Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Un Sistema Multi-Agente basato su reti neurali per la ricerca veloce di pagine HTML

G. Pilato, S. Vitabile, G. Vassallo, V. Conti, F. Sorbello

RT-ICAR-PA-03-04 **novembre 2003**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*
– Sezione di Palermo, Viale delle Scienze, 90128 Palermo, URL: *www.pa.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Un Sistema Multi-Agente basato su reti neurali per la ricerca veloce di pagine HTML

G. Pilato[1], S. Vitabile[1], G. Vassallo[2], V. Conti[3], F. Sorbello [1-3]

Rapporto Tecnico N.4: **RT-ICAR-PA-03-04**

Data: **novembre 2003**

---

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sezione di Palermo
[2] CRES – Centro per la Ricerca Elettronica in Sicilia
via Regione Siciliana 49, 90046 Monreale, Palermo, Italia
[3] Università degli Studi di Palermo Dipartimento di Ingegneria Informatica

Sommario

Questo lavoro è incentrato su un sistema multi-agente basato su un classificatore neurale per rapide ricerche su pagine Html. Il sistema è basato sull'architettura EaNet, una rete neurale capace di apprendere le funzioni di attivazione delle sue unità nascoste e avente una buona capacità di generalizzazione.
Lo scopo di questo sistema è la ricerca di documenti che soddisfino una richiesta e che appartengano ad una determinata classe (per esempio, voglio trovare tutti i documenti che contengono la parola "football" e che parlano di "sport").
Il sistema è stato sviluppato usando le caratteristiche di base fornite dalla piattaforma Jade per la creazione degli agenti, la loro coordinazione e il controllo. Il sistema è composto da quattro agenti: il "Trainer Agent", il "Neural Classifier Mobile Agent", l'"Interface Agent" ed il "Librarian Agent".
La conoscenza sub-simbolica del Neural Classifier Mobile Agent è aggiornata automaticamente ogni volta che dall'esterno un qualunque utente richiede dei documenti la cui classe di appartenenza non è tra quelle già considerate.
Il Neural Classifier Mobile Agent interagisce sia con il Librarian Agent per la ricerca dei documenti nei database, sia con l'Inteface Agent per la comunicazione con l'utente.
Il sistema proposto è particolarmente utile per la classificazione dei documenti immagazzinati in database di reti private che, per varie ragioni (per esempio, privacy, sicurezza, ecc.), non possono essere indicizzati da un motore di ricerca esterno.
Il sistema è molto efficiente: i risultati sperimentali preliminari mostrano che nel miglior caso di classificazione l'errore percentuale ottenuto è del 9.98%.

INDICE

1 Introduzione

Con il recente incremento delle librerie digitali è nata la necessità di progettare sistemi automatici che gestissero la ricerca di informazioni nascoste. Il raggruppamento e la classificazione di documenti sono due complesse applicazioni del text-mining [8][11][16], e molti ricercatori hanno proposto vari approcci e tecniche per trattare questo problema [1][7][14].
I documenti possono essere presenti su un database distribuito in una rete di calcolatori, è quindi importante progettare sistemi per aiutare e supportare gli utenti che vogliono informazioni in breve tempo e senza sovraccaricare la rete.
Le piattaforme di sistemi multi-agente provvedono a veloci approcci per il progetto tool per l'analisi di dati intelligente per ambienti distribuiti [5][13][17]. Inoltre, gli agenti mobili possono essere usati per cercare risorse nella rete, eseguire programmi localmente e ritornare i risultati ottenuti. D'altra parte, le architetture di reti neurali possono essere usate per classificare e ricercare documenti; in più, tali architetture neurali, possono essere aggiornate per trattare anche nuove classi di documenti fino a quel momento sconosciute al sistema.
In questo lavoro è presentato un veloce sistema basato sia sull'utilizzo del paradigma degli agenti mobili sia sulle reti neurali per la ricerca automatica e parallela di pagine Html. La principale caratteristica del sistema è la capacità di trattare database contenenti documenti Html le cui classi non sono note e fissate prima. In altre parole, il sistema è capace di auto-addestrare una rete neurale al fine di riconoscere un nuovo tipo di documenti appartenenti ad una classe non trattata fino a quel momento.
Il sistema è composto da quattro agenti, il "Trainer Agent", l'"Interface Agent", il "Librarian Agent", e il "Neural Classifier Mobile Agent".
Il sistema proposto è stato sviluppato usando le caratteristiche di base fornite dalla piattaforma Jade [5], che è conforme alle specifiche FIPA [17], per la creazione, il coordinamento ed il controllo degli agenti.
Nella fase di startup del sistema, il Trainer Agent controlla se il Neural Classifier Mobile Agent è registrato nella piattaforma, se non lo è un processo autonomo, capace di addestrare la rete Neurale EaNet [4], parte su un fissato numero e tipo di classi. Per una questione di pura convenienza, le classi usate per la fase di training sono un sotto insieme di quelle di Google [18]. Questo motore di ricerca è stato scelto dagli autori solo per la disponibilità delle sue API Java che possono così essere usate con la piattaforma Jade [5]. Lo stesso ruolo potrebbe essere preso da un altro motore di ricerca o un insieme di pre-classificati documenti Html. La rete neurale è successivamente racchiusa all'interno del Neural Classifier Mobile Agent.
L'utente richiede, tramite l' Interface Agent di cercare tutti i documenti Html che soddisfino una data query e appartengano ad una data classe (per esempio, tutti i documenti contenenti la parola "football" e appartenenti alla classe "sport"). Se la classe è stata già trattata, allora il Neural Classifier Mobile Agent riceve il messaggio dall'Interface Agent e si clona nel database di documenti registrato nella piattaforma. Al contrario, il Neural Classifier Mobile Agent chide al Trainer Agent di aggiornare la sua rappresentazione sub-simbolica della conoscenza circa la nuova classe. Questo task è legato alla ricerca su internet di un insieme di pre-classificati documenti Html e all'addestramento di una nuova rete neurale EaNet capace di gestire anche la nuova classe. Successivamente, ciascun clone del Neural Classifier Mobile Agent interagisce con il Librarian Agent presente in ciascun database al fine di trovare tutti i documenti

da classificare. Dopo il processo di classificazione, ciascun clone manda un messaggio all'Interface Agent che mostra il risultato all'utente.
Le capacità del sistema sono state verificate con differenti numeri di classi appartenenti all'intervallo 1 – 7. In questo lavoro sono riportati i risultati sperimentali riguardanti le seguenti 7 classi di Google: *Arts, Business, Computers, Games, Health, Science, Sports*.
L'addestramento del sistema è stato eseguito usando un insieme casuale di 1400 pagine Html pre-classificate di Google e il test è stato fatto su altri 1400 nuovi documenti, uniformemente distribuiti rispetto alle classi utilizzate. I documenti sono stati distribuiti su tre computer e i risultati sperimentali preliminari mostrano che il miglior errore di classificazione è di 9.98%.

2 Descrizione della Rete Neurale

L'architettura della rete neurale usata è la EaNet [2][4]. La sua capacità di generalizzazione è stata misurata usando il classico metodo del test-set e il metodo dei fattori di qualità [12].
Subito dopo è riportata una breve trattazione sui fattori di qualità e sull'architettura dell' EaNet.

