UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

DIPARTIMENTO DI INFORMATICA

SPERIMENTAZIONE DI APPRENDIMENTO

Sistemi FOIL e ALEPH

a.a. 2011/2012

Autore:

Francesco Serafino - N° Matr. 590143

INDICE

1. OBIETTIVO DEL CONFRONTO

La valutazione dei sistemi di apprendimento è utile e necessaria per scegliere il sistema più adatto per un particolare task. In questo caso si valuteranno, comparandoli, due sistemi di apprendimento dedicati al task di classificazione al fine di scegliere il più adatto tra i due.

Poiché possono essere commessi degli errori nel training set, che comprometterebbero i risultati facendo apparire come performante un sistema che in realtà non lo è, si utilizzeranno tre diversi dataset a partire dai quali si costruirà un training set con le tecniche che saranno mostrate più avanti e successivamente si forniranno dei test statistici per avere la certezza che le differenze non siano dovute al caso. Il tutto sarà ripetuto per entrambi i sistemi di apprendimento.

I sistemi di apprendimento da analizzare sono stati sviluppati entrambi con la tecnica dell'ILP (Inductive Logic Programming). La ILP è un tipo di programmazione logica che sfrutta la potenza espressiva della logica del prim'ordine (Tramite le clausole di Horn) per la rappresentazione delle osservazioni, delle ipotesi e della teoria di fondo ed inoltre eredita gli obiettivi dell'apprendimento induttivo.

Di seguito si mostreranno le caratteristiche dei due sistemi di apprendimento, dei dataset scelti per la sperimentazione, delle tecniche scelte per la conduzione dell'esperimento e l'interpretazione dei risultati.

2. SCELTA DEI SISTEMI DI APPRENDIMENTO

2.1 FOIL (First Order Inductive Learner)

FOIL è un sistema di apprendimento di tipo batch ispirato a CN2 che apprende da programmi Datalog, di conseguenza non ammette i funtori all'interno dei termini. Utilizza inoltre un database espresso in forma estensionale. La strategia di FOIL per la classificazione è Top-Down che quindi lavora su specializzazione. FOIL riceve in input un insieme di relazioni descritte sotto forma di clausola di Horn secondo una sua ben definita sintassi, ognuna di queste relazioni è formata da un set di tuple di esempi come positivi e per la CWA (Ipotesi del mondo chiuso) FOIL dà per scontato che qualunque altra tupla sia negativa. A scelta è possibile invece aggirare la CWA e fornire una serie di tuple che FOIL deve considerare come negative.

Come già detto il sistema parte con un insieme di esempi positivi ed esempi negativi e inizializza una nuova clausola di Horn che spiega alcune delle tuple, successivamente rimuove dall'insieme delle tuple positive quelle coperte dalla clausola appena inizializzata e continua la ricerca per la prossima clausola. Quando le clausole coprono tutti gli esempi positivi, le clausole vengono revisionate e vengono eliminate quelle che dovessero risultare ridondanti e vengono riordinate le clausole ricorsive in modo che il passo base si trovi sempre prima del passo ricorsivo. FOIL inizia dal lato sinistro della clausola e la specializza aggiungendo letterali nella parte destra, poi si ferma quando nessun esempio negativo viene coperto da quella clausola oppure quando la clausola inizia a diventare troppo lunga e complessa (FOIL è dotato di una funzione euristica che limita la lunghezza delle clausole per ridurre la complessità del sistema. Ciò comporta che l'ipotesi non sarà completa, ma almeno sarà coerente).

Presa dunque una relazione R del training set T con k argomenti (V1, V2, … VK), l'algoritmo del sistema è il seguente:

```
Inizializza la clausola nel seguente modo: R(V1,V2,…VK) :-. ;
Rimuovi dal Training set T tutte le tuple coperte in una precedente
```

```
clausola;
      Mentre T contiene tuple negative e non è troppo complesso;
            Trova un letterale L da aggiungere al lato destro della
            clausola;
            Crea un nuovo tranining set T';
            Per ogni tupla t in T;
                  Per ogni binding b di ogni nuova variabile
                  introdotta dal letterale L;
                  Se la tupla t.b ottenuta concatenando t
                        e b soddisfa L allora;
                              Aggiungi t.b a T' con la stessa
                              etichetta positiva o negativa che
                              aveva t;
                        Fine Se;
                  Ripeti;
            Ripeti;
            Sostituisci T con T';
      Ripeti;
Elimina dalla clausola tutti i letterali non necessari.
```

Per stabilire come fare a determinare gli appropriati letterali da aggiungere a destra di una clausola, FOIL utilizza due criteri: accetta letterali nella parte destra se e solo se un letterale può essere utile per dire che una tupla esempio negativo deve essere esclusa, oppure il nuovo letterale deve introdurre nuove variabili che saranno utili per nuovi letterali. I letterali del primo tipo sono detti *gainful* mentre i secondi sono detti *determinate*.

2.2 ALEPH (A Learning Engine for Proposing Hypothesis)

ALEPH è un sistema che implementa un algoritmo basato su PROGOL sviluppato da Muggleton nel 95. ALEPH è scritto in Prolog e come PROGOL, è un sistema di tipo batch basato su una strategia ibrida, cioè in grado sia di specializzare che generalizzare le clausole di Horn.

Ad ogni passo ALEPH apprende una clausola alla volta e rimuove gli esempi positivi coperti da quella clausola.

