Università degli Studi di Modena e Reggio Emilia

Facoltà di Ingegneria – Sede di Modena

Corso di Laurea in Ingegneria Informatica – *Nuovo Ordinamento*

TUCUXI
Un agente intelligente per la ricerca di sorgenti informative in Internet

Relatore:
Prof. Sonia Bergamaschi

Candidato:
Gozzi Daniele

Anno Accademico 2003-2004

Indice generale

1 Introduzione....5
1.1 Descrizione delle problematiche affrontate....5
1.2 Modalità di soluzione proposte....6
2 Componenti utilizzate nello sviluppo....7
2.1 Linguaggio di programmazione....7
2.2 Servizi....7
2.2.1 DataBase Management Server (DBMS)....7
2.2.2 WordNet....8
2.2.3 SOAP....10
2.3 Toolkit e librerie....13
2.4 Apache Ant....13
2.4.1 JADE - Java Agent DEvelopment Framework....14
2.4.1.1 Note per l'installazione di Jade....15
2.4.2 Part Of Speech (POS) tagger....17
3 Descrizione degli algorimi....18
3.1 Dall'Ontologia ad un Common Thesaurus....18
3.2 Disambiguazione lessicale....19
3.2.1 Premessa....19
3.2.2 Procedimento....20
3.2.2.1 Elaborazione preliminare....21
3.2.2.2 Disambiguazione dei lemmi....22
3.3 Costruzione delle catene lessicali....25
3.4 Calcolo dell'affinità semantica....28
3.4.1 Ricerca di parole chiave....28
3.4.1.1 Procedimento....29
3.4.2 Ricerca attraverso un Common Thesaurus....33
3.4.2.1 Procedimento....34
4 Deployment - Utilizzo dell'agente hunter....36
4.1 Installazione....36
4.2 Esecuzione....37
4.2.1 Applet di controllo....37
4.2.1.1 Crawling ricorsivo....39
4.2.1.2 Ricerca di parole chiave....39
4.2.1.3 Ricerca attraverso un Common Thesaurus....40
4.2.1.4 Scheda di configurazione....40
4.2.2 SocketProxyAgent....41
5 Prove di funzionamento....44
5.1 Crawling ricorsivo ed estrazione delle catene lessicali....44
5.2 Ricerca di parole chiave....46
5.3 Ricerca basata su un Common Thesaurus....48
5.4 Tempi di esecuzione....49
6 Conclusioni....49
7 Bibliografia....51

1 Introduzione

Durante lo svolgimento della tesi è stato realizzato un agente per la ricerca di informazioni pertinenti a un certo dominio di appartenenza. Tale agente è stato identificato con il nome di Agente Hunter (in inglese Hunter Agent, abbreviato HA).

Gli agenti hunter si inseriscono nell'ambito dei framework di accesso e integrazione dell'informazione (II). Il progetto MIKS [1]– MOMIS è un esempio di tale organizzazione: il suo scopo è l'estrazione di informazioni provenienti da sorgenti dati strutturate e semistrutturate. Per compiere l'estrazione viene introdotto un linguaggio orientato agli oggetti dotato di una propria semantica basata su una Description Logics. Tale linguaggio, che deriva dallo standard ODMG, è chiamato ODL-I3. L'integrazione delle informazioni viene compiuta in modo semi-automatico, utilizzando la conoscenza presente in un Common Thesaurus (definito utilizzando il framework), le sottostanti descrizioni ODL-I3 degli scemi sorgenti e alcune tecniche di clustering e di Description Logics. Il processo di integrazione definisce una vista virtuale integrata degli scemi sottostanti (il Global Schema) nella quale sono specificate regole di mapping e vincoli di integrità per la gestione delle eterogeneità.

I progetti di questo tipo mirano a fornire un accesso unificato all'informazione attraverso una Vista Virtuale Globale (Global Virtual View, GVV), che sostanzialmente fornisce un accesso uniforme a insiemi di dati eterogenei tra loro. Per ottenere questo scopo è evidente che i nomi delle colonne nelle tabelle di dati non sono un supporto sufficiente. L'integrazione è possibile solo se i dati disponibili vengono catalogati *semanticamente*, non solo letteralmente come sarebbe possibile fare con un qualsiasi DBMS[1].

Uno dei passi da compiere per fornire questo tipo di servizio è l'integrazione delle sorgenti (eterogenee) che sono rese disponibili da terze parti. Tali sorgenti potrebbero essere siti web, database aziendali o anagrafici, archivi di ogni tipo. Una descrizione delle finalità e delle funzioni del sistema di integrazione è descritta approfonditamente in [2].

1.1 Descrizione delle problematiche affrontate

Un requisito fondamentale per un sistema di Integrazione dell'Informazione è la capacità di identificare l'area di semantica a cui fa riferimento il contenuto delle sorgenti di dati con cui entra in contatto.

Occorre quindi un componente che sia in grado di **effettuare ricerche a**

1 DBMS – DataBase Management System, o *motore* database. Si tratta di una categoria di programmi software progettati per archiviare grandi quantità di informazioni e consentirne la consultazione, la modifica e la cancellazione.

livello semantico in grandi archivi di testi non strutturati, con particolare riferimento al Web.

1.2 Modalità di soluzione proposte

L'agente hunter è in grado di effettuare ricerche e fornire risultati coerenti con il contesto che ha originato la ricerca, non solo con le parole chiave eventualmente fornite. Tale contesto viene rappresentato dal concetto di *ontologia.*

Con l'appoggio di un motore di ricerca “tradizionale”, l'agente utilizza un limitato insieme di parole chiave, volutamente generico, per ottenere un primo elenco di pagine Web che hanno una limitata probabilità di essere correlate all'ontologia di ricerca. Successivamente, l'agente raffina la ricerca utilizzando un algoritmo euristico che è in grado di ricavare gruppi di lemmi attinenti tra loro da ciascuna pagina e calcolare un punteggio di affinità nei confronti dell'ontologia di ricerca. I risultati “letterali” del motore di ricerca, così filtrati vengono presentati all'utente del servizio di ricerca.

Dal momento che l'agente hunter provvede al reperimento di dati per un sistema di Information Integration, esso deve necessariamente possedere la capacità di ottenere informazioni di livello semantico da insiemi di dati che non dispongono di un tale tipo di catalogazione. L'operazione che deve quindi essere in grado di svolgere è quella che una persona compie quando, inserita una stringa in un motore di ricerca online, essa valuta il grado di interesse che ciascun risultato suscita nei confronti dell'idea che ha originato la ricerca.

A causa del costante aumento di materiale disponibile nel Web, diventa sempre più difficile la ricerca di pagine di interesse basata su semplici parole chiave. Un approccio semantico al Web consente una ricerca più accurata e riesce a eliminare le pagine non pertinenti tra quelle contenenti le parole chiave desiderate. In questo modo diventa possibile discriminare le pagine “trappola” realizzate appositamente per comparire tra i risultati del maggior numero di ricerche, includendo nella pagina dei veri e propri dizionari di termini frequentemente cercati.

2 Componenti utilizzate nello sviluppo

2.1 Linguaggio di programmazione

Il linguaggio selezionato per l'implementazione è Java, proprietà di Sun Microsystems. La scelta è stata operata tenendo conto dell'alta portabilità che tale linguaggio offre e della semplicità che lo contraddistingue. Una semplicità che nella costruzione di sistemi software complessi non è assolutamente negativa, ma anzi, privilegiando la modularità del codice spinge al riutilizzo, al risparmio e alla creazione di ampia documentazione.
Tutto il progetto MIKS è stato scritto con Java, quindi le librerie rese disponibili da tale progetto sono facilmente utilizzabili da un agente implementato in Java.
Il paradigma di programmazione ad agenti prevede la mobilità degli stessi agenti, questo significa che il loro stesso programma (oltre ai blocchi di memoria allocata) deve potere migrare da un nodo della piattaforma ad agenti ad un altro, spesso attraverso una rete informatica. L'unico linguaggio di programmazione che fornisce una completa compatibilità per il codice binario compilato è Java. Il *bytecode* Java può essere eseguito su qualsiasi sistema che disponga di una Java Virtual Machine senza alcuna modifica. Questo rende possibile la migrazione degli agenti software, che non devono preoccuparsi della compatibilità binaria nei confronti del sistema su cui migrano.
Il framework Jade per sistemi ad agenti è disponibile solo nella sua implementazione Java. Benché sia possibile creare una interfaccia tra Java e altri linguaggi di programmazione (ad esempio tramite JNI, *Java Native Interfaces*), la scelta di gran lunga preferibile è stata implementare l'agente usando Java.

2.2 Servizi

2.2.1 DataBase Management Server (DBMS)

L'archiviazione di grandi quantità di dati in modo indipendente dal file system su cui si trovano è la caratteristica che rende oggi utile utilizzare un DBMS quando ci si trova a dovere gestire medie e grandi basi di dati. L'interfaccia attraverso un linguaggio di alto livello come SQL, convogliato su socket in rete, rende l'accesso ad un DBMS estremamente trasparente nei confronti delle applicazioni, che non devono necessariamente risiedere sullo stesso host su cui il server DBMS è in esecuzione.
A causa della fama di grandi prestazioni su query semplici e grandi tabelle di dati si è scelto di utilizzare il DBMS MySQL [3].

L'interfaccia dal codice Java dell'agente hunter al database sfrutta le classi JDBC, distribuite da MySQL, che rendono l'esecuzione di codice SQL sul database assolutamente trasparente, valorizzando la velocità che questo particolare DBMS garantisce.
Il DBMS rende disponibile anche l'accesso al dizionario WordNet (cap. 2.2.2), che come vedremo è usato per assegnare un significato ai termini di un testo (una operazione che tecnicamente è chiamata annotazione).
L'interfacciamento attraverso un DBMS permette di mettere in atto particolari accorgimenti che velocizzano l'elaborazione che l'agente hunter compie, perciò è stata scelta tra le due alternative proposte. Questi accorgimenti vanno dal mantenimento di una connessione al database, che diminuisce il tempo di latenza di ciascuna ricerca, all'ottimizzazione di query per raggruppamento e all'ordinamento preventivo delle tabelle in funzione del campo ricercato. La velocità nell'esecuzione delle query è un fattore di importanza critica per l'agente hunter, perché costituisce da solo la parte più gravosa computazionalmente. La possibilità di produrre query complesse viene pagata però con la necessità di installare, configurare e tenere in esecuzione un server per database.

2.2.2 WordNet

WordNet è un grande database lessicale che comprende tutti i lemmi della lingua inglese. Il dizionario è reso disponibile in due modalità: una con interfaccia da shell attraverso opportuni comandi e una come dump SQL da caricare in un DBMS, in questo modo le interrogazioni possono essere effettuate attraverso normali query SQL.
La principale caratteristica che rende WordNet indispensabile in una applicazione semantica è che la catalogazione dei lemmi è stata effettuata tenendo conto dei molteplici significati che a volte una parola può assumere. Ciascun significato è identificato attraverso un numero naturale e viene indicato con il termine ***synset***.
Al nome “computer”, ad esempio, corrispondono due possibili synset:

```
mysql> SELECT LEM.LEMMA, SYN.WN_SYNSET_ID, SYN.GLOSS FROM WN_LEMMA LEM,
WN_LEMMA_SYNSET LEMSYN, WN_SYNSET SYN WHERE LEM.WN_LEMMA_ID = LEMSYN.WN_LEMMA_ID
and LEMSYN.WN_SYNSET_ID = SYN.WN_SYNSET_ID and SYN.SYNTACTIC_CATEGORY = 1 and
LEM.LEMMA="computer";
```

LEMMA	WN_SYNSET_ID	GLOSS
computer	14866	a machine for performing calculations automatically
computer	44124	an expert at calculation (or at operating calculating machines)

```
2 rows in set (0.00 sec)
```

Inoltre WordNet comprende una tabella che descrive tutte le relazioni note tra i synset, rendendo in questo modo possibile stabilire le eventuali affinità semantiche che legano due parole. Le relazioni sono orientate e sono applicabili solo ad alcuni tipi di synset (il tipo di un synset è indicato dalla colonna SYNTACTIC_CATEGORY nella tabella WN_SYNSET).
I tipi di relazione classificati da WordNet, sono riassunti nella tabella seguente:

Relazione	***Tipi interessati***	***Simmetria***
Antonym	nomi, verbi, aggettivi, avverbi	√
Hypernym	nomi, verbi	√
Hyponym	nomi, verbi	√
Member Meronym	nomi	√
Substance Meronym	nomi	√
Part Meronym	nomi	√
Member Holonym	nomi	√
Substance Holonym	nomi	√
Part Holonym	nomi	√
Attribute	nomi, aggettivi	√
Entailment	verbi	X
Cause	verbi	X
Also see	verbi, aggettivi	X
Verb Group	verbi	X
Similar to	aggettivi	√
Participle of verb	aggettivi	X
Pertainym	riguarda nomi, aggettivi	X

Ogni relazione esistente tra due synset è di uno dei tipi elencati. La proprietà *simmetria* indica che la presenza di una relazione tra il synset A e il synset B comporta necessariamente la presenza di una relazione (eventualmente di diverso tipo) tra B e A.
Complessivamente, WordNet comprende 152'059 lemmi, che corrispondono a 115'424 synset, per un totale di 203'145 coppie lemma-synset.

2.2.3 SOAP

I servizi offerti dall'agente hunter sono relativi a ricerche di sorgenti dati appartenenti ad un determinato ambito semantico, ma come è logico non è possibile iniziare una ricerca di questo tipo con un crawling completo del Web. Il procedimento, almeno inizialmente, ha la necessità di avere un insieme di parole chiave da ricercare letteralmente medianto un motore di ricerca online. I risultati così ottenuti, solo potenzialmente attinenti con l'ontologia di ricerca, vengono raffinati con algoritmi euristici più complessi (descritti in seguito nel capitolo 3.2). Nello scegliere un motore di ricerca esterno adatto allo scopo, la scelta è caduta su Google [4], grazie al fatto che fornisce una comoda interfaccia SOAP [5] (oltre a una definizione WSDL [6] del servizio) per applicazioni che come l'agente hunter richiedono una ricerca nel web.

SOAP è il protocollo attraverso cui Google mette a disposizione le proprie "Web APIs" [4] : un servizio web avviato in via sperimentale che consente a chi lo desidera di potere eseguire ricerche nel web all'interno delle proprie applicazioni. Google inoltre mette a disposizione una libreria per Java che si accolla tutta la gestione di RPC[2] necessaria e il parsing XML, semplificando ulteriormente il compito di chi desidera una interfaccia al motore di ricerca nella propria applicazione.