2.1 Fattori di Qualità

I te fattori di qualità [12], danno una misura, senza l'uso di alcun test-set, della capacità di generalizzazione di una rete neurale feed-forward. In breve, i fattori di qualità sono riferiti ad una tipica architettura feed-forward:

- Q_L è il "Learning Qualità Factor" e quantifica la capacità di addestramento della rete rispetto al solo training set; il suo valore dovrebbe essere zero dopo il processo di training.
- Q_G è il "Generalization Qualità Factor" ed è relativo al gradiente della funzione d'uscita della rete nei punti di training e dà un valore della capacità di generalizzazione della rete; il suo valore dovrebbe essere più basso possibile dopo la fase di training.
- Q_P è il "Production Cost Quality Factor" e stima il costo computazionale come numero di connessioni tra le unità della rete durante la fase di produzione; il suo valore dovrebbe essere il più basso possibile.

2.2 L'architettura EaNet

La rete neurale EaNet è una architettura neurale feed-forward con buona capacità di generalizzazione e capace inoltre di imparare le funzioni di attivazione delle sue unità nascoste durante la fase di training.
La rete dà un basso valore sia di Q_G che di Q_P confrontati con i relativi valori di una tradizionale rete feed-forward che usa le funzioni di attivazione sigmoidali per i suoi neuroni nascosti [4][12]. Questa caratteristica è stata ottenuta tramite la combinazione dell'algoritmo di ottimizzazione CGD con le condizioni di restart di Powell [15] e la formula di regressione di Hermite [6], usando un numero R di funzioni ortonormali di hermite per rappresentare le funzioni di attivazione. Le R funzioni ortonormali di hermite iniziali sono state fissate a priori e poi dinamicamente affinate usando il criterio dei fattori di qualità.
Con queste caratteristiche EaNet è un'architettura neurale molto flessibile e con buona capacità di generalizzazione [2][4][12].

3 Il sistema proposto

Il cuore del sistema è un agente mobile, basato su un'architettura neurale EaNet, che è capace di clonarsi in ciascun computer su cui sono stati memorizzati dei documenti Html e spedire all'utente le informazioni riguardanti solamente i documenti di interesse nascosti.
L'architettura neurale EaNet è usata per classificare automaticamente le pagine Html sulla base di parole contenute al loro interno e che costituiscono un globale sub-simbolico modello di conoscenza circa l'appartenenza a delle classi.
La soluzione proposta ha il vantaggio, rispetto a soluzioni analoghe centralizzate, che l'analisi dei documenti è eseguita da diversi agenti concorrenti, quindi è minore sia il tempo che le risorse utilizzate rispetto al sistema centralizzato [1].
Inoltre, il sistema è particolarmente utile per classificare documenti immagazzinati in database di reti private, che per vari motivi (per esempio, privacy, sicurezza, ecc.) non possono essere indicizzati tramite un motore di ricerca esterno.
Il sistema multi-agente è composto da un insieme di agenti ciascuno dei quali avente una specifica funzione:

1. il Trainer Agent esegue la fase di preprocessing delle pagine Html, la selezione delle caratteristiche e la trasformazione dei documenti in vettori tramite un vocabolario di parole. Tali vettori sono poi utilizzati per addestrare e aggiornare l'architettura EaNet;
2. l'Interface Agent è l'interfaccia tra l'utente e il sistema, esso acquisisce la coppia *(query,class)* e ritorna i documenti nascosti;
3. il Librarian Agent fornisce le informazioni circa la locazione dei database dei documenti;
4. il Neural Classifier Mobile Agent è capace di muoversi tra i database di documenti della rete, e dopo che la rete EaNet è stata addestrata e racchiusa in esso, cerca i documenti che soddisfano la richiesta dell'utente *(query,class).* Esso interagisce con l'Interface e con il Librarian Agent.

Nella fase di startup del sistema, il Trainer Agent controlla se la rete EaNet è stata addestrata; se non è stata addestrata, l'agente cerca in internet o in database locali un insieme di pagine web pre-classificate. Le pagine possono appartenere ad una lista di iniziali pre-classificate classi o ad una nuova classe che è stata richiesta dall'utente. I documenti sono poi usati per addestrare il classificatore neurale. L'architettura EaNet è poi racchiusa nel Neural Classiefier Mobile Agent e l'intero sistema è pronto per essere usato.
L'utente introduce una coppia *(query,class)* tramite l'Interface Agent per la ricerca di tutti i documenti appartenenti alla classe *class* e soddisfacenti la *query*. Se la classe introdotta non stata ancora usata, cioè non è presente all'interno del modello di conoscenza dell'agente mobile, un autonomo processo di aggiornamento per la rete EaNet parte. Tale processo consiste nello scaricare nuovi documenti di addestramento, nell'estrazione delle caratteristiche con conseguente aggiornamento della topologia dell'architettura. L'Interface Agent spedisce un messaggio per il servizio richiesto al Neural Classifier Mobile Agent, che controlla dove sono localizzati i database di documenti nella rete. Questa informazione è fornita dal Jade Directory Facilitator (DF). Conseguentemente il Neural Classifier Mobile Agent si clona sui computer dove vi sono i database. Ciascun clone spedisce un messaggio al

Librarian Agent per ottenere l'informazione circa la posizione del database di documenti locale. Il Librarian Agent comunica l'informazione richiesta spedendo un messaggio all'agente mobile. Il clone, quindi, cerca i documenti locali che soddisfano la richiesta dell'utente. Quindi esso spedisce il risultato all'Interface Agent. Alla fine, l'Interface Agent riunisce tutte le risposte dei vari cloni e mostra il risultato all'utente.

3.1 Il Trainer Agent

Il Trainer Agent è automaticamente invocato durante la fase di startup del sistema o durante la fase di aggiornamento della rete neurale (cioè quando un utente chiede documenti appartenenti ad una nuova classe). Il suo lavoro è costruire un training set per l'architettura neurale EaNet che sia adatto alle classi che devono essere considerate.
Nella fase di startup, è scelto un insieme iniziale di classi. Successivamente, il numero di classi è aumentato in accordo alle richieste dell'utente.
Il processo di training è composto dai seguenti passi:

1. si collezionano i documenti html pre-classificati (da Google o da un database locale);
2. preprocessing delle pagine Html e conteggio delle parole;
3. costruzione del dizionario delle caratteristiche;
4. costruzione del training set;
5. addestramento del classificatore neurale.

3.1.1 Acquisizione dei documenti pre-classificati

La prima ipotesi iniziale è che i documenti di training siano pre-classificati. Questa informazione è necessaria per la costruzione del training set. Un'altra ipotesi è che le pagine considerate siano scritte in Inglese.
Per una questione di pura convenienza, l'insieme di documenti pre-classificati è stato scaricato da internet usando le classi del motore di ricerca di Google [18], il quale è provvisto di una directory di documenti classificati tramite classi di appartenenza ed un insieme di API Java che possono essere facilmente racchiuse nella piattaforma Jade. Usando queste caratteristiche, un insieme di documenti è automaticamente scaricato da internet ed è etichettato con la sua classe di appartenenza.
I documenti di training scaricati sono memorizzati in un database locale per future fasi di aggiornamento della rete EaNet.