Si mostra di seguito il funzionamento dell'algoritmo base di Aleph:

```
Per ogni esempio da generalizzare;
      Scegli un esempio;
      Costruisci la clausola più specifica (bottom clause);
      Trova una clausola più generale della bottom clause costruendo
      sottoinsiemi di letterali della bottom clause che hanno il
      miglior punteggio (Punteggio calcolato da Aleph);
      Aggiungi la clausola con il miglior punteggio alla teoria e
      rimuovi gli esempi ridondanti;
Ripeti;
```

ALEPH inoltre possiede una gestione del Test Set leggermente diversa rispetto a FOIL. Infatti una volta avviato il sistema e caricata la background knowledge ALEPH costruisce la sua teoria sugli esempi di training e successivamente gli si forniscono i documenti di test per valutare i risultati dell'apprendimento. Su FOIL invece questo non è possibile, il sistema viene avviato dandogli in input sia il test set che il training set e fornisce direttamente i risultati. Maggiori dettagli sono approfonditi nel capitolo successivo relativo alla fase di riscrittura dei dataset e nel capitolo relativo alla conduzione dell'esperimento vero e proprio.

3. SCELTA DEI DATASET

I dataset che sono stati presi in esame per testare i sistemi sono relativi alla Document Image Understanding. Sono stati scelti quelli della categoria Classification, quindi il sistema dovrà apprendere da una serie di documenti di esempio descritti sotto forma di clausole di Horn (opportunamente formattate per ogni sistema) e classificare i documenti nella categoria corretta. Si analizzano di seguito nel dettaglio i dataset scelti.

3.1 ECAI

Dataset di documenti suddivisi come segue:

- Esempi Positivi 28/122
- Esempi Negativi 94/122

Ogni clausola rappresentante gli esempi positivi ha nella testa il predicato *class_ecai/1*, mentre per gli esempi negativi è applicato il not al letterale nella testa.

3.2 ELSEVIER

Dataset di documenti suddivisi come segue:

- Esempi Positivi 38/122
- Esempi Negativi 84/122

Ogni clausola rappresentante gli esempi positivi ha nella testa il predicato *class_elsevier/1*, mentre per gli esempi negativi è applicato il not al letterale nella testa.

3.3 ICML

Dataset di documenti suddivisi come segue:

- Esempi Positivi 20/122
- Esempi Negativi 102/122

Ogni clausola rappresentante gli esempi positivi ha nella testa il predicato *class_icml/1*, mentre per gli esempi negativi è applicato il not al letterale nella testa.

3.4 Caratteristiche comuni dei Dataset

I tre dataset descritti prima contengono tutti documenti descritti tramite una serie di proprietà che questi posseggono e vengono di seguito riportate. A sinistra c'è il nome della proprietà così come stabilita sotto forma di predicato e la relativa arietà.

Nome Proprietà	Concetto Rappresentato
numero_pagine/2	Numero di pagine contenute nel documento
pagina_1/2	Prima pagina del documento
larghezza_pagina/2	Larghezza della pagina
altezza_pagina/2	Altezza della pagina
centrali_pagine/1	Pagina centrale
frame/2	Cornice trovata in una pagina del documento
ascissa_rettangolo/2	Ascissa di una cornice
ordinata_rettangolo/2	Ordinata di una cornice
larghezza_rettangolo/2	Larghezza di una cornice
altezza_rettangolo/2	Altezza di una cornice
tipo_testo/1	Cornice che contiene Testo
on_top/2	Esprime quale cornice si trova sopra a quale altra cornice
to_right/2	Esprime quale cornice si trova a destra di quale altra cornice
allineato_al_centro_verticale/2	Esprime che una cornice è verticalmente allineata con un'altra

allineato_al_centro_orizzontale/2	Esprime che una cornice è allineata orizzontalmente con un'altra
prime_pagine/1	Prime pagine del documento
penultima_pagina/1	Penultima pagina del documento
tipo_vuoto/1	Cornice che è vuota
tipo_misto/1	Cornice che contiene contenuto di più tipi differenti
tipo_immagine/1	Cornice che contiene una immagine
tipo_linea_obbliqua/1	Cornice che contiene una linea obliqua
tipo_linea_orizzontale/1	Cornice che contiene una linea orizzontale

3.5 Riscrittura dei Dataset

I dataset così come sono stati forniti sono generali e devono essere adattati allo specifico sistema ILP affinché possa acquisirli come esempi di training e possa utilizzarli anche come documenti di test. Di seguito si specifica come è necessario modificare i dataset per adattarli sia a FOIL che a ALEPH.

3.5.1 Riscrittura per FOIL

FOIL prevede un solo file di input con estensione "*.d" contenente tutti gli esempi sia positivi che negativi. Inoltre prevede che il file sia strutturato in tre parti separate da una linea bianca:

- Una prima sezione contenente i tipi di dati utilizzati
- Una seconda sezione con la definizione delle relazioni
- Una terza parte con gli esempi di test che deve utilizzare

Per quanto riguarda i tipi di dati essi possono essere definiti in maniera intensionale quando essi sono di tipo numerico, altrimenti devono essere definiti in maniera estensionale. Per raccogliere i tipi di dati estensionali è stato utilizzato un semplice programma prolog costituito di fatto dall'unione delle clausole contenute nel file di esempi positivi e di quelle contenute nel file di esempi negativi. Gli esempi negativi sono stati resi positivi eliminando il not che era loro posto davanti tramite l'editor di testo notepad++. Con lo stesso editor sono stati trasformati tutti i letterali di tutte le regole in semplici fatti sostituendo le virgole e i simboli di inferenza ":-" con un punto. In questo modo

è possibile interrogare la base di conoscenza tramite delle findall e ottenere i valori dei tipi di dati non numerici. Inoltre in questo file prolog è stato definito un predicato *mfa/2* che sta per "my findall", è un predicato che velocizza le operazioni e oltre a richiamare la findall di yap, in più effettua l'ordinamento e la rimozione dei duplicati dei valori, successivamente chiama un altro predicato per stampare a video i risultati aggiungendo un "*" prima di ogni valore.

```
mfa(X, P) :-
      findall(X, P, L),
      sort(L, Lsorted),
      stampaLista(Lsorted).

stampaLista([]).
stampaLista([H|T]):-
      write('*'),
      write(H),
      write(', '),
      stampaLista(T).

... Rest of dataset file...
```