Questo tipo di accesso ad una sorgente di dati è solo uno dei tre principali modi con cui è possibile accedere ad una sorgente di dati. Tali modalità sono:

- Analisi (*parsing*) di contenuti preesistenti
- Accesso diretto a database
- Servizi web

All'ultimo tipo appartiene l'interazione con Google attraverso le Web API utilizzando il protocollo SOAP, questo tipo di comunicazione verso una sorgente dati è in pratica un accesso indiretto ad un database, mediato da *middleware*, cioè software impiegato ad un livello intermedio tra database e server.

SOAP fornisce la definizione dello standard per i dati XML che devono essere scambiati tra host cooperanti in un sistema software distribuito. Tale protocollo non prevede l'esistenza di stati nella comunicazione, la quale viene condotta in un solo senso, benché le applicazioni possano definire funzionamenti più avanzati.

SOAP non definisce la semantica del contenuto a livello applicativo dei messaggi che trasporta, quindi per ciascuna realizzazione che utilizza questo

2 RPC – Remote Procedure Call, o chiamata di procedure da remoto. Si tratta di un sistema per richiedere l'esecuzione di una particolare computazione da parte di un servizio residente su un host diverso dal richiedente. Tale sistema comprende il trasferimento di eventuali parametri da passare alla funzione richiesta e l'invio della risposta contenente il risultato elaborato.

protocollo sarà necessario progettare la semantica che il contenuto dei messaggi deve osservare.
Uno dei possibili utilizzi di SOAP è la comunicazione per RPC. Attraverso RPC è possibile che un programma in esecuzione su un host richieda ad un secondo host l'esecuzione di una funzione, passando i parametri e ricevendo i risultati con SOAP.
I messaggi scambiati con SOAP sono costituiti da un contenitore principale, il quale contiene una intestazione opzionale e il corpo del messaggio, obbligatorio. Attraverso l'intestazione è possibile estendere il funzionamento di SOAP implementando comunicazioni bidirezionali, frammentazione del contenuto in più messaggi e quant'altro.
La comunicazione attraverso SOAP può essere diretta tra mittente e destinatario o può passare attraverso uno o più intermediari. Ciascun intermediario elabora i messaggi in arrivo secondo le indicazioni presenti nelle intestazioni e può ritrasmetterli modificati come ritiene opportuno, ad esempio aggiungendo alcuni valori. Il comportamento da seguire è determinato dal *ruolo* che un nodo assume nell'ambito di una particolare comunicazione. Esistono tre ruoli predefiniti: *none*, *next* (per i nodi intermedi) e *ultimateReceiver* (destinatario).
A titolo di esempio, viene riportato di seguito parte del payload di un pacchetto TCP intercettato mentre veniva mandato all'host di Google deputato a eseguire le richieste delle Google API (il pacchetto contiene dati HTTP inviati con il metodo POST).

```
<?xml.version='1.0'.encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:doGoogleSearch.xmlns:ns1="urn:GoogleSearch"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <key.xsi:type="xsd:string">
        [omissis]
      </key>
      <q.xsi:type="xsd:string">
        &quot;mozilla&quot; AND &quot;png&quot; AND &quot;overflow&quot;
      </q>
      <start.xsi:type="xsd:int">0</start>
      <maxResults.xsi:type="xsd:int">10</maxResults>
      <filter.xsi:type="xsd:boolean">true</filter>
      <restrict.xsi:type="xsd:string"></restrict>
      <safeSearch.xsi:type="xsd:boolean">false</safeSearch>
```

```
        <lr.xsi:type="xsd:string"></lr>
        <ie.xsi:type="xsd:string">UTF-8</ie>
        <oe.xsi:type="xsd:string">UTF-8</oe>
      </ns1:doGoogleSearch>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Si nota che effettivamente il contenuto inviato è in formato XML e che in questo particolare messaggio SOAP viene omessa l'intestazione facoltativa, mentre viene riportato il corpo del messaggio.
All'interno del contenitore *Body* vengono specificati nome, tipo e valore di ciascuno dei parametri necessari alla ricerca.
Similarmente, nel messaggio di risposta vengono elencati i risultati della ricerca, specificando per ognuno i dati che lo caratterizzano: URL, frammento di testo che ha dato luogo alla corrispondenza, *encoding*...

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/1999/XMLSchema">
   <SOAP-ENV:Body>
      <ns1:doGoogleSearchResponse
         xmlns:ns1="urn:GoogleSearch"
         SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <return.xsi:type="ns1:GoogleSearchResult">
         <directoryCategories
            xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
            xsi:type="ns2:Array"
            ns2:arrayType="ns1:DirectoryCategory[0]">
         </directoryCategories>
         <documentFiltering.xsi:type="xsd:boolean">true</documentFiltering>
         <endIndex.xsi:type="xsd:int">10</endIndex>
         <estimateIsExact.xsi:type="xsd:boolean">false</estimateIsExact>
         <estimatedTotalResultsCount
            xsi:type="xsd:int">
            4350
         </estimatedTotalResultsCount>
         <resultElements
         xmlns:ns3="http://schemas.xmlsoap.org/soap/encoding/"
         xsi:type="ns3:Array"
         ns3:arrayType="ns1:ResultElement[10]">
            <item xsi:type="ns1:ResultElement">
               <URL xsi:type="xsd:string">
                  http://www.k-otik.com/exploits/08112004.libpng.c.php
               </URL>
               <cachedSize.xsi:type="xsd:string">25k</cachedSize>
```

```
            <directoryCategory xsi:type="ns1:DirectoryCategory">
               <fullViewableName xsi:type="xsd:string"></fullViewableName>
               <specialEncoding xsi:type="xsd:string"></specialEncoding>
            </directoryCategory>
            <directoryTitle xsi:type="xsd:string"></directoryTitle>
            <hostName xsi:type="xsd:string"></hostName>
            <relatedInformationPresent.xsi:type="xsd:boolean">
               true
            </relatedInformationPresent>
            <snippet xsi:type="xsd:string">
               LibPNG.Graphics.Library.Remote.Buffer.&lt;b&gt;Overflow&lt;/b&gt;.Exploit
               ../*.*.exploit.&lt;b&gt;...&lt;/b&gt;.line..*.the&lt;br&gt;.output.is.not
               .an.enire.&lt;b&gt;png&lt;/b&gt;,.just.enough.to.trigger.the.bug..&lt;b&g
               t;...&lt;b&gt;..
            </snippet>
            <summary xsi:type="xsd:string"></summary>
            <title xsi:type="xsd:string">
               KOTik.:.LibPNG.&lt;b&gt;Mozilla&lt;/b&gt;.Firefox.Graphics.Library.Remote
               .Buffer.&lt;b&gt;...&lt;/b&gt;
            </title>
         </item>
[...]
      </resultElements>
   </ns1:doGoogleSearchResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2.3 Toolkit e librerie

2.4 Apache Ant

Proprietà di: The Apache Software Foundation

Licenza d'uso: Apache License, Version 2.0

Ant è un tool che gestisce la compilazione, l'esecuzione e l'installazione dei programmi scritti in Java. Le funzionalità di questo tool sono simili a quelle del tradizionale Make impiegato nei sistemi Unix da sempre, ma la sintassi dei file contenenti le istruzioni è completamente diversa. L'autore del tool non approva la grande rigidità del formato che i classici Makefile devono avere, egli ha quindi deciso di scrivere un software in Java i cui file di istruzioni per la compilazione sono testi XML.

Il successo di questo approccio è stato tale che Ant è ormai lo standard di fatto per la gestione dei sorgenti scritti in Java. Una prestigiosa compagnia del calibro di Sun Microsystems ha avallato questa supremazia includendo un supporto completo per Ant all'interno dei propri IDE (Integrated Development Environment, ambiente di sviluppo integrato) per Java.

All'interno del progetto sviluppato nel corso della tesi, Ant è stato impiegato

con successo per gestire la compilazione dell'agente hunter.

2.4.1 JADE - Java Agent DEvelopment Framework

JADE è un ambiente software orientato allo sviluppo di agenti software conformi agli standard FIPA [7] (Foundation for Intelligent Physical Agents), la fondazione che emana le specifiche inerenti la tecnologia ad agenti.
Il progetto JADE è gestito da Telecom Italia Lab [8], che lo rende disponibile con licenza LGPL (Library General Public Licensee) a chiunque voglia sviluppare i propri agenti. Il tipo di licenza adottato rende possibile disporre del codice sorgente di JADE, eventualmente modificarlo e distribuirlo nuovamente, ma solo con la stessa licenza. La libertà di modifica contestuale all'obbligo di mantenere “libero” il software è la principale caratteristica della filosofia Open Source, da cui deriva la licenza LGPL.
La funzione di JADE è quella di implementare i servizi necessari alla creazione e al mantenimento di un sistema multi-agente, fornendo al programmatore una semplice API che consente di gestire le comunicazioni e il ciclo di vita degli agenti: istanziazione, messa in opera, comportamenti, migrazione, terminazione. In quest'ottica JADE viene classificato come componente *middleware*.
Gli agenti vengono raggruppati in *container*, ciascuno dei quali corrisponde a una JVM[3] in esecuzione. I container a loro volta vengono eseguiti all'interno di una *piattaforma*. Una singola piattaforma può essere distribuita su host diversi, questa caratteristica unita alla capacità degli agenti di migrare tra i container rende possibile distribuire il carico di lavoro di un sistema multi-agente su diversi nodi di una rete.
La caratteristica che contraddistingue gli agenti dai normali programmi software è la **socialità** (capacità di comunicare tra di loro), la **proattività** dei propri comportamenti e l'**autonomia decisionale** nelle proprie azioni.
I sistemi ad agenti sono realizzati intrinsecamente con una struttura *da pari a pari* (in inglese *peer to peer*), che recentemente ha iniziato ad affermarsi nei confronti della struttura tradizionale *client/server* non solo nel campo dei protocolli di rete, ma anche nelle architetture dei database [9].
In un sistema peer-to-peer l'organizzazione dei componenti è decentralizzata e tutte le comunicazioni avvengono tra gli appartenenti al sistema in modo diretto, mentre in una organizzazione client/server esiste un nodo centrale che

3 JVM – Java Virtual Machine: l'ambiente di runtime che è in grado di eseguire il bytecode Java, cioè il risultato della compilazione dei programmi scritti in Java. La quasi totalità dei dispositivi dotati di microprocessore non è in grado di eseguire nativamente il bytecode Java, perciò esiste una JVM diversa per ciascun dispositivo.

gestisce la comunicazione e viene utilizzato come intermediario (come avviene con il protocollo IRC).
La più importante conseguenza della scelta di dotare gli agenti di autonomia decisionale è che essi assumono in questo modo una forte identità: possono rifiutarsi di operare, considerano la comunicazione come una qualsiasi azione e non dipendono da essa per il proprio funzionamento, inoltre i messaggi inviati devono necessariamente possedere un contenuto semantico affinché gli agenti li comprendano.
La comunicazione tra gli agenti si svolge principalmente grazie a due protocolli:

- ACL - Agent Communication Language, che definisce la semantica dei messaggi ed è stato formalizzato da FIPA.
- MTP – Message Transport Protocol, che gestisce il trasferimento dei messaggi su una rete informatica e di cui esistono varie implementazioni.

Si è scelto di utilizzare il tipo di MTP che effettua la comunicazione attraverso il protocollo HTTP[4], questo considerando l'elevata permeabilità che in genere ogni rete informatica offre a tale tipo di traffico.
Gli agenti presenti in una piattaforma JADE vengono catalogati da un particolare agente, preposto unicamente a tale funzione: il *Directory Facilitator* (DF). Nel momento in cui un agente viene istanziato, esso viene presentato al DF, che lo inserisce nel proprio catalogo.

2.4.1.1 Note per l'installazione di Jade

Requisiti

- Java Runtime Environment version 1.2
- Decompressore ZIP (UnZip della suite InfoZip, PKUNZIP di PKWARE, ...)

Queste note sono basate sulla release 3.1 di Jade; versioni successive potranno avere differenti requisiti.

Procedimento

1. Fare il download del pacchetto contenente JADE dal sito jade.cselt.it, dove è possibile trovare sia l'ultimo release stabile che lo snapshot del tree di sviluppo. `JADE-all-<versione>.zip` contiene tutte le componenti di JADE: i binari eseguibili, la documentazione, alcuni esempi di utilizzo e il codice sorgente.
2. Decomprimere l'archivio ZIP appena scaricato. Si ottengono altri quattro archivi ZIP contenenti le componenti sopra menzionate
   `JADE-bin-<versione>.zip`

4 HTTP – HyperText Transfer Protocol: protocollo comunemente usato per il Web.

JADE-doc-<versione>.zip
JADE-examples-<versione>.zip
JADE-src-<versione>.zip)

3. Per il normale utilizzo di JADE i sorgenti non dovrebbero essere necessari, quindi ci limiteremo a decomprimere i primi tre archivi, tutti nella stessa directory perché le sovrapposizioni nei nomi dei file estratti corrispondono in realtà agli stessi file ripetuti più volte nei diversi archivi (quindi quando viene richiesto se si intende sovrascrivere i file già esistenti è indifferente premere “sì” o “no”).
4. Per il funzionamento di JADE è richiesto il Java Runtime Environment 1.2.
5. È necessario aggiungere al `CLASSPATH` (variabile d'ambiente) alcuni jar perché JADE funzioni.
   Ad esempio, con Windows:
   ```
   set CLASSPATH=%CLASSPATH%;.;c:\jade\lib\jade.jar;
   c:\jade\lib\jadeTools.jar; c:\jade\lib\Base64.jar;
   c:\jade\lib\iiop.jar
   ```
   e con Linux, supponendo di avere estratto JADE in `/opt/jade`:
   ```
   export CLASSPATH=”$CLASSPATH:./:/opt/jade/lib/jade.jar:/opt/jade
   /lib/jadeTools.jar:/opt/jade/lib/Base64.jar:/opt/jade/lib/iiop.j
   ar”
   ```
6. Supposto che l'eseguibile java sia nel path di sistema, JADE si avvia con
   `jade.Boot -gui`.

Librerie aggiuntive

Nel caso del mio progetto viene utilizzato anche l'MTP via HTTP, che va installato a parte. Per la versione 3.1 di JADE il file che lo contiene si chiama `HttpMTPUABAddOn-3.1.zip`.

Per la compilazione è necessario ANT, facente parte del progetto Jakarta.

Nella directory dove è stato decompresso l'archivio zip basterà eseguire

```
ant lib
```

perchè il jar richiesto venga compilato.