3.1.2 Preprocessing

I documenti scaricati sono elaborati come segue:si estraggono da essi solo il testo utile (cioè senza punteggiatura, codice Html e JavaScript, ecc.), mentre le altre parole come gli articoli, aggettivi, avverbi, ecc. sono scartati.

3.1.3 Costruzione delle caratteristiche

I documenti sono codificati in vettori multidimensionali, è quindi necessario identificare un insieme F, chiamato dizionario, di parole significative; queste parole saranno usate per codificare i documenti in vettori, in accordo al paradigma "bag-of-words" [10]. Per selezionare le più appropriate parole che costituiranno il dizionario, è stato usato una versione modificata dell'approccio proposto in [3]. Il metodo considera l'appartenenza di specifiche parole ad una specifica classe.
Diciamo w_i essere la i-esima parola del documento d_j, diciamo w_{tot} essere il numero di differenti parole nella pagina web d_j che appartiene alla classe c_k; per ciascun c_k e per ciascun d_j il termine T_f è definito come segue:

$$TF(w_i, d_j, c_k) = \frac{w_i}{w_{tot}} \tag{1}$$

Per ciascuna classe c_k e per ciascuna parola w_i il più alto valore T_{fmax} (w_i,c_k) di T_f (w_i,d_j,c_k) e la percentuale P_f(w_i,c_k) di documenti appartenenti alla k-esima classe c_k sono calcolati.
L'insieme di tutte le parole w_i estratte dalle pagine di una specifica classe c_k può essere usato per descrivere lo specifico soggetto del documento appartenente alla classe c_k. Per ciascuna parola w_i è calcolata la quantità:

$$\begin{aligned} &maxTF.PF(w_i, c_k) = \\ &= maxTF(w_i, c_k) \cdot PF(w_i, c_k) \end{aligned} \tag{2}$$

Conseguentemente, le parole più rappresentative per una specifica classe c_k hanno il più alto valore di $T_{fmax} \cdot P_f$. Tuttavia, alcune parole che caratterizzano la classe c_k possono essere molto frequenti nell'uso della lingua inglese, quindi possono anche apparire in altre classi. Per questa ragione queste parole dovrebbero essere eliminate. Il processo di selezione quindi applica al termine $T_{fmax} \cdot P_f$ con un peso pari al quadrato dell'inverso del numero di dizionari per classe (Cf) in cui la parola w_i è presente. Questo termine definito $T_f \cdot P_f \cdot IC_f$ è calcolato come segue:

$$TF.PF.ICF(c_k, w_i) = \frac{MaxTF.PF(c_k, w_i)}{CF(w_i)^2} \tag{3}$$

Le parole w_i caratterizzanti ciascuna classe c_k hanno il più alto valore di $T_f \cdot P_f \cdot IC_f$, quindi, per la k-esima classe è selezionato un numero T di parole che la rappresentano. Queste parole costituiscono un insieme F_k i cui elementi caratterizzano i documenti appartenenti alla classe c_k. Diciamo N essere il numero totale di classi di documenti. Il numero T è lo stesso per tutte le N classi.
L'insieme F, definito come segue:

$$F = \bigcup_{i=1}^{N} F_i \tag{4}$$

Costituisce il dizionario delle caratteristiche che sarà usato per codificare i documenti in vettori.

3.1.4 Costruzione del Training Set

I documenti preelaborati sono codificati come vettori multidimensionali in accordo al paradigma "bag-of-words" usando la nota formula Term Frequency Inverse Document Frequency (TFIDF) pesata [10].
Diciamo w_i^* essere una parola del dizionario delle caratteristiche F, diciamo |D| essere il numero dei documenti e $Df(w_i)$ il numero delle pagine in cui la parola w_i è presente, allora la IDF è calcolata come segue:

$$IDF(w_i^*) = \log\left(\frac{|D|}{DF(w_i^*)}\right) \tag{5}$$

Questo valore è piccolo se la i-esima parola è presente in molti documenti, mentre è grande se la parola è presente in pochi documenti. Per il documento di training d_j, il peso $d_j^{(i)}$, di una parola w_i^* contenuta in esso, è dato da:

$$d_j^{(i)} = TF(w_i^*, d_j) \cdot IDF(w_i^*) \tag{6}$$

Quindi, il documento sarà codificato in un vettore multidimensionale come nella formula che segue:

$$\forall w_i^* \in F, doc_j = \left[d_j^{(1)}, d_j^{(2)}, \cdots, d_j^{(m)}\right] \tag{7}$$

Il vettore ottenuto è poi normalizzato.

3.1.5 L'addestramento della Rete EaNet

Dopo l'elaborazione dei documenti di training, la creazione del dizionario F e il corrispondente training set, la rete EaNet è costruita e addestrata.
L'architettura ha un numero di unità di uscita uguale al numero delle classi e un numero di unità di ingresso pari al numero delle caratteristiche presenti nel dizionario F. Il numero delle unità nascoste invece è funzione del numero delle unità di ingresso e di uscita: è calcolato usando una look-up-table determinata sperimentalmente.
La procedure di training consiste nell'addestrare un insieme di architetture EaNet scegliendo quella che soddisfa i seguenti vincoli [12]:

- il valore del fattore Q_L dovrebbe essere zero;
- i valori dei fattori Q_G e Q_P dovrebbero essere i più bassi possibili.

3.2 Il Neural Classifier Mobile Agent

Nella fase di startup del sistema il Neural Classifier Mobile Agent carica al suo interno sia il dizionario delle caratteristiche sia la rete neurale addestrata.
Dopo aver ricevuto un messaggio dall'Interface Agent contenente la coppia *(query,class)*, il Neural Classifier Mobile Agent verifica se la classe introdotta è già stata usata prima in un'altra richiesta di un utente. Se la classe scelta non è già stata usata, allora il Neural Classifier Mobile Agent chiede al Trainer Agent di provvedere al processo di aggiornamento della rete.
Dopo aver verificato che l'architettura EaNet è pronta per l'uso, il Neural Classifier Mobile Agent chiede al Directory Facilitator della piattaforma la locazione dei Librarian Agents. Se non ci sono Librarian Agents disponibili, il Neural Classifier Mobile Agent manda un messaggio all'Interface Agent affinché comunichi all'utente che non è possibile soddisfare la sua richiesta; altrimenti, se ci sono Librarian Agents disponibili, il Neural Classifier Mobile Agent si clona nei corrispondenti computer dove vi sono i database. Ciascun clone chiede al corrispondente Librarian Agent la locazione del database dei documenti, trova i documenti che soddisfano la query dell'utente e li trasforma in vettori multidimensionali. Il documento codificato è quindi dato come ingresso alla rete neurale EaNet affinché verifichi la sua appartenenza alla classe desiderata. Dopo il processo di ritrovamento, ciascun clone spedisce all'Interface Agent la lista dei documenti trovati con la loro rispettiva posizione.
Nel task di classificazione è necessario considerare che le parole di un nuovo documento non contribuiscono all'aggiornamento del dizionario delle caratteristiche F. Per questa ragione è stata introdotta la formula Inverse External Document Frequency (IEDF) di una parola w_i^*, definita come segue:

$$IEDF(w_i^*) = \log\left(\frac{|D|+1}{DF(w_i^*)+1}\right) \tag{8}$$

Avendo introdotto questo termine, è possibile classificare una nuova pagina web che non sia mai stata considerata prima. Il documento è elaborato e codificato in un vettore usando la seguente formula pesata:

$$d_j^{(i)} = TF(w_i^*, d_j) \cdot IEDF(w_i^*) \tag{9}$$

$$\forall w_i^* \in F, doc_j = \left[d_j^{(1)}, d_j^{(2)}, \cdots, d_j^{(m)}\right] \tag{10}$$

Le unità di uscita sono caratterizzate dal più alto valore determinato.

