TESINA OPEN DATA
MERCATI STORICI SICILIANI

Valeria Maria Apelle (0683019)
Pietro Attardi (0679170)

Indice

1 Idea di progetto

L'idea di progetto è stata quella di porre un'analisi sui mercati storici presenti in Sicilia.L'obiettivo è quello di fornire ai turisti, o a chi di interesse, uno strumento attraverso il quale è possibile ottenere informazioni in tempo reale sui mercati storici presenti nel territorio siciliano. Per far questo si è partiti da un dataset di livello a 3 stelle (file GeoJSON) per giungere alla realizzazione di un file a 5 stelle (LOD). Inoltre è stato creato un BOT telegram che mostra le informazioni di ciascun mercato storico e la propria locazione. Il fruitore del servizio può infatti ricercare ciascun mercato storico specificando la provincia, il comune o semplicemente la denominazione del mercato presso cui ci si vuole recare. Inoltre il BOT fornisce anche delle informazioni sulle condizioni meteo, come temperatura attuale/percepita/minima e massima, della località in cui è situato ciascun mercato. Ciò è stato realizzato utilizzando un'apposita Weather API.

2 Dataset

Per realizzare questo progetto è stato utilizzato un dataset pubblicato dalla regione Sicilia in formato GeoJSON, che descrive i mercati storici siciliani e la loro posizione nelle varie località siciliane. Viene descritta la tipologia di ciascun mercato e una breve descrizione. È possibile ottenere il dataset completo presso il seguente sito web:
`http://dati.regione.sicilia.it/dataset/mercati-storici-siciliani`.

2.1 Analisi e operazioni di pulizia sul dataset

Il Dataset pubblicato dalla regione Sicilia considerato è in formato GeoJSON. L'obiettivo di questa fase è, dopo aver applicato operazioni di pulizia sul dataset per eliminare tutte le informazioni non necessarie per l'analisi in questione e correggere errori presenti, avere un file di output privo di errori e con dati coerenti.

2.1.1 Operazioni di pulizia del Dataset

In questa fase è stato utilizzato il software **Open Refine**, mediante il quale è stato creato un nuovo progetto con il nostro file come input, per ottenere come output un file completo e privo di errori. Il file di output però è stato esportato in formato CSV. Inizialmente nel dataset erano presenti alcuni errori di battitura (e.x. "Agrifgento"), che sono stati opportunamente corretti. Inoltre è stato riorganizzato il contenuto del file, alcune colonne sono state rimosse e altre sono state riorganizzate (e.x. split della colonna "coordinates" in due colonne "longitudine" e "latitudine").

2.2 Visualizzazione dataset su Google Maps

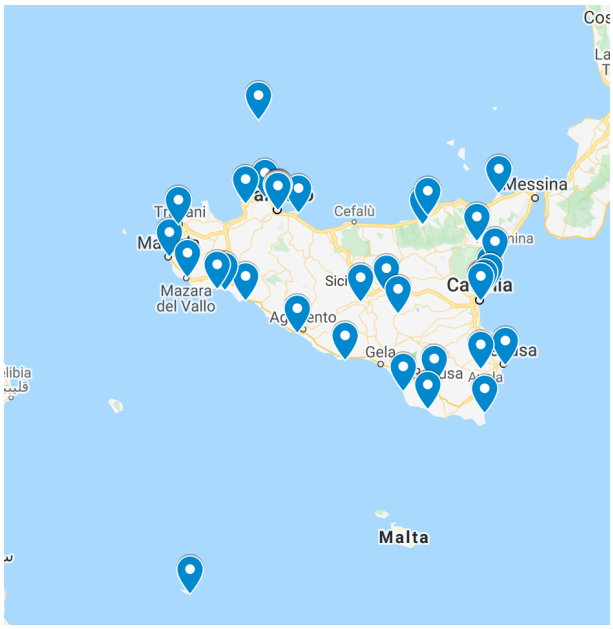


Figure 1: Mappa mercati storici

Essendoci note le coordinate in cui si trova ciascun mercato, è stato possibile creare un'apposita mappa, da poter osservare in dettaglio cliccando sul seguente link: `https://www.google.com/maps/d/u/0/edit?mid=1CGvD17oPHGKdewHH83yzq7kEm7EBFK3d&usp=sharing`. Per realizzare ciò è stato necessario esportare il file di output da formato GeoJSON a CSV, in modo tale che **Google Maps** potesse generare la mappa. Ciò è stato possibile grazie alle operazioni effettuate con il tool **Open Refine**.