```
cp lib/http.jar /opt/jade/lib/
```

ora è necessario modificare il classpath aggiungendo anche tale jar.

Note

Per esigenze di praticità può essere comodo mantenere sempre online una piattaforma per agenti e un *Main Container* in modo da caricare e terminare gli agenti che si sottopongono a debug, senza dovere riavviare Jade ogni volta che si compilano gli agenti. Per fare questo è necessario che gli agenti si trovino in un proprio container, diverso dal principale. Ciò si può ottenere con l'opzione

-container.
ad esempio

```
java jade.Boot -container me0:examples.mine.MyAgent
```

Terminato l'utilizzo dell'agente è sempre possibile nell'interfaccia grafica di Jade fare un *kill* del container interessato, rilasciando le classi dell'agente che può così essere modificato e compilato nuovamente.
Per informazioni più dettagliate, è possibile fare rifermento alla documentazione di JADE (JADE-doc-<versione>.zip), che include anche informazioni su come integrare JADE in un ambiente di sviluppo integrato (IDE).

2.4.2 Part Of Speech (POS) tagger

Dall'inglese è possibile tradurre POS in “parti del discorso”, intendendo con ciò la catalogazione grammaticale delle parole di un testo. Il tagger POS è un programma che, una volta sottoposto ad una fase di apprendimento, è in grado di posporre a ogni parola di un insieme di frasi un tag (o *segnaposto*) che ne indica il tipo (nome, verbo, aggettivo...).
Per esempio, la frase

```
For information on a feature of the J2SE platform, click on
the component in the diagram below.
```

Viene tradotta in

```
For_IN information_NN on_IN a_DT feature_NN of_IN the_DT
J2SE_NNP platform,_NN click_NN on_IN the_DT component_NN in_IN
the_DT diagram_NN below._.
```

Ovviamente il processo non è esente da errori, nel caso dell'esempio “`click_NN`” è sbagliato, in quanto “click” in questo contesto è un verbo.
L'apprendimento del tagger è una attività completamente scollegata dal suo utilizzo, poiché l'algoritmo impiegato non è genetico.
Normalmente l'apprendimento di un tagger avviene tramite una operazione di tagging manuale su di un testo di riferimento. Interpretando le istruzioni fornite, il programma di tagging è in grado di ripetere l'operazione su testi diversi da quello usato per il *training* mantenendo una discreta precisione.
Il tagger POS selezionato in via definitiva all'interno del progetto dell'agente hunter è stato MontyTagger [10], di Hugo Liu, studente al MIT.

3 Descrizione degli algorimi

3.1 Dall'Ontologia ad un Common Thesaurus

> **ontologia**, s.f. (filos.) parte della metafisica che studia il concetto e la struttura dell'essere in genere, e non le peculiari caratteristiche dei singoli esseri particolari ¶ Comp. Di *onto-* e *-logia.*
>
> **ònto-**, primo membro di parole composte, dal gr. ō'n óntos, p.pr. di êinai 'essere'; in filosofia vale 'essere, esistenza' (*ontologia*), in biologia 'organismo, essere vivente' (*ontogenesi*).
>
> **-logìa**, secondo membro di parole composte di origine greca o di formazione moderna, dal gr. *-logia*, deriv. di *-lógos* (V. *-logo*); significa a) studio, trattazione (*psicologia*, *teologia*); b) modo di parlare, discorso, espressione (*analogia*, *brachilogia*).
>
> [Dizionario Garzanti della lingua italiana, ed. XXII, 1983]

Nel campo dell'Intelligenza Artificiale, la parola *ontologia* assume il significato di **esplicita specificazione di una concettualizzazione** [11] [12], che può essere rappresentata con un formalismo dichiarativo attraverso un vocabolario contenente oggetti e le possibili relazioni tra di essi. La definizione formale di ontologia prevede molti altri tipi di costituenti nella struttura di tale vocabolario, ma esistono differenti interpretazioni al riguardo, e la più semplice prevede le sole componenti di *concetti* e *relazioni*.

La necessità di approssimare il concetto di ontologia con un oggetto rappresentabile attraverso formalismi informatici ha portato all'elaborazione del *Common Thesaurus* [1]: un insieme di relazioni tra schemi utilizzate come riferimento nell'identificazione delle classi individuate in una sorgente di dati. Il CT verrà impiegato come riferimento nel momento in cui si costruirà una rappresentazione globale di tali classi, in un'ottica di Integrazione dell'Informazione.

A differenza di come viene utilizzato normalmente, per introdurre una classificazione che arricchisce la definizione di classi altrimenti slegate, un Common Thesaurus può essere usato per cercare nuove sorgenti, che contengono dati coerenti con i concetti espressi al suo interno.

Esistono tre principali categorie di sorgenti di dati soggette all'integrazione:

- sorgenti contenenti dati strutturati, come un database ad accesso pubblico
- sorgenti contenenti dati non strutturati, come una pagina web contenente del testo
- sorgenti contenenti dati semistrutturati, come un documento XML

L'agente sviluppato durante lo svolgimento di questa tesi si occupa delle

sorgenti del secondo tipo. Benchè sia possibile che caricando documenti sconosciuti vengano trovate delle tabelle (strutturate), l'agente si dedica unicamente ad una **analisi semantica globale** del testo.
Per aggiungere questo scopo esiste più di un metodo messo a punto nel campo dell'analisi del linguaggio naturale, in questo caso si sfrutterà la creazione di catene lessicali ([13], [14]).
Nel tipo di analisi che si rende necessaria per l'identificazione corretta di una sorgente di dati si presentano tre principali problemi:

1. **Eliminazione delle ambiguità nei significati dei lemmi. Identificazione del senso di ciascun lemma.**
2. **Creazione di gruppi di significati in relazione tra di loro che riassumano con un certo grado di approssimazione il senso generale di un testo.**
3. **Determinazione quantitativa dell'attinenza di un testo ad un particolare ambito semantico, identificato da insiemi di parole chiave.**

L'algoritmo impiegato per il primo caso è una diversa implementazione di quanto descritto in [15], si tratta di un algoritmo definito di ***disambiguazione lessicale***.
Per il secondo caso è stato necessario sviluppare un algoritmo di ***concatenamento lessicale*** (*lexical chaining*).
Infine, per la soluzione del terzo problema si è utilizzata una estensione degli algoritmi descritti in precedenza. Tale algoritmo viene presentato con il nome di ***calcolo dell'affinità semantica***.

3.2 Disambiguazione lessicale

3.2.1 Premessa

Dato un insieme di lemmi ottenuto da un particolare testo e l'insieme di tutti i synset contenuti in WordNet [16], lo scopo del procedimento di disambiguazione è quello di stabilire una applicazione biunivoca tra questi due insiemi. L'applicazione ha valore solo per il particolare testo in esame e il mapping ottenuto rappresenta per ciascun lemma il senso (synset) che un lettore dotato di giudizio autonomo assegnerebbe.
Sarebbe teoricamente possibile eseguire una disambiguazione di tutti i lemmi presenti in un dato testo, ma considerato che ciò che occorre è una analisi semantica globale, limitando il procedimento ai soli nomi si ottiene una significativa riduzione della complessità del problema senza inficiare

l'affidabilità del procedimento nel suo complesso.
Questo algoritmo può essere esteso in vari altri modi, per esempio considerando la vicinanza reciproca dei lemmi nel testo sarebbe in grado di dare un peso maggiore al significato dei lemmi adiacenti durante la determinazione del significato di ciascun lemma.
Non essendo significativo controllare l'esistenza di relazioni dirette prendendo due a due i lemmi dell'insieme fornito, si cerca una corrispondenza tra tutti i synset collegati ad una parola e le relazioni che questi hanno nei confronti di ogni altro synset. Successivamente tra le relazioni individuate si considerano quelle che essi hanno nei confronti dei soli synset collegati alle altre parole in qualità di possibili significati. Allo scopo di allargare il campo di ricerca mantenendo però una buona efficienza dell'algoritmo si esegue una ricerca di tipo ricorsivo. Per gli scopi del software che si è realizzato, la verifica dell'esistenza di una relazione entro due livelli di correlazione tra i nomi è risultata adatta, ma il procedimento è estendibile facilmente a qualsiasi numero di livelli (i metodi presenti nel codice sorgente prevedono un parametro intero a questo scopo).
Viene ora descritto nel dettaglio l'algoritmo utilizzato per la disambiguazione lessicale.

3.2.2 Procedimento

L'algoritmo realizzato fa uso di una struttura dati del tipo *grafo* la cui implementazione è realizzata attraverso una matrice sparsa.

> Un grafo elementare è costituito dalla coppia G = (V, E), dove V ed E sono insiemi finiti, e ogni elemento di E è un sottoinsieme di due elementi di V (cioè una coppia non ordinata di elementi disgiunti di V). Gli elementi di V si chiamano *nodi*, mentre gli elementi di E si chiamano *archi*. Se $e \in E$ allora $e = \{a, b\}$ per qualche $a, b \in V$. In questo caso, possiamo più semplicemente indicare *e* come *e*=*ab*=*ba*. Si dice che l'arco *e* *collega* i due nodi *a* e *b*, che *e* è *incidente* sui nodi *a* e *b*, che *a* e *b* sono gli estremi dell'arco *e*, che *a* e *b* sono tra loro *adiacenti*.[...]
>
> Si definisce *percorso dal nodo a al nodo b* di un grafo la sequenza di archi (a_0a_1, a_1a_2, ..., $a_{k-1}a_k$), dove a_0=*a* e a_k=*b*, ovvero una sequenza di archi (e_1, e_2, ..., e_k) nella quale l'arco e_i collega un nodo al nodo a_i, l'arco successivo e_{i+1} collega il nodo a_i ad un altro nodo, formando così una catena di nodi collegati tra loro che va da *a* a *b*. La lunghezza *k* di un percorso è data dal numero di archi che lo compongono. Si può anche deifnire un percorso come la sequenza dei nodi che gli archi collegano tra loro; il percorso

(a_0a_1, a_1a_2, ..., $a_{k-1}a_k$) può essere indicato più brevemente da $a_0a_1a_2...a_{k-1}a_k$, dove ogni coppia (a_{i-1},a_i) è un arco del grafo. [...]

Si definisce *cammino* ("*path*") un percorso i cu inodi sono tutti distinti tra loro.[17]

Un grafo è rappresentabile attraverso una matrice *n* per *n*, se agli indici delle righe e delle colonne si fa corrispondere univocamente i nodi del grafo. Gli elementi della matrice corrispondono agli archi. Un approccio di questo tipo è più flessibile rispetto alla definizione matematica di grafo, poiché gli elementi della matrice possono non essere solamente numeri qualsiasi. È possibile rappresentare non solo una informazione booleana come la presenza o l'assenza di un arco, ma si può costruire per esempio un grafo in cui gli archi presentano differenti "lunghezze" o dispongono di una o più proprietà.
Una matrice sparsa è concepita come un vettore i cui elementi sono altri vettori. Se viene utilizzata per rappresentare un grafo si può mantenere la sola informazione relativa agli archi presenti nel grafo, interpretando l'assenza di un elemento come l'assenza di un arco. Dal punto di vista informatico è evidente come una matrice sparsa è preferibile nel caso si trattino grafi in cui sono presenti relativamente pochi archi, perché le risorse impiegate durante l'esecuzione dipendono direttamente dalla complessità del grafo. Trascurando lo *overhead* necessario a codificare l'assenza di valori in una cella della matrice, solo nel caso in cui ogni nodo fosse connesso a tutti gli altri si avrebbe una occupazione di risorse equivalente a quella di una matrice tradizionale. Dato che tale condizione sarebbe quantomeno anomala relativamente al caso che prenderemo in esame, la scelta di una matrice sparsa è decisamente preferibile.

3.2.2.1 Elaborazione preliminare

1. Dato un testo eventualmente contenente elementi di markup, come codice HTML esso viene "ripulito" nella formattazione e da tutti i tag estranei al solo corpo del testo. Le frasi vengono isolate basandosi sulla punteggiatura. Al termine dell'elaborazione si ottiene un testo contenente le stesse parti discorsive, suddivise in una frase per riga.
2. Il testo ottenuto viene passato aa un tagger POS che identifica grammaticalmente ogni parola riportandone il tipo in un tag suffisso. Ai nomi, ad esempio, viene suffisso "`/NN`"
3. Dopo avere identificato i soli nomi nel testo, essi vengono estratti e passati attraverso un processo di rimozione dei suffissi (come la "s" del plurale inglese). La classe che è deputata a questo compito è distribuita dalla

Apache Software Foundation, ed è basata sul concetto espresso in un articolo del 1980 di Porter [18]. In seguito ad alcuni test preliminari è emerso che la rimozione del suffisso ha cattivi risultati quando si opera con dei nomi, poiché l'algoritmo realizzato da Porter riconduce le parole inglesi alla sola radice, che non è sempre una parola di senso compiuto. Si è preparato quindi un algoritmo più semplice in grado di eliminare il plurale nei nomi seguendo le regole grammaticali della lingua inglese. L'applicazione di tale algoritmo ha dato risultati soddisfacenti.