3.3 Il Librarian Agent

Il Librarian Agent è in esecuzione su ciascun computer della rete in cui vi sono memorizzati documenti Html non classificati. Il task di questo agente è comunicare al Neural Classifier Mobile Agent la locazione del database di documenti.

3.4 L’Interface Agent

L’Interface Agent è in esecuzione su ciascun sito accessibile all’utente. Il task di questo agente è ottenere dall’utente la sua richiesta come coppia *(query,class)* (cioè, voglio trovare tutti i documenti che contengano la parola football e appartengano alla classe sport) e spedirla al Neural Classifier Mobile Agent che provvederà alla classificazione dei documenti. Alla fine tale agente unirà le risposte calcolate dai cloni del Neural Classifier Mobile Agent nella rete e mostrerà i risultati all’utente.

4 Risultati Sperimentali

Il sistema è stato implementato usando la piattaforma Jade [5] grazie alla sua versatilità:

- Jade semplifica lo sviluppo di applicazioni multi-agenti ed è conforme alle specifiche FIPA [17];
- usando Jade è possibile costruire una piattaforma distribuita, cioè, un singolo sistema ad agenti suddiviso in differenti host di una rete;
- la piattaforma Jade permette la mobilità degli agenti: è possibile progettare agenti capaci di migrare e di clonarsi in altri host, quindi è molto facile distribuire il lavoro per riconoscimento semantico [9];
- Jade ha un insieme di tool per lo sviluppo e la gestione del sistema creato.

Per la nostra prova abbiamo usato una singola piattaforma Jade distribuita su una rete di quattro computer aventi come sistemi operativi sia Linux che Windows NT (Figura 1). Uno dei computer assume il ruolo di "host principale", dove deve girare il Main Container di Jade. Nel Main Container è presente il Trainer Agent, il Neural Classifier Mobile Agent e i relativi agenti di default di Jade. I container remoti ed il Librarian Agent girano sugli altri tre computer. L'Interface Agent è presente solo nel quarto computer.
Dal momento in cui la piattaforma è attivata, gli agenti di default AMS, DF ed RMA sono lanciati.
L'Agent Managment System (AMS) è l'agente che supervisiona l'accesso e l'uso della Agent Platform (AP). Il Directory Facilitator (DF) è l'agente che provvede al servizio di pagine gialle nella piattaforma. Il Remote Monitoring Agent (RMA) controlla il ciclo di vita dell'AP e di tutti gli agenti registrati nella piattaforma.
I risultati sperimentali mostrati qui sono stati ottenuti con un training set casuale di 1400 documenti Html scaricati da internet usando il motore di ricerca di Google ed appartenenti alle seguenti sette classi: Arts, Business, Computers, Games, Health, Science e Sport.

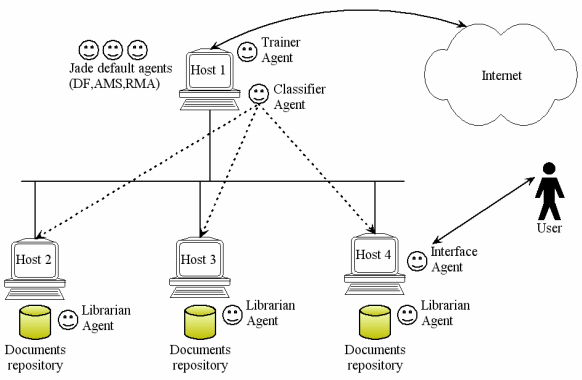


Figura 1. L'architettura proposta

Per la fase di test, un nuovo insieme di 1400 documenti, uniformemente distribuiti sulle sette classi, sono stati divisi nei tre computer: 2,3,4. Il numero T di

caratteristiche per classe è stato fissato sperimentalmente; il numero di unità di ingresso è uguale al numero di caratteristiche presenti nel dizionario F; il numero di unità nascoste è funzione del numero di unità di ingresso e di uscita: esso è determinato tramite un look-up-table.
Le prove sperimentali mostrano che un buon numero di caratteristiche per classe è T=150. La procedura automatica di aggiornamento dell'architettura neurale ha determinato un numero di 978 ingressi (cioè 72 caratteristiche sono condivise tra le sette classi), 7 uscite e 10 unità nascoste con una media di 6 funzioni ortonormali di Hermite per unità nascosta.
In tabella 1 sono mostrati i dettagli degli errori di classificazione per le sette classi. La differenza tra alcuni errori di classificazioni tra classi è dovuta alla scelta casuale dei documenti di training che possono aver sporcare l'insieme di esempi illustrativi delle differenti classi.

Class	**EαNet Classification Error**
Arts	9.98 %
Business	17.87 %
Computers	18.18 %
Games	16.13 %
Health	10.94 %
Science	29.16 %
Sports	16.04 %

5 Conclusioni

In questo lavoro è stato proposto un piccolo sistema, basato sia sul paradigma degli agenti mobili che sulle reti neurali, per il ritrovamento parallelo e automatico di pagine Html. La principale caratteristica del sistema è la sua potenza con i database di documenti Html, le cui classi di appartenenza non sono né note né fissate a priori. In altre parole, il sistema è capace di auto addestrare una rete neurale al fine di riconoscere nuove classi non incluse in un primo insieme di classi. Per le prime prove sperimentali sono stati usati un insieme casuale di 1400 documenti Html appartenenti alle classi di Google per il sistema di addestramento e un nuovo insieme di altri 1400 documenti, uniformemente distribuiti nelle sette classi considerate, per la fase di test. I risultati sperimentali mostrano che il sistema proposto è molto efficiente: i preliminari risultati sperimentali mostrano che il miglior errore di classificazione è di 9.98%.
Il sistema proposto è particolarmente utile per la classificazione dei documenti memorizzati in database distribuiti in reti private che, per vari motivi, (per esempio, privacy, sicurezza, ecc.), non possono essere indicizzati da un motore esterno di ricerca. Inoltre, il sistema non richiede alcuna fase di indicizzazione e, grazie al paradigma degli agenti mobili, i calcoli sono eseguiti localmente e non sovraccaricano la banda della rete.

