RICERCA SEMANTICA MULTIDOMINIO SUL WEB

By

Gianluca Di Fiore

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
AT
DALHOUSIE UNIVERSITY
HALIFAX, NOVA SCOTIA
JANUARY 2003

DALHOUSIE UNIVERSITY
DEPARTMENT OF
MATHEMATICS STATISTICS AND COMPUTING SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "**Ricerca semantica multidominio sul WEB**" by **Gianluca Di Fiore** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: January 2003

Supervisor: _____

Readers: _____

_____

DALHOUSIE UNIVERSITY

Date: **January 2003**

Author: **Gianluca Di Fiore**

Title: **Ricerca semantica multidominio sul WEB**

Department: **Mathematics Statistics and Computing Science**

Degree: **M.Sc.** Convocation: **May** Year: **2003**

Signature of Author

A mia madre e Daria

Table of Contents

List of Tables

List of Figures

Introduzione

Il *World Wide Web* è diventato progressivamente una vasta risorsa di informazioni; a causa della complessità nell'organizzazione dei dati e della quantità di informazioni presenti, la ricerca sul WEB di informazioni davvero utili per un utente è diventata decisamente complessa: anche se un motore di ricerca Web assiste nella scoperta di risorse, esso è lontano dal soddisfare, per la sua scarsa precisione, le richieste di una ricerca mirata. Infatti trovare le informazioni richieste nei risultati di un motore di ricerca si rivela fruttuoso solo in presenza di argomenti di una certa notorietà e importanza, e di query molto precise; negli altri casi questo lavoro può implicare una considerevole perdita di tempo che un utente può pagare in termini di costi di collegamento alla rete.

Questo avviene perché i motori di ricerca tradizionali effettuano ricerche di tipo **sintattico**: essi restituiscono le pagine che contengono le keywords presenti nelle query degli utenti, indipendentemente dal contesto in cui esse vengono utilizzate. Se ciò da un lato è conveniente in termini di velocità di reperimento delle pagine e restituzione dei risultati, dall'altro lato porta spesso a risultati errati o imprecisi, dato

che vengono restituite molte pagine non attinenti al contesto della query dell'utente.

Ad esempio, un utente che voglia cercare pagine relative alla pesca, inserendo in un motore come keyword la parola *"pesca"*, oltre a trovare pagine attinenti al dominio ittico troverà sicuramente pagine inerenti al dominio ortofrutticolo. Sarebbe conveniente avere a disposizione un sistema in grado di capire di cosa parla una pagina valutando la sua attinenza con i domini di interesse per l'utente. Una ricerca di tal tipo è detta **semantica** in quanto non restituisce semplicemente pagine che contengono le keywords, ma pagine che hanno anche un contenuto semantico aderente al dominio desiderato dall'utente.

Un problema analogo, relativo però alle informazioni contenute nei database, si ha con il *Data Mining*; infatti esso è usato per identificare modelli validi, nuovi, potenzialmente utili e comprensibili, da una raccolta di dati in una collettività di database.

Il lavoro che invece si interessa delle informazioni non strutturate ed eterogenee presenti sul WEB viene spesso identificato con il nome di **Web Mining**. Anche se non c'è un accordo generale sulla definizione di Web Mining possiamo dire, in analogia con la definizione di Data Mining, che esso *rappresenta l'attività di identificazione di modelli impliciti in una grande raccolta di documenti presenti sul WEB.*

Il progetto sviluppato si colloca nella teoria del Web Mining e si basa sugli **Intelligent Search Agents (ISA)**. Questi sono programmi usati per filtrare le informazioni

ritrovate sul WEB in base agli interessi dell'utente, memorizzandone le scelte e costruendone il profilo. Ad essi si stanno interessando molte tra le più importanti aziende del settore informatico, da Netscape a Microsoft, da IBM a AT&T.

Tale lavoro si prefigge lo scopo di estendere un prodotto già realizzato precedentemente, che valuta l'attinenza delle pagine reperite ad un dominio applicativo prefissato, al fine di renderlo multidominio, cioè capace di apprendere nuovi domini e valutare l'attinenza delle pagine a tutti i domini conosciuti, considerando anche eventuali gradi di correlazione tra i domini. Il sistema, come il prodotto precedente, consente di reperire pagine Web interfacciandosi con i motori di ricerca più utilizzati dalla comunità Web ma consente di aggiungere nuovi motori di ricerca nel caso in cui si presenti la necessità.

0.1 Organizzazione della tesi

Nel primo capitolo è descritto lo stato dell'arte del Web Mining ed è mostrata una sua possibile tassonomia; vengono altresì illustrati alcuni sistemi già realizzati.

Nel secondo capitolo viene spiegato l'approccio teorico sul quale si basa il seguente lavoro di tesi, in particolare

- si chiarisce il concetto di *Agente Software* e viene presentata una panoramica sulle diverse tipologie di agenti esistenti in letteratura
- si definisce cosa si intende per *Semantic Network*

- si introduce il *modello formale* alla base del sistema realizzato
- si definisce la *metrica* mediante la quale vengono giudicate le pagine Web

Nel terzo capitolo viene illustrato il sistema realizzato in ogni sua componente. In tale capitolo vengono anche presentati gli agenti sviluppati per mezzo di uno specifico linguaggio di descrizione degli agenti: il *Service Description Language (SDL)*. Vengono infine mostrate le strutture del *Web Repository* e della *Semantic Knowledge Base*.

Nel quarto capitolo vengono illustrati i risultati sperimentali ed i confronti effettuati con i risultati dei motori di ricerca.

Chapter 1

Stato dell'arte

Nel corso del capitolo, dopo aver spiegato cosa si intende per Web Mining, verranno illustrati alcuni sistemi che affrontano il problema dell'identificazione di informazioni nascoste nel WEB.

1.1 Una definizione di Web Mining

Il **Web Mining** è un nuovo problema di ricerca sotto discussione, che ha molto interesse per molte comunità. Attualmente non c'è accordo su cosa debba intendersi con il termine Web Mining, ma una sua possibile definizione è la seguente [4]

Definition 1.1.1 (Web Mining). Il Web Mining è l'attività di identificazione di modelli impliciti in una grande raccolta di documenti C, che può essere denotata con un mapping $\xi : C \rightarrow p$.

Poiché il Web Mining deriva dal Data Mining, la sua definizione è molto simile alla nota definizione di Data Mining[1].

[1]Si definisce Data Mining l'attività usata per identificare modelli validi, nuovi, potenzialmente utili e comprensibili da una collettività di database [4].

Ciononostante, il Web Mining ha molte caratteristiche uniche se confrontato con il Data Mining:

- Le sorgenti del Web Mining sono documenti Web
- Il WEB è assimilabile ad un grafo orientato i cui nodi sono i documenti ed i cui archi sono i link. Per questo motivo, il modello identificato può riguardare il contenuto dei documenti oppure la struttura WEB
- I documenti Web sono semi-strutturati o non strutturati, mentre i dati su cui opera il Data Mining sono strutturati in database[2]

1.2 Una tassonomia per il Web Mining

Spesso c'è una forte ambiguità quando si parla di Web Mining, poiché con la stessa locuzione si può far riferimento a due campi di applicazione molto diversi tra loro:

- **Web Content Mining**: descrive il processo di ricerca di informazioni o risorse da milioni di fonti attraverso il Web
- **Web Usage Mining**: è il processo di mining dei file log di accesso al Web su uno o più siti

[2]Si noti che il Web Mining è differente dal Web Information Retrieval in quanto hanno differenti obiettivi. Sebbene il Web Mining è più del Web Information Retrieval esso non è inteso per rimpiazzarlo, invece sono due tecnologie che si integrano. D'altro canto il Web Mining può essere utilizzato per aumentare la precisione del Web Information Retrieval e migliorare l'organizzazione del risultato del recupero delle informazioni [4].

In questi due campi, poi, occorre fare un ulteriore sforzo di classificazione per poter capire bene tutte le possibili applicazioni cui spesso ci si riferisce semplicemente con la locuzione "Web Mining".

Una possibile tassonomia è rappresentata nella figura 1.1.

