**Plant-Aid-KBS: Sistema Diagnostico Ibrido per la Cura delle Piante**

Gruppo di lavoro

Donato Cancellara, matricola 776972, donato.cancellara03@gmail.com

URL Repository

https://github.com/donato2403/plant-aid-kbs

**AA 2024-2025**

**Introduzione**

Il dominio di interesse di questo progetto è la **fitopatologia diagnostica**, specificamente applicata a un contesto hobbistico e semi-professionale. L'obiettivo è assistere utenti non esperti (es. coltivatori amatoriali) nell'identificazione di malattie comuni per tre piante specifiche: **Olivo (Olea europaea)**, **Rosa (Rosa spp.)** e **Basilico (Ocimum basilicum)**. La diagnosi in questo campo è complessa a causa della sovrapposizione dei sintomi (es. l'ingiallimento fogliare è comune a molteplici patologie) e della necessità di considerare fattori contestuali (es. stagione, condizioni ambientali).

**Sommario del KBS**

Per affrontare questa complessità, è stato sviluppato "Plant-Aid-KBS", un **Sistema Basato sulla Conoscenza (KBS) ibrido**. Il sistema orchestra quattro moduli distinti per integrare conoscenza formale, ragionamento euristico e apprendimento statistico, aderendo al modello "ML+OntoBK"4.

Come descritto nella proposta 5, il flusso diagnostico (main\_cli.py) interroga tre motori in parallelo:

1. Un **Motore Simbolico (Datalog/ASP)** per applicare la conoscenza esperta.
2. Un **Classificatore Statistico (SVM)** per il riconoscimento di pattern dai dati.
3. Un **Motore Probabilistico (Rete Bayesiana)** per la gestione dell'incertezza.

I risultati vengono poi aggregati da un orchestratore che calcola un "Fattore di Confidenza" finale e arricchisce la diagnosi con dati estratti da un'**Ontologia OWL** formale.

**Elenco argomenti di interesse**

Il progetto dimostra competenze sui seguenti argomenti del programma di Ingegneria della Conoscenza:

1. **Rappresentazione della Conoscenza e Ragionamento Automatico**: Modellazione della conoscenza di dominio tramite Ontologie (OWL) e formalizzazione della conoscenza euristica tramite regole logiche (Datalog/ASP).
2. **Apprendimento Supervisionato (SVM)**: Utilizzo di Support Vector Machines per un task di classificazione basato su feature.
3. **Ragionamento con Conoscenza Incerta (Reti Bayesiane)**: Modellazione dell'incertezza diagnostica e inferenza probabilistica tramite Reti Bayesiane.

**Sezione 1: Rappresentazione della Conoscenza e Ragionamento Automatico**

**Sommario**

Per rappresentare la conoscenza del sistema, è stato adottato un approccio ibrido che separa la conoscenza tassonomica da quella inferenziale6.

1. **Conoscenza Tassonomica (TBox e ABox)**: Un'ontologia formale (data/plant\_care.owl) definisce la struttura gerarchica del dominio. Definisce classi (es. Pianta, Malattia, Sintomo), sottoclassi (es. Olivo ⊆ Pianta) e proprietà (es. colpisce, presenta\_sintomo, richiede\_trattamento). L'ontologia è anche popolata con individui (es. Occhio\_di\_Pavone, Trattamento\_Rame) che rappresentano la conoscenza fattuale.
2. **Conoscenza Euristica (Base di Regole)**: Una base di conoscenza separata (kbs\_engine/datalog\_engine.py) formalizza la logica diagnostica degli esperti in formato ASP (Answer Set Programming), compatibile con Datalog.

**Strumenti utilizzati**

* **Owlready2**: Libreria Python utilizzata per caricare, interrogare e manipolare l'ontologia OWL (ontology\_manager.py).
* **Clingo (Potassco)**: Solver ASP utilizzato dal datalog\_engine.py per eseguire il ragionamento automatico (inferenza) sulla base di regole.

**Decisioni di Progetto**

La scelta architetturale chiave è stata quella di non usare l'ontologia per il ragionamento diagnostico pesante (es. tramite SWRL), ma di usarla come un *Knowledge Graph* formale e validato7. Il ragionamento diagnostico "complesso" è delegato a Datalog/ASP.

La base di conoscenza Datalog è stata progettata per superare il "semplice pattern matching" 8 implementando:

* **Regole di Esclusione (Ragionamento non monotono)**: Utilizzo della negazione (not) per la diagnosi differenziale (es. Regola 9: Peronospora della Rosa), che si attiva solo se i sintomi chiave di altre malattie (macchie\_nere\_foglie, muffa\_biancastra) sono assenti.
* **Modulazione della Confidenza**: Le regole assegnano livelli di confidenza simbolici (es. media, alta, critica) basati sulla completezza dei sintomi osservati.
* **Logica Contestuale**: Le regole integrano fatti esterni come stagione\_corrente(primavera) per aumentare o diminuire la probabilità di una diagnosi.