Bibliografia

1. Brewington B., Gray R., Moizumi K., Kotz D., Cybenko G., Rus D.: “Mobile agents in distributed information retrieval” – Intelligent Information Agents- Springer-Verlag 1—32
2. Cirasa, G. Pilato, F. Corbello, and G. Vassallo: “EaNet: A Neural Solution for Web Pages Classificatio”. Proc. Of 4th World Multi Conference on Systemics, Cybernetics and Informatics, pages 192-197, 2000
3. Esposito F., Malerba D., Di Pace L., Leo P. “A learning intermediary for automated classification of web pages”. Proc. of 16th Int. ICML99 Workshop on Machine Learning in Text Data Analysis, Bled, Slovenia, pages 3746, 1999.
4. Gaglio S., Pilato G., Sorbello F., Vassallo G. “Using the Hermite regression formula to design a neural architecture with automatic learning of the 'hidden' activation functions”. Lecture Notes in Artificial Intelligence 1792, pages 226-237, Jan 2000 Springer-Verlag.
5. *http://jade.cselt.it*
6. Hwang J. N.,You S.,Lay, S.Jou “What's wrong with a cascaded correlation learning network: a projection pursuit learning perspective” ftp://ftp.cis.ohiostate.edu/pub/ neuroprose
7. Kosala R., Blockeel H.: “Web Mining research: a survey” – ACM SIGKDD Explorations, 2000, Vol 2, No 1, 1—15
8. J. Liang, D. Doermann, M.Ma, and J. Guo, “Page Classification Through Logical Labeling”, Proc. Of 16th International Conference on Pattern Recognition, 2002
9. Liu J., Chua T.. Building semantic perceptron net for topic spotting. Proceedings of 37th Meeting of Association of Computational Linguistics (ACL’2001), Toulouse, France, Jul 2001. pp 370-377.
10. Mladenic D., Institute J. S. “Text learning and related intelligent agents: A survey.” IEEE Intelligent Systems, (4):4454, 1999.
11. S. K. Pal, V. Talwar, and P. Mitra: “Web Mining in Soft Computing Framework: Relevance, State of the Art and Future Direction”. IEEE Transaction on Neural Networks, 13 September 2002.
12. Pilato G., Sorbello F., Vassallo G. “An innovative way to measure the quality of a neural network without the use of the test set.” International Journal of Artificial Computational Intelligence, Vol. 5 No 1, 2001, pp:31-36
13. G. Pilato, G. Vassallo, S. Vitabile, V. Conti, and F. Sorbello: “A Concurrent Neural Classifier for Html Documents Retrieval” Lecture Notes in Computer Science, Sprinter Verlag, in press. 2003
14. Porter M. “An algorithm for suffix stripping.” Program (Automated Library and Information Systems), Vol.14 No.3 pages 130137 1980.
15. Powell M.J.D. “Restart procedures for the conjugate gradient method.” Mathematical Programming, 12:241254, 1977.
16. N. Soonthrnphisaj and B. Kijirikul: “The Effects of Different Feature Sets on the Web Page Categorization Problem Using the Iterative Cross-Training Algorithm”. Proc. Of the International Conference on Enterprise and nformation System, ICEIS, July 2001.
17. www.fipa.org - FIPA Specification and Documentation 1997/2000
18. www.google.com

6 Appendice A: Manuale d'uso

La verifica del sistema presentato è stata fatta su una rete di 2 PC, i cui sistemi operativi erano rispettivamente Windows 2000 Professional (Host A) e Linux RedHat 7.3 (Host B). Su entrambi i sistemi occorre installare sia la piattaforma di sviluppo Java (http://java.sun.com/j2se/) sia la piattaforma Jade (http://sharon.cselt.it/projects/jade/).

Nel seguito viene descritto un caso completo di utilizzo del sistema. Prima di cominciare ad immettere le query bisogna installare ed avviare il sistema. Nelle prove effettuate, il sistema è stato distribuito nel modo seguente:

- installare (copiare) l'agente *Interface Agent* (agenteR.class) nell'Host A, in particolare nella cartella "mioagente" che si trova nella directory c:\Jade;

- installare (copiare) gli agenti nell'Host B, in particolare l'agente *Librarian Agent* (agenteS.class) e l'agente *Interface Agent* (agenteR.class) nella cartella "mioagente" che si trova nella directory /SistemaIAT/Jade/, mentre gli agenti *Trainer Agent* (AgenteAddestra) e *Neural Classifier Mobile Agent* (AgenteClone) vanno installati (copiati) nella cartella "miomobile" nella directory /SistemaIAT/Jade/mioagente;

- avviare il sistema.

Descrizione di un caso di utilizzo del sistema
Il caso si riferisce ad una prova completa di utilizzo del sistema, in cui vi è anche la fase di startup del sistema, cioè la rete neurale ancora non è stata addestrata e quindi si deve provvedere a farlo.

Ecco la sequenza dei comandi da digitare su una prima shell dell'host linux:
cd SistemaIAT/Jade/mioagente
java jade.Boot –gui Trainer:AgenteAddestra
(questo permetterà di lanciare la piattaforma JADE e i suoi agenti di default; inoltre l'agente *Trainer Agent* avvierà la fase di startup del sistema con l'addestramento della rete neurale).

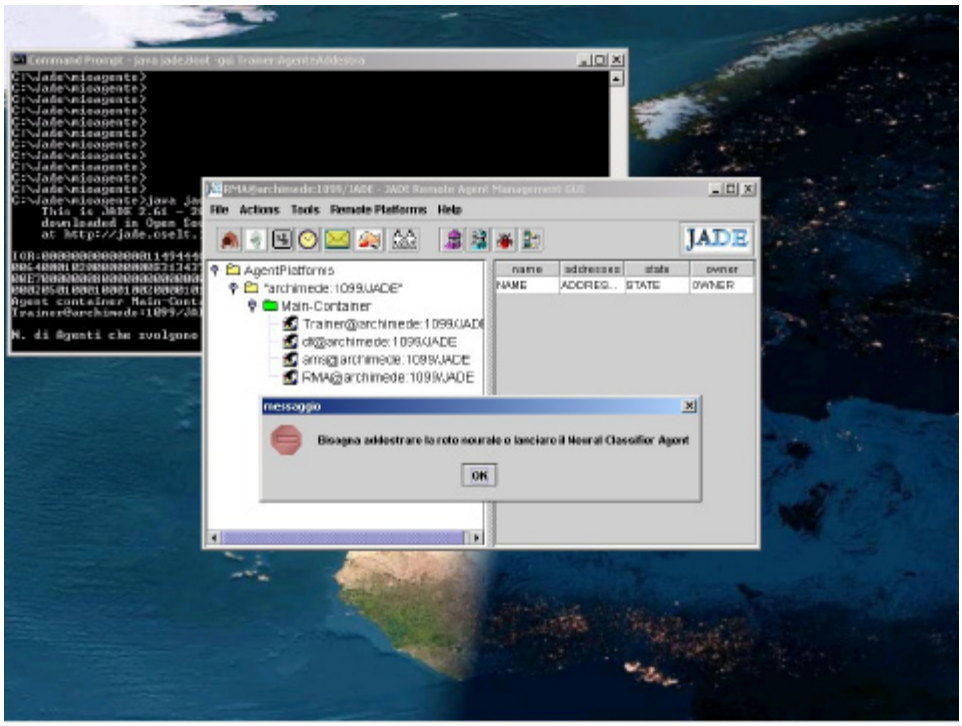
In una seconda shell dello stesso sistema si digiterà:
java jade.Boot –container NeuralClassifier:miomobile.AgenteClone
(si avvierà direttamente l'agente mobile che starà in attesa di una richiesta da parte dell'agente Interface, mentre l'agente Trainer rimane in attesa di eventuali upgrade).