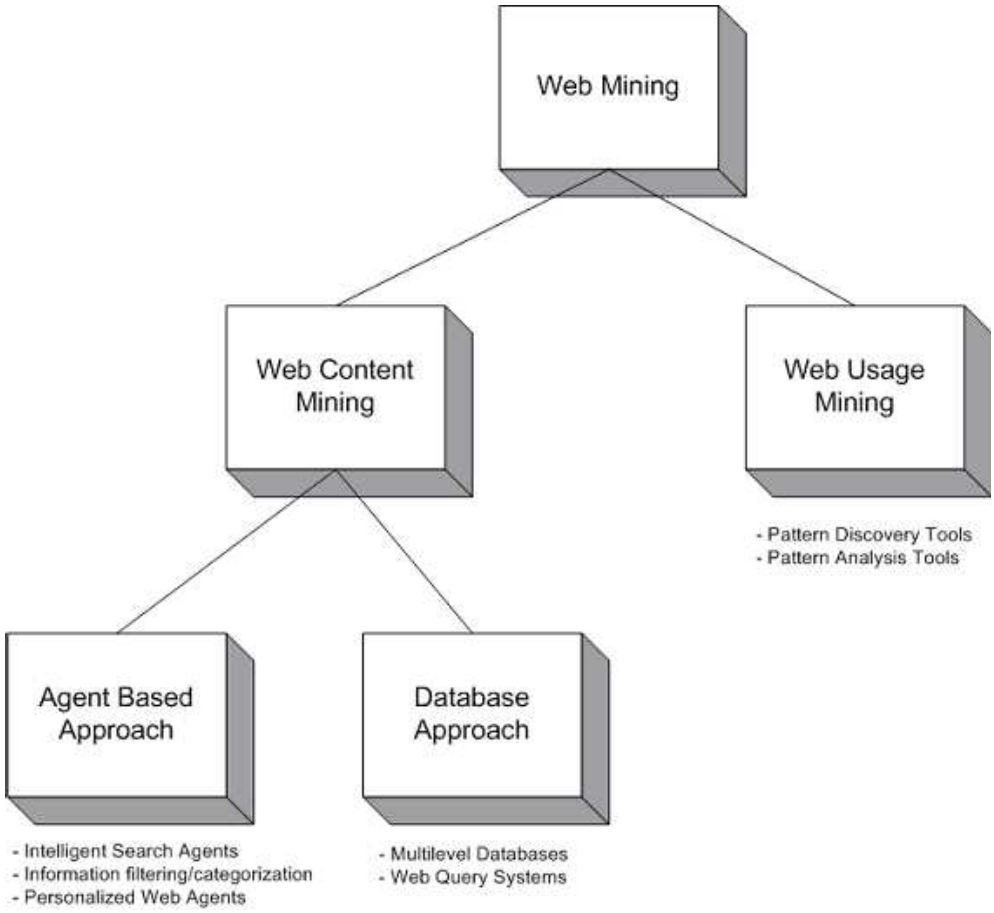

Figure 1.1: Tassanomia per il Web Mining

1.3 Web content mining

L'eterogeneità e la mancanza di struttura che caratterizza le fonti di informazione sul WEB ne rende difficile la scoperta automatica, l'organizzazione e la gestione delle informazioni. Gli strumenti tradizionali di ricerca e indicizzazione di Internet e del World Wide Web, come Lycos, Altavista, WebCrawler e altri, costituiscono un buon supporto per l'utente, ma generalmente non forniscono informazioni strutturate e non categorizzano, filtrano o interpretano i documenti.

Negli anni recenti c'è stato un grosso impulso da parte della ricerca per sviluppare strumenti di information retrieval più "intelligenti", come, ad esempio, agenti Web intelligenti o l'estensione di tecniche di database e di Data Mining per fornire un più alto livello di organizzazione per i dati semi-strutturati disponibili sul WEB.

Gli approcci generalmente proposti in letteratura sono quelli "Agent Based" e quelli "Data Base Approach". Nel seguito verranno discussi i due approcci.

1.3.1 Approccio Agent-Based

Di solito, i sistemi di **Web Mining Agent-Based** si possono dividere in tre categorie:

- **Agenti di ricerca intelligente**: tali agenti ricercano le informazioni rilevanti sulla base di caratteristiche del *"dominio"* e dei *"profili utente"* per organizzare e interpretare le informazioni trovate. Esistono agenti per informazioni di domini pre-specificati per cercare particolari tipi di documenti (ad esempio **Harvest** [2] [41], **FAQ-Finder** [42], **Information Manifold** [43]) ed agenti che imparano la struttura di informazioni sconosciute (ad esempio **ShopBot** [44])

- **Agenti per il filtraggio/categorizzazione delle informazioni**: usano varie

tecniche di *Information Retrieval* e le caratteristiche ipertestuali del WEB per reperire, filtrare e categorizzare i documenti (ad esempio **HyPursuit** [45] che clusterizza i documenti sulla base di informazioni insite nelle strutture dei link e del contenuto dei documenti)

- **Agenti Web personalizzati**: imparano le preferenze dell'utente e sulla base di esse (e di preferenze di utenti con interessi analoghi) ricercano le informazioni sul WEB (ad esempio **WebWatcher** [46])

1.3.2 Database Approach

Tale approccio consente di organizzare i dati semi-strutturati del WEB in collezioni di risorse più strutturate, usando i meccanismi standard di interrogazione dei database e tecniche di Data Mining per l'analisi. Sono state proposte due soluzioni:

- **Database multilivello**: in tali database il livello più basso contiene informazioni semi-strutturate memorizzate in diversi *Web Repositories*, ad esempio documenti ipertestuali. Ai livelli più alti sono estratti i metadati che vengono organizzati in collezioni strutturate di dati, come i database relazionali. Ad esempio, il sistema **ARANEUS** [60] estrae informazioni rilevanti dai documenti ipertestuali e li integra in *Ipertesti Web Derivati* (*Derived Web Hypertexts*) di più alto livello che rappresentano generalizzazioni del concetto di vista di un database

- **Sistemi Web Query**: utilizzano i linguaggi standard di interrogazione dei database, ad esempio SQL, informazioni strutturali e perfino il linguaggio naturale per sottomettere query sul WEB. Ad esempio, **W3QL** [47] combina query riguardanti la struttura, basate sull'organizzazione degli ipertesti, e query riguardanti il contenuto dei documenti, basate su tecniche di Information Retrieval.

1.4 Web Usage Mining

Il **Web Usage Mining** consiste nella scoperta automatica dei pattern di accesso degli utenti ai siti Web a partire dai *log* memorizzati nei server Web. Le organizzazioni hanno a disposizione ingenti quantità di dati generate automaticamente dai server Web e raccolte nei *log*. Il Web Usage Mining consiste nell'applicazione di tecniche di Data Mining agli *usage logs* di grandi repository di dati Web in modo da analizzare come un sito viene utilizzato. L'analisi di utilizzo include statistiche dirette, come la frequenza di accesso alle pagine, e forme più sofisticate di analisi, come trovare *paths* di attraversamento comuni attraverso un sito Web (*common traversal paths*).

Le informazioni sull'utilizzo possono essere usate per ristrutturare un sito Web in modo da servire meglio l'utente:

- Percorsi di attraversamento lunghi e complessi o il basso utilizzo di una pagina con informazioni importanti possono suggerire che i link del sito e le informazioni non sono posizionate in modo intuitivo

- Il progetto del layout fisico dei dati o lo schema di caching per un server Web possono essere migliorati dalla conoscenza di come gli utenti tipicamente navigano attraverso il sito

- Le informazioni sull'uso possono anche essere usate per aiutare direttamente la navigazione del sito fornendo una lista di destinazioni preferite da una particolare pagina Web

Distinguiamo due tipologie di tools: **pattern discovery tools** e **pattern analysis tools**.

1.4.1 Pattern discovery tools

Sono tools che usano tecniche sofisticate di intelligenza artificiale, Data Mining, psicologia e teoria dell'informazione per "scovare" conoscenza da collezioni di dati. Ad esempio, **WEBMINER** [48] è in grado di scovare automaticamente regole d'associazione e pattern sequenziali dai log.

1.4.2 Pattern analysis tools

Sono tools per interpretare e visualizzare i pattern trovati. Alcuni tools fanno uso di *tecniche OLAP* come i *data cube* per semplificare l'analisi. **WEBMINER** [48] propone un meccanismo di query simile a SQL per interrogare i pattern scoperti.

1.5 L'architettura del sistema WebTMS

Per meglio comprendere gli approcci descritti, nel seguito illustriamo un certo numero di sistemi presentati in letteratura.

