" + str(number_report)], namespace["hasReporType"], Literal(arrest_reports_dataset[i][1], datatype=XSD.string)))
        graph.add((namespace["report" + str(number_report)], namespace["hasArrestType"], Literal(arrest_reports_dataset[i][12], datatype=XSD.string)))
        graph.add((namespace["report" + str(number_report)], namespace["hasDispositionDescription"], Literal(arrest_reports_dataset[i][15], datatype=XSD.string)))
```

Code Snippet 2.1: Adding Arrest Reports data to the RDF graph

```python
# set up the people class
people_age = set(graph.subjects(predicate = namespace["hasAge"], object=Literal(arrest_reports_dataset[i][7], datatype=XSD.integer)))
people_sex = set(graph.subjects(predicate = namespace["hasSex"], object=Literal(arrest_reports_dataset[i][8], datatype=XSD.string)))
people_decendent = set(graph.subjects(predicate = namespace["hasDescendent"], object=Literal(arrest_reports_dataset[i][9], datatype=XSD.string)))
# returns the number of persons for the naming scheme
number_person = len(list(graph.subject_objects(predicate=namespace["hasAge"])))
# give all the triples with age, sex, and decendent
person = list(people_age & people_sex & people_decendent)
if(len(person) == 0): |
    graph.add((namespace["Person" + str(number_person)], RDF.type, namespace["Person"]))
    graph.add((namespace["Person" + str(number_person)], namespace["hasAge"], Literal(arrest_reports_dataset[i][7], datatype=XSD.integer)))
    graph.add((namespace["Person" + str(number_person)], namespace["hasSex"], Literal(arrest_reports_dataset[i][8], datatype=XSD.string)))
    graph.add((namespace["Person" + str(number_person)], namespace["hasDescendent"], Literal(arrest_reports_dataset[i][9], datatype=XSD.string)))
    person = namespace["Person" + str(number_person)]
else:
    person = person[0]
```

```python
#add to report
graph.add((namespace["report" + str(number_report)], namespace["hasPerson"], person))
graph.add((namespace["report" + str(number_report)], namespace["hasLocation"], location))
graph.add((namespace["report" + str(number_report)], namespace["hasBooking"], booking))
graph.add((namespace["report" + str(number_report)], namespace["hasCharge"], charge))
```

Code Snippet 3: Adding Crime Reports data to the RDF graph

```python
158  # Premise
159  premiseCodeList = df['PremiseCode']
160  premiseDescriptionList = df['PremiseDescription']
161
162  for i in range(0,len(premiseCodeList)):
163    premiseCode = set(g.subjects(predicate = namespace["hasPremiseCode"], object=Literal(premiseCodeList[i], datatype=XSD.integer)))
164    premiseDesc = set(g.subjects(predicate = namespace["hasPremiseDescription"], object=Literal(premiseDescriptionList[i], datatype=XSD.string)))
165
166    # returns the number of location for the naming scheme
167    number_premise = len(list(g.subject_objects(predicate=namespace["hasPremiseCode"])))
168
169    # give all the triples with age, sex, and decendent
170    premises = list(premiseDesc & premiseCode)
171
172    if(len(premises) == 0):
173      # add to Location
174      g.add((namespace["Premise" + str(number_premise)], RDF.type, namespace["Premise"]))
175      g.add((namespace["Premise" + str(number_premise)], namespace["hasPremiseCode"], Literal(premiseCodeList[i], datatype=XSD.integer)))
176      g.add((namespace["Premise" + str(number_premise)], namespace["hasPremiseDescription"], Literal(premiseDescriptionList[i], datatype=XSD.string)))
177      premise = namespace["Premise" + str(number_premise)]
178    else:
179      premise = premises[0]
180    g.add((namespace["Report" + str(i)], namespace["hasPremise"], premise))
```

Technical Challenges

Challenge 1: Synonym Labels of Data

Problem:

Even though both datasets come from a single source (https://data.lacity.org/), there are multiple labels used to define similar data.

Ex: Report ID (Arrest Reports) == DR_NO (Crime Reports)

Solution:

We examines all data and unify similar data under a single label.

Ex: Report ID (Arrest Reports) == DR_NO (Crime Reports) == ID (RDF Graph)

Challenge 2: Inconsistent Data Format

Problem:

    In both datasets, some data do not match the format of XML Schema.

Ex:

    XSD: time ⇒ hh:mm:ss

    Data from dataset ⇒ "0935"

Solution:

    We format all data to match XML Schema format

Ex: "0935" ⇒ 09:35:00

Challenge 3: Missing Data

Problem:

    Since datasets are generated by real life events, there are a lot of missing data values.

Solution:

    For completeness' sake, we will represent those missing values with black nodes