In una terza shell dello stesso sistema si digiterà:
java jade.Boot –container Librarian:AgenteS
(questo aprirà un container nella piattaforma dove risiederà l'agente *Librarian Agent*).

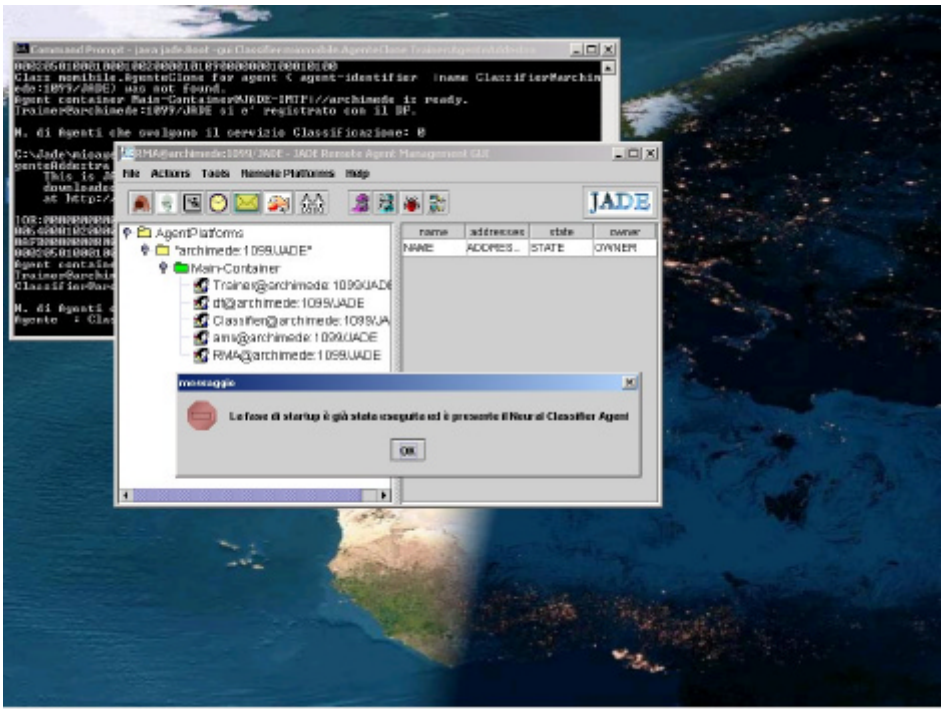
Nell'host Windows 2000 Pro si digiterà (in una finestra dos) la seguente linea di comando:
cd jade\mioagente
java jade.Boot –container –host <indirizzo ip dell'host linux> Interface:AgenteR

(questo permetterà il lancio dell'agente che interagisce con l'utente (*Interface Agent*) e quindi di poter procedere con il ritrovamento dei documenti.

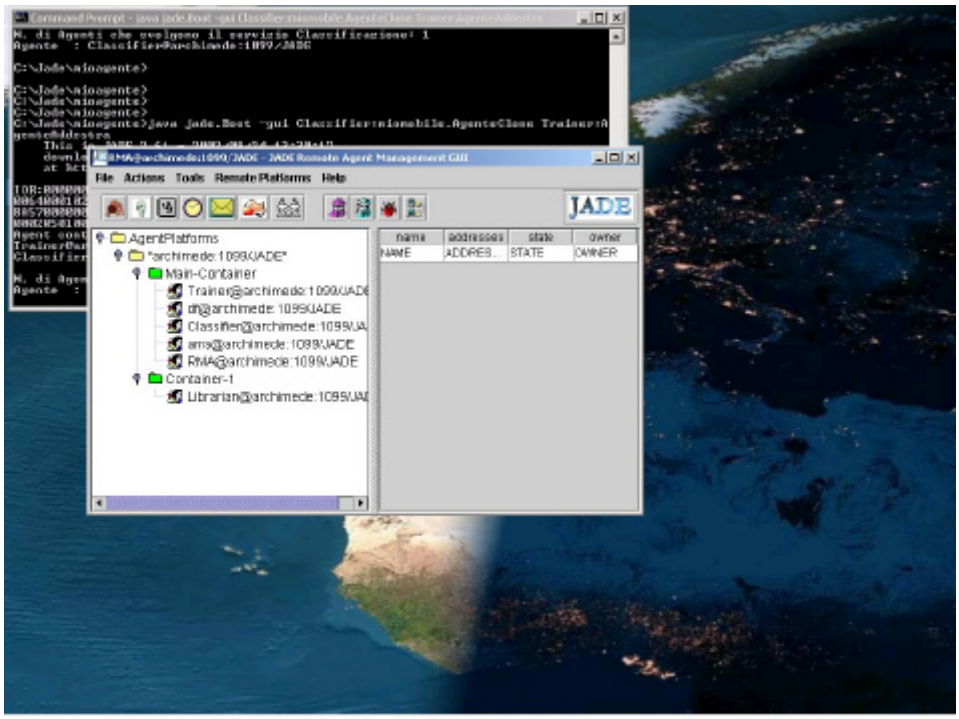
Adesso verranno mostrate le schermate di funzionamento del sistema nel caso sopra descritto.
In questa prima schermata l'utente ha lanciato il primo degli agenti, il *Trainer Agent*, che si occuperà della fase di startup del sistema, cioè addestrerà la rete su un numero di classi predefinito.

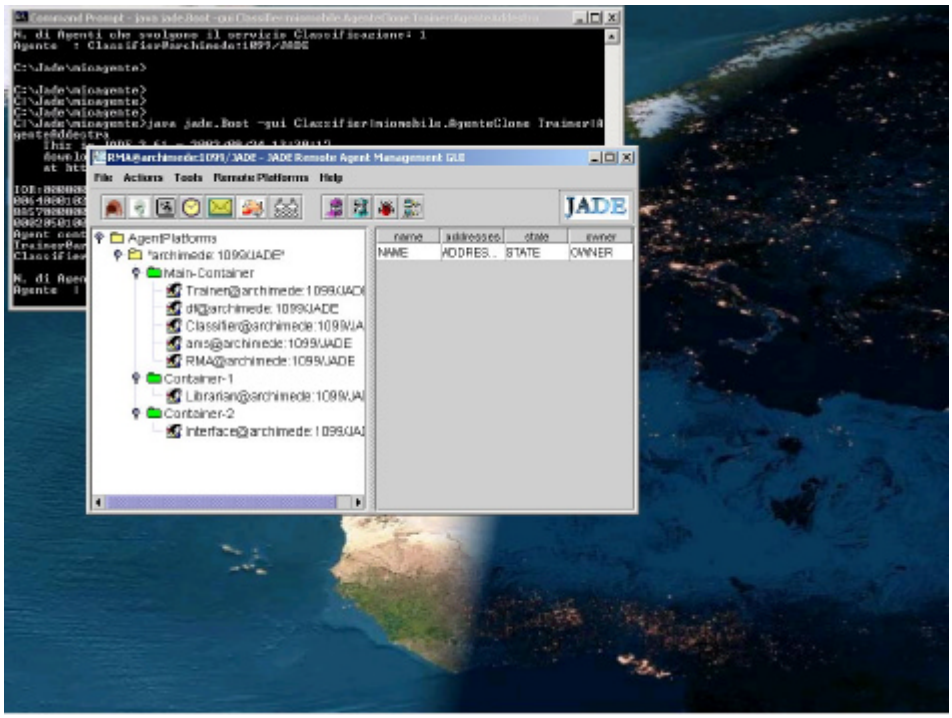


Come si può vedere, è stata avviata la piattaforma Jade con i suoi agenti di default, mentre l'utente ha creato l'agente *Trainer Agent* il cui compito è quello di addestrare la rete neurale e passare le informazioni dell'architettura all'agente *Neural Classifier Mobile Agent*.
Nella figura seguente vediamo infatti la fine del processo di addestramento e la comparsa del *Neural Classifier Mobile Agent* nella piattaforma.