WebTMS [38] (*Web Text Mining System*), sviluppato dal Dipartimento di Computer Science and Technology dell'Università di Nanjing, è un sistema multi-agente che combina il Text Mining con l'analisi multidimensionale dei documenti per aiutare l'utente nel mining dei documenti HTML reperibili sul WEB. Lo scopo è quello di identificare pattern validi e potenzialmente utili in una collezione di dati presenti in una Web Warehouse.

L'architettura del sistema viene mostrata sinteticamente in figura 1.2.

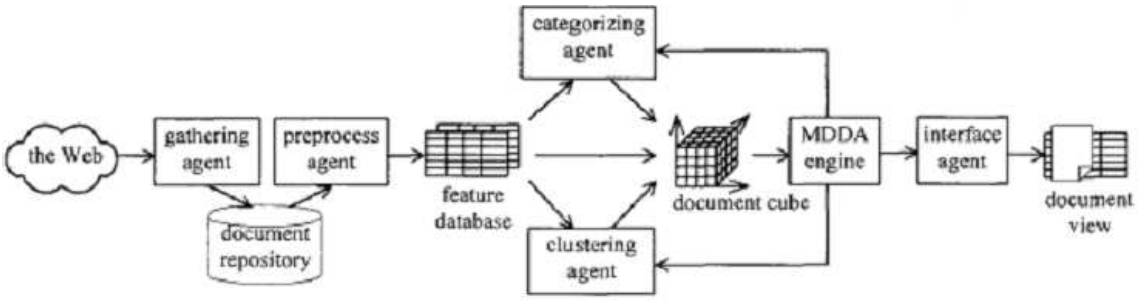

Figure 1.2: L'architettura del sistema WebTMS

Analizziamo di seguito i singoli componenti dell'architettura. Si noti che ognuno degli agenti può completare il lavoro in modo relativamente autonomo. Questi agenti possono essere messi in un computer o in diversi computer distribuiti in una rete. Inoltre, nuovi componenti possono essere facilmente aggiunti in WebTMS vista l'alta modularità.

1.5.1 Gathering agent

Tale agente, illustrato in figura 1.3, recupera i documenti, che possono essere distribuiti su parecchi Server Web, su cui effettuare il mining.

Tali documenti sono immagazzinati in opportuni repository del sistema. I documenti vengono raccolti a partire da una lista iniziale di URL immessi dall'utente.

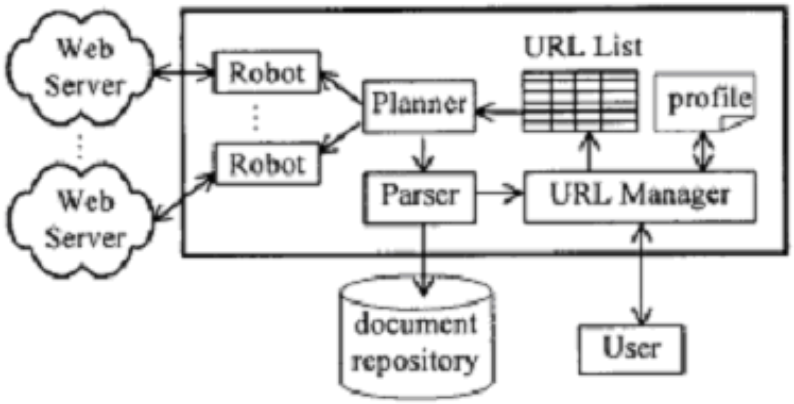

Figure 1.3: Il Gathering Agent

1.5.2 Prepocessing Agent

Nella figura 1.4 è rappresentato lo schema del Preprocessing Agent.

A partire da regole euristiche, vengono estratti metadati che poi vengono utilizzati per dedurre una rappresentazione del documento.

I metadati e la rappresentazione dei documenti sono tutti memorizzati, come dati strutturati nel database delle caratteristiche, per un uso futuro nel Text Mining.

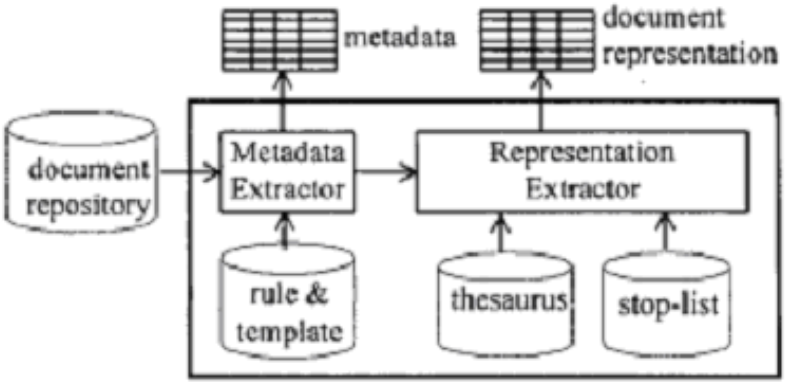

Figure 1.4: Il Preprocessing Agent

1.5.3 Categorization Agent

L'agente è mostrato in figura 1.5.

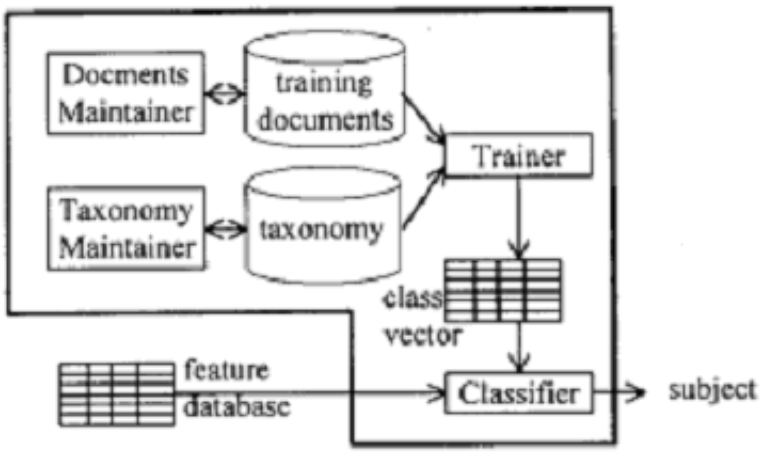

Figure 1.5: Il Categorization Agent

A partire da una tassonomia predefinita vengono divisi i documenti in classi, viene in altri termini realizzata una categorizzazione.

1.5.4 Clustering Agent

L'agente è raffigurato in figura 1.6. A differenza del *categorization agent*, tale agente non fa uso di una tassonomia predefinita per clusterizzare i documenti.

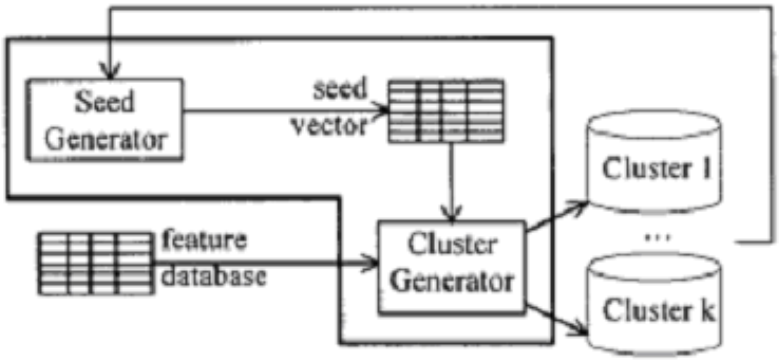

Figure 1.6: Il Clustering Agent

L'obiettivo è di dividere una raccolta di documenti in un insieme di cluster tali che è massimizzata la somiglianza intra-cluster e minimizzata quella inter-cluster.

1.5.5 MDDAE (Multi-Dimension Document Analysis Engine)

WebTMS usa una tecnologia di analisi multidimensionale basata su un *document cube*, per fornire viste multi-dimensione dei documenti per gli utenti.

Definition 1.5.1 (Document cube). Dato un documento D da analizzare, si definisce *"document cube"* la tupla:

$$CD = (d_1, d_2, ..., d_m, D)$$

dove D è il centro attorno al quale i metadati (di dimensioni d_i) girano.

Un esempio di document cube è mostrato in figura 1.7.