**Valutazione**

La valutazione di questo modulo è qualitativa e si basa sulla sua capacità di gestire casi di test ambigui, dimostrata durante i test del main\_cli.py:

* **Test 3 (Diagnosi Differenziale)**: Con sintomi ambigui (ingiallimento\_foglie, caduta\_foglie) sulla Rosa, il motore Datalog ha correttamente scartato la Ticchiolatura (a cui mancava il sintomo chiave macchie\_nere\_foglie) e ha inferito Peronospora della Rosa (confidenza "alta"), dimostrando una logica più raffinata rispetto ai modelli statistici.
* **Test 4 (Caso Incoerente)**: Con input incoerente (Pianta: Olivo, Sintomi: Muffa [Rosa]), il motore Datalog ha correttamente prodotto **zero** diagnosi, agendo come "guardiano" contro i falsi positivi.

**Sezione 2: Apprendimento Supervisionato (SVM)**

**Sommario**

Questo modulo implementa la diagnosi come un task di classificazione statistica9. L'obiettivo è fornire una predizione parallela basata esclusivamente sui pattern appresi dai dati storici.

La conoscenza è rappresentata come un vettore di feature. L'input dell'utente (sintomi + pianta) viene trasformato in un vettore numerico multi-dimensionale (one-hot encoding per sintomi e tipo di pianta) dal \_prepara\_feature() in svm\_model.py.

**Strumenti utilizzati**

* **Scikit-learn**: Utilizzato per l'implementazione del SVC (Support Vector Classifier), il StandardScaler (per la normalizzazione) e le utility di valutazione (cross\_val\_score, classification\_report).
* **Pandas/Numpy**: Utilizzati per la manipolazione dei dati e la vettorizzazione.

**Decisioni di Progetto**

* **Modello**: È stato scelto un SVC (Support Vector Classifier) con kernel rbf (Radial Basis Function), efficace nel trovare iperpiani di separazione non lineari, adatti alla complessità delle combinazioni di sintomi.
* **Parametri**:
  + probability=True: Abilitato per ottenere stime di probabilità (confidenza) da utilizzare nel modulo di aggregazione finale, invece di una semplice predizione di classe.
  + class\_weight='balanced': Utilizzato per contrastare lo sbilanciamento delle classi (es. 'Occhio di Pavone' ha molti più esempi di 'Oidio' nel training\_data.csv), evitando che il modello sia affetto da bias verso la classe maggioritaria.
* **Dati**: Per l'addestramento è stato utilizzato un dataset sintetico generato dalla funzione crea\_dataset\_sintetico (svm\_model.py), in quanto il training\_data.csv fornito copre solo 4 delle 8 malattie del dominio.

**Valutazione**

Come richiesto dalle linee guida10, la valutazione non si basa su un singolo run, ma su una **Validazione Incrociata (K-Fold)**. Lo script svm\_model.py implementa una cross\_val\_score (CV) a 5 fold sul set di addestramento.

I risultati medi della CV (eseguiti su dataset sintetico di 400 campioni) mostrano una buona generalizzazione:

| **Metrica (Media CV)** | **Punteggio** |
| --- | --- |
| Accuracy (CV) | 0.985 |
| Deviazione Std. (CV) | +/- 0.012 |

L'elevata accuracy con bassa deviazione standard indica che il modello è stabile e generalizza bene sui dati (sintetici) visti.

**Sezione 3: Ragionamento con Conoscenza Incerta (Reti Bayesiane)**

**Sommario**

Questo modulo affronta l'incertezza intrinseca della diagnosi11. Utilizza un modello probabilistico grafico (una Rete Bayesiana) per calcolare la probabilità a posteriori di ciascuna malattia, data l'evidenza (i sintomi osservati)12121212.

La conoscenza è rappresentata come un grafo aciclico diretto (DAG) e un insieme di Tabelle di Probabilità Condizionata (CPD). Il DAG implementa una struttura "Naive Bayes": un nodo radice malattia ha come figli tutti i nodi sintomo. Le CPD, apprese dai dati, quantificano P(Sintomo | Malattia).

**Strumenti utilizzati**

* **pgmpy**: Libreria Python per la modellazione e l'inferenza con modelli grafici probabilistici (BayesianNetwork, BayesianEstimator, VariableElimination).

**Decisioni di Progetto**

* **Stima dei Parametri**: Invece del MaximumLikelihoodEstimator (MLE), si è scelto di usare un BayesianEstimator. Questa scelta è cruciale perché i dati di training (training\_data.csv) sono sparsi e alcune combinazioni sintomo-malattia potrebbero avere probabilità zero.
* **Smoothing**: È stato applicato uno smoothing di Laplace (impostando pseudo\_conteggio=1.0 nel BayesianEstimator) per evitare probabilità nulle e rendere il modello più robusto a combinazioni di sintomi non viste.
* **Inferenza**: È stato utilizzato l'algoritmo VariableElimination per l'inferenza, efficiente per una rete con una struttura semplice come quella adottata.
* **Limitazione (emersa dai test)**: La struttura scelta (\_definisci\_struttura) non include la pianta come variabile. La BN basa l'inferenza solo sui sintomi.

