Documentazione del Progetto:

Dataset Builder Ontology

Vincenzo Buttari

5 febbraio 2025

Indice

Dataset Builder Ontology

Nome: Vincenzo Buttari

GitHub:
https://github.com/zeltarave/dataset_builder_ontology

Matricola: 776274

5 febbraio 2025

Prefazione

Questo documento descrive in dettaglio il progetto **Dataset Builder Ontology e Modello Predittivo**. Il progetto nasce con l'obiettivo di:

- Popolare automaticamente un'ontologia OWL con dati realistici.
- Estrarre dati strutturati dall'ontologia per costruire un dataset.
- Applicare tecniche di NLP per arricchire le feature testuali dei corsi.
- Addestrare modelli predittivi per la classificazione, confrontando un approccio base con uno ottimizzato mediante Grid Search e cross-validation.
- Fornire interfacce sia a riga di comando (CLI) che tramite una web app sviluppata con Flask.

Il documento illustra la struttura del progetto, i moduli principali, gli esempi di esecuzione e le possibili direzioni di sviluppo futuro.

Capitolo 1

Struttura del Progetto

La struttura delle cartelle è organizzata in modo modulare per separare le responsabilità.
La gerarchia delle cartelle è la seguente:

```
dataset_builder_ontology/
 data/
    ontology.owl       % Ontologia OWL popolata (output)
    dataset.csv        % Dataset generato (output)
 src/
    owl/
       logger_config.py         % Configurazione del logging
       populate_ontology.py     % Popolamento dell'ontologia
       ontology_manager.py      % Classe OntologyManager per gestione ontologia
       dataset_generator.py     % Estrazione dei dati dall'ontologia
    predictive_model/
       predictive_model.py      % Modello predittivo base
       grid_search_model.py     % Ottimizzazione tramite Grid Search
       compare_models.py        % Confronto tra modelli
    app.py                       % Interfaccia web con Flask
    cli.py                       % Interfaccia a riga di comando
 templates/
    index.html
    dataset.html
    train.html
 requirements.txt                 % Dipendenze del progetto
 README.md
```

Capitolo 2

Gestione dell'Ontologia

2.1 La Classe `OntologyManager`

Il modulo `ontology_manager.py` definisce la classe `OntologyManager`, che incapsula le seguenti operazioni:

- **load()**: Carica l'ontologia dal file specificato o, se il file non esiste, crea una nuova ontologia.
- **populate()**: Definisce classi (ad es. `Person` e `Course`), proprietà (ad es. `has_name`, `has_age`, `teaches`, `takes`, `course_title`, `course_description`) e popola l'ontologia con individui generati automaticamente.
- **reason()**: Esegue il ragionamento sull'ontologia tramite il ragionatore di `Owlready2`.
- **extract_person_data()** e **extract_course_data()**: Estrae rispettivamente i dati delle classi `Person` e `Course`.

2.1.1 Esempio di Utilizzo

```
from owl.ontology_manager import OntologyManager
import os

ontology_file = os.path.abspath(os.path.join("data", "ontology.owl"))
om = OntologyManager(ontology_file)
om.load()
om.populate()
om.reason()

person_data = om.extract_person_data()
course_data = om.extract_course_data()
```

Listing 2.1: Utilizzo della classe OntologyManager

Capitolo 3

Modelli Predittivi e Ottimizzazione

3.1 Modello Base

Il modulo `predictive_model.py` addestra un modello di regressione logistica con parametri fissi utilizzando:

- Le feature `age` e il numero di corsi seguiti.
- Un target `teacher` definito in base alla presenza di corsi insegnati.

3.2 Ottimizzazione tramite Grid Search

Il modulo `grid_search_model.py` implementa una procedura di Grid Search con 5-fold cross-validation per:

- Ottimizzare i parametri del modello (ad es. il parametro C e il tipo di penalizzazione: `l1` o `l2`).
- Trovare la configurazione ottimale in base all'accuracy media ottenuta durante la cross-validation.

Esempio di addestramento con Grid Search:

```
from predictive_model.grid_search_model import train_with_grid_search

dataset_path = "data/dataset.csv"
best_model = train_with_grid_search(dataset_path, random_state=42)
```

Listing 3.1: Addestramento del modello con Grid Search

3.3 Confronto delle Prestazioni

Il modulo `compare_models.py` confronta le prestazioni tra il modello base e quello ottimizzato:

- Suddivide il dataset in training e test set.

- Calcola l'accuracy e il classification report per ciascun modello.
- Stampa a video i risultati per facilitarne il confronto.

Capitolo 4

Interfacce Utente e Esempi di Esecuzione

4.1 Interfaccia a Riga di Comando (CLI)

Il file `src/cli.py` fornisce un'interfaccia a riga di comando per eseguire le seguenti operazioni:

- `populate`: Popola l'ontologia.
- `extract`: Estrae il dataset dall'ontologia.
- `train`: Addestra il modello predittivo base.
- `grid_search`: Addestra il modello predittivo ottimizzato tramite Grid Search.
- `compare`: Confronta le prestazioni dei modelli.

4.1.1 Esempi di Esecuzione dal CLI

Per eseguire le operazioni via CLI, aprire il terminale nella root del progetto ed eseguire:

```
# Popola l'ontologia:
python src/cli.py populate

# Estrae il dataset e lo salva in data/dataset.csv:
python src/cli.py extract

# Addestra il modello base:
python src/cli.py train

# Addestra il modello ottimizzato con Grid Search:
python src/cli.py grid_search

# Confronta le prestazioni tra il modello base e quello ottimizzato:
python src/cli.py compare
```

4.2 Interfaccia Web

L'applicazione web (definita in `app.py`) consente di eseguire le stesse operazioni tramite un'interfaccia grafica. Per avviare l'app:

```
python app.py
```

Successivamente, aprire il browser e navigare all'indirizzo:

```
http://127.0.0.1:5000/
```

Da qui è possibile interagire con l'applicazione per visualizzare il dataset estratto e i risultati dell'addestramento del modello.

Capitolo 5

Istruzioni per l'Installazione

5.1 Prerequisiti

- Python 3.6 o superiore.
- Le dipendenze elencate in `requirements.txt` (installabili con `pip install -r requirements.txt`).

5.2 Esecuzione

- Per eseguire il progetto via CLI, utilizzare i comandi indicati nella sezione "Esempi di Esecuzione dal CLI".
- Per avviare l'interfaccia web, eseguire `python app.py`
  e accedere all'indirizzo `http://127.0.0.1:5000/`.

Capitolo 6

Conclusioni e Sviluppi Futuri

Il progetto **Dataset Builder Ontology** rappresenta una solida base per la costruzione di dataset a partire da ontologie e per l'addestramento di modelli predittivi. Possibili sviluppi futuri includono:

- L'implementazione di tecniche di interpretabilità dei modelli (es. SHAP o LIME).
- L'integrazione di dashboard interattive per il monitoraggio delle prestazioni.
- Miglioramenti nella gestione e validazione dei dati nell'ontologia.
- L'automazione del processo di test e deployment attraverso sistemi di integrazione continua (CI/CD).

Bibliografia

[1] R. Pelletier, *Owlready2: A module for ontology-oriented programming in Python.* `https://pypi.org/project/Owlready2/`

[2] Scikit-learn Developers, *Scikit-learn: Machine Learning in Python.* `https://scikit-learn.org/`

[3] Scikit-learn Developers, *GridSearchCV Documentation.* `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`

[4] Armin Ronacher, *Flask: A lightweight WSGI web application framework.* `https://flask.palletsprojects.com/`