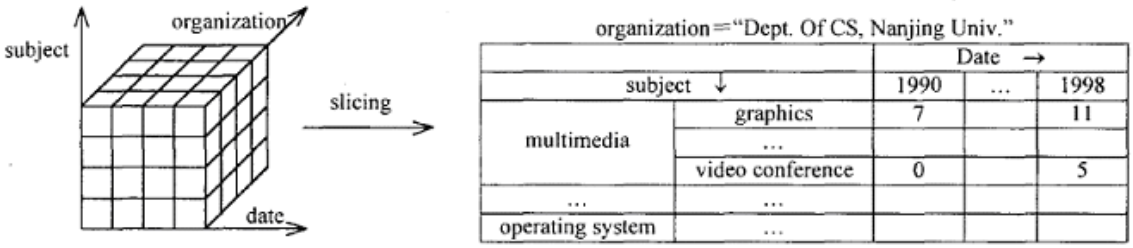

Figure 1.7: Un esempio di Document Cube

Si possono effettuare operazioni di slicing, dicing, rotation, drill down, roll up per dare all'utente diversi punti di vista del documento.

MDDAE offre anche funzionalità di analisi statistica per rivelare le distribuzioni o le tendenze nei documenti.

1.5.6 User Interface Agent (UIA)

È un'interfaccia visuale tra gli utenti e il MDDAE. Esso traduce le richieste degli utenti in un linguaggio specializzato e le passa ai moduli MDDAE, Document Categorizazion Agent o Document Clustering Agent, e poi mostra le viste dei documenti multi-dimensione e i documenti ritornati per gli utenti.

1.6 Il meta-motore GSA

GSA (**Global Search Agent**) [1] è un meta-motore di ricerca semi-adattativo sviluppato dal *DEIS* dell'Università della Calabria congiuntamente con le società

Pitagora S.p.A. e CM Sistemi Sud S.r.l.

GSA è un'applicazione stand-alone che offre un'interfaccia in cui l'utente può sottoporre una query specificando un set di keywords. GSA sottopone la stessa query, opportunamente adattata, ad un certo numero di motori di ricerca, mettendone insieme i risultati restituiti e dando a ciascuno di essi un punteggio (*rank*).

1.6.1 L'architettura di GSA

L'architettura di GSA è mostrata in figura 1.8.

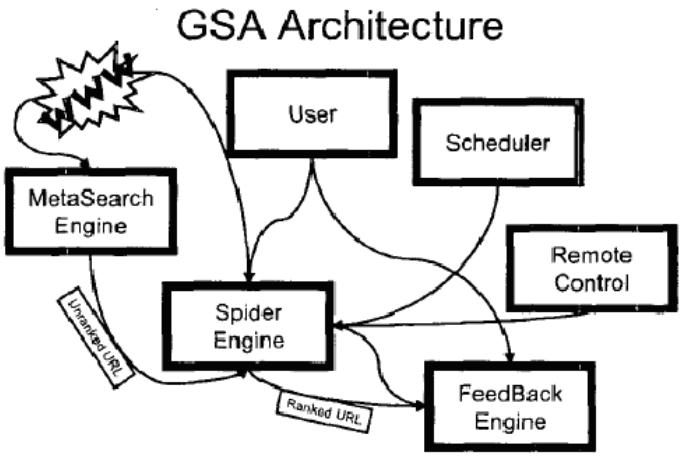

Figure 1.8: L'architettura del sistema GSA

Il sistema inizia la propria attività quando viene sottoposta una query dall'utente o dallo scheduler o da un'istanza remota di GSA. Contemporaneamente può essere dato un URL di partenza. GSA attiva lo *Spider Engine (SE)* ed il *Metasearch Engine (ME)*.

I due sottosistemi lavorano in parallelo: SE si occuperà dello spidering a partire dall'URL dato dall'utente e ME si occuperà di sottoporre ai motori di ricerca le parole

chiave inserite dall'utente.

ME estrae i singoli risultati dai motori di ricerca appena essi sono disponibili e chiede a SE di iniziare lo spidering delle pagine relative agli URL restituiti dai motori di ricerca. SE si occupa dello spidering e del parsing degli URL e di assegnare ad ogni pagina un punteggio (rank). Inoltre, SE decide se l'URL è abbastanza interessante così da mostrare soltanto gli URL che sembrano rilevanti per l'utente e decide se è il caso di effettuare ulteriori operazioni di spidering di pagine a profondità maggiore.

La ricerca termina quando ME ed SE non hanno più documenti da analizzare, cioè quando non occorre più effettuare alcuna operazione di spidering.

Il *Feedback Engine (FE)* lavora off-line. L'utente può specificare la sua opinione su quali sono i documenti effettivamente di suo interesse e quali non lo sono. FE a questo punto restituisce un set di parole rilevanti così che l'utente possa rifinire la ricerca.

1.6.2 Il ranking delle pagine

GSA utilizza una funzione di ranking molto interessante, che ha dato buoni risultati in fase di sperimentazione, pertanto è utile illustrarla, anche perché costituisce un buon confronto con la metrica che sarà definita più avanti per il sistema sviluppato.

Siano:

- **D** un documento
- **W** l'insieme delle parole chiave
- **w** = card(W)
- $\mathbf{K_0}$ e $\mathbf{K_1}$ due costanti (il cui ruolo sarà chiarito in seguito)

- $\mathbf{K_2}$, $\mathbf{K_3}$ e $\mathbf{K_4}$ opportuni pesi dei tre contributi specificati in seguito
- $\mathbf{K_5}$ la massima distanza (in parole) considerata significativa per due occorrenze di termini in W
- $\mathbf{N_p}$ la somma dei valori di *presenza* di ogni termine (per *presenza* si intende la massima *somiglianza* trovata per un certo termine nel testo considerato, così realizzando una particolare tecnica di stemming)
- $\mathbf{N_t}$ la somma delle occorrenze delle parole di W, ognuna pesata con il corrispondente valore di *somiglianza*
- $\mathbf{N_s}$ il numero di parole di W con un significativo valore di *presenza*
- **d(i, j)** la minima distanza trovata tra due occorrenze significative delle parole i e j

Si calcolino le seguenti quantità:

1. $\mathbf{b_0} = K_2 \cdot \frac{N_p}{w}$
2. $\mathbf{f_0} = K_3 \cdot \frac{N_t}{w}$
3. $\mathbf{d_0} = K_4 \cdot \frac{K_5 - \frac{\sum_{i=0}^{N_s-1} \sum_{j=i+1}^{N_s} min(d(i,j),k_5)}{\sum_{k=0}^{N_s-1}(N_p-k)}}{K_5}$
4. $\mathbf{f} = b_0 + f_0 + d_0$

Ed infine la funzione di ranking relativa al documento D ed al set di keywords W viene calcolata come segue:

$$\mathbf{r(D,W)} = k_0 \cdot \frac{f}{f + k_1}$$

Le costanti k_0 e k_1 hanno l'unico scopo di controllare la forma della funzione di ranking. Osserviamo che la funzione di ranking di GSA:

- Incorpora una tecnica di *"stemming"*
- Esprime le distanze in parole e non in caratteri
- Assume valori compresi tra 0 e K_0

Lo **stemming** è il processo di trasformazione di una parola nella sua radice ("stem"). Per esempio, considerando la lingua inglese, i diversi termini *learns*, *learned*, *learning* si possono ricondurre alla stessa radice learn.

1.6.3 Come GSA impara dall'utente

L'utente di GSA può esprimere una preferenza di tipo booleano (interessante o non interessante) su ogni documento restituito dall'agente, oppure ignorarne alcuni; dopodiché può chiedere al sistema di tener conto delle sue preferenze.

GSA effettua un parsing dei documenti interessanti (*hot*) e non (*cold*) ed estrae due insiemi di vocaboli: un insieme *good* ed un insieme *bad*.

Per capire se un termine è *good* o *bad* viene applicata un'euristica, dopo aver eliminato le *stop-words* dai documenti, ossia parole molto comuni che non inficiano il contenuto semantico del documento.

L'idea alla base dell'euristica è una funzione di ranking delle singole parole e di considerare *good* soltanto i termini il cui *rank* supera una certa soglia.