```
public static String simpleNounStem(String noun_suffix) {
      if (noun_suffix.endsWith("ss")) // miss, glass, success, ...
            return noun_suffix;
      if (noun_suffix.endsWith("ies")) // parties, frequencies, ...
            return noun_suffix.replaceAll("ies\\z","y");
      if (noun_suffix.endsWith("sses")) // misses, glasses, ...
            return noun_suffix.replaceAll("sses\\z","ss");
      if (noun_suffix.endsWith("s")) // other regular words
            return noun_suffix.replaceAll("s\\z","");
      return noun_suffix; // return the same word if singular
}
```

La struttura ottenuta è un insieme di nomi ricondotti alla loro forma base, ciascuno presente una o più volte nel testo di partenza.

4. Per ogni nome distinto reperito in questo modo viene fatta una query al database di WordNet, che ritorna i synset dei possibili significati del lemma, per esempio il nome “mouse” conduce[5] a due risultati:

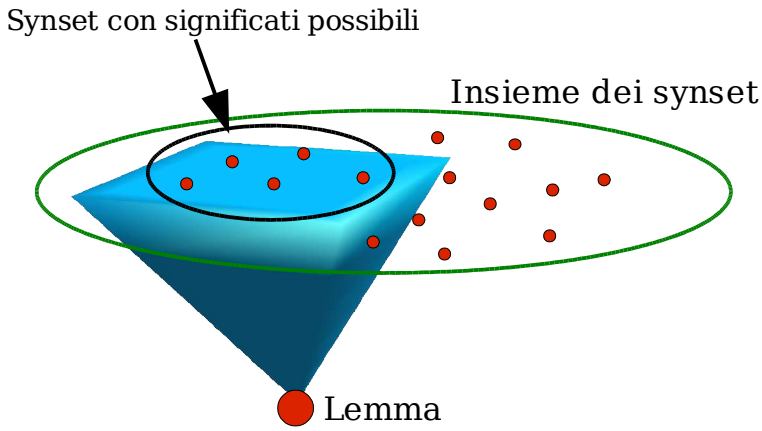
   10785 any of numerous small rodents typically resembling diminutive rats having pointed snouts and small ears on elongated bodies with slender usually hairless tails

   18391 a hand-operated data input device that moves the cursor on a computer screen

3.2.2.2 Disambiguazione dei lemmi

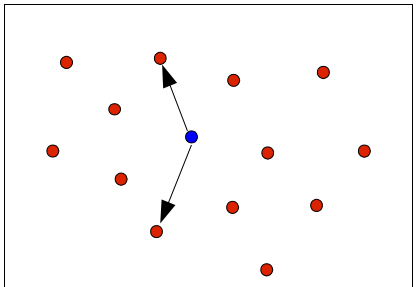
I synset risultanti al termine della fase precedente saranno l'oggetto della disambiguazione. Sarà necessario scoprire a quale significato il lemma fa realmente riferimento, quindi selezionare uno tra i synset corrispondenti.

5 mysql> SELECT SYN.WN_SYNSET_ID, SYN.GLOSS FROM WN_LEMMA LEM, WN_LEMMA_SYNSET LEMSYN, WN_SYNSET SYN WHERE LEM.WN_LEMMA_ID = LEMSYN.WN_LEMMA_ID and LEMSYN.WN_SYNSET_ID = SYN.WN_SYNSET_ID and SYN.SYNTACTIC_CATEGORY = 1 and LEM.LEMMA="mouse";

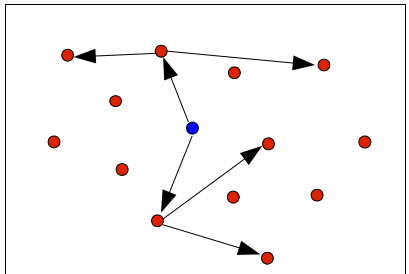


Disegno 1, un lemma e i possibili synset che rappresenta

1. Per ogni insieme di synset corrispondenti a un solo lemma viene fatta una nuova query al database di WordNet dove vengono ritornati tutti i synset collegati da qualche tipo di relazione con qualsiasi elemento dell'insieme. Per ogni synset ritornato si effettua un'altra ricerca dei synset ad esso collegati. Tale procedimento può essere esteso ad un numero arbitrario di livelli, ma l'algoritmo risulta efficace già con due ricorsioni (corrispondenti a cinque livelli di relazioni: nonno, padre, persona, figlio, nipotino).



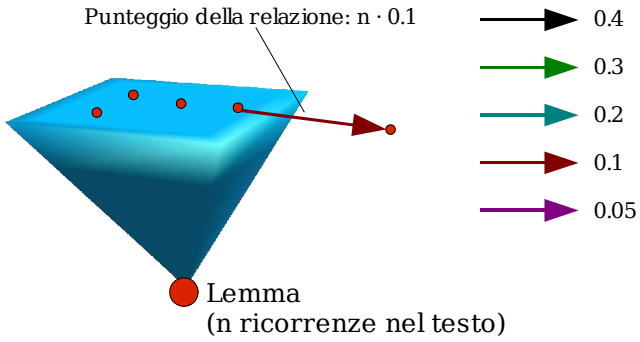
Disegno 2, primo livello di relazioni tra i synset



Disegno 3, primo e secondo livello di relazioni tra i synset

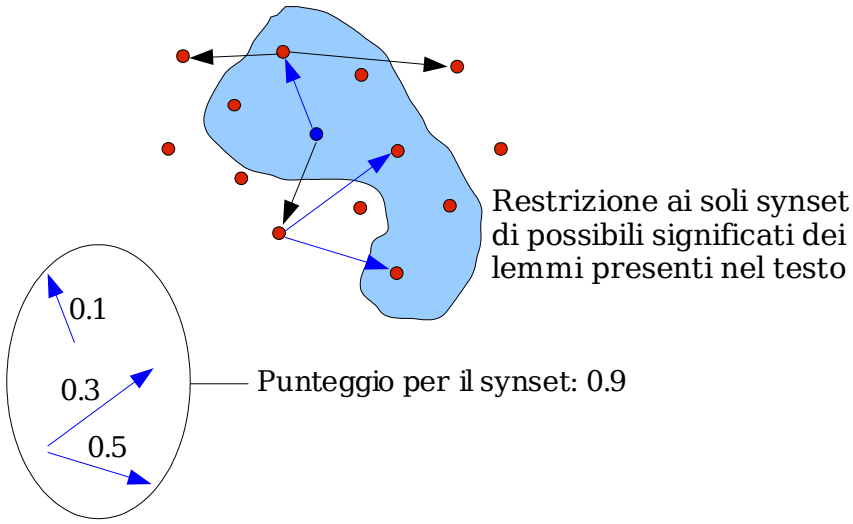
2. A ogni tipo di relazione presente in WordNet (si veda la tabella nel capitolo 2.2.2, pag. 9) è stato assegnato un punteggio arbitrario, che costituisce il principale parametro da variare per ottenere una buona affidabilità da questo procedimento euristico. Viene creata una matrice sparsa che rappresenta le relazioni tra i synset. La struttura che si ottiene è quella di un grafo, dove ogni synset è un nodo e le relazioni sono archi aventi due proprietà: il tipo (di relazione) e il punteggio "specifico" di tale relazione moltiplicato per il numero di ricorrenze nel testo del lemma corrispondente al synset a cui l'arco fa capo.

Se tra due synset esistono due tipi diversi di relazione, viene mantenuta la relazione più significativa, cioè quella il cui tipo conduce a un punteggio maggiore indipendentemente dal numero di ricorrenze del lemma.



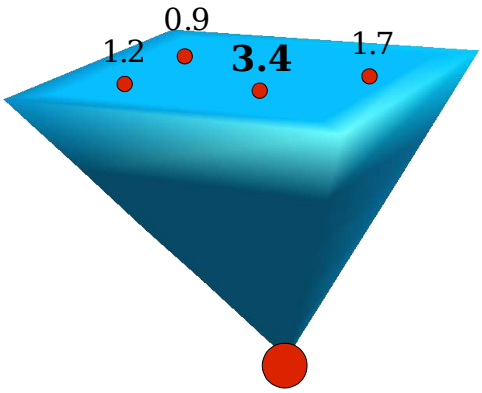
Disegno 2, relazione tra un synset di possibile significato per un lemma e un altro synset estraneo al lemma

3. Esiste quindi un modo per calcolare il punteggio attribuibile ad un singolo synset. Il punteggio è determinato attraverso una restrizione della matrice di relazioni tra synset ai soli synset rappresentanti i “sensi” dei lemmi presenti nel testo.



Disegno 3, calcolo del punteggio di un synset al fine di disambiguare il lemma a cui fa capo

4. Per ogni lemma si considerano in sequenza i synset rappresentanti i possibili significati corrispondenti. Il punteggio iniziale di ciascun synset è zero. Per ognuno si seguono gli archi che ne dipartono, se i nodi di arrivo di tali archi sono contenuti nell'insieme di tutti i possibili synset di tutti i lemmi del testo,



Disegno 4, selezione del synset più probabile in base al punteggio

si somma per ciascuno il punteggio dell'arco ad esso collegato al punteggio totale del synset. Questo procedimento è estendibile ricorsivamente proseguendo la navigazione del grafo attraverso i nodi connessi da un cammino lungo **n**. L'operazione di *disambiguazione* di un lemma viene quindi ricondotta a trovare il synset corrispondente dal punteggio più alto.

3.3 Costruzione delle catene lessicali

Lo scopo del processo di creazione delle catene lessicali è quello, dato un insieme di synset, di estrarre un certo numero di *catene lessicali* [13] [14]. Una catena lessicale è un insieme di synset che secondo WordNet [16] si trovano in relazione (si veda il capitolo 2.2.2 per maggiori informazioni riguardo alle relazioni in WordNet).

Questa particolare implementazione di un algoritmo per l'estrazione delle catene lessicali riutilizza la struttura a grafo creata durante l'algoritmo di disambiguazione.

A questo punto del procedimento si dispone di un insieme di lemmi a ciascuno dei quali corrisponde un solo synset che ne identifica presumibilmente il significato.

Le catene lessicali che si vogliono ottenere devono essere corredate da un punteggio di auto-coesione che descriva sinteticamente l'intensità delle relazioni tra i lemmi che le compongono.

Tramite tale punteggio è possibile determinare quali catene lessicali sono più rilevanti al fine di determinare l'oggetto di un testo in maniera generica.

L'algoritmo prevede i seguenti passi:

1. Si crea un insieme dei synset unici (disambiguati) corrispondenti ai lemmi presenti nel testo.
2. Si mantiene un insieme delle catene lessicali individuate, inizialmente vuoto.

   Per ogni lemma vengono ripetuti i punti 3-5:
3. Si verifica se nell'insieme delle catene lessicali individuate fino a questo punto ne esiste una in cui tutte le coppie lemma/synset secondo il grafo creato ai punti precedenti risultano in relazione con il lemma in esame. Tutto

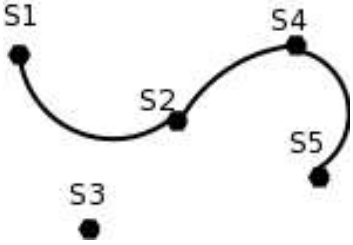
ciò operando in una opportuna restrizione del grafo corrispondente ai soli synset disambiguati (si veda il punto 1, in proposito).

4. Se non esistono catene lessicali rispondenti a questo requisito ne viene creata una nuova contenente il lemma/synset in esame, il punteggio di auto-coesione per questa catena è posto a zero.
5. Se esistono catene lessicali che rispondono ai requisiti, la coppia lemma/synset in esame vi viene aggiunta. Si somma al punteggio della/e catena/e il punteggio delle singole relazioni che legano il lemma/synset agli altri lemmi presenti nella catena lessicale.

Al termine dell'iterazione:

6. Si ha un certo numero di catene lessicali, ciascuna avente un proprio punteggio di auto-coesione.

Nota: Le relazioni disponibili in WordNet hanno una proprietà denominata *REFLEX*, traducibile in *simmetria*. Se i lemmi esaminati sono solo nomi, tutte le relazioni che è possibile trovare sono simmetriche. Nell'insieme di catene lessicali individuate non esisteranno lemmi condivisi tra più catene se tutte le relazioni estratte da WordNet sono simmetriche. Vedremo però che questa soluzione non è completamente libera da problemi e si renderà necessario introdurre l'esistenza di catene lessicali con termini condivisi.
Se in futuro si volesse estendere il procedimento anche a lemmi di altre parti del discorso (POS), si potrebbero ottenere lemmi comuni a più catene lessicali se le relazioni che li coinvolgono fossero in parte non simmetriche, come nel caso dei tipi *Entailment*, *Cause*, *Also see*, *Verb Group*, *Participle of verb* e *Pertainym* (che coinvolgono verbi e aggettivi).
È necessario un ulteriore approfondimento per quanto riguarda il punto 3. Intendiamo che due synset sono in relazione se essi hanno un cammino lungo al più (N-1)/2 che li congiunge nel grafo, dove N è il numero di livelli desiderato nell'analisi delle relazioni lessicali, normalmente 5. La costruzione delle catene lessicali nel modo che è stato esposto si presta a produrre risultati diversi a seconda dell'ordine in cui vengono elaborati i synset.
Avendo a disposizione una struttura come quella rappresentata nel disegno 5, immaginando di procedere iterativamente dal nodo S1 al nodo S5 nella costruzione delle catene lessicali otterremmo:



Disegno 5, esempio di grafo risultante

1. catena {S1, S2, S4}
2. catena {S3}
3. catena {S5}

a questo punto è evidente che se invece avessimo proceduto nell'ordine contrario, cioè dal nodo S5 al nodo S1, avremmo ottenuto:

1. catena {S5, S4,S 2}
2. catena {S3}
3. catena {S1}

Per risolvere questa inconsistenza logica nell'algoritmo sono state elaborate due diverse soluzioni:

- Se il concetto di *relazione* tra synset venisse esteso a cammini di qualsivoglia lunghezza si otterrebbero le due catene lessicali {S1,S2,S4,S5}, {S3}
- Se si ammettesse la non univocità della relazione synset-catena lessicale, si potrebbe, qualora si creasse una nuova catena lessicale, aggiungere tutti i synset che risultano “in relazione” al synset esaminato correntemente (mantenendo il significato originale di *relazione*). In questo modo si otterrebbero le catene lessicali {S1, S2, S4}, {S3}, {S2, S4, S5}. Attraverso questa soluzione purtroppo vengono individuate catene lessicali diverse che possono avere alcuni termini in comune.

Il risultato “ideale” che si potrebbe ottenere sviluppando una soluzione ottimale al problema senza modificarne alcun termine (mantenendone la complessità non-polinomiale) è un clustering che assegna a ciascun synset la catena lessicale in cui esso aumenta maggiormente il punteggio di coesione.
Il primo approccio espone l'algoritmo al problema di creare catene lessicali eccessivamente lunghe e generiche. Inoltre, adottando questa soluzione, se si aumentasse il livello di ricorsione nella ricerca di relazioni tra i synset si otterrebbe solo un aumento della genericità delle catene lessicali e non una maggiora accuratezza. Adottando invece la seconda soluzione proposta, un aumento nel livello di ricorsione introduce necessariamente meno genericità nelle catene lessicali che l'algoritmo individua, perchè non ci si espone alla creazione accidentale di lunghi cammini tra synset semanticamente distanti. Quindi l'effettiva implementazione software prevede l'eventuale costruzione di catene lessicali diverse aventi uno o più elementi in comune, anche se tale risultato è relativamente poco comune e quindi poco incidente rispetto a una “vera” soluzione di clustering ottimizzato, in cui i termini del problema sono

stati mantenuti invariati.

3.4 Calcolo dell'affinità semantica

Nell'ambito delle ricerche semantiche nel Web sono state individuate due principali modalità di utilizzo di un agente hunter:

- Ricerca semantica di parole chiave
- Ricerca semantica di una ontologia (o meglio della sua rappresentazione in forma di *Common Thesaurus*, secondo le specifiche del progetto MOMIS)

In riferimento al secondo caso, è opportuna una breve descrizione della rappresentazione di un *Common Thesaurus* (CT).

Un CT è descritto attraverso un documento XML [19] che rappresenta un particolare dominio semantico. La struttura dati astratta (ADS[6]) che rispecchia il modello di un documento XML è l'albero.

Una misura precisa dell'affinità semantica di un documento con l'oggetto della ricerca consente di discriminare tra i risultati ottenibili attraverso la semplice ricerca di una corrispondenza di parole chiave, fornendo un risultato dove i documenti non significativi sono stati rimossi e i rimanenti sono ordinati in base alla loro pertinenza. L'affinità semantica è un tipo di misurazione (relativa) della pertinenza di un testo nei confronti di un particolare dominio di significati. Le formule matematiche utilizzate nel codice sorgente del progetto per il calcolo di tale grandezza sono fortemente parametrizzate, dato che i termini del problema sono decisamente confusi e la soluzione individuata è necessariamente una semplificazione del reale problema.

Inoltre, l'affinità semantica non ha un senso assoluto, ma il punteggio calcolato è *locale* a un determinato testo e a un gruppo di parole chiave, o a un Common Thesaurus. Quindi l'affinità semantica può solo fornire una misura, relativamente a più testi, di quale tra essi sia probabilmente il più significativo rispetto all'ontologia desiderata.

3.4.1 Ricerca di parole chiave

La ricerca di parole chiave è l'azione basilare che l'agente hunter compie e consiste nella ricerca di pagine web legate all'ambito semantico di alcune parole chiave fornite dal *client* di questo servizio.

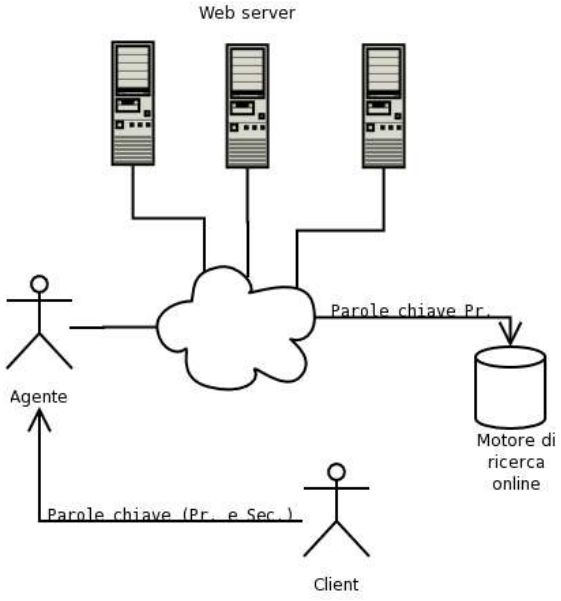
A causa della enorme estensione del Web non è stato possibile realizzare un sistema di ricerca completamente semantico, ma almeno nelle prime fasi di ciascuna ricerca è necessario l'appoggio ad un motore di ricerca online, che è in grado unicamente di fare confronti tra pagine web archiviate e parole

6 ADS – Abstract Data Structure

chiave.
La ricerca è quindi composta di due fasi principali: una puramente “letterale”, che ottiene una lista di pagine contenenti una o più parole chiave *principali*; mentre la fase successiva esegue un raffinamento dei risultati ottenuti dal motore di ricerca.
Questa seconda fase necessita di un altro gruppo di parole chiave, che vengono classificate come *secondarie*.

3.4.1.1 Procedimento

L'insieme complessivo delle parole che chiave, sia primarie che secondarie, viene sottoposto a disambiguazione lessicale. Le catene lessicali così ottenute verranno utilizzate per il calcolo dell'affinità in seguito.
Le parole chiave principali vengono fornite ad un motore di ricerca online (nel caso specifico Google) con l'indicazione di ritornare un elenco di pagine contenenti tutte le parole chiave.

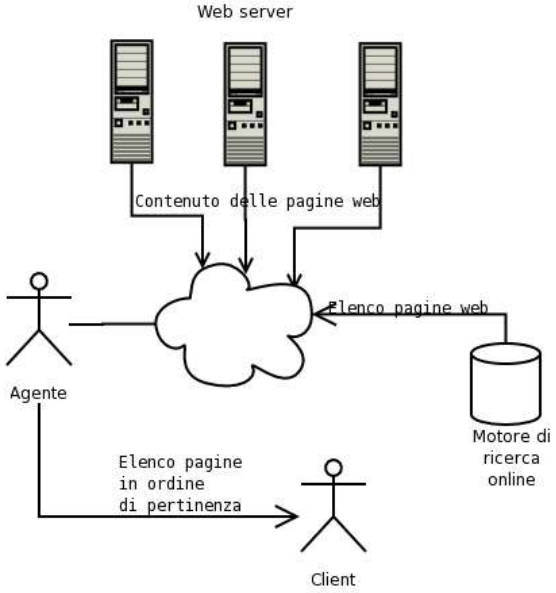


Disegno 6, Il client fornisce le parole chiave e l'agente contatta il motore di ricerca

Per ogni risultato che viene individuato dal motore di ricerca si carica il documento corrispondente e si procede ad una normale disambiguazione dei lemmi e alla costruzione di catene lessicali.
In seguito, viene calcolato il punteggio di coesione tra tali catene lessicali e i synset che derivano dalla disambiguazione delle parole chiave fornite dal client.
A seconda del punteggio ottenuto si possono presentare tre diversi casi:

- La pagina web è pertinente all'ambito delle parole chiave
- La pagine web non è pertinente
- La pagina web è pertinente, ma con un basso punteggio, tale da consigliare la ricerca anche nelle pagine immediatamente collegate ad essa.

Nel terzo caso viene eseguito un crawling di livello 1 dalla pagina in esame e le pagine direttamente collegate vengono analizzate nello stesso modo, fatto salvo che la loro analisi non dà luogo ad altri crawling, per ragioni di performance. Le pagine collegate che ottengono un punteggio di coesione maggiore della pagina di partenza vengono inserite nei risultati insieme a quelle ritornate dal motore di ricerca online, successivamente verranno presentate al client ad elaborazione conclusa.



Disegno 7, l'agente procede al caricamento delle pagine, alla loro analisi e ritorna al client la lista delle pagine pertinenti

Determinare per ciascuna pagina le azioni da intraprendere è una operazione equivalente a dividere l'intervallo dei punteggi di coesione possibili in tre parti. A seconda dell'intervallo in cui il punteggio di ciascuna pagina cade, viene determinato quale è la procedura da seguire.

Disegno 8, classificazione del grado di interesse di un documento in base al punteggio di coesione con le chiavi di ricerca

Nel caso di problemi legati al linguaggio naturale (NL), una delle soluzioni più efficaci è un approccio *fuzzy* come quello appena descritto.
Nello scegliere i punteggi che determinano l'appartenenza di ciascun caso ad una delle tre categorie citate si è reso necessario in pratica stabilire due formule, poiché il punteggio di coesione di una pagina molto lunga nei confronti di un certo insieme di synset risulta necessariamente più alto di quello di una pagina breve, a parità di pertinenza. Le formule individuate dipendono quindi dal numero di lemmi presenti nella pagina.

$$\underset{average}{Rel\ Score} = \frac{\sum_{t}^{Noun\ Rel\ Types} Scores_t \cdot Rel\ Number_t}{\sum_{t}^{Rel\ Types} Rel\ Number_t}$$

Formula 1

$$\underset{Random\ Relationships}{Score} = (\underset{average}{Rel\ Number}^{1} + \underset{average}{Rel\ Number}^{1})^{2} \cdot \underset{average}{Rel\ Score}^{2} \cdot \underset{of\ nouns}{Number}^{3} \cdot \underset{of\ keywords}{Number}^{4}$$

Formula 2

Tale valore indica il punteggio che si otterrebbe da semplice “rumore di fondo” semantico tra il testo analizzato e i synset delle parole chiave, cioè il punteggio che un testo di analoga lunghezza potrebbe ottenere a causa di corrispondenze casuali con le chiavi. Sostanzialmente ciò serve a eliminare gran parte dei possibili falsi positivi che verrebbero mostrati nel caso in cui non venisse trovata una vera corrispondenza: ordinare in base al punteggio di coesione i documenti individuati dal motore di ricerca non è sufficiente a comprendere se effettivamente ne esistono di attinenti all'ontologia di ricerca.
Nella formula due parametri derivano da calcoli statistici e uno dipende dalla pagina in esame.

1. Numero medio di relazioni che dipartono dai nomi in WordNet (media tra tutti i nomi)
2. Punteggio medio di una relazione. Si ottiene come mostrato nella formula 1.
3. Numero di lemmi di tipo *nome* presenti nel testo
4. Numero di parole chiave nei confronti delle quali verrà calcolata la coesione.

$$\underset{Intresting}{Score} = \underset{Random\ Relationships}{Score} \cdot \underset{Intresting}{Threshold}$$

Formula 3

$$\underset{Worth\ Crawling}{Score} = \underset{Random\ Relationships}{Score} \cdot \underset{Worth\ Crawling}{Threshold}$$

Formula 4

Come sempre nelle soluzioni utilizzanti metodiche *fuzzy*, vengono determinate

delle costanti arbitrarie che distinguono i vari casi in cui l'inferenza *fuzzy* è suddivisa.

Codice SQL utilizzato