Per ottenere ad esempio tutti i nomi dei documenti sarà sufficiente effettuare una consult sul file del dataset così modificato e interrogare il predicato mfa come segue:

```
?- mfa(X, class_ecai(X)).
```

Il sistema mostrerà una lista dei nomi dei documenti preceduti dal simbolo "*" e separati da virgola. Questa sintassi di output è utile appunto per FOIL e va utilizzata se ci sono dei dati espressi in forma estensionale da considerare come costanti. Il discorso è analogo per gli altri tipi di dati testuali come Frame e Pagina che indicano rispettivamente i nomi di tutti i frame del dataset e i nomi di tutte le pagine del dataset.

Durante la sperimentazione ci si è resi conto che specificare i dati come costanti faceva bloccare il sistema di apprendimento e non riusciva a trovare una teoria che coprisse gli esempi positivi. Per questo motivo si sono rimossi tutti gli

asterischi davanti a questi tre tipi di dati in modo da consentire a FOIL una corretta esecuzione dell'algoritmo.

Per tutti gli altri tipi numerici è stato sufficiente specificare un tipo di dato continuo (continuous), in particolare sono stati usati i tipi: NumeroPagine, AltezzaPagina, LarghezzaPagina, AltezzaRettangolo, LarghezzaRettangolo, AscissaRettangolo e OrdinataRettangolo.

La sezione successiva è quella che riguarda le relazioni, queste si costruiscono a partire dai predicati usati all'interno del dataset con i rispettivi valori che essi assumono all'interno dell'intero dataset. Per quanto riguarda il predicato class_[nome_dataset] che discrimina se il documento appartiene a quella classe o meno, esso costituirà di fatto il training set, quindi nella definizione estensionale della relazione saranno separati i documenti positivi e i documenti negativi dal carattere ";". Il sistema dovrà costruire la teoria su questa relazione, dunque per tutte le altre relazioni costruite sui predicati rimanenti si specificherà esplicitamente con un simbolo di "*" prima del nome di ogni relazione, che per quella relazione non si dovrà costruire la teoria. Anche se per queste relazioni non è necessario costruire la teoria, esse sono indispensabili come supporto per la costruzione di altre teorie (in questo caso per class_[nome_dataset]).

Infine, l'ultima sezione è quella dei documenti di test. Dopo un'altra linea bianca per separare le relazioni si scrivono i nomi dei documenti da testare uno su ogni riga e si indica accanto se l'esempio di test è positivo o negativo con la sintassi:

: + (per gli esempi positivi)

: - (per gli esempi negativi)

In questa maniera FOIL sa già quali sono positivi e quali sono negativi e autonomamente è in grado di dire subito quanti ne ha sbagliati e quanti ne ha classificati correttamente.

Tutte le operazioni di conversione sono state effettuate manualmente ed in tempi abbastanza rapidi grazie all'utilizzo dell'editor notepad++.

3.5.2 Riscrittura per ALEPH

Aleph essendo scritto in prolog ha una sintassi prolog-like pertanto la specifica dei dataset avviene in maniera più intuitiva e necessita di una operazione di riscrittura più rapida.

Aleph necessita di tre tipi di file di input, tutti con lo stesso nome ma con estensioni diverse:

- Un file con estensione ".f" contenente gli esempi di training positivi
- Un file con estensione ".n" contenente gli esempi di training negativi
- Un file con estensione ".b" contenente la conoscenza di fondo

I primi due file contengono il predicato di testa delle clausole dei dataset con la sintassi: *class_[nome_dataset]([nome_documento])*.

Per quanto riguarda invece il file ".b" esso può essere considerato come suddiviso in 5 sottosezioni:

- **Parametri di avvio:** Questa sezione contiene una serie di settaggi che riguardano le prestazioni dell'algoritmo. Nello specifico sono stati impostati i seguenti parametri:

:- set(nodes, 50000).	Aumenta il livello di profondità nell'albero di decisione per raggiungere la soluzione. (Default 5000).
:- set(clauselength, 15).	Imposta la lunghezza massima delle clausole che costruiscono la teoria a 15.
:- set(thread, 2).	Imposta i thread del processore a 2 per elaborare le clausole più velocemente.
:- set(i, 15).	Imposta il numero massimo di variabili che può contenere una clausola ad un

	valore più alto, infatti di default è impostato a 2 che è troppo basso.
:- set(minpos, 2)	Serve ad escludere teorie che coprono meno di due esempi. Fissandolo a 2 il sistema non fornirà teorie ground, che infatti sono da evitare.
:- set(noise, 0)	Soglia che migliora la teoria in quanto evita che essa copra qualunque esempio negativo. In altri termini si sta dicendo che la teoria può coprire al massimo zero esempi negativi.

- **Modi:** I modi si distinguono in due tipi: i *modeh* che identificano la testa della clausola, quindi servono per la formulazione di una teoria, nel nostro caso class_[nome_dataset] e i *modeb* che identificano i letterali nel corpo delle clausole e sono dunque di supporto per la costruzione della teoria.
  In entrambe le definizioni modeb e modeh si inserisce il nome del letterale, quante volte questo potrà essere presente nelle clausole della conoscenza di fondo (1 o *) e il tipo dei suoi argomenti accompagnato da un + o da un - per indicare se esso è di input o di output. Se il tipo è numerico allora si possono utilizzare dei costrutti di aleph indicando ad esempio #int o #real per indicare che è un numero intero o un numero reale.