Siano:

- $\mathbf{G}$ l'insieme dei documenti *hot*
- $\mathbf{B}$ l'insieme dei documenti *cold*
- $\mathbf{t}$ il termine di cui si vuole calcolare il rank
- $\mathbf{o_i}$ il numero di occorrenze del termine t nel documento i
- $\mathbf{W}$ l'insieme delle parole della query di partenza
- $\mathbf{w_k} \in W$ il k-esimo termine della query di partenza
- $\mathbf{D}(\mathbf{w_k}, \mathbf{t})$ la minima distanza (in parole) tra una occorrenza significativa del termine w_k ed il termine t
- $\mathbf{w = card(W)}$

- $\mathbf{g = card(G)}$
- $\mathbf{b = card(B)}$

Si calcolino le seguenti quantità:

1. $\mathbf{O_i} = \min_{k=1}^{w}(d(w_k, t) \cdot \sum_{j=0}^{O_i - 1}(\frac{1}{2^j})$
2. $\mathbf{r} = \sum_{i=1}^{g} \frac{O_i}{g} - \sum_{i=1}^{b} \frac{O_i}{b}$

in cui r è il rank del termine t. Ogni termine incrementa il suo punteggio se appare in un documento *hot* e decrementa il suo punteggio se appare in un documento *cold*. Altre occorrenze di un termine in un documento oltre la prima non alterano più di tanto il valore di r.

1.7 Il motore di ricerca Mondou, basato sul Text Mining

Il motore di ricerca giapponese **Mondou** [39] è stato sviluppato dal *Department of Systems Science* presso la Graduate School of Informatics dell'Università di Kyoto.

Nonostante si tratti di un motore di ricerca di prima generazione, le tecniche di Data Mining sono adottate sin dal 1995. Una delle tecniche più importanti utilizzate dal motore è quella basata sulle regole d'associazione.

Il motore dà agli utenti una lista di *associative keywords*, che costituiscono un feedback per l'utente mediante il quale può raffinare la sua ricerca.

1.7.1 L'architettura di Mondou

Mondou è costituito da tre moduli fondamentali: il *Database*, il *Query Server* ed i *Web Robots*.

In figura 1.9 è mostrata l'architettura del sistema.

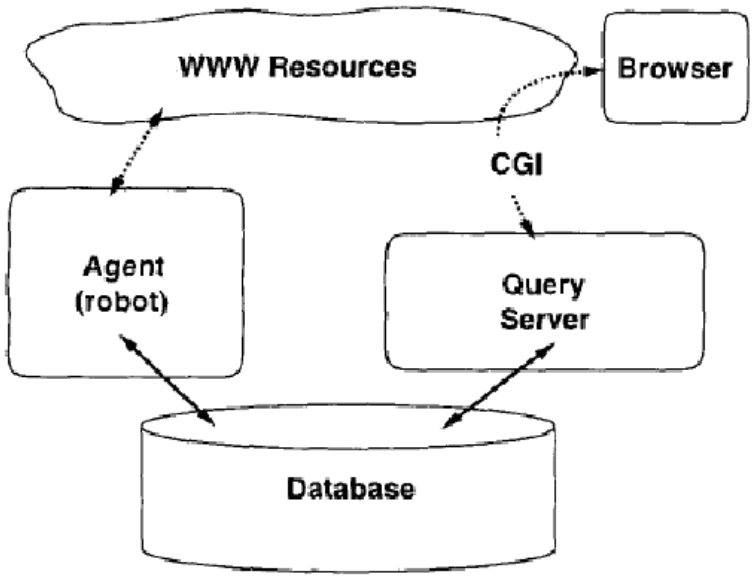

Figure 1.9: L'architettura del sistema Mondou

Il *modulo Database* non memorizza semplicemente il testo delle pagine Web, ma anche le relazioni tra i link.

Il *Query Server* è il modulo CGI più importante e si occupa di restituire i risultati delle ricerche, le associative keywords e i valori degli attributi per l'interfaccia grafica.

Il *modulo Web Robot* colleziona i dati dai server Web su Internet o su Intranet e li memorizza nel database. Oltre alle solite funzioni dei normali Web Robots, questo robot effettua il parsing dei documenti e sceglie le parole chiave più importanti mediante operazioni euristiche basate su analisi testuali del linguaggio naturale e dà

pesi appropriati ai dati.

1.7.2 Il data mining nel motore di ricerca Mondou

Mondou applica molti algoritmi di data mining per rendere più efficace la ricerca di informazioni sul WEB. Il più importante è quello volto a trovare le *associative keywords*.

Definition 1.7.1. *Dato un set di documenti, ognuno costituito da un certo numero di* keywords*, una* ***regola d'associazione*** *è un'espressione*

$$X \rightarrow Y$$

con X e Y insiemi di keywords.

Il significato intuitivo di un tal tipo di regola è che i documenti che contengono le keywords X tendono a contenere anche le Y. X si dice *testa* della regola o *premessa*; Y si dice *corpo* della regola o *conseguenza* [40].

La validità di una regola d'associazione è misurabile mediante due quantità:

- **Supporto**: è la probabilità che in una osservazione siano presenti sia la testa sia il corpo di una regola
- **Confidenza**: è la probabilità che in una osservazione sia presente la testa di una regola, essendo già presente il suo corpo.

Intuitivamente, il supporto misura l'importanza di una regola (quanto spesso premessa e conseguenza sono presenti contemporaneamente) mentre la confidenza ne

misura l'affidabilità (quante volte, essendo presente la premessa, è presente anche la conseguenza).

Keywords	**Number of URLs**	**Related Keywords**
Applications	975	windows, speech, computer, internet, helper, commercial, user, audio, network
Informatics	462	genome, medical, departement, center, school, faculty
Japan	20,366	japan, country, webec, japan, japanese, back
Engine	2,836	lycos, yahoo, dragon, search, creative, japan
Disney	183	tokyo, disney land, walt, world, parks, records
Mining	134	akita, geology, college, net, data
Mondou	28	rcaau, search

Table 1.1: Esempi di *associative keywords*

Mondou estende l'algoritmo di ricerca delle regole di associazione per considerare la frequenza e l'importanza delle keywords che appaiono nei documenti Web. Il Web Robot di Mondou dà l'appropriato valore di peso alle keywords in base alla frequenza ed al tipo di tag html. Esempi di associative keywords sono mostrati in Tabella 1.1.

In particolare, Mondou suddivide i tipi di attributi in tre categorie:

- **Characteristic** include keywords che mostrano le caratteristiche fondamentali di un documento e che sono specificate dall'autore o dall'editore del documento stesso, come ad esempio titolo e keywords

- **Description** include frasi che descrivono il contenuto, come ad esempio sommario, titolo, etc.
- **Supplement** include informazioni addizionali, come ad esempio autore, editore, dimensioni, data, etc.

Category	**Keyword space**	**INSPEC**	**Average**
Characteristics	Small	Title	11
		Keywords	30
Description	Small $\sim$ Large	Abstract	183
Supplement	Small	Author	3

Table 1.2: Le dimensioni degli spazi delle keywords

Come mostrato dalla tabella 1.2, lo spazio delle keywords delle categorie *Characteristic* e *Supplement* include un numero di keywords troppo piccolo per poter effettuare significative operazioni di mining basate sulle regole d'associazione.

La soluzione proposta dai ricercatori giapponesi è quella di estendere l'algoritmo base aumentando il numero delle keywords introducendo relazioni tra attributi differenti, così come mostrato in figura 1.10.

Ad esempio, si introduce una relazione tra il titolo A_u e l'autore A_v per espandere le keywords del titolo, considerando titoli di altri documenti dello stesso autore.