```
1
     mysql> create table tmp(no INT);
     Query OK, 0 rows affected (0.42 sec)

     mysql> insert into tmp select COUNT(*) from
     WN_RELATIONSHIP, WN_SYNSET, WN_LEMMA, WN_LEMMA_SYNSET
     where WN_RELATIONSHIP.WN_SOURCE_SYNSET_ID =
     WN_SYNSET.WN_SYNSET_ID and
     WN_LEMMA_SYNSET.WN_SYNSET_ID = WN_SYNSET.WN_SYNSET_ID
     and WN_LEMMA_SYNSET.WN_LEMMA_ID = WN_LEMMA.WN_LEMMA_ID
     and WN_LEMMA.SYNTACTIC_CATEGORY = 1 GROUP BY
     WN_RELATIONSHIP.WN_SOURCE_SYNSET_ID;
     Query OK, 66071 rows affected (14.55 sec)
     Records: 66071  Duplicates: 0  Warnings: 0

     mysql> select avg(no) from tmp;
     +---------+
     | avg(no) |
     +---------+
     |  4.9501 |
     +---------+
     1 row in set (0.04 sec)
```

2

```
mysql> select WN_RELATIONSHIP_TYPE_ID, COUNT(*) from
WN_RELATIONSHIP group by WN_RELATIONSHIP_TYPE_ID;
+-------------------------+----------+
| WN_RELATIONSHIP_TYPE_ID | COUNT(*) |
+-------------------------+----------+
|                       1 |     7708 |
|                       2 |    78501 |
|                       3 |    78502 |
|                       4 |    11849 |
|                       5 |      709 |
|                       6 |     6885 |
|                       7 |    11849 |
|                       8 |      709 |
|                       9 |     6885 |
|                      10 |     1300 |
|                      11 |      427 |
|                      12 |      224 |
|                      14 |      523 |
|                      15 |    21901 |
+-------------------------+----------+
14 rows in set (1.76 sec)
```

3.4.2 Ricerca attraverso un Common Thesaurus

Per le ricerche nel Web, l'agente riceve una copia del Common Thesaurus specifico per la ricerca da effettuare. Per ottenere un certo numero di URL in cui cercare relazioni con il CT l'agente utilizza un normale motore di ricerca online (Google). Iniziare una ricerca senza questo tipo di informazioni comporterebbe la necessità di esplorare tutto il Web. Attraverso un processo descritto in dettaglio in seguito, dal CT vengono estratte una o più parole chiave *principali*, che vengono utilizzate per la ricerca con il motore di ricerca esterno. Le parole chiave principali vengono fornite al motore con l'indicazione di ritornare i soli URL delle pagine che le contengono tutte.

Dal Common Thesaurus vengono estratte anche alcune parole chiave secondarie, che verranno utilizzate nell'analisi dei risultati forniti dal motore di ricerca per scremare le pagine semanticamente non coerenti con l'oggetto della ricerca.

Iterando attraverso tutti i nodi di un albero è possibile ottenere un vettore di nodi terminali o “foglie” dell'albero, cioè nodi che non hanno “figli”. La fondamentale proprietà di un albero è l'esistenza di un solo cammino tra ciascun nodo e la radice dell'albero. Per ogni foglia può essere quindi individuato il cammino tra essa e la radice, creando così un vettore di cammini

che complessivamente “coprono” tutto l'albero, dalla radice a ciascuna foglia.
La struttura di un documento XML come un Common Thesaurus del progetto MOMIS ricalca quella di un albero. Nella fattispecie, ad ogni nodo corrisponde una *classe*, che può o non può possedere determinati *attributi,* a seconda del tipo di classe.
La specifica struttura che un documento XML può assumere è definita nel DTD (Document Type Definition), che è proprio di una categoria di documenti XML e descrive la logica con cui possono essere aggregate le classi descritte nel documento.
La struttura di un Common Thesaurus necessita di attributi logicamente complessi, che non possono rientrare nella definizione normale di attributi in XML, perché sono costituiti da valori composti di altri “sotto-attributi”. Il problema è stato aggirato utilizzando delle classi che modellano tali attributi, una soluzione perfettamente coerente con la sintassi XML, che tende a creare un po' di confusione tra attributi XML propri e classi-attributo così definite, specifiche del Common Thesaurus.
La ricerca di sorgenti dati nel Web che corrispondano semanticamente almeno con una parte del Common Thesaurus deve necessariamente passare per l'estrazione di alcune parole chiave, da usare in un motore di ricerca online per ottenere un insieme di pagine rilevanti, tali pagine verranno analizzate semanticamente per stabilirne l'attinenza con l'ontologia della ricerca.
Si è scelto di utilizzare il supporto per la ricerca di parole chiave descritto nella sezione precedente, considerando come parole chiave principali i nomi delle classi, mentre come parole chiave secondarie i nomi degli attributo.
La totalità dei nomi di classe e di attributo viene sottoposta a disambiguazione lessicale, come descritto nel capitolo precedente, servirà per stabilire l'attinenza con le pagine ritornate dal motore di ricerca.

3.4.2.1 Procedimento

Come detto in precedenza, dalla struttura ad albero del documento XML rappresentante il Common Thesaurus è possibile ottenere un vettore di cammini, ciascuno avente per estremi la radice dell'albero e uno dei nodi foglia.
Per ogni cammino vengono estratte le parole chiave principali proprie dei nodi che attraversa, cioè il contenuto degli attributi *name* dei soli nodi che non descrivono attributi nella logica del Common Thesaurus. In questo modo si ottiene un elenco di gruppi di parole chiave, ciascuno legato ad un particolare percorso tra la radice del documento XML ed una delle foglie.

I gruppi di parole chiave principali così ottenuti vengono impiegati in ricerche analoghe a quelle di parole chiave, come descritte nel paragrafo 3.4.1. Per il calcolo dell'affinità, il gruppo di parole chiave secondarie utilizzate comprende tutte le stringhe contenute negli attributi *name* del Common Thesaurus, sia di classi in senso proprio che di classi-attributo. L'affinità viene perciò calcolata nei confronti dell'intero Common Thesaurus, non delle sole parole chiave presenti nel cammino che di volta in volta viene utilizzato per le ricerche attraverso il motore di ricerca online. Il calcolo viene effettuato in questo modo poiché in linea di principio si suppone di realizzare una ricerca che come risultati fornisca pagine web attinenti all'ontologia complessivamente rappresentata nel Common Thesaurus.
Questo tipo di attività rappresenta il più lungo e articolato comportamento dell'agente hunter che è stato realizzato, a causa della grande quantità di analisi che richiede esso impiega molto tempo per essere portato a termine.

4 Deployment - Utilizzo dell'agente hunter

Il sistema su cui è stato eseguito lo sviluppo e il test del software, è un host Linux con kernel serie 2.6, tutto il software utilizzato è disponibile anche in forma di porting per Windows di Microsoft.
Inoltre sono stati utilizzati:

- MySQL Ver 12.21 Distrib 4.0.15a, for slackware-linux (i486)
- Apache/2.0.49 (Unix)

4.1 Installazione

Per creare una installazione funzionante dell'agente hunter è necessario disporre del tool Apache Ant e di una installazione di Jade.
Dopo avere eventualmente decompresso i sorgenti del programma in una propria directory, il file `build.xml` può essere utilizzato con Ant per eseguire uno dei compiti in esso definiti.
Prima di eseguire alcunché, è necessario modificare `build.xml` perché rifletta le peculiari caratteristiche del nostro sistema. Le due proprietà marcate da commenti riguardano la posizione nel file system dell'eseguibile applet-viewer e dell'installazione di Jade.
Altre opzioni di configurazione sono disponibili nel file sorgente `Configuration.java`, quali il path in cui il server web mantiene le pagine pubblicate, e parametri per l'accesso al database WordNet (che può trovarsi su un diverso host, all'occorrenza).
Una volta apportate le modifiche necessarie, se vogliamo compilare tutti i sorgenti del progetto possiamo usare il comando

