

# Documentazione del Progetto: Sistema di Monitoraggio della Qualità dell'Aria, Airlytics

## Gruppo di lavoro




Achille Carbonara, 778109, a.carbonara30@studenti.uniba.it  
Federico Di punzio, 775488, f.dipunzio3@studenti.uniba.it  
Domenico Marsico, 778283, d.marsico4@studenti.uniba.it

Repository: <https://github.com/Achille3287/Progetto-Icon-24-25>

Anno Accademico: 2024–2025

## 📌 Introduzione

Il progetto nasce per costruire un **sistema intelligente per il monitoraggio della qualità dell'aria**, fondato su tre paradigmi dell'Ingegneria della Conoscenza:

-  **Modellazione probabilistica** con HMM
-  **Rappresentazione logica** (proposizionale e relazionale)
-  **Ontologia semantica** per il ragionamento concettuale

Tale sistema è in grado di:

- Simulare scenari ambientali
- Stimare probabilisticamente la qualità dell'aria
- Effettuare inferenze simboliche
- Rispondere a interrogazioni semantiche sul dominio

## Argomento 1 – Modello di previsione con HMM

Il **modello di Markov nascosto (HMM)** è una rappresentazione probabilistica di un sistema con:

- Stati nascosti  $S = \{\text{buona, moderata, cattiva}\}$
- Osservazioni  $O = \{\text{PM10\_basso, PM10\_medio, PM10\_alto}\}$
- Matrici di:
  - **Transizione:**  $P(S_t|S_{t-1})P(S_t \mid S_{t-1})P(S_t|S_{t-1})$
  - **Emissione:**  $P(O_t|S_t)P(O_t \mid S_t)P(O_t|S_t)$

 **Riferimento:** Dispense ICon, Capitolo 9

## Applicazione al progetto

Il sistema genera sequenze di osservazioni (dati simulati) e usa il **filtro HMM** per stimare la distribuzione sugli stati nascosti:

$$P(S_t \mid O_{1:t}) = \alpha \cdot P(O_t \mid S_t) \cdot \sum_{S_{t-1}} P(S_t \mid S_{t-1}) \cdot P(S_{t-1} \mid O_{1:t-1})$$

Implementato in markov\_chain.py, con simulazione e inferenza.

## Argomento 2 – Rappresentazione Logica Proposizionale (Cap. 4)

### Teoria

La **logica proposizionale** permette di esprimere fatti e regole in forma di **clausole di Horn**:

$$p \leftarrow q_1 \wedge q_2 \wedge \dots \wedge q_n$$

 **Riferimento:** Dispense ICon, Capitolo 4

- Rappresentazione simbolica
- Inferenza tramite forward chaining / backward chaining

### Applicazione al progetto

- Nel modulo **KB simbolica** (kb\_engine.py), abbiamo:
- prolog
- Copia codice
- `qualita_buona(X) :- PM10_basso(X), NO2_basso(X).`

Esempio:

```
PM10_basso(stazione1).  
NO2_basso(stazione1).  
⇒ qualita_buona(stazione1)
```

## Argomento 3 – Rappresentazione Relazionale (Cap. 5)

### Teoria

La **logica del primo ordine (FOL)** consente:

- **Quantificatori** ( $\forall$ ,  $\exists$ )
- **Relazioni** tra entità
- **Classi e oggetti** modellati formalmente

 **Riferimento:** Dispense ICon, Capitolo 5

- Differenza tra proposizionale e relazionale
- Descrizione di domini con relazioni complesse

### **Applicazione al progetto**

Il modulo **ontologia OWL** (ontology/):

- Definisce classi: Stazione, Inquinante, QualitàAria
- Proprietà: hasMeasurement, hasValue
- Relazioni tra entità:

Stazione  $\sqsubseteq \exists \text{hasMeasurement. QualitàAria}$

Query di esempio:

```
SELECT ?x WHERE {  
  ?x rdf:type :Stazione .  
  ?x :hasLevel :Cattiva .  
}
```

### **Architettura del sistema**

[Dataset PM10 + simulazione]

↓

[HMM] → Predizione dello stato

↓

[KB Logica] → Inferenza su regole ambientali

↓

[Ontologia OWL] → Query e ragionamento semantico

### **Esecuzione del progetto**

#### **Requisiti**

python >= 3.9

pip install numpy owlready2 pytholog

## Setup

```
python -m venv venv
```

```
source venv/bin/activate
```

```
pip install -r requirements.txt
```

## Avvio dei moduli

```
python KB/markovChain/markov_chain.py    # HMM
```

```
python KB/kb_engine.py                   # KB simbolica
```

```
python ontology/semantic_query.py        # Ontologia
```

## 📁 Struttura del progetto

Progetto-Icon-24-25-main/

```
|
|
|└─ dataset/           # Dati ambientali grezzi (es. PM10.csv)
|
|
|└─ KB/
|  |└─ markovChain/
|  |  |└─ markov_chain.py  # Simulatore HMM e filtro
|  |  └─ libs/
|  |      └─ HMM.py        # Libreria HMM base
|  └─ kb_engine.py        # Motore logico per KB (regole + inferenza)
|
|
|└─ ontology/
|  |└─ air_quality.owl    # File OWL definito con Protégé
|  └─ semantic_query.py   # Interrogazioni semantiche con Owlready2
|
|
|└─ interface/          # (Opzionale) Interfaccia utente CLI/testuale
|
|
|└─ utils/               # Funzioni di supporto (lettura dati, parsing)
|
```

```
└─ main.py          # Entry point per l'integrazione dei moduli
|
└─ README.md        # Descrizione progetto e istruzioni
└─ requirements.txt  # Librerie necessarie
└─ .gitignore       # File esclusi dal controllo versione
```

## Applicazioni future

Il progetto, pur già funzionante, può essere **esteso o adattato a diversi contesti applicativi**, sia accademici che industriali. Di seguito, alcune direzioni possibili:

### 1. Integrazione con dati reali (API o IoT)

Attualmente i dati sono simulati o statici. È possibile:

- Collegare sensori reali (PM10, NO<sub>2</sub>, O<sub>3</sub>)
- Usare API open data (es. ARPA, OpenAQ)
- Aggiornare dinamicamente la KB e l'ontologia con dati live

### 2. Apprendimento supervisionato delle matrici HMM

Le probabilità di transizione/emissione ora sono fissate manualmente. In futuro:

- Si può usare **apprendimento supervisionato** per stimarle da dataset reali
- Adottare **algoritmi di training HMM** (es. algoritmo di Baum-Welch)

### 3. Aggiunta di un'interfaccia utente intelligente

Possibile sviluppo di:

- Un **chatbot** (es. con NLU) che interroga la KB e l'ontologia in linguaggio naturale
- Una **dashboard web** per visualizzare gli stati previsti, le regole attivate e la rete semantica

## Riferimenti bibliografici

- [1] Dispense Ingegneria della Conoscenza – UniBA (2024)
- [2] Rabiner, L. R. – Hidden Markov Models
- [3] Russell & Norvig – Artificial Intelligence: A Modern Approach

[4] Baader et al. – Description Logic Handbook