- **Determinazioni:** La sezione determination è utile per dichiarare quali predicati sono utili per la costruzione di una teoria. Nel nostro caso tutti i predicati tranne class_[nome_dataset] che rappresenta la testa, sono stati ritenuti utili per la costruzione della teoria. Se non vengono dichiarate le determination, aleph non costruirà nessuna teoria.

- **Tipi di dati:** I tipi di dati non numerici, che quindi non sono stati rappresentati come #int o #real devono essere espressi in maniera estensionale. Per far questo è sufficiente inserire nel file una serie di fatti con sintassi nome_tipo(Valore).

- **Base di conoscenza estensionale:** Questa è l'ultima sezione nella quale viene rappresentata in maniera estensionale la conoscenza di fondo. La conoscenza di fondo non è altro che il dataset stesso privato di tutti i letterali nella testa di ogni clausola, il cui corpo è stato convertito da un insieme congiunto di letterali ad una serie di fatti.

Come si è potuto notare, a differenza di FOIL, nei file così creati non si specifica il test set. Per effettuare il test dell'algoritmo di apprendimento sarà necessario specificare un ulteriore file da dare in input ad aleph dopo che questo avrà costruito la teoria sui documenti di training già in suo possesso. In questo file saranno contenuti uno dopo l'altro gli esempi positivi e negativi senza specificare quali sono negativi e quali positivi, sarà l'utente a verificare dall'output se questi sono stati classificati correttamente o meno.

Anche per la riscrittura dei dataset di Aleph è stato utilizzato l'editor di testo notepad++.

4. SCELTA DELLA METODOLOGIA SPERIMENTALE

Per la conduzione dell'esperimento si è scelto di utilizzare la tecnica della K-fold Cross Validation Stratified con K = 10. Si è deciso di impostare k a tale valore vista l'evidenza teorica nota in letteratura riguardo al numero ideale di fold in cui suddividere gli esempi. Sebbene il dataset degli esempi sia abbastanza piccolo si è deciso comunque di utilizzare la K-fold Cross Validation e non si è usata la Leave-One-Out Cross Validation vista la sua complessità computazionale elevata e l'impossibilità di effettuare la stratificazione.

Anche la tecnica di bootstrap non è stata utilizzata poiché essa non garantisce l'utilizzo completo degli esempi nella fase di test. Infatti è possibile che la selezione random degli esempi di test e di training possa far confluire nel test set di fold diverse, più volte uno stesso esempio e allo stesso tempo non farvi confluire mai uno o più esempi che sarebbero presi in tutte le fold come esempi di training.

4.1 Creazione delle k fold

Mostriamo ora come decomporre i dataset in K=10 fold mantenendo le proporzioni tra le classi positiva e negativa usando la stratificazione. Per formare le fold si estrarranno a caso dai dataset un certo numero di esempi positivi ed in proporzione un certo numero di esempi negativi. Il numero di esempi è dettato dalla dimensione del dataset di partenza, in particolare con i dataset scelti le fold saranno costituite come segue:

FOLD		01	02	03	04	05	06	07	08	09	10	Totale	Tot P+N
Ecai	+	3	3	3	3	3	3	3	3	3	1	28	122
	-	9	9	9	9	9	9	9	9	9	13	94	
Elsevier	+	4	4	4	4	4	4	4	4	4	2	38	122
	-	8	8	8	8	8	8	8	8	8	12	84	
Icml	+	2	2	2	2	2	2	2	2	2	2	20	122
	-	10	10	10	10	10	10	10	10	10	12	102	

Come si può vedere l'ultima fold di ogni dataset non è proporzionata rispetto alle altre, pertanto eventuali problemi trovati in queste fold saranno da considerare marginali rispetto ad altri eventuali problemi trovati sulle altre fold. Le fold così costruite saranno le stesse da usare per testare entrambi i sistemi, esse cambieranno soltanto nella sintassi con cui saranno rappresentate.

4.2 Misure di prestazione

Per analizzare i risultati forniti dal test dei due sistemi sui tre dataset saranno utilizzate delle misure di prestazione. Le misure standard di efficacia per i classificatori sono la precision (corrisponde alla coerenza) e la recall (corrisponde alla completezza), queste misure si calcolano in base ai risultati forniti dal classificatore espressi in termini di True Positive, True Negative, False Positive e False Negative.

Categoria class_dataset		**Etichettamento dell'esperto**	
		SI	**NO**
Etichettamento del classificatore	**SI**	TP	FP
	NO	FN	TN

Le misure di precision (π) e recall (ρ) sono rispettivamente calcolate come:

$$\pi = \frac{TP}{TP + FP} \qquad \rho = \frac{TP}{TP + FN}$$

Se la precision e la recall sono entrambe uguali ad 1 significa che il sistema è perfetto ed è equivalente all'esperto che etichetta gli esempi, ma questo risultato è difficilmente ottenibile. In genere all'aumentare della precision diminuisce la recall e viceversa.

Un'altra misura di prestazione che si valuta solitamente è l'accuratezza. Questa misura è una delle più importanti perché è in grado di dire tramite un solo

valore quanto il sistema di classificazione è accurato nel predire i risultati. Essa si calcola come:

$$\mathbf{ACC} = \frac{\textbf{Numero di esempi classificati correttamente}}{\textbf{Totale degli esempi nel Test Set}}$$

Un'altra misura strettamente correlata all'accuratezza, anch'essa largamente usata, è l'error rate. Si calcola come segue:

$$\mathbf{ERR} = \frac{\textbf{Numero di esempi classificati erroneamente}}{\textbf{Totale degli esempi nel Test Set}}$$

Poiché ognuna delle due misure è ottenibile dall'altra, si è deciso di riportare come misura di prestazione, soltanto l'error rate. Infatti le due misure sono definibili anche come:

$$\mathbf{ACC = 1 - ERR} \qquad\qquad \mathbf{ERR = 1 - ACC}$$

Si mostrano di seguito delle tabelle riepilogative che evidenziano le differenze tra i due sistemi FOIL e ALEPH rispetto ai tre dataset, tramite le misure di prestazione appena definite. Per ogni fold sono riportate le misure e infine se ne calcola la media accanto al simbolo **μ**.