```
$ ant all
```

In questo modo l'agente verrà compilato e compresso in unico file jar, mentre l'applet di controllo verrà anch'esso compilato e compresso in un altro archivio jar.
Le librerie necessarie per l'esecuzione dell'agente sono incluse nella stessa distribuzione dei sorgenti, per semplicità.
All'archivio jar contenente l'applet viene apposta una forma digitale al momento della creazione, tale firma viene generata casualmente se non ne viene trovata una nel file `keystore`. Le firme digitali generate casualmente non sono riconosciute da alcuna autorità garante, per questo al momento di caricare l'applet in un browser viene mostrato un messaggio di avvertimento circa la non affidabilità del codice che si sta per eseguire.

A questo punto l'agente è pronto per l'esecuzione.

4.2 Esecuzione

Perché si possa eseguire l'agente è necessaria la presenza di altri servizi:

- Il web server
- MySQL
- Una piattaforma Jade
- Un SocketProxyAgent all'interno di tale piattaforma

Per automatizzare tutto ciò è stato scritto un breve script per la shell Bash che pone in esecuzione tutti questi servizi (deve essere eseguito dal superutente).

```
# ./system_run
```

L'agente hunter è stato sviluppato sfruttando JADE, questo approccio si riflette nel modo in cui l'agente viene posto in esecuzione: anziché avviare l'agente come processo autonomo, è necessario che JADE venga avviato passando come argomento il nome della classe dell'agente (che deve trovarsi nel *classpath*).

La messa in opera dell'agente è però semplificata grazie al file `build.xml`, dato che il comando