2.3 Licenza

La licenza con cui viene rilasciato il dataset considerato è di tipo Creative Commons Attribuzione (CC-BY), la quale permette di distribuire, modificare e sviluppare anche commercialmente il dataset, riconoscendo sempre l'autore originale.

3 Pipeline di elaborazione

L'obiettivo di questa fase è quello di creare un file machine readable (livello 4 e a seguire 5 stelle) iniziando dalla creazione di un'ontologia relativa al dataset considerato. L'ontologia relativa al dataset in questione è stata generata mediante delle triplette (Soggetto-Predicato-Oggetto) in RDF creando così un file turtle, il quale è stato aperto e visualizzato in dettaglio con il software Protegè. Le classi definite sono: HistoricalMarket e GeoPoint con le rispettive proprietà.

@prefix rdfs: <http://www.w3.org/2000/01/rdf−schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#> .

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rso : <http://www.regionesiciliana.it/ontology/> .

rso:HistoricalMarket a rdfs:Class .
rso:GeoPoint a rdfs:Class .

rso:name a rdf:Property .
rso:latitude a rdf:Property .
rso:longitude a rdf:Property .
rso:localization a rdf:Property .
rso:type a rdf:Property .
rso:description a rdf:Property .
rso:town a rdf:Property .
rso:province a rdf:Property .
rso:id a rdf:Property .


rso:name rdfs:domain rso:HistoricalMarket .
rso:latitude rdfs:domain rso:GeoPoint .
rso:longitude rdfs:domain rso:GeoPoint .
rso:type rdfs:domain rso:HistoricalMarket .
rso:localization rdfs:domain rso:HistoricalMarket .
rso:description rdfs:domain rso:HistoricalMarket .
rso:id rdfs:domain rso:HistoricalMarket .

rso:isLocated rdfs:domain rso:HistoricalMarket ; rdfs:range rso:GeoPoint .
```

3.1 Generazione file RDF (livello 4 stelle)

In questa fase viene descritto come è stato convertito il file di input dal JSON ad un file a livello superiore (4 stelle) in grado di fornire dei servizi efficienti. La seguente funzione è stata utilizzata per creare l'URI con la corretta formattazione.

```
def urify(ns, testo):
    testo=testo.replace(" ","_").replace(".","")
    return ns+urllib.parse.quote(testo)
```

Si crea l'URI della risorsa da utilizzare per l'ontologia relativa al nostro dataset.

```
g = Graph()
rso = Namespace("http://www.regionesiciliana.it/ontology/")
base_uri="http://www.regionesiciliana.it/resource/"
g.bind("rso", rso)
```

Dopo l'apertura del file CSV contenente il dataset, si procede con la lettura riga per riga del file e si eseguono le seguenti operazioni sui dati:

```
#Creazione URI necessarie
    uri_HistoricalMarket=urify(base_uri , str(row['ObjectID']))
    uri_GeoPoint= urify(base_uri, "coordinates" + str(row['ObjectID']))
```

Dopo la creazione delle URI, si procede con la definizione delle triple (Soggetto - Predicato - Oggetto)