4.3 Risultati ECAI

ECAI	FOIL						ALEPH					
Fold	TP	FP	FN	π	ρ	ERR	TP	FP	FN	π	ρ	ERR
01	3	0	0	1	1	0	3	0	0	1	1	0
02	3	0	0	1	1	0	3	0	0	1	1	0
03	3	0	0	1	1	0	3	0	0	1	1	0
04	3	0	0	1	1	0	3	0	0	1	1	0
05	2	0	1	1	0,66…	0,083…	2	0	1	1	0,66…	0,083…
06	3	0	0	1	1	0	3	0	0	1	1	0
07	3	0	0	1	1	0	3	0	0	1	1	0
08	3	0	0	1	1	0	3	0	0	1	1	0
09	3	0	0	1	1	0	3	0	0	1	1	0
10	1	0	0	1	1	0	1	0	0	1	1	0
μ	2,7	0	0,1	1	0,966..	0,0083…	2,7	0	0,1	1	0,966..	0,0083…

Come si può vedere dalla tabella, il dataset ecai ha dato problemi in entrambi i sistemi soltanto in una fold, ossia quella che contiene il documento “get_tr” nel test set. Questo documento non viene coperto dalla teoria che è stata formulata sia da Aleph che da FOIL:

Teoria di FOIL: `class_ecai(A) :- pagina_1(A,B), centrali_pagine(B).`

Teoria di Aleph: `class_ecai(A) :- numero_pagine(A,5).`

Sebbene le teorie formulate dai due sistemi siano diverse, il task di classificazione è stato portato a termine da entrambi nello stesso modo.

4.4 Risultati ELSEVIER

ELSEVIER	FOIL						ALEPH					
Fold	TP	FP	FN	π	ρ	ERR	TP	FP	FN	π	ρ	ERR
01	4	1	0	0,8	1	0,083…	4	0	0	1	1	0
02	4	0	0	1	1	0	4	0	0	1	1	0
03	4	0	0	1	1	0	4	0	0	1	1	0
04	4	0	0	1	1	0	4	0	0	1	1	0
05	4	0	0	1	1	0	4	1	0	0,8	1	0,083…
06	4	0	0	1	1	0	4	0	0	1	1	0
07	4	0	0	1	1	0	4	0	0	1	1	0
08	4	1	0	0,8	1	0,083…	4	0	0	1	1	0
09	4	2	0	0,6..	1	0,16…	3	0	1	1	0,75	0,083…
10	1	0	1	1	0,5	0,07142857	1	0	1	1	0,5	0,07142857
μ	3,7	0,4	0,1	0,92	0,95	0,039742857	3,6	0,1	0,2	0,98	0,925	0,023742857

Come si può vedere dalla tabella, il dataset elsevier ha dato problemi in entrambi i sistemi sulle ultime due fold, mentre ha dato problemi nella prima e nella settima fold con FOIL e nella quinta fold per ALEPH. I documenti su cui i sistemi si sono comportati in maniera differente sono gli stessi ad eccezione del documento “d04032611212504982” (positivo) che è stato classificato male da entrambi.

Per quanto riguarda le teorie i due sistemi si sono comportati entrambi in maniera molto differente al variare delle fold costituenti il training set:

Teoria di FOIL: Foil ha addirittura formulato delle teorie diverse ad ogni avvio del sistema su una nuova fold. Quasi tutte le teorie formulate, però, contengono la seguente clausola:

```
class_elsevier(A) :- pagina_1(A,B), penultima_pagina(B).
```

Allo stesso modo in molte fold testate, la teoria formulata dal sistema ha mostrato anche la seguente clausola:

```
class_elsevier(A) :- pagina_1(A,B), prime_pagine(B), frame(B,C),
to_right(C,D),
                tipo_immagine(D).
```

In alcuni casi quest'ultima clausola è apparsa con altri letterali nel corpo che l'hanno così specializzata ulteriormente.

Teoria di ALEPH: Anche aleph ha formulato diverse teorie durante le 10 esecuzioni delle fold, ma si è dimostrato più costante nella loro formulazione. Tutte le fold hanno esibito la seguente teoria:

```
class_elsevier(A) :- pagina_1(A,B), frame(B,C),
                     XXX_rettangolo(C,COSTANTE).
```

dove al posto della stringa XXX sono variate da fold a fold esclusivamente le proprietà relative ai frame, cioè ascissa, ordinata, altezza e larghezza. COSTANTE rappresenta invece un valore numerico che varia di volta in volta ed è stato scoperto da aleph come valore caratterizzante per la classe elsevier.

4.5 Risultati ICML

ICML	FOIL						ALEPH					
Fold	TP	FP	FN	π	ρ	ERR	TP	FP	FN	π	ρ	ERR
01	1	0	1	1	0,5	0,083…	2	1	0	0,6...	1	0,083…
02	2	0	0	1	1	0	2	0	0	1	1	0
03	2	0	0	1	1	0	2	0	0	1	1	0
04	2	1	0	0,6…	1	0,083…	1	0	1	1	0,5	0,083…
05	2	1	0	0,6…	1	0,083…	1	0	1	1	0,5	0,083…
06	2	0	0	1	1	0	2	0	0	1	1	0
07	2	0	0	1	1	0	2	0	0	1	1	0
08	2	0	0	1	1	0	2	0	0	1	1	0
09	2	1	0	0,6…	1	0,083…	2	0	0	1	1	0
10	2	0	0	1	1	0	2	0	0	1	1	0
μ	1,9	0,3	0,1	0,88	0,95	0,03…	1,8	0,1	0,2	0,96	0,9	0,025

Il dataset ICML si è comportato in maniera abbastanza simile su entrambi i sistemi ILP. I problemi sono sempre situati nella prima, nella quarta e nella quinta fold a differenza di FOIL che classifica male anche un esempio nella nona fold. I problemi trovati nelle stesse fold, però, riguardano documenti diversi e non lo stesso documento come si potrebbe invece immaginare.

