Apprendimento di programmi logici da dati RDF

L'Inductive Logic Programming, è una classe di algoritmi di apprendimento supervisionato che rappresentano l'insieme degli esempi e la backgroung knowledge nella forma della logica del primo ordine. In particolare, consistono nell'apprendimento del version space, dati un insieme di esempi positivi e di esempi negativi, dove gli elementi del version space sono programmi logici che implicano gli esempi positivi.
In questo sistema, scritto in python, una volta fornito in input un dataset in formato RDF, viene utilizzato l'ILP per apprendere una nuova relazione, con arietà massima di due, non definita nel dataset ma dall'utente, che ne fornisce il nome, gli esempi positivi e gli esempi negativi. Per apprendere la relazione è stato utilizzato Metagol, un'implementazione in prolog che utilizza metaregole per definire le clausole permesse in un'ipotesi. Le metaregole utilizzate dal sistema sono:

- [P,Q] , [P,A,B] , [[Q,A,B]]
- [P,Q] , [P,A,B] , [[Q,B,A]]
- [P,Q] , [P,A,B] , [[Q,C,A] ,[Q,C,B]]
- [P,Q] , [P,A,B] , [[Q,A,C] , [Q,B,C]]
- [P,Q,R] , [P,A,B] , [[Q,A,C] , [R,C,B]]

I simboli P, Q ed R denotano variabili del secondo ordine, le variabili per cui metagol trova un valore, e le variabili A, C e B denotano variabili del primo ordine. L'implementazione originale è stata modificata in modo tale da scrivere il risultato dell'algoritmo su un file .pl.

Come esempio dimostrativo l'algoritmo viene applicato sul dataset wine per l'apprendimento della relazione, che stabilisce se due vini sono dello stesso colore.

Innanzitutto il sistema espande il dataset attraverso il reasoning, aggiungendo tutte le possibili triple al grafo. L'implementazione del reasoning è fornita dalla libreria owl rl, che implementa il forward chaining su grafi RDF. Per eseguire Metagol, è stato utilizzato pyswip un Python - SWI-Prolog bridge che consente l'esecuzione di programmi prolog in python.

Successivamente il sistema scrive un file in prolog che definisce l'input di Metagol: viene acquisito dall'utente il numero massimo di clausole da generare, vengono raccolte tutte le proprietà su tutti gli individui dal dataset, che andranno a rappresentare la backgroung knowledge per l'algoritmo, viene definito lo spazio di ricerca attraverso le metaregole, e infine viene acquisita la relazione da apprendere insieme agli esempi positivi e negativi.

Dopo aver selezionato una parte degli esempi totale come training set, l'algoritmo viene addestrato su questi ultimi e, se l'addestramento è andato a buon fine, il sistema può restituire un numero di programmi logici che varia da uno al numero inserito, che sono in grado di stabilire se la relazione appresa tra due individui è vera o falsa. In particolare, i programmi appresi vengono memorizzati su un file .pl, da cui il sistema li acquisisce e li carica come clausole prolog prima di eliminare il file. Inoltre, il programma seleziona tutti gli esempi non utilizzati per l'addestramento come insieme di test e applica i programmi logici ottenuti su questi ultimi, classificando ognuno di essi come vero o falso, per verificare se il programma ottenuto è valido su esempi non visti.

Il sistema costruito fa assunzione di conoscenza completa, dovuta anche all'utilizzo di un algoritmo scritto in prolog, data l'assunzione di conoscenza completa insita nel linguaggio. Questo implica che se

il sistema non ha osservato degli individui e le relazioni associate, il sistema darà per scontato che la relazione appresa, per questi ultimi, sia falsa, sebbene nell'interpretazione intesa dell'utente sia vera.

Di seguito, viene presentata un' istanza di esecuzione:

all'avvio, il sistema, dopo aver aumentato i dati con il reasoning, chiede all'utente il numero massimo di clausole da generare:

```
inserisci il numero di clausole(MAX 10):6
```

Dopo di che viene chiesto il nome della relazione da apprendere (in questo caso viene eseguito l'esempio dimostrativo)

```
inserire la relazione da apprendere(arietà MAX 2)
('esDim' per un esempio dimostrativo):esDim
```

e gli esempi positivi e negativi.

```
inserire gli esempi positivi('fine positivi' per inserire i negativi)
samebody(mountedenvineyardednavalleychardonnay,mountadamriesling)
```

Infine il sistema restituisce in output la relazione appresa e le prestazioni sul test set

```
% learning samecolor/2
% clauses: 1
['samecolor(A,B):-hascolor(A,C),hascolor(B,C).']
on positive example
samecolor(pulignymontrachetwhiteburgundy,ventanacheninblanc) true
samecolor(schlossrothermeltrochenbierenausleseriesling,corbansprivatebinsauvignonblanc) true
samecolor(formancabernetsauvignon,chateaulafiterothschildpauillac) true
samecolor(mariettaoldvinesred,pagemillwinerycabernetsauvignon) true
on negative example
samecolor(chateaulafiterothschildpauillac,mountadamriesling) false
samecolor(elysezinfandel,selakssauvignonblanc) false
samecolor(chateauchevalblancstemilion,corbansdrywhiteriesling) false
samecolor(mariettacabernetsauvignon,mountedenvineyardednavalleychardonnay) false
samecolor(chateaudemeursaultmeursault,taylorport) false
samecolor(selakssauvignonblanc,cotturizinfandel) false
```

Studente: Angelo Ignomeriello

Matricola: 669986