```
#Impostazione localizzazione, denominazione, coordinate, descrizione di un singolo mercato
    g.add([URIRef(uri_HistoricalMarket), rso.localization, Literal(row['Localizzazione'])])
    g.add([URIRef(uri_HistoricalMarket), rso.name, Literal(row['Denominazione'])]])
    g.add([URIRef(uri_HistoricalMarket) , rso.isLocated, URIRef(uri_GeoPoint)])
    g.add([URIRef(uri_GeoPoint), rso.latitude, Literal(row['Latitudine'])])
    g.add([URIRef(uri_GeoPoint), rso.longitude, Literal(row['Longitudine'])])
    g.add([URIRef(uri_HistoricalMarket), rso.description, Literal(row['Descrizione'], lang='it' )])
```

3.2 Generazione file LOD (livello 5 stelle)

In questa fase, dopo aver creato l'ontologia relativa al dataset utilizzato, si procede con l'interlink dei dati con la base di conoscenza *dbpedia*.
Per la fase di interlinking, per ogni proprietà, si controlla se già esiste una risorsa analoga nella base di conoscenza di dbpedia: se sì, si procede l'interlinking con la risorsa analoga esistente. Per verificare che la risorsa cercata da collegare esiste, sono state effettuate delle query ASK alla base di conoscenza mediante la funzione **existsInDBPedia()** descritta successivamente.

```
#Impostazione Provincia
#verifichiamo se esite la risorsa per la provincia su dbpedia
province_boolean = existsInDBPedia(dictionary, str(row["Provincia"]))
if (province_boolean):
    formatted_resource_name = urify("",str(row["Provincia"])).title().replace(" ", "_")
    dbpedia_asked_uri = urify("http://dbpedia.org/resource/",formatted_resource_name)
    print(uri_HistoricalMarket + " rso:province " + dbpedia_asked_uri)
    g.add([URIRef(uri_HistoricalMarket) , rso.province, URIRef(dbpedia_asked_uri)])
else:
#non esiste su dbpedia, viene associato un literal
    print(uri_HistoricalMarket + " rso:province " + uri_MarketProvince)
    g.add([URIRef(uri_HistoricalMarket) , rso.province, Literal(row['Provincia'])])
#Controlliamo che esista la risorsa mercato storico su dbpedia
boolean = existsInDBPedia(dictionary, str(row["Denominazione"]))
dbpedia_asked_uri = urify("http://dbpedia.org/resource/",str(row["Denominazione"]))
if(boolean):
    print(uri_HistoricalMarket + " owl:sameAs " + dbpedia_asked_uri)
    g.add([URIRef(uri_HistoricalMarket), OWL.sameAs, URIRef(dbpedia_asked_uri)])
```

La seguente funzione utilizza un metodo di programmazione dinamica basato su una lookup table, con lo scopo di minimizzare il numero di query ASK alla base di

conoscenza. È stato implementato questo metodo con l'utilizzo di una struttura dati di tipo dizionario, in cui vengono memorizzare le informazioni fino a quel momento acquisite.