Per quanto riguarda la definizione della teoria nei due sistemi, si è notato che FOIL ha cambiato la definizione della teoria per ogni fold, mentre aleph ha formulato sempre la stessa teoria ad eccezione di un'unica fold la cui teoria aveva un letterale aggiuntivo nel corpo della clausola.

Teoria per FOIL: In circa la metà delle fold compaiono insieme all'interno della teoria le seguenti clausole:

```
class_icml(A) :- pagina_1(A,B), prime_pagine(B), frame(B,C),
                 to_right(D,C), tipo_immagine(D), on_top(E,D).
```

```
class_icml(A) :- pagina_1(A,B), frame(B,C), to_right(D,C),
                 tipo_linea_orizzontale(D), on_top(D,C),
                 allineato_al_centro_verticale(E,C), to_right(E,C).

class_icml(A) :- pagina_1(A,B), prime_pagine(B), frame(B,C),
                 allineato_al_centro_orizzontale(C,D), on_top(D,C).
```

Per quanto riguarda l'altra metà delle fold, quasi sempre la teoria formulata conteneva le seguenti clausole:

```
class_icml(A) :- pagina_1(A,B), prime_pagine(B), frame(B,C),
                  on_top(D,C), to_right(D,E), tipo_testo(E),
                  on_top(C,F), tipo_linea_orizzontale(D),
                  on_top(F,G), to_right(D,C),
                  allineato_al_centro_verticale(H,C), to_right(C,E).

class_icml(A) :- pagina_1(A,B), prime_pagine(B), frame(B,C),
                  on_top(C,D), tipo_immagine(C), on_top(E,C).
```

Teoria di ALEPH: Anche con il dataset ICML, Aleph è stato più costante nella definizione della teoria. Per quasi tutte le fold la teoria costruita è stata la seguente:

```
class_icml(A) :- pagina_1(A,B), frame(B,C),
                  altezza_rettangolo(C,0.0012626263),
                  allineato_al_centro_verticale(C,D),
                  prime_pagine(B).
```

L'unica fold per cui aleph ha costruito una teoria differente è stata la prima, infatti per questa fold la teoria risultante è stata la stessa di prima a meno del letterale "allineato_al_centro_verticale(C,D)" che non è presente nel corpo della clausola.

5. SCELTA DEL TEST DI VALUTAZIONE

Per poter analizzare i dati così raccolti e capire quale dei due sistemi risulta essere migliore nel task di classificazione è necessario utilizzare un test delle ipotesi che sia in grado di confermare o rigettare l'ipotesi nulla.

Innanzi tutto possiamo definire le due ipotesi H_0 e H_a in maniera formale come:

H_0 (ipotesi nulla): Non c'è una significativa differenza statistica tra i sistemi di apprendimento FOIL e ALEPH.

H_a (ipotesi alternativa): C'è una significativa differenza statistica tra i sistemi di apprendimento FOIL e ALEPH.

Poiché per ogni dataset è stata ripetuta la sperimentazione, il test delle ipotesi sarà condotto per ognuno dei tre dataset separatamente.

Si è scelto di utilizzare come test delle ipotesi il test non parametrico denominato "Test di Wilcoxon dei ranghi con segno", chiamato più semplicemente "Wilcoxon" (da non confondersi con il Wilcoxon-Mann-Whitney).

In un primo momento si è pensato di utilizzare il paried t-test, infatti per ogni dataset, la tecnica della k-fold cross validation è stata condotta utilizzando le stesse identiche fold su entrambi i sistemi; ciò ha reso dipendenti i due campioni e l'uso del paired t-test avrebbe fornito una maggiore significatività. Il problema nell'uso del paired t-test è che essendo un tipo di test parametrico esso richiede che siano soddisfatte due condizioni abbastanza stringenti: la prima è che la misura da utilizzare sia almeno su una scala ad intervallo e la seconda è che la distribuzione dei dati sia una distribuzione normale. Per quanto riguarda la prima condizione non ci sono problemi, infatti l'error rate è un valore percentuale riconducibile ad un numero razionale che è su scala ratio. La seconda condizione invece non è stata soddisfatta, infatti è possibile considerare una distribuzione approssimata ad una distribuzione normale solo se per questa sono forniti un gran numero di campioni (tipicamente > 100), ma nel caso in esame i campioni sono soltanto 10.

Ricapitolando, non potendo applicare il paired t-test si è deciso di utilizzare un test di tipo non parametrico ed in particolare il Wilcoxon poiché esso è l'alternativa non parametrica al paired t-test.

Per l'applicazione del test Wilcoxon si è utilizzato il motore computazionale di conoscenza WolframAlpha. Per eseguire il test si inserisce nella barra di ricerca online il nome del test seguito dalle distribuzioni dei due campioni sotto forma di vettori e al termine della computazione si ottiene il p-value, un indicatore che se risulta essere maggiore di 0,05 indica che bisogna accettare l'ipotesi nulla H_0, oppure in caso contrario, rigettarla.