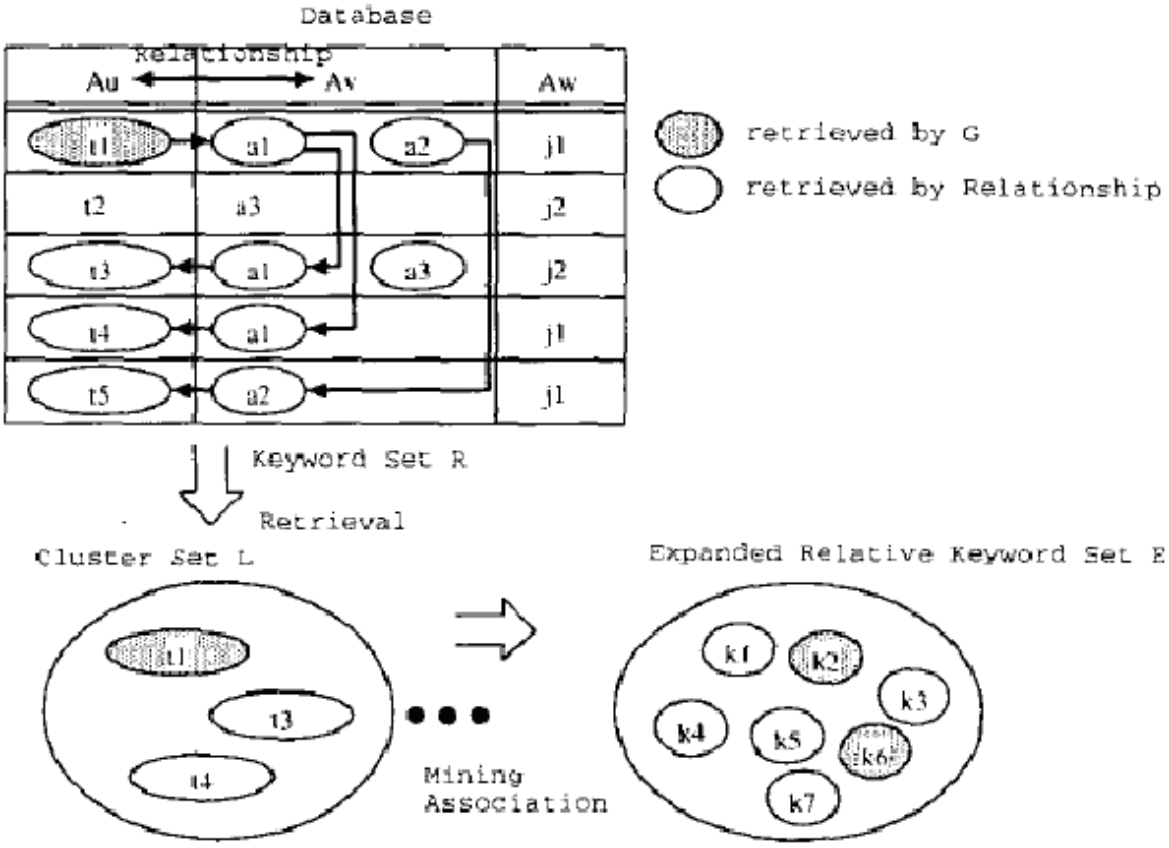

Figure 1.10: Relazioni tra attributi differenti

1.8 Il progetto Harvest

Scopo del progetto **Harvest Information Discovery and Access System** [2][41] non è soltanto il reperimento dei dati, ma anche il filtraggio degli stessi: filtrare enormi masse di dati per trarne informazioni concise riguardo il loro contenuto. Harvest è stato annunciato nel novembre del 1994 da un team di ricercatori universitari e del mondo dell'industria e fu rilasciato nell'agosto del 1996.

Lo scopo primario del progetto era di facilitare l'accesso alle diverse risorse di informazioni disponibili su Internet. L'idea di base era un'indicizzazione delle risorse disponibili sul WEB, dando la possibilità di cercare negli indici in modo flessibile e replicare tali indici su Internet così da ridurre il traffico ridondante di informazioni inutili sulla rete e facilitare il reperimento delle sole informazioni di interesse dell'utente.

Alcuni prodotti commerciali hanno fatto uso dei tool di Harvest, soprattutto in virtù della sua buona abilità nel caching delle pagine. Alcuni esempi sono il *Netscape Catalogue Server*, oppure *Glimpse*, un'applicazione per Unix per la creazione di motori di ricerca.

Il progetto, però, nasceva con ambizioni ben più grandi ed uno degli auspici dei suoi promotori era quello di essere utilizzato in larga scala dai maggiori motori di ricerca. Ciò non avvenne principalmente a causa del fatto che Harvest fu rilasciato troppo tardi, quando i motori di ricerca più importanti già esistevano ed avevano la propria tecnologia e non erano ben disposti a cambiarla in modo così radicale.

Chapter 2

Approccio teorico

2.1 Gli Agenti Software

Nei decenni passati gli sviluppatori di software hanno realizzato un'ingente quantità di programmi software monolitici caratterizzati dal fatto che ciascuno di essi realizzava un ampio e variegato insieme di task. Negli ultimi anni, al contrario, c'è stato un cambiamento nella modalità di sviluppo: invece di programmi con milioni di linee di codice, ci si è orientati verso pezzi di codice più piccoli e modulari, dove ogni modulo realizza un task ben definito (o, al limite, un insieme limitato di task) piuttosto che migliaia di task differenti, come si era soliti fare, per esempio, nei legacy systems.

Gli agenti software rappresentano l'ultimo ritrovato tecnologico in questa tendenza a suddividere sistemi software complessi in componenti. Un agente software è un "pezzo" di software avente le seguenti caratteristiche [5]:

- È in grado di agire in modo tale da portare a termine task per conto degli utenti
- Mette a disposizione uno o più servizi che altri agenti possono usare sotto certe condizioni

- Include una descrizione dei servizi offerti
- È in grado di agire autonomamente senza eventuali indicazioni da parte di un essere umano
- Descrive in che modo un agente decide quale azione intraprendere (anche se questa descrizione può essere tenuta nascosta agli altri agenti)
- È in grado di interagire con altri agenti (incluso esseri umani)

È da notare che non tutti gli agenti software debbono possedere le proprietà menzionate anche se ogni paradigma di programmazione basata su agenti dovrebbe dare la possibilità di sviluppare agenti con tutte o, eventualmente, alcune di queste proprietà.

2.1.1 Tipologie di Agenti

In questo paragrafo si tenterà di collocare i vari agenti esistenti in diverse classi in modo tale da poter definire una precisa tipologia degli agenti software [2] .

In primo luogo, gli agenti possono essere classificati in base alla loro mobilità (intesa come capacità di "muoversi" su una qualche rete): in base a ciò gli agenti possono suddividersi in **Statici** o **Mobili**.

In secondo luogo, possono suddividersi in **Deliberativi** o **Reattivi**. I primi derivano dal paradigma di pensiero deliberativo: posseggono al loro interno un modello di ragionamento simbolico e sono in grado di agire, pianificare e negoziare in modo tale da coordinarsi con altri agenti. Gli agenti di tipo reattivo, al contrario, non posseggono alcun modello simbolico dell' ambiente in cui si trovano e agiscono sulla base di un modello di comportamento stimolo/risposta, ossia rispondendo allo stato presente dell'ambiente in cui sono collocati [6].

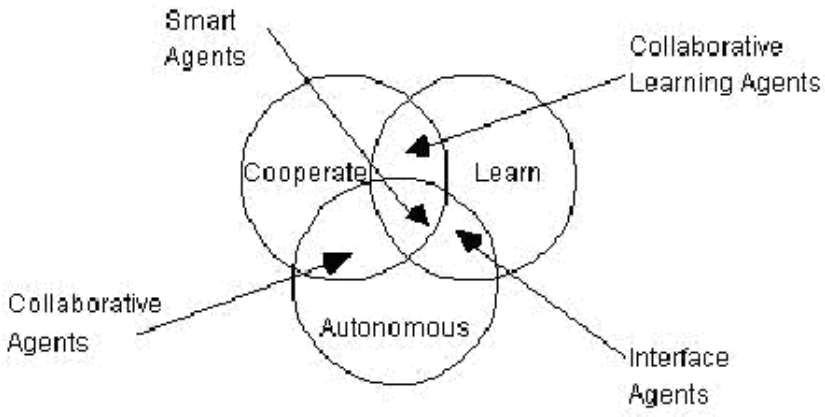

Figure 2.1: Una tipologia di agenti

In terzo luogo, gli agenti possono essere classificati in base ad alcuni fondamentali attributi che essi dovrebbero possedere. Tre sono gli attributi principali: *Autonomia*, *Apprendimento* e *Cooperazione*.

Con il termine **autonomia** si intende la capacità degli agenti di agire senza la guida dell'uomo. Tali agenti posseggono degli stati interni e degli obiettivi e agiscono in modo tale da realizzare i loro scopi nell'interesse dei propri utenti. Elemento chiave dell'autonomia è la cosidetta *"proactiveness"*, ossia la capacità di prendere iniziative piuttosto che agire semplicemente in risposta agli stimoli dell'ambiente circostante [7].

Con il termine **cooperazione**, si intende la capacità di interagire con altri agenti. È un attributo fondamentale per poter avere agenti multipli piuttosto che uno solo che lavora su un determinato task. Per poter cooperare, gli agenti devono possedere quella che viene definita capacità sociale ossia la capacità di interagire con altri agenti o esseri umani tramite qualche linguaggio di comunicazione.

Ultimo attributo è l'**apprendimento**; con esso si intende la capacità di un agente di imparare come esso reagisce e/o interagisce con l'ambiente esterno.