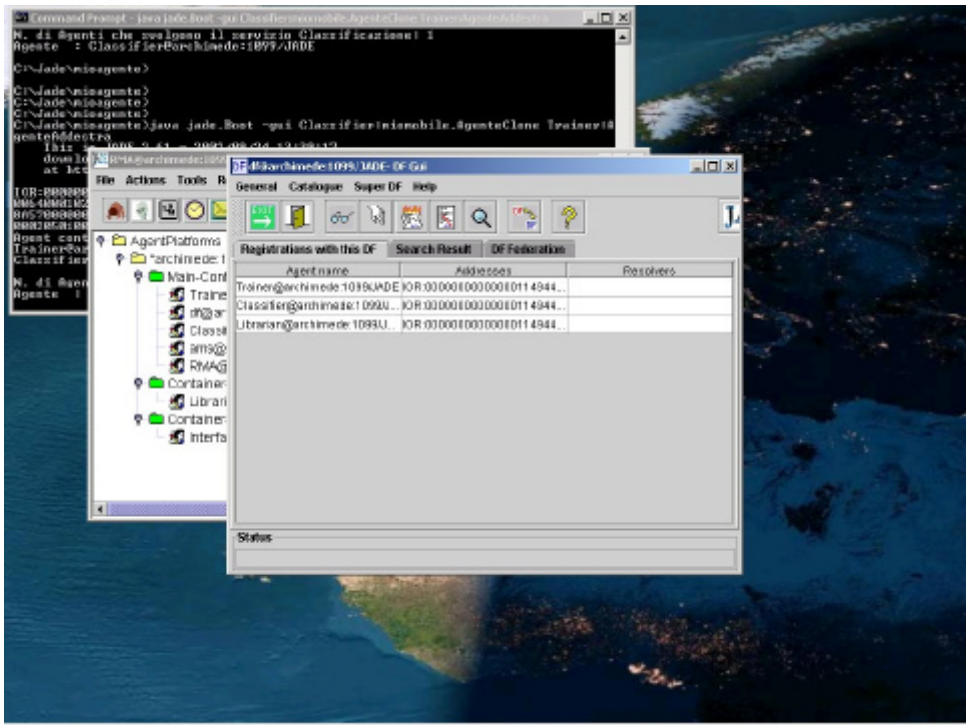


Nelle due successive figure, si possono vedere gli altri due agenti, *Librarian Agent* e *Interface Agent*, in due contenitori distinti. Il *Librarian Agent* è stato avviato dall'host linux e si occuperà di individuare i repository, mentre l' *Interface Agent* si occuperà di interagire sia con l'utente esterno che con il *Neural Classifier Mobile Agent*.