I comandi digitati per ottenere i tre p-value sono:

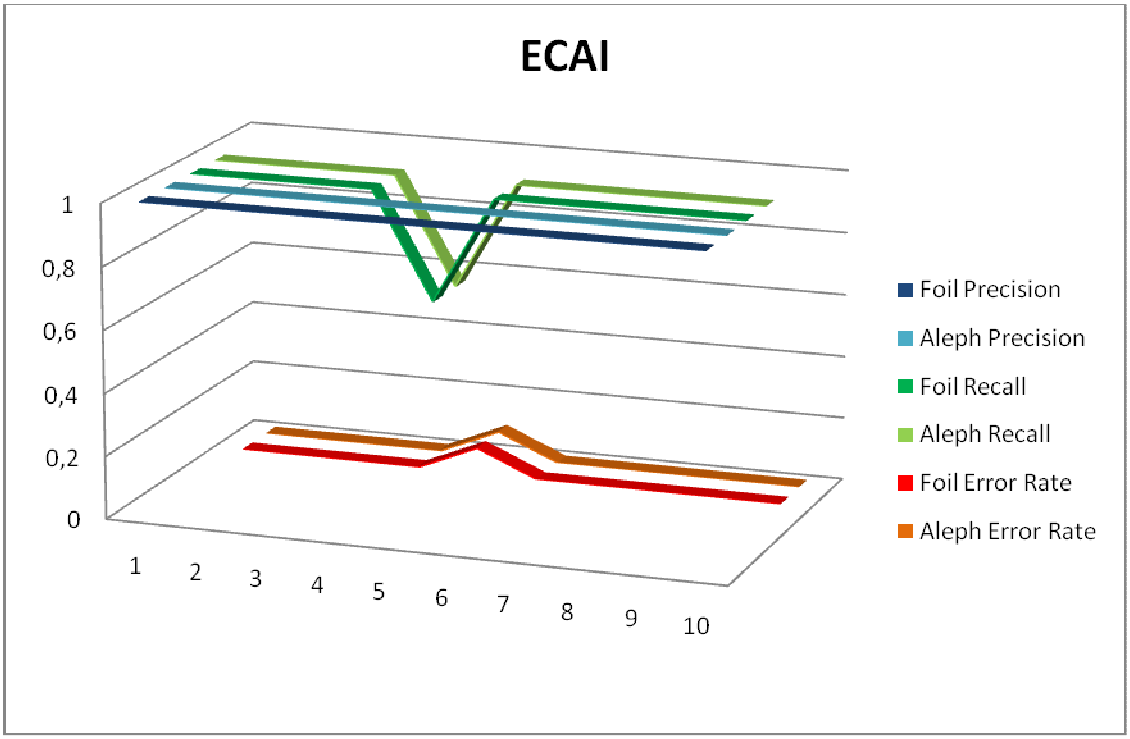
```
SignedRankTest[{(0, 0, 0, 0, 0.83, 0, 0, 0, 0, 0),
                (0, 0, 0, 0, 0.83, 0, 0, 0, 0, 0)]}

SignedRankTest[{(0.083, 0, 0, 0, 0, 0, 0, 0.083, 0.16, 0.07142857),
                (0, 0, 0, 0, 0.083, 0, 0, 0, 0.083, 0.07142857)]}

SignedRankTest[{(0.083, 0, 0, 0.083, 0.083, 0, 0, 0, 0.083, 0),
                 (0.083, 0, 0, 0.083, 0.083, 0, 0, 0, 0, 0)]}
```

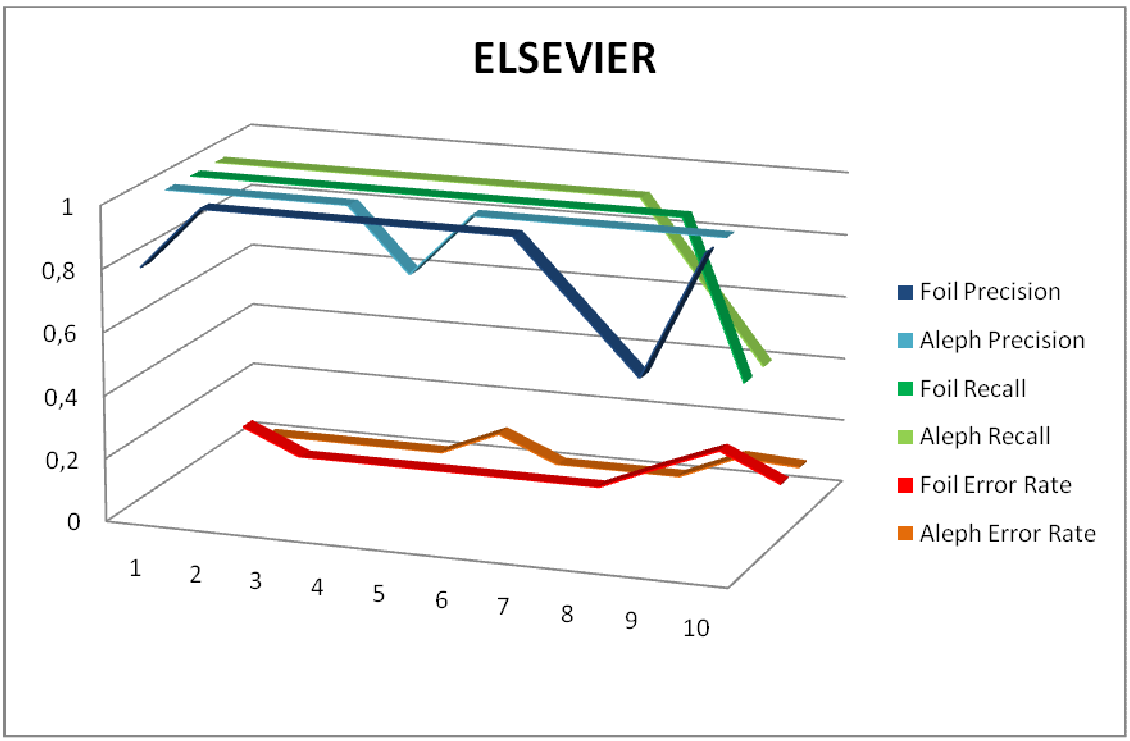
Si mostra di seguito una tabella riepilogativa di come si sono comportati i due sistemi di apprendimento sui tre dataset, indicando per ogni dataset se risulta vera H_0 oppure H_a.