```
def existsInDBPedia(dictionary,resource_name):
#serve solamente per avere il formato desiderato
    formatted_resource_name = urify("", resource_name).title().replace(" ", "_")
    dbpedia_asked_uri = urify("http://dbpedia.org/resource/",formatted_resource_name)
    resource = dictionary.get(formatted_resource_name)
    if (resource != None):
        return resource
    else:
        sparql.setQuery("ASK { <http://dbpedia.org/resource/"
                        + formatted_resource_name + "> ?p ?o }")
        sparql.setReturnFormat(JSON)
        results = sparql.query().convert()
        dictionary[formatted_resource_name] = results["boolean"]
        return results["boolean"] #restituisce "True" o "False"
```

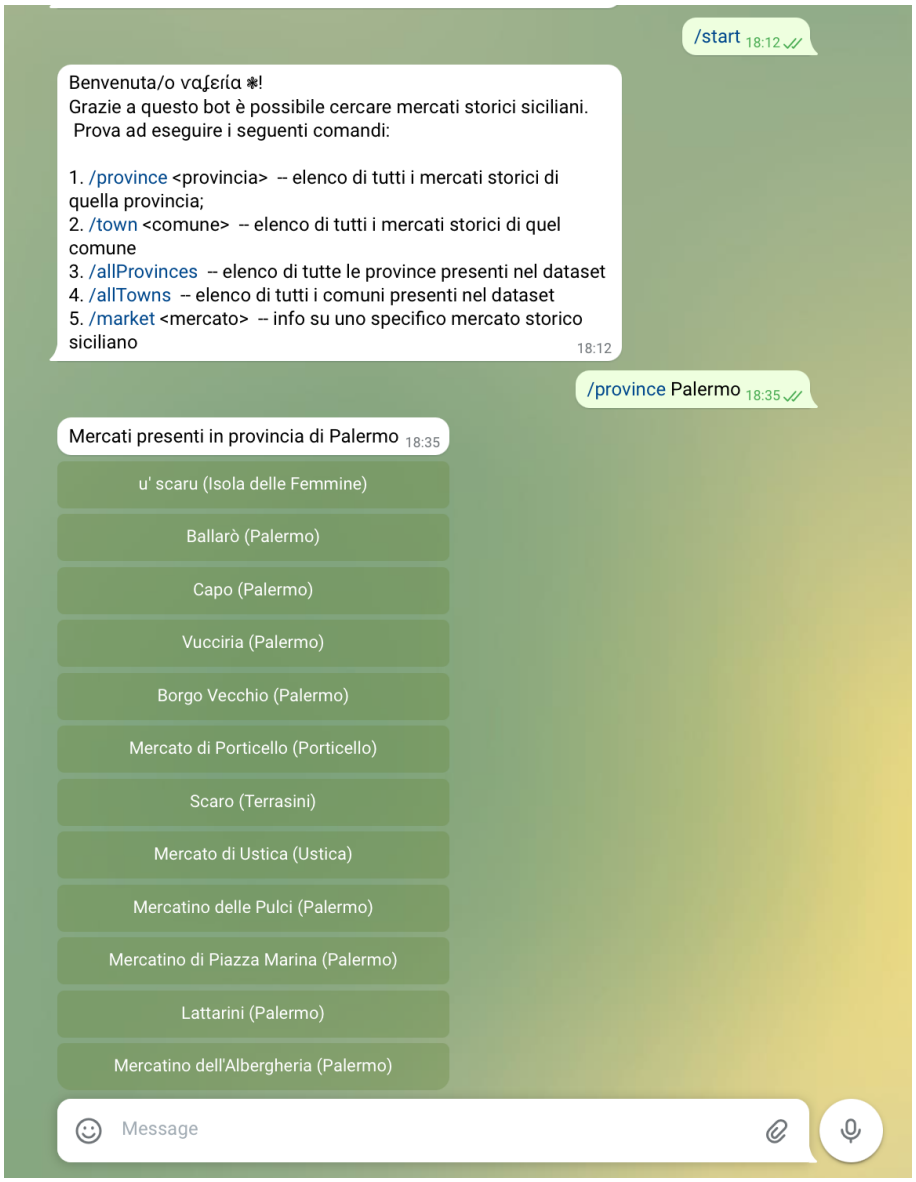
Il risultato di questa fase è un file di livello 5 stelle, *linked open data*.

4 BOT Telegram

Un'applicazione del dataset consiste nella realizzazione di un Bot telegram. Il Bot permette di ottenere informazioni sui mercati storici siciliani, filtrando i risultati di ricerca per provincia, comune e denominazione del mercato che si vuole visitare. Il Bot è stato realizzando mediante il linguaggio di programmazione Python, con l'ausilio della libreria **pyTelegramBotAPI**.

4.1 Struttura BOT e funzioni

Questo Bot, dopo essere stato avviato, richiede di effettuare una ricerca sui mercati storici siciliani da poter visitare, mediante una ricerca filtrata.



Dopo aver selezionato una opzione di ricerca dal menù, è possibile cliccare e in seguito visualizzare i dettagli relativi ad un mercato storico selezionato. Inoltre sarà possibile ottenere informazioni dettagliate sulla condizione meteo attuale in quella località, ottenute mediante la weather API, passando come parametro le coordinate del mercato storico selezionato.