**Valutazione**

La valutazione della BN è qualitativa, basata sulla sua capacità di gestire l'incertezza nei test del main\_cli.py:

* **Test 1 (Caso Netto)**: Con sintomi chiari (macchie\_circolari\_grigie, ecc.), la BN ha fornito una probabilità molto alta (97.3%) per la diagnosi corretta, mostrando alta confidenza quando giustificato.
* **Test 3 (Diagnosi Differenziale)**: Con sintomi ambigui (ingiallimento\_foglie, caduta\_foglie), la BN ha correttamente riflettuto l'incertezza, assegnando una probabilità bassa (48.6%) alla malattia top, segnalando l'ambiguità.
* **Test 4 (Caso Incoerente)**: Il test ha rivelato la limitazione della struttura scelta. Ignorando la pianta (Olivo), la BN ha assegnato una probabilità alta (87.2%) all'Oidio della Rosa basandosi solo sul sintomo muffa\_biancastra. Questo risultato, sebbene errato, è stato mitigato con successo dall'aggregazione ibrida (che ha prodotto una confidenza finale bassa del 43.6%).

**Conclusioni**

Il progetto "Plant-Aid-KBS" ha raggiunto con successo l'obiettivo di implementare un sistema diagnostico ibrido13. L'integrazione di un'ontologia formale, un motore di regole simboliche, un classificatore statistico e una rete probabilistica ha dimostrato, attraverso test qualitativi, una robustezza superiore a quella dei singoli moduli14.

Come emerso dai test (Test 3 e 4), il sistema è in grado di gestire sia l'**ambiguità** (sintomi comuni a più malattie) sia il **conflitto** (quando i moduli statistici e simbolici sono in disaccordo), producendo una confidenza finale bassa che riflette correttamente l'incertezza del sistema.

**Sviluppi Futuri**

* **Miglioramento BN**: La Rete Bayesiana (bn\_model.py) dovrebbe essere estesa per includere la pianta come variabile genitore (influenzando sia la malattia che i sintomi), risolvendo l'anomalia emersa nel Test 4.
* **Espansione della KB**: L'ontologia e la base di regole possono essere facilmente estese per includere nuove piante e malattie senza modificare il codice dei motori di inferenza.
* **Input Utente**: L'interfaccia CLI potrebbe essere sostituita da un'interfaccia che accetti input in linguaggio naturale, richiedendo un ulteriore modulo NLP per mappare l'input utente ai sintomi formali dell'ontologia.

**Riferimenti Bibliografici**

Riferimenti Tecnici del Progetto

[1] Owlready2: Jean-Baptiste Lamy. Owlready: A Python package for ontologies. https://owlready2.readthedocs.io/

[2] Clingo (Potassco): Roland Kaminski, et al. Clingo User Guide. https://potassco.org/clingo/user-guide/

[3] pgmpy: Ankur Ankan, Abinash Panda. pgmpy: A Python library for Probabilistic Graphical Models. https://pgmpy.org/

[4] Scikit-learn: Fabian Pedregosa, et al. Scikit-learn: Machine Learning in Python. https://scikit-learn.org/

[5] La base di conoscenza simbolica (ASP/Datalog) è stata sviluppata con il supporto consulenziale di Google Gemini Pro per la formalizzazione delle euristiche diagnostiche a partire dalle fonti di dominio.

Riferimenti Dominio Botanico (Fonti Esterne)

[6] CABI Plant Sciences: https://www.cabidigitallibrary.org/product/QC

[7] International Plant Names Index (IPNI): https://www.ipni.org

[8] Plants of the World Online (POWO): https://powo.science.kew.org

[9] World Flora Online (WFO): https://wfoplantlist.org

[10] Coltivazione Biologica - Malattie Rose: https://www.coltivazionebiologica.it/malattie-delle-rose

[11] Rose Barni - Malattie Fungine: https://www.rosebarni.it/malattia-fungine-piante-quali-sono-i-migliori-trattamenti-preventivi

[12] Regione Veneto - Malattie Olivo: https://www.regione.veneto.it/web/fitosanitario/malattie-olivo

[13] Orto da Coltivare - Problemi Olivo: https://www.ortodacoltivare.it/guide/problemi-ulivo

[14] Orto da Coltivare - Fusarium Basilico: https://www.ortodacoltivare.it/difesa/malattie/fusarium-basilico

[15] AgroNotizie - Basilico: https://agronotizie.imagelinenetwork.com/colture/basilico/257