Sulla base di tali caratteristiche si può derivare una prima tipologia di agenti. Essi sono rappresentati nella figura 2.1 e sono[1]:

1. **Collaborative agents**
2. **Collaborative learning agents**
3. **Interface agents**
4. **Smart agents**

In quarto luogo, gli agenti sono talvolta classificati sulla base dei ruoli svolti; per esempio, si può parlare di **World Wide Web Information Agents** intendendo così agenti che permettono di gestire le informazioni presenti su Internet, oppure di **Testing Agents** intendendo agenti che facilitano il testing del software. Altri tipi di agenti sono i **Presentation Agents** e gli **Analysis and Design Agents**, tutti agenti classificati in base ai compiti svolti.

Un' ultima tipologia di agenti sono gli **agenti ibridi** i quali uniscono due o più diverse caratteristiche in un singolo agente.

Oltre agli attributi menzionati, ce ne sono altri che vengono però considerati secondari ai fini della determinazione di una tipologia di agenti, quali:

- **Versatilità**, ossia la capacità di avere più obiettivi
- **Mendacità**, ossia la capacità di essere o no sempre veritieri

[1]Tali agenti verranno spiegati nel dettaglio successivamente

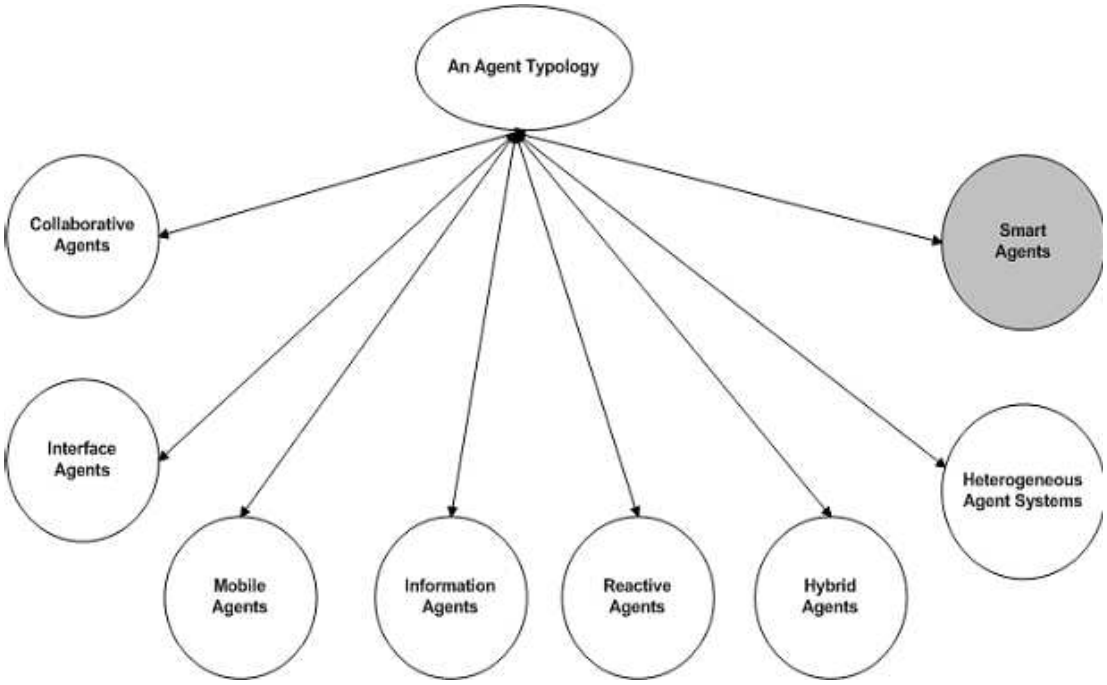

Figure 2.2: Classificazione di Agenti Software

Sulla base di quanto esposto, è possibile identificare sette tipi di agenti, i quali verranno spiegati nel dettaglio nei paragrafi successivi:

1. **Collaborative agents**
2. **Interface agents**
3. **Mobile agents**
4. **Information/Internet agents**
5. **Reactive agents**
6. **Hybrid agents**

7. **Heterogeneous agents system**

8. **Smart agents**

Si noti che rispetto alla figura 2.2 non sono presenti i *Collaborative Learning Agents*. Ciò è motivato dal fatto che non si è ancora a conoscenza di agenti che collaborano e apprendono, ma non sono autonomi.

C'è da dire che esistono diverse applicazioni che combinano agenti appartenenti a due o più categorie; a tali applicazioni ci si riferisce con il termine *Sistemi ad Agenti Eterogenei* (*heterogenous agent systems*). Tali applicazioni, però, non sono così diffuse come quelle elencate precedentemente.

Un'altra nota è che gli agenti non devono essere necessariamente benevoli gli uni con gli altri; infatti, è possibile avere agenti in competizione tra loro o, addirittura, antagonisti. Comunque, si può vedere agli agenti competitivi come ad una potenziale sottoclasse di quelli visti. In tal modo è possibile avere *Collaborative Agents* di tipo competitivo, *Interface Agents* di tipo competitivo e via dicendo.

Collaborative Agents

Come mostrato in figura 2.1, tali agenti pongono l'accento sulle caratteristiche di autonomia e cooperazione (con altri agenti) al fine di realizzare task per gli utenti [8].

Essi possono eventualmente anche apprendere ma ciò non è un aspetto fondamentale. Al fine di collaborare, possono avere necessità di effettuare negoziazioni per accordarsi su diversi aspetti dei task in cui sono coinvolti.

Alcuni ricercatori nel campo dell'*Intelligenza Artificiale* (AI) stanno tentando di

fornire delle definizioni più adeguate per tali agenti ponendo l'accento su alcuni attributi di tipo mentale come convinzioni, desideri e intenzioni (beliefs, desires e intentions, da cui l'acronimo di collaborative agents di tipo BDI).

Brevemente, le caratteristiche fondamentali di tali agenti sono *autonomia*, *capacità sociale*, *comprensione* e *proactiveness*. In tal modo sono in grado di agire in maniera razionale ed autonoma in un ambiente multi-agente. In genere sono statici e possono essere benevoli e veritieri o una qualche combinazione di essi.

Ci sono diversi motivi che possono portare allo sviluppo di sistemi basati su tale tipo di agenti:

- Risolvere problemi troppo complessi da realizzare tramite un sistema basato su un singolo agente centralizzato
- Permettere l'interconnessione di legacy system esistenti come sistemi esperti o sistemi di supporto alle decisioni
- Risolvere problemi di natura distribuita come sistemi di sensori distribuiti o di controllo del traffico aereo [9]
- Affrontare problemi con sorgenti di informazione distribuite
- Ottenere modularità (che riduce la complessità), velocità (dovuta al parallelismo), affidabilità (dovuta alla ridondanza), flessibilità (nuovi task possono essere affrontati tramite nuovi agenti).

Un esempio di sistema basato su agenti di tipo collaborativo è quello sviluppato presso il CMU: il **Pleiades Project** [10][11]. Tale sistema applica tale tipo di agenti

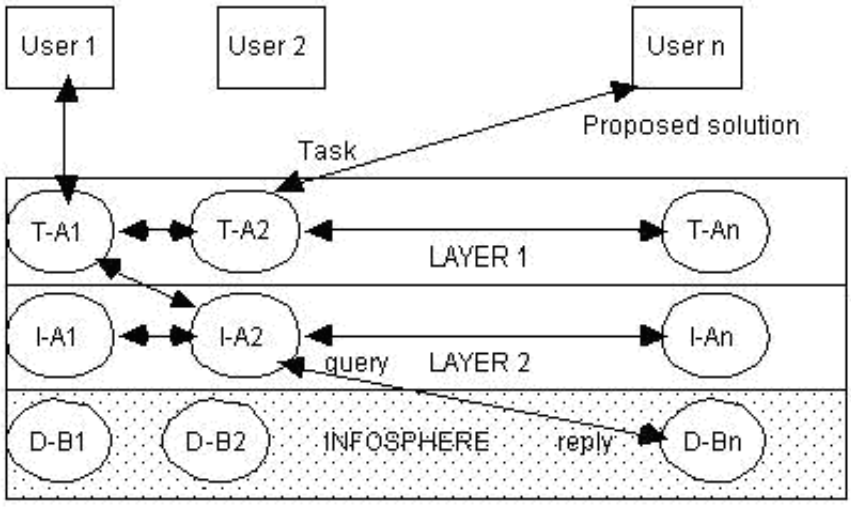

Figure 2.3: Architettura Pleiades

al problema di effettuare decisioni di tipo organizzativo. Esso è raffigurato in figura 2.3 e si basa su un'architettura ad agenti a due livelli: il primo livello contiene i *Tasks-Specific Collaborative Agents*, che realizzano specifici task come organizzare appuntamenti e incontri.