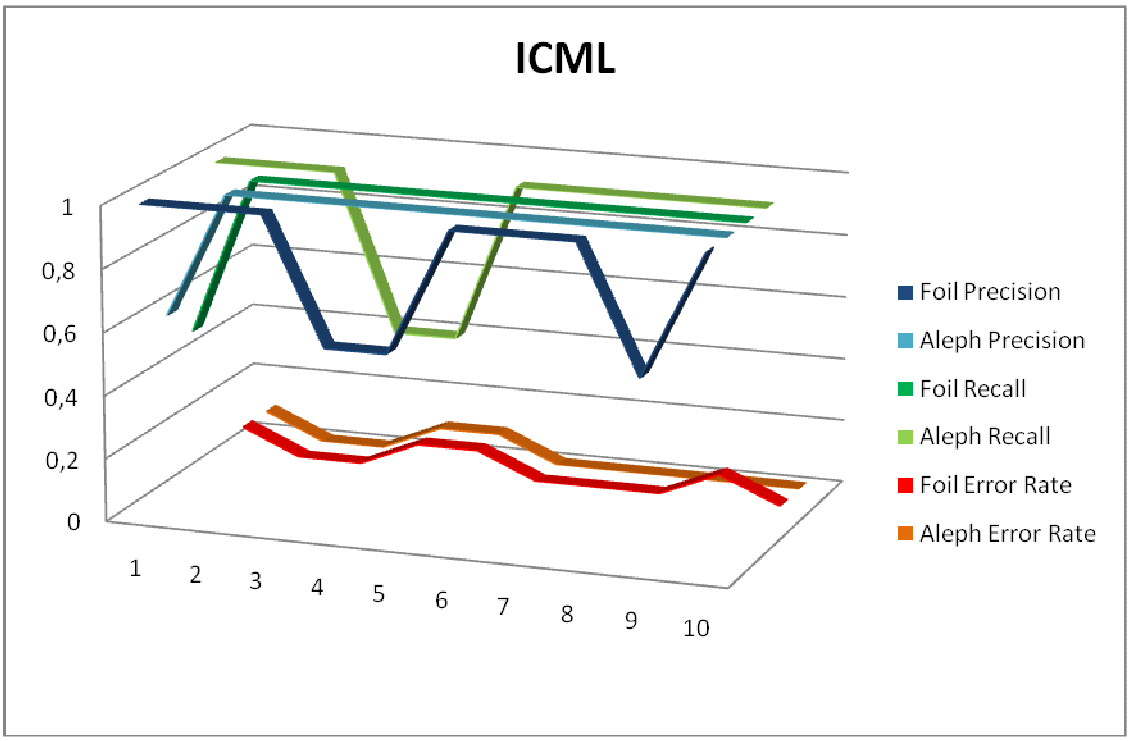
Media Error Rate	FOIL	ALEPH	p-value	H_0	H_a
Ecai	**0,0083**	**0,0083**	**1**	**Accettata**	Rigettata
Elsevier	**0,039742857**	**0,023742857**	**0,57075**	**Accettata**	Rigettata
Icml	**0,03**	**0,25**	**1**	**Accettata**	Rigettata

Analizzando i risultati così ottenuti si può notare che per tutti i tre dataset, il p-value risulta essere maggiore di 0,05. Ciò vuol dire che non c'è una significativa differenza tra i due sistemi di apprendimento e per questo motivo l'ipotesi H_0 non viene rigettata in nessuno dei tre casi.

Per completezza e per avere una visione istantanea dei risultati ottenuti dai due sistemi sui tre dataset, si mostrano una serie di grafici che mettono in relazione le misure di prestazione di Precision, Recall e Error Rate ottenute dai due algoritmi.






ICML
1
0,8
0,6
0,4
0,2
0
1
2
3
4
5
6
7
8
9
10
Foil Precision
Aleph Precision
Foil Recall
Aleph Recall
Foil Error Rate
Aleph Error Rate

6. DISCUSSIONE CONCLUSIVA

In conclusione si può affermare che tutti e tre i dataset testati hanno confermato che non c'è stata una significativa differenza tra i sistemi FOIL e ALEPH. Dalla sperimentazione si è però evinto che ALEPH ha un comportamento più stabile nella definizione delle teorie ed è più flessibile in quanto tramite la specifica dei parametri iniziali è possibile migliorarne le prestazioni bilanciando il tempo di esecuzione rispetto alla bontà dei risultati ottenuti. ALEPH inoltre formula spesso teorie che contengono letterali ground a differenza di FOIL che nei vari test ha dimostrato una preferenza per l'utilizzo di variabili piuttosto che di costanti.

Per quanto riguarda le misure di Precison e Recall per le quali non si è tenuto conto nel test statistico sono state utili in quanto ci consentono di confermare ancora una volta quelli che sono i risultati ottenuti, infatti in tutti i dataset e con tutti i sistemi la precison e la recall non sono mai state inferiori allo 88% (in media i valori si sono aggirati intorno al 95%), un valore abbastanza alto considerato che il caso ideale è che siano entrambe al 100% e che solitamente all'aumentare dell'una, l'altra decresce.

Infine si precisa che per quanto riguarda le prestazioni in termini di tempo di esecuzione si è notato che FOIL è più veloce di ALEPH nella formulazione della teoria, anche se talvolta, su alcune fold, la differenza è stata minima. In ogni caso si parla di tempi di attesa inferiori al minuto.