```
$ ant runAgent
```

carica Jade e l'agente hunter.

4.2.1 Applet di controllo

Il controllo dell'agente in JADE viene tradizionalmente effettuato attraverso una interfaccia grafica inserita direttamente nel codice dell'agente. Tale scelta normalmente non comporta grossi problemi, ma in un'ottica di mobilità (gli agenti devono necessariamente non superare una certa dimensione per migrare agevolmente tra i container) si è scelto di separare l'agente dalla sua interfaccia grafica. Ciò serve principalmente per limitare il carico di lavoro dello stesso agente, che non deve effetture *polling* degli eventi prodotti dall'interfaccia grafica. Inoltre, grazie a questa soluzione viene consentito il controllo dell'agente da parte di un host remoto.

La scelta che è stata fatta nella separazione dell'agente dalla propria interfaccia è quella di creare un applet Java che comunichi all'agente le azioni da intraprendere.

L'applet può risiedere all'interno dei contenuti di un server web in esecuzione sullo stesso host dell'agente, ma ciò non è strettamente necessario: la pagina contenente l'applet può risiedere in qualunque parte del web. Esiste naturalmente una limitazione imposta dalla rete sulla quale l'agente hunter si

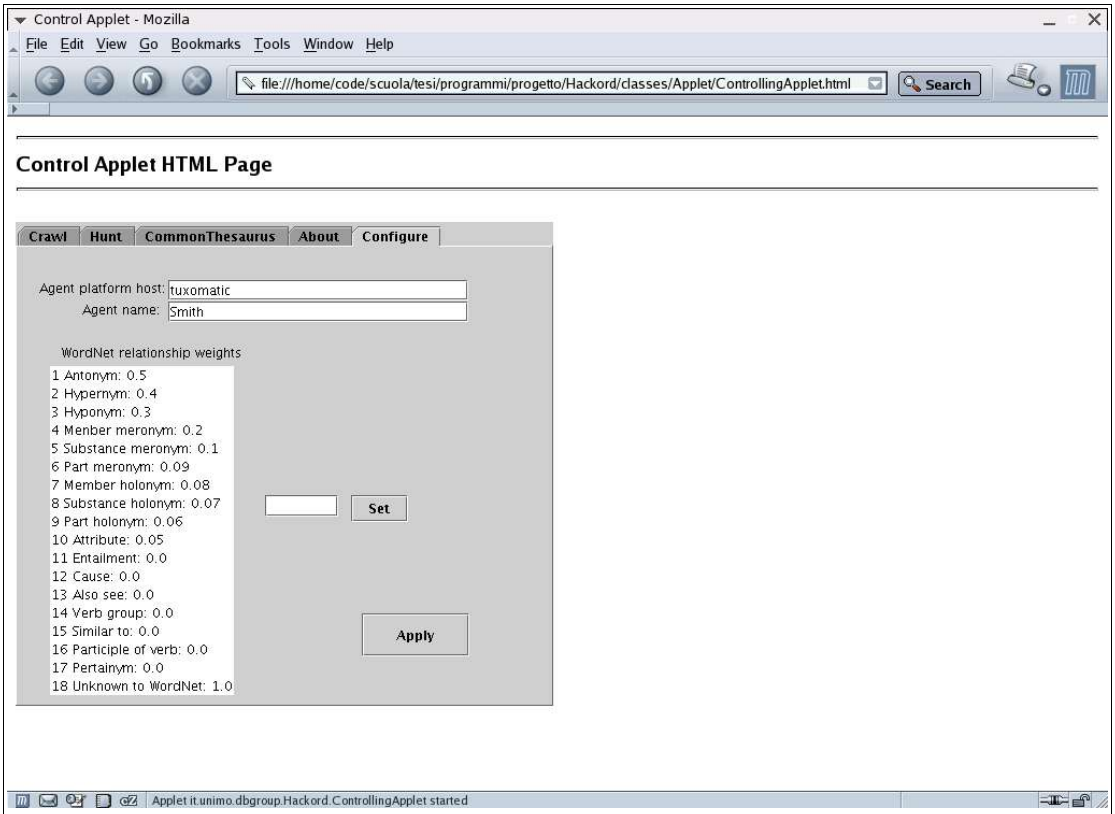


Illustrazione 1, l'applet di controllo dell'agente caricato dal browser insieme ad una pagina HTML

trova: la comunicazione tra l'host che carica l'applet e l'agente stesso deve essere possibile, come mostrato dal disegno 9 (pag. 42).

La struttura dell'applet è suddivisa in quattro schede principali, tre corrispondono alle **tre funzioni** che l'agente hunter è in grado di eseguire:

- **Crawling ricorsivo** con partenza da una pagina specificata e creazione di catene lessicali per ogni pagina incontrata
- **Ricerca** basata su due gruppi di **parole chiave**, principali e secondarie
- **Ricerca** basata su di un **Common Thesaurus** di cui viene fornito l'URL, l'agente provvederà a caricarlo e a intraprendere la ricerca.

L'ultima scheda contiene i **parametri di configurazione** che è possibile modificare da parte dell'utente.

In ciascuna delle schede che consentono di azionare l'agente, quando viene premuto il bottone di conferma, il browser viene ridiretto alla pagina contenente i risultati dell'elaborazione. La pagina inizialmente contiene un messaggio di attesa e l'istruzione di effettuare automaticamente il refresh di se stessa ogni pochi secondi. Al momento del termine delle operazioni, l'agente hunter sostituirà il contenuto della pagina di attesa con i risultati della propria ricerca, che verranno mostrati al client al momento del refresh successivo.

4.2.1.1 Crawling ricorsivo

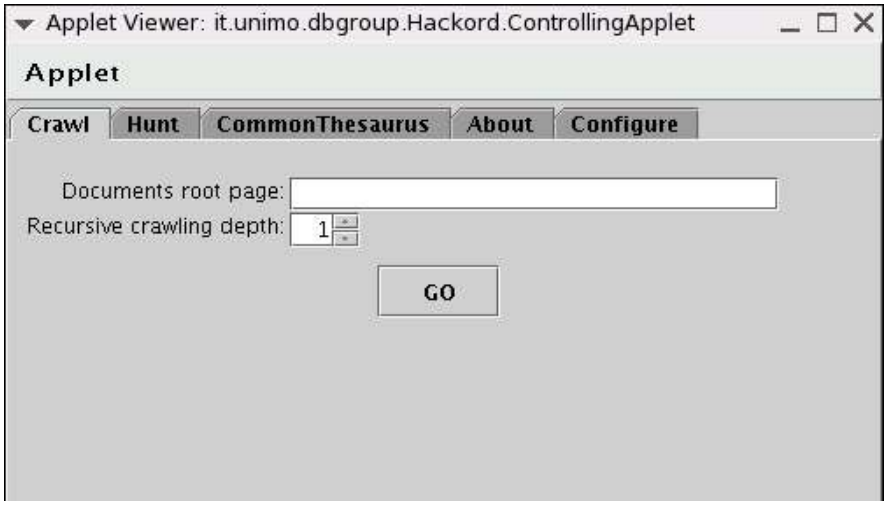


Illustrazione 2, scheda dell'applet dedicata alla funzione di crawling ricorsivo

L'interfaccia grafica dedicata a questa particolare funzione dell'agente hunter è relativamente semplice: viene fornita una casella di testo dove immettere l'URL della pagina web da cui iniziare il crawling, una ulteriore casella in cui immettere il massimo livello di ricorsione nel seguire i collegamenti, e naturalmente un bottone di conferma.

4.2.1.2 Ricerca di parole chiave

Essendo necessari due gruppi di parole chiave per iniziare questo tipo di ricerca, all'utente vengono presentate due caselle di testo in cui immettere le parole chiave principali e quelle secondarie.

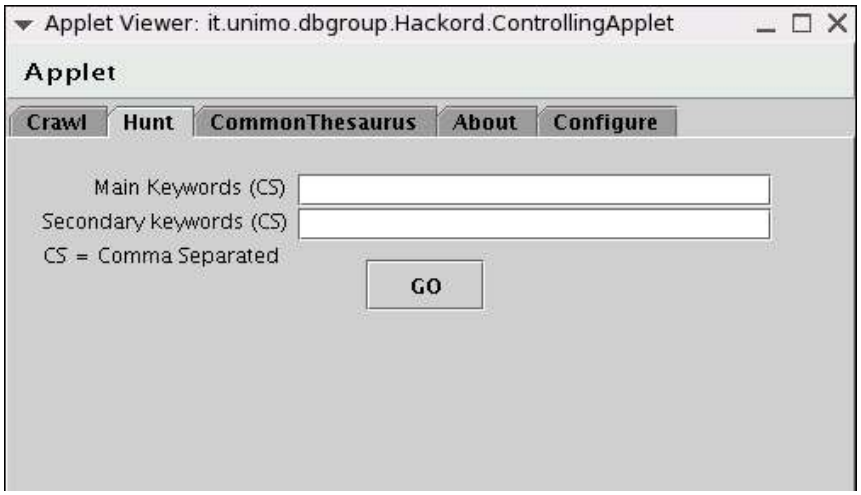


Illustrazione 3, scheda dell'applet utilizzata per iniziare una ricerca di parole chiave

Le parole chiave devono essere separate da virgole, la presenza di eventuali spazi è ignorata.

4.2.1.3 Ricerca attraverso un Common Thesaurus

L'unico parametro da inserire per fare sì che l'agente hunter carichi il contenuto di un Common Thesaurus e inizi a cercare documenti correlati è l'URL del file XML contenente il CT.

Illustrazione 4, scheda dell'applet da cui avviare la ricerca del contenuto di un Common Thesaurus

Ciò comporta che il Common Thesaurus risieda su qualche server web. Questa può sembrare una scelta discutibile, ma i CT sono normalmente documenti ingombranti e l'esecuzione di un upload da parte del client è spesso inapplicabile. Esistono comunque modi per predisporre funzioni di upload di contenuti all'interno dei documenti di un server web, per esempio appoggiandosi a linguaggi di scripting come PHP.

4.2.1.4 Scheda di configurazione

La configurazione consiste in alcuni parametri necessari all'applet per comunicare con l'agente e in altri impiegati dall'agente nel corso delle proprie elaborazioni.

All'avvio dell'applet tutti questi campi vengono inizializzati con opzioni di default, parte costanti e parte specificate nella pagina web che collega l'applet.

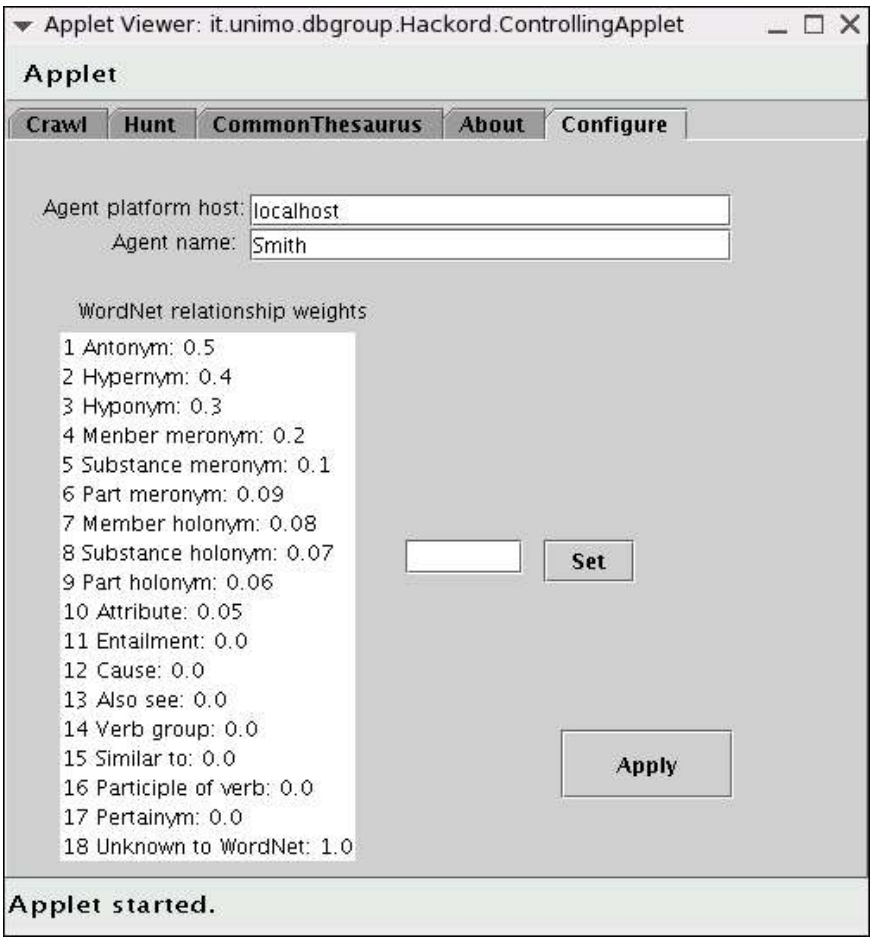


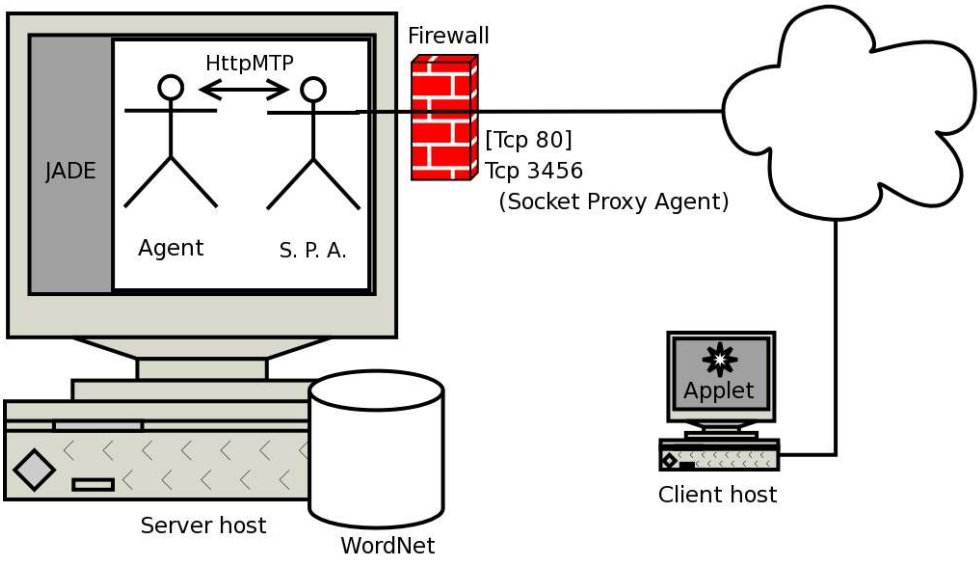
Illustrazione 5, scheda dell'applet dedicata alla configurazione della comunicazione con l'agente e ai pesi dei vari tipi di relazioni

È possibile specificare lo Hostname su cui risiede il SocketProxyAgent impiegato come *proxy* nella comunicazione.
La seconda casella di testo contiene il nome dell'agente hunter all'interno della piattaforma Jade su cui si trova, servirà per indirizzare correttamente i messaggi contenenti i comandi.
Infine, vi è una lista che per ogni tipo di relazione tra synset classificato in WordNet mostra il punteggio assegnato. Tali valori servono per fare pesare alcuni tipi di relazioni più di altri nei calcoli, benchè ciascuno di essi preso singolarmente sia privo di significato.
Quando l'utente seleziona un elemento della lista, il punteggio corrispondente compare nella piccola casella di testo sulla destra. Una volta modificato tale valore, premendo il pulsante Set viene confermato il nuovo punteggio.

4.2.2 SocketProxyAgent

Si è ritenuto opportuno utilizzare un particolare sistema per il trasferimento dei messaggi tra agenti, nella implementazione dell'agente hunter: il Socket Proxy

Agent (in realtà tale scelta si riflette solamente nella scrittura dell'applet già descritto). Tale agente viene fornito dagli stessi produttori di JADE e consente di fare un passo indietro nella virtualizzazione che il protocollo di trasferimento per i messaggi fornisce (per informazioni riguardo al funzionamento di JADE si rimanda al capitolo 2.4.1).



Disegno 9, schema del collegamento tra client e agente

Il Socket Proxy Agent riceve messaggi nel formato di comunicazione tra agenti attraverso un socket TCP[7] (realizzato utilizzando una porta a piacere) e li ridirige ad un agente specificato nei propri file di configurazione. Questa scelta fa sì che per l'agente hunter non debba cambiare nulla, mentre l'applet di controllo non necessita di una piattaforma ad agenti per comunicare, ma può semplicemente collegarsi con un socket TCP al proxy e trasferire i messaggi in un formato piuttosto semplice basato su testo ASCII.

Questa scelta consente l'utilizzo di un applet di dimensioni ridotte, quando l'alternativa sarebbe stata quella di avere una piattaforma Jade funzionante sul lato del client. Ciò avrebbe comportato di passare al client più librerie, dall'occupazione complessiva di 1.4MB.

L'unico svantaggio di questo approccio è la minore capacità di attraversare le reti che la connessione ad un SocketProxyAgent ha. L'HTTP-MTP di JADE è a tutti gli effetti assimilabile a traffico HTTP e un gateway che fosse dotato di funzioni di *inspection* dei protocolli non avrebbe difficoltà a fare passare i messaggi (supponendo che il traffico HTTP sia permesso). La connessione al Socket Proxy Agent da parte dell'applet di controllo non usa il protocollo HTTP ed è soggetta alle limitazioni che un *proxy* HTTP può causare. Tuttavia, finora questo approccio non ha causato difficoltà perché i test dell'agente sono stati

7 TCP – Transmission Control Protocol. Protocollo *connection-oriented* comunemente utilizzato nelle reti TCP/IP, quale è Internet.

condotti utilizzando l'applet sullo stesso host, facendo avvenire la comunicazione localmente. Se in futuro si dovesse rendere necessario utilizzare HTTP anche per le comunicazioni tra applet e agente hunter, si dovrà considerare la necessità di inserire una piattaforma ad agenti nell'applet.
Dal momento dell'avvio dell'interfaccia grafica di JADE è possibile vedere gli agenti che si trovano in esecuzione ed eseguire alcune operazioni di debugging.

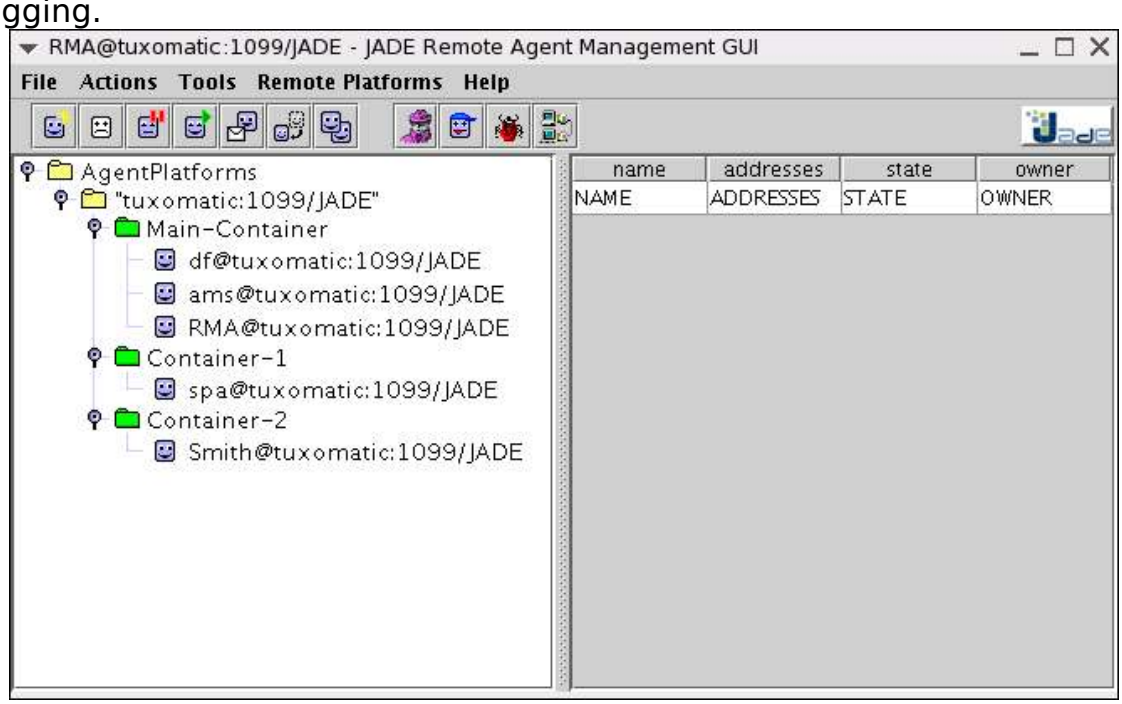


Illustrazione 6, cattura della finestra Remote Monitoring Agent di JADE

Come è possibile vedere dall'illustrazione 6, l'interfaccia grafica dell'RMA mostra due nuovi container distinti dal principale, in uno viene eseguito il Socket Proxy Agent, mentre nell'altro è attiva una istanza dell'agente hunter (l'interfaccia non mostra il nome della classe che viene eseguita, ma solo il nome dell'istanza).
Sullo stesso host dell'agente hunter si trova in esecuzione il DBMS (MySQL nel caso specifico) su cui è stato precedentemente caricato il database di WordNet. Se per particolari esigenze il DBMS dovesse essere situato su un altro host, sarebbe comunque possibile fare funzionare l'agente, anche se a velocità probabilmente ridotta dalla latenza della rete.

5 Prove di funzionamento

5.1 Crawling ricorsivo ed estrazione delle catene lessicali

Questo comportamento dell'agente a grandi linee rappresenta una eventuale indicizzazione preventiva secondo criteri semantici di un gruppo di pagine web collegate. Potrebbe venire impiegato in modo proattivo per velocizzare il reperimento di contenuti sulla base delle loro caratteristiche di significato.
L'efficacia nell'estrazione delle catene lessicali viene in parte inficiata dal gergo tecnico, che il tagger POS non è in grado di comprendere e i cui lemmi non sono presenti all'interno di WordNet. Quest'ultimo problema può essere corretto creando apposite estensioni al database di WordNet, mentre l'inefficienza del tagger comporta effetti collaterali più gravi, poiché vengono riconosciuti come nomi dei lemmi che in realtà apparterrebbero a una diversa POS.
Considerando il solo algoritmo di disambiguaizone lessicale ed estrazione delle catene lessicali, la performance appare decisamente buona. Le catene lessicali individuate e contengono comunque sempre lemmi legati semanticamente tra di loro, benché a volte si abbia l'impressione che nel procedimento di disambiguazione sia stato travisato il significato di alcuni nomi.
Segue parte dell'output dell'agente:

[...]
[http://www.slackware.com/announce/10.0.php]
(1390.6) -> {
device-15392: an instrumentality invented for a particular purpose; "the device is small enough to wear on your wrist"; "a device intended to conserve water",
support-21222: any device that bears the weight of another thing; "there was no place to attach supports for a shelf",
buffer-13965: a cushion-like device that reduces shock due to contact,
keyboard-17520: set of keys on a piano or organ or typewriter or typesetting machine or computer or the like,
drive-15674: (computer science) a device that writes data onto or reads data from a storage medium,
machine-17947: any mechanical or electrical device that transmits or modifies energy to perform or assist in the performance of human tasks
}
(1012.9) -> {
info-30530: a message received and understood that reduces the recipient's uncertainty,
list-29839: a database containing an ordered array of items (names or topics),
information-30530: a message received and understood that reduces the recipient's uncertainty
}
(945.0) -> {
info-30530: a message received and understood that reduces the recipient's uncertainty,
information-30530: a message received and understood that reduces the recipient's uncertainty,
offering-33033: something offered (as a proposal or bid); "noteworthy new offerings for investors

included several index funds"
}
(926.0) -> {
	info-30530: a message received and understood that reduces the recipient's uncertainty,
	propaganda-30693: information that is spread for the purpose of promoting some cause,
	information-30530: a message received and understood that reduces the recipient's uncertainty
}
(921.5) -> {
	info-30530: a message received and understood that reduces the recipient's uncertainty,
	update-30563: news that updates your information,
	information-30530: a message received and understood that reduces the recipient's uncertainty
}
(921.0) -> {
	info-30530: a message received and understood that reduces the recipient's uncertainty,
	information-30530: a message received and understood that reduces the recipient's uncertainty,
	book-30537: a compilation of the known facts regarding something or someone; "Al Smith used to say, `Let's look at the record'"; "his name is in all the recordbooks"
}
(515.3) -> {
	device-15392: an instrumentality invented for a particular purpose; "the device is small enough to wear on your wrist"; "a device intended to conserve water",
	printer-19422: a machine that prints,
	keyboard-17520: set of keys on a piano or organ or typewriter or typesetting machine or computer or the like,
	machine-17947: any mechanical or electrical device that transmits or modifies energy to perform or assist in the performance of human tasks
}
(488.2) -> {
	device-15392: an instrumentality invented for a particular purpose; "the device is small enough to wear on your wrist"; "a device intended to conserve water",
	connection-14906: something that connects; "he soldered the connection"; "he didn't have the right connector between the amplifier and the speakers",
	system-21313: a combination of interrelated interacting artifacts designed to work as a coherent entity; "he bought a new stereo system"; "the unit consists of a motor and a small computer"
}
(403.9) -> {
	utility-30272: (computer science) a program designed for general support of the processes of a computer; "a computer system provides utility programs to perform the tasks needed by most users",
	interpreter-30252: (computer science) a program that translates and executes source language statements one line at a time,
	compiler-30239: (computer science) a program that decodes instructions written in a higher order language and produces a machine language program,