I Tasks-Specific Collaborative Agents collaborano tra di loro per risolvere conflitti e integrare informazioni. Per ottenere le informazioni necessarie, essi le richiedono agli agenti di secondo livello: gli *Information-Specific Agents*. Anch'essi possono collaborare tra loro ed attingono informazioni da specifici database di sistema. Alla fine, i Tasks-Specific Collaborative Agents propongono soluzioni ai diversi utilizzatori del sistema.

Sebbene ci siano altri esempi di sistemi basati su agenti di tipo collaborativo, questi ultimi sono stati poco sviluppati anche se la situazione sta mutando, come dimostrano i sistemi di controllo dei malfunzionamenti dello Space Shuttle o i simulatori di volo

usati presso la Royal Australian Airforce [12]. Ci sono, infatti, alcuni problemi che si sta cercando di risolvere. Se ne possono menzionare alcuni anche se essi non sono tipici solamente di tale categoria di agenti:

- Cercare di ingegnerizzare lo sviluppo dei sistemi [13]
- Migliorare la coordinazione Inter-Agenti; la coordinazione è essenziale affinchè gruppi di agenti risolvano in maniera corretta i problemi. Senza un'adeguata teoria sulla coordinazione, c'è il rischio che anarchia o arresti non desiderati prendano il sopravvento [14]
- Stabilire eventuali tecniche di apprendimento che non portino all'instabilità dei sistemi
- Stabilire tecniche e metodologie per integrare agenti e legacy system
- Stabilire dei metodi per la verifica e la validazione dei sistemi che permettano di capire se le specifiche funzionali sono rispettate

In conclusione si può affermare che, nonostante i problemi ancora da risolvere, sono molti i campi di interesse in cui possono trovare applicazione tali tipi di agenti.

Interface Agents

Gli **Interface Agents**, come si nota nella figura 2.1, pongono l'accento sulle caratteristiche di autonomia e apprendimento al fine di portare a termine i task per i loro utilizzatori.

Pattie Maes, che rappresenta uno dei maggiori ricercatori in questo campo, ha evidenziato quella che è la metafora alla base di tali agenti: *un assistente personale*

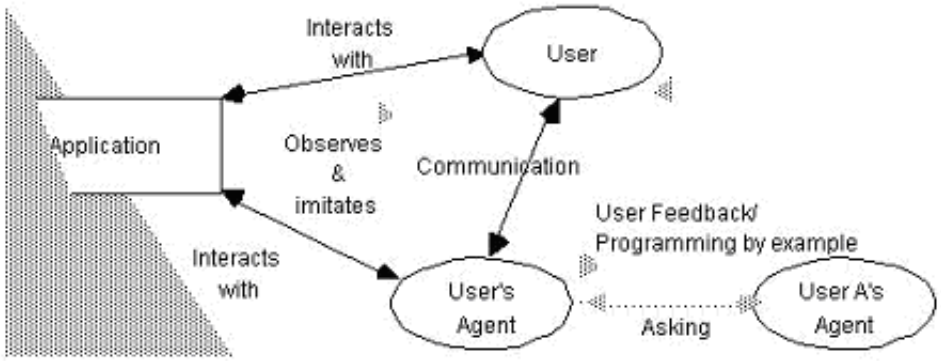

Figure 2.4: Interface Agents

che collabora con l'utente nello stesso ambiente di lavoro [15]. Da ciò si evince la sottile differenza tra questo tipo di agenti e quelli descritti nel paragrafo precedente: questi ultimi collaborano tra di loro mentre gli Interface Agents collaborano esclusivamente con gli utenti. In figura 2.4 viene mostrata la modalità di funzionamento di tali agenti.

In sintesi, tali agenti forniscono generalmente assistenza ad un utente che sta utilizzando una particolare applicazione (per esempio, fogli di calcolo o sistemi operativi). Tali agenti monitorano le azioni intraprese dagli utenti, apprendono nuovi shortcuts e suggeriscono in tal modo le migliori modalità per portare a termine un lavoro. Perciò tali agenti possono essere visti come assistenti personali che collaborano con l'utente nella realizzazione dei task in un'applicazione.

Tali tipi di agenti posseggono la capacità di apprendere al fine di assistere meglio gli utenti; ciò viene fatto in 4 modi [15]:

1. Osservando e imitando gli utenti

2. Ricevendo feedback positivi e negativi dagli utenti

3. Ricevendo esplicite istruzioni dagli utenti

4. Chiedendo ad altri agenti

Riassumendo, si può affermare che, sotto certe ipotesi, un tale tipo di agente è in grado di programmarsi ossia di acquisire la conoscenza necessaria per assistere gli utenti. L'agente è dotato di una conoscenza minima che può aumentare osservando gli utenti oppure chiedendo ad altri agenti [15].

La cooperazione con altri agenti è limitata esclusivamente alla richiesta di consigli e non ad eventuali negoziazioni al fine di collaborare al compimento di un task (come si verificava nei Collaborative Agents).

L'osservazione alla base dello sviluppo di tali agenti, è che le interfacce della maggior parte dei sistemi sono di tipo passivo, ossia eseguono compiti solamente quando sono sollecitate. Con lo sviluppo di tali agenti si cerca di avere, invece di un utente che impartisce comandi ad un'interfaccia, una modalità di funzionamento basata sulla collaborazione uomo-agente, in cui entrambi possono iniziare una comunicazione con l'altro, realizzare task e monitorare eventi. La chiave, quindi, è questa cooperazione tra l'uomo e l'agente [16]. Pertanto, l'obiettivo è la migrazione da una metafora basata sulla diretta manipolazione ad una che deleghi alcuni task a degli Interface Agents magari per aiutare utenti alle prime armi.

Sono tre i principali benefici che derivano dall'utilizzo di tali agenti [15]. In primo luogo rendono meno faticoso il lavoro degli utenti in quanto task noiosi o faticosi possono essere delegati ad essi (per esempio la gestione delle mail vecchie, la schedulazione degli appuntamenti, la ricerca di informazioni da un grosso insieme). In secondo luogo, l'agente può adattarsi nel tempo alle abitudini e le preferenze dei propri utenti.

In ultimo, la conoscenza all'interno di una comunità di utenti può essere condivisa (ciò accade, per esempio, quando un agente acquisisce conoscenza da un suo pari). Ci sono diversi esempi di sistemi basati su tali agenti:

- *Calendar Agent* [51], sviluppato da Kozierok & Maes, che assiste gli utenti nella schedulazione degli appuntamenti apprendendo le loro preferenze
- *Letizia Agent* [**?**], sviluppato da Liebermann, che assiste gli utenti nella navigazione sul WEB effettuando ricerche di pagine Web basandosi sui comportamenti degli utenti
- *The Remembrance Agent* [19], sviluppato da Rhodes & Starner, che restituisce, durante la scrittura di una e-mail, le cinque e-mail più attinenti a quella che si sta scrivendo contenute nel proprio archivio
- *The Kasbah Agent* [20], sviluppato da Chavez & Maes, che filtra gli annunci pubblicitari sul WEB e li seleziona in base agli interessi degli utenti

Concludiamo il paragrafo indicando quelli che sono i principali oggetti della ricerca nel campo degli Interface Agents:

- Dimostrare che la conoscenza acquisita tramite tali agenti sia davvero utile a ridurre il lavoro degli utenti e dimostrare, invece, che tali utenti, davvero li vogliano come collaboratori
- Portare a compimento centinaia di esperimenti usando varie tecniche di apprendimento per capire quale sia la migliore su un certo dominio e perchè

- Analizzare gli effetti delle diverse tecniche di apprendimento sulle risposte fornite dagli agenti agli utenti
- Dare la possibilità a tali agenti di negoziare con altri alla pari
- Estendere il campo di applicazione di tali agenti in nuove aree innovative come l'intrattenimento, come gli agenti ALIVE e HOMR stanno facendo

Al di là di tali osservazioni, non si può negare il fatto che gli Interface Agents sono destinati ad un continuo sviluppo e ad una continua applicazione poichè sono semplici, lavorano in domini limitati e