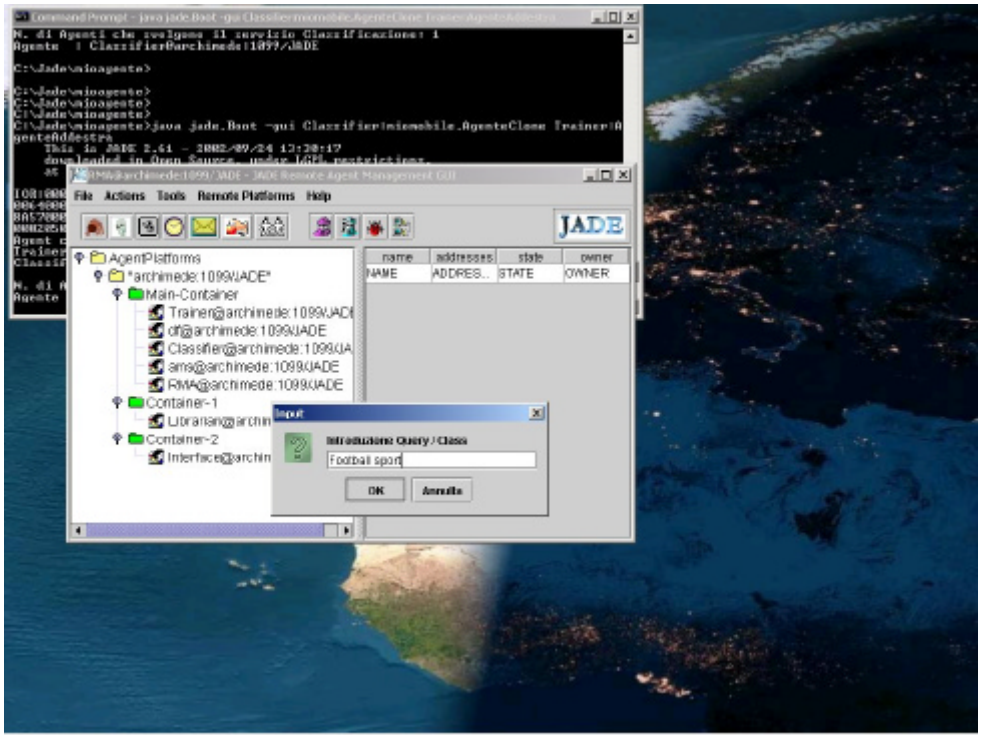




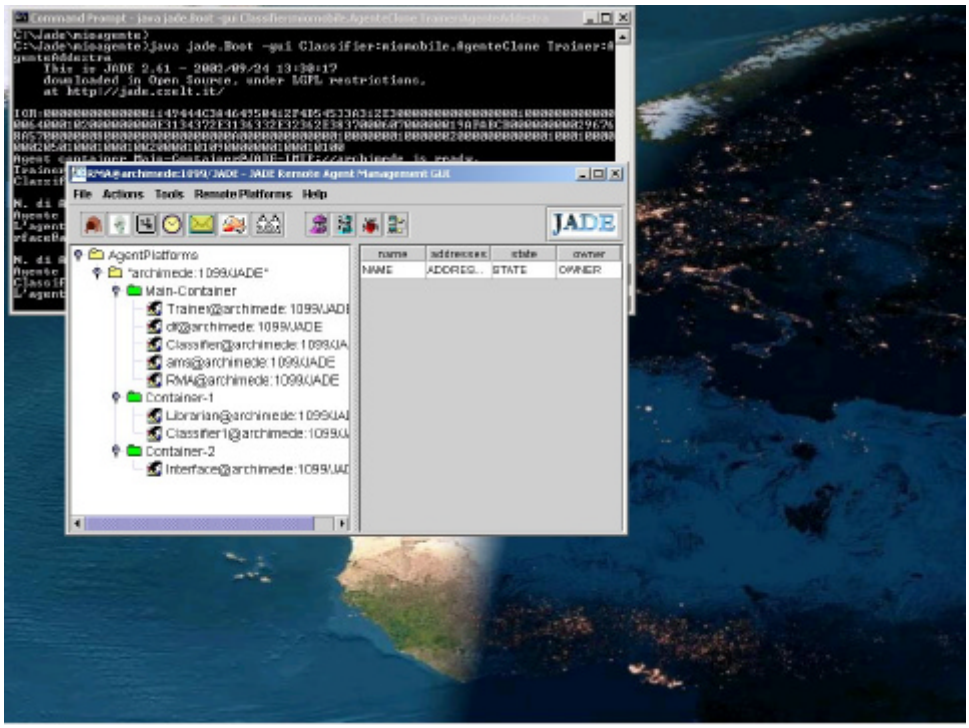
Nella figura successiva è mostrato il DF della piattaforma, cioè l'elenco di tutti gli agenti che offrendo un servizio si registrano nel sistema delle pagine gialle di Jade. Come si può vedere gli agenti che offrono dei servizi sono il *Trainer Agent*, il *Librarian Agent* e l'*Interface Agent*.



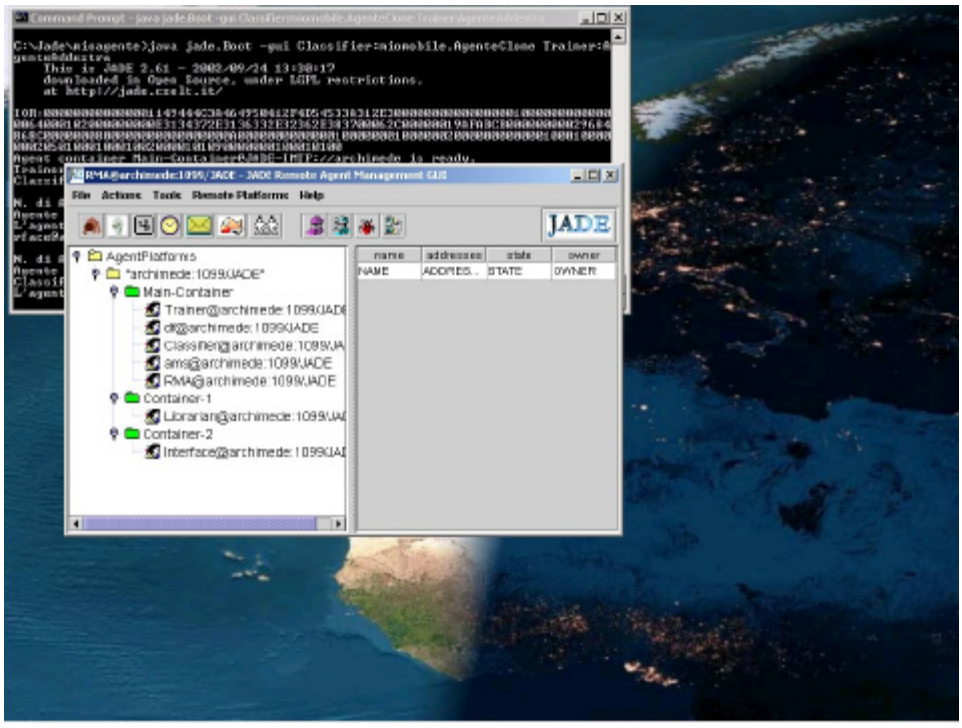
A questo punto può iniziare la fase che riguarda direttamente l'utente, il quale introduce la sua *(query,class)*, cioè chiede al sistema di cercare in tutti i repository disponibili nella rete, tutti i documenti HTML che contengono, ad esempio, la parola "**football**" e che parlino di "**sport**".



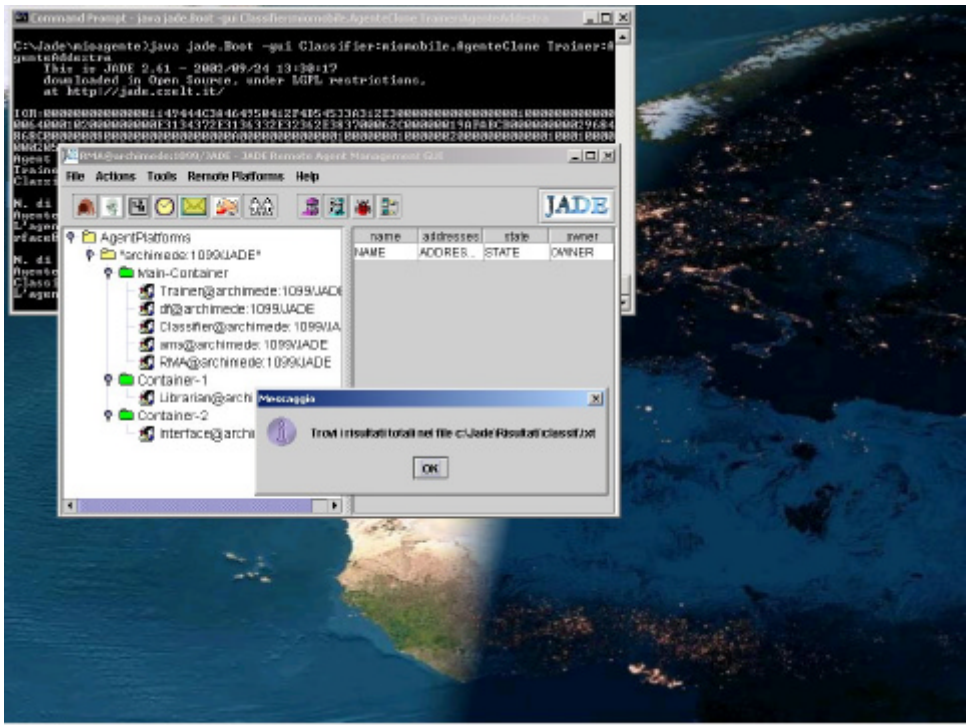
Acquisite le informazioni, l'agente *Interface Agent* passa la richiesta al *Neural Classifier Mobile Agent* che interroga il DF per verificare se esistono e dove sono i repository di documenti HTML, cioè se nel DF vi è registrato qualche *Librarian Agent*.
A questo punto l'agente *Neural Classifier Mobile Agent* si clona in tutti gli host in cui vi è la presenza di *Librarian Agent* ed esegue la ricerca dei documenti di interesse.
Nella seguente figura infatti si può notare che l'agente *Neural Classifier Mobile Agent* si è clonato nel container in cui si trova l'agente *Librarian Agent*, passandogli la richiesta.



Il *Neural Classifier Mobile Agent* ottenuto un risultato, lo comunica all'*Interface Agent* e si distrugge, avendo, questo clone, terminato la sua missione.



Non appena l'*Interface Agent* riceve i risultati, li mostra all'utente, che quindi conosce dove si trovano i documenti di interesse.



Nell'ultima figura si può vedere il file creato con i relativi container e path per trovare i documenti richiesti.