	software-30213: (computer science) written programs or procedures or rules and associated documentation pertaining to the operation of a computer system and that are stored in read/write memory,
	program-30226: (computer science) a sequence of instructions that a computer can interpret and execute; "the program required several hundred lines of code",
	application-30227: a program that gives a computer instructions that provide the user with tools to accomplish a task; "he has tried several different word processing applications"
}
[...]

Visited pages: [http://www.slackware.com/announce/9.1.php, http://www.slackware.com/index.php, http://www.slackware.com/announce/10.0.php, http://www.slackware.com/book/, http://www.slackware.com/lists/, http://www.slackware.com/contact/, http://www.slackware.com/pb/, http://www.slackware.com/about/, http://www.slackware.com, http://www.slackware.com/support/, http://www.slackware.com/security/, http://alphageek.dyndns.org/linux/ slackware-mirrors.shtml, http://www.slackware.com/info/, http://slackware.com/trademark/trademark.php, http://www.mnstate.edu, http://store.slackware.com, http://www.slackware.com/zipslack/, http://www.slackware.com/announce/1.0.php, http://www.slackware.com/ports/, http://www.slackware.com/config/, http://www.slackware.com/faq/, http://www.slackware.com/torrents/index.html, http://www.slackware.com/install/, http://www.slackware.com/changelog/i386/ChangeLog-stable.txt, http://www.slackware.com/~msimons/slackware/grfx/, http://www.slackware.com/links/, http://www.slackware.com/changelog/, http://www.slackware.com/announce/9.0.php, http://www.slackware.com/getslack/]
Run time: 965272

5.2 Ricerca di parole chiave

Con il seguente test si è cercato di osservare se la ricerca di parole chiave descritta nel capitolo 3.4.1 è realmente in grado di aggiungere una componente semantica alla ricerca sul web che è possibile effettuare con un comune motore di ricerca.
Al momento dell'esecuzione del test era appena stata rilevata una vulnerabilità di tipo overflow al codice della libreria libpng, che normalmente viene usata per la gestione delle immagini nel formato Portable Network Graphics. Tale problema colpiva sia i programmi che fanno uso della libpng in forma dinamica che i programmi compilati con un link statico a tale libreria. Uno dei programmi che contengono libpng staticamente è il largamente diffuso browser Mozilla.
Allo scopo di ottenere maggiori informazioni al riguardo, le parole chiave utilizzate per la ricerca perciò sono state:

- Parole chiave principali: mozilla, png, overflow
- Parole chiave secondarie: fix,patch,bug,crash,security,bad

I risultati ritornati dall'agente sono stati i seguenti:

> Searching: **mozilla png overflow**
>
> Run time: 263580
>
> Keywords: mozilla bad patch fix crash security overflow bug png
>
> http://www.xatrix.org/advisory.php?s=4389
> Score: 123.8

... gtk2 imagemagick imlib latex2html libwmf **mozilla** netpbm perl-gd perl-tk php **png** pstoedit qt **...** security advisory [0], a buffer **overflow** vulnerability was **...**

http://www.monkey.org/openbsd/archive/ports/0408/msg00204.html
Score: 85.0
... how to update **mozilla** for latest **png** buffer-**overflow** fix? To: ports@openbsd.org; Subject: how to update **mozilla** for latest **png** buffer-**overflow** fix? **...**

http://www.monkey.org/openbsd/archive/ports/0408/msg00246.html
Score: 48.0
... Re: how to update **mozilla** for latest **png** buffer-**overflow** fix? **...** Prev by thread: Re: how to update **mozilla** for latest **png** buffer-**overflow** fix? **...**

http://annevankesteren.nl/archives/2004/08/overflow
Score: 42.8
... Mouse wheel scrolling on a DIV using **overflow** didn't work (either **...** have been added already and it seems that **Mozilla** is going to support **PNG** images and **...**

Per avere un riscontro circa la effettiva validità dei metodi di analisi del linguaggio naturale impiegati è stata eseguita una query di ricerca presso Google con le stesse parole chiave impiegate dall'agente hunter e si è data una valutazione soggettiva dei singoli risultati. Sostituendo il procedimento automatizzato di analisi del testo con questo tipo di valutazione si è cercato di scoprire se i risultati riportati dall'agente hunter come interessanti sono effettivamente soddisfacenti dal punto di vista dei contenuti. Ciò valutando se il contenuto delle pagine è effettivamente relativo all'ambito semantico delle parole chiave fornite all'agente al momento di iniziare la ricerca.

URL	P	A	S	P/A	V.R.	P/A/S
http://www.monkey.org/openbsd/archive/ports/0408/msg00246.html	48.00	35.48	8	1.35	0.34	0.17
http://www.monkey.org/openbsd/archive/ports/0408/msg00204.html	85.00	23.85	9	3.56	0.75	0.40
http://www.k-otik.com/exploits/08112004.lbpng.c.php	7.80	101.21	8	0.08	10.67	0.01
http://www.xatrix.org/advisory.php?s=4389	123.80	110.52	7	1.12	0.20	0.16
http://xforce.iss.net/xforce/xfdb/9287	19.00	54.09	8	0.35	1.56	0.04
http://archivist.incutio.com/viewlist/css-discuss/24514	2.50	18.61	5	0.13	5.70	0.03
http://archivist.incutio.com/viewlist/css-discuss/24510	9.00	47.11	5	0.19	3.71	0.04
http://www.symantec.com/techsupp/bulletin/archive/netrecon/082004sva.html	34.00	127.38	9	0.27	2.37	0.03
http://annevankesteren.nl/archives/2004/08/overflow	42.80	41.30	4	1.04	0.13	0.26
P = punteggio risultante						
A= punteggio minimo per risultare rilevante						
S = interesse valutato soggettivamente (su 10)						
V.R. = Variazione rispetto alla media (riferito alla colonna precedente)						

Delle sei pagine ritenute rilevanti da una persona in grado di leggere la lingua inglese, solo tre sono state individuate dall'agente. Resta comunque

significativo il fatto che tra i risultati vi è un solo falso positivo e che la risposta dell'agente ha risparmiato all'utente la lettura delle dieci pagine restituite da Google.
Volendo produrre una operazione analoga sarebbe stato possibile valutare i risultati di Google basando il giudizio dell'utente sullo *snippet*, la piccola porzione della pagina riportata dove vengono messe in evidenza le parole chiave. Tra le pagine riportate riportate di interesse in seguito alla lettura, almeno in due casi questo sarebbe stato impossibile dato che lo snippet non contiene frasi di senso compiuto. Uno di questi due casi è stato riportato dall'agente come rilevante, risparmiando all'utente la lettura del documento.
Allo scopo di avere più informazioni riguardo l'accuratezza dell'agente hunter nella ricerca di parole chiave, una attività che sta alla base di tutti i suoi comportamenti più complessi, si è deciso di eseguire un ultariore test al riguardo. Le parole chiave utilizzate in questo caso sono state le seguenti:

- Parole chiave principali: pcmcia, card, wireless
- Parole chiave secondarie: attaching, external, wi-fi, 802.11b, radio, communication, signal, noise, power, gain, decibel, lan

Omettendo la lista dei risultati, si riporta in forma tabellare l'analisi della performance.

URL	P	A	S	P/A	V.R.	P/A/S	V.R.
http://linux.oldcrank.com/tips/wpc11/	80.9	77.56	7	1.04	0.37	0.15	0.44
http://www.trendware.com/products/TEW-PC16.htm	156.96	104.7	7	1.5	0.04	0.21	0
http://www.compusa.com/products/product_info.asp?product_code=287284&pfp=BROWSE	91.2	185.16	6	0.49	1.91	0.08	1.61
http://catalog.belkin.com/IWCatProductPage.process?Merchant_Id=&Section_Id=201523&pcount=&Product_Id=122691	9.9	6.79	6	1.46	0.02	0.24	0.12
http://www.linksys.com/products/product.asp?prid=156&grid=19	75.1	50.41	6	1.49	0.04	0.25	0.14
http://www.utexas.edu/its/wireless/install/install_pcmcia.html	29.56	123.12	7	0.24	4.97	0.03	5.24
http://forums1.itrc.hp.com/service/forums/bizsupport/questionanswer.do?threadId=687415	47	130.87	6	0.36	2.99	0.06	2.58
http://www.goonda.org/wireless/	133	87.25	8	1.52	0.06	0.19	0.12
http://www.techadvice.com/help/Products/w/wireless-pcmcia-cards.asp	438.76	120.21	9	3.65	0.61	0.41	0.47
http://www.experts-exchange.com/Hardware/Notebooks_Wireless/Q_21062798.html	271.3	105.67	5	2.57	0.44	0.51	0.58
P = punteggio risultante							
A= punteggio minimo per risultare rilevante							
S = interesse valutato soggettivamente (su 10)							

In questo caso gli algoritmi di analisi del linguaggio naturale sembrano avere funzionato meglio che nel caso precedente, poiché nei sette risultati riportati dall'agente uno solo è un falso positivo. Inoltre, all'interno dei dieci risultati di Google solo uno ritenuto rilevante da un utente umano è stato tralasciato dall'agente.

5.3 Ricerca basata su un Common Thesaurus

I tempi lunghi che caratterizzano questo tipo di ricerca non hanno consentito

l'esecuzione di test approfonditi. È stato accertato che anche dopo ore di funzionamento l'agente hunter non causa una eccessiva occupazione di memoria e la sua performance non degrada.
Realizzando un servizio basato sulla ricerca di Common Thesaurus, la proattività che caratterizza gli agenti si renderebbe oltremodo necessaria, poiché il tempo di risposta che si otterrebbe eseguendo la ricerca in linea sarebbe scoraggiante per ogni tipo di utenza.

5.4 Tempi di esecuzione

Nel corso dell'esecuzione dei test si è avuto modo di constatare che il tempo impiegato da una ricerca è scarsamente influenzato dalla velocità con cui è possibile effettuare il download delle pagine web da sottoporre ad analisi. In due test eseguiti fornendo le stesse parole chiave, uno operando da una LAN 10baseT (10 Mbps, condivisi con altri host) e l'altro da una connessione dialup su linea PSTN (56 kbps, dedicati) il tempo di esecuzione è stato pressoché identico.

$t_{Dialup} = 719262\,ms\,;\, t_{LAN} = 263580\,ms$

Speedup globale $s = \frac{t_{Dialup}}{t_{Dialup} - t_{LAN}} \simeq 1.58$

Speedup della connessione $s_{net} = \frac{10\,Mbps}{56\,kbps} \simeq 183$

Risulta evidente come gran parte del tempo impiegato viene speso nella computazione dei punteggi relativi a ciascuna pagina e nell'attesa delle risposte del DBMS, non nel download.

6 Conclusioni

> *Any sufficiently advanced technology is indistinguishable from a rigged demo.*
>
> (Anonimo)

Il naturale evolversi delle tecnologie dell'Informazione non potrà prescindere dallo studio di avanzati algoritmi di analisi del linguaggio naturale. La quantità di informazioni che viene ogni giorno prodotta e destinata all'archiviazione non sarà accessibile per sempre con metodiche “letterali” di ricerca di espressioni. Si giungerà ad un punto in cui le *keyword* potranno produrre così tanti match da rendere inutilizzabili le metodiche di ricerca tradizionali. Dato che non è possibile ricercare interi paragrafi e la maggior parte delle persone non sa coniare espressioni regolari per le proprie ricerche, un metodo di archiviazione

e ricerca semantico sarà un giorno necessario.
Gli agenti software rappresentano un affascinante paradigma di programmazione, che rappresenta una svolta forse paragonabile al passaggio dal sistema procedurale a quello ad oggetti. Gli aspetti più innovativi che caratterizzano gli agenti sono sicuramente l'autonomia e la proattività. L'idea che un programma software possa anticipare le reali necessità di un utente sembra fantascientifica se paragonata a ciò che l'informatica poteva offrire fino a non molti anni fa, cioè comupter che per quanto potenti erano destinati ad eseguire applicazioni completamente *passive*. Ma gli agenti rappresentano un grande passo avanti anche rispetto alla programmazione di software interattivo: non vengono più richieste le decisioni dell'utente.
Con la realizzazione di un progetto, seppur dimostrativo, che combina queste due tecnologie si spera di avere messo in evidenza la versatilità e la potenza che esse offrono.
I test condotti sull'agente hunter hanno mostrato degli esiti concordi alle aspettative e fanno ben sperare circa il riutilizzo in progetti futuri del codice sorgente scritto in questa occasione.

Durante lo svolgimento di questa tesi di laurea è stato prodotto, oltre a questo stesso testo, il codice sorgente di una nuova implementazione di TUCUXI, su cui per la prima volta sono stati effettivamente realizzati gli algoritmi presentati al capitolo 3 (pag. 18). Tale sorgente consiste in

➔ 22 classi Java

per un totale di

➔ 4603 righe di codice sorgente.

Il codice sorgente del progetto sarà reso disponibile presso il sito web del DataBase Group di questa stessa Università:

http://dbgroup.unimo.it

Daniele Gozzi

7 Bibliografia

1: D. Benevantano, S. Bergamaschi, G. Gelati, F. Guerra, M. Vincini, "MIKS: an agent framework supporting information access and integration",

2: S. Bergamaschi, S. Castano, M. Vincini, D. Beneventano, "Semantic Integration and Query of Heterogeneous Information Sources", 1999

3: MySQL, Homepage, http://www.mysql.org

4: Google, Google WEB APIs, http://www.google.com/apis/index.html

5: W3C, Web Services Description Language (WSDL), http://www.w3.org/TR/wsdl

6: W3C, SOAP Version 1.2 Part 0: Primer, http://www.w3.org/TR/soap12-part0/

7: FIPA, Foundation for Intelligent Physical Agents Homepage, http://www.fipa.org

8: Telecom Italia Lab, Homepage, http://www.telecomitalialab.com

9: Beneventano, Bergamaschi, Fergnani, Guerra, Vincini, Montanari, "A peer-to-peer information system for the semantic web", 2003

10: Hugo Liu, Homepage di MontyTagger, http://web.media.mit.edu/~hugo/montytagger

11: Francesco Guerra, Dai Dati all'Informazione: il sistema MOMIS, 2004

12: T. R. Gruber, A translation approach to portable ontology specification (vol. 5), Cap. ,1993

13: Jane Morris, Graeme Hirst, "Lexical Cohesion Computed by Thesaural Relations as an Indicator of [...]",

14: Manabu Okumura, Takeo Honda, "Word Sense Disambiguation and Text Segmentation Based on Lexical Cohesion",

15: R. Benassi, S. Bergamaschi, M. Vincini, "TUCUXI: the intelligent hunter agent for concept understanding and [...]", 2004

16: Graeme Hirst, David St. Onge, WordNet: An electronic lexical database and some of its applications, Cap. ,

17: John R. Hubbard, Strutture dati in Java, Cap. 16,2001

18: Porter, Program, Vol. 14, Cap. 3,

19: W3C, Extensible Markup Language, http://www.w3.org/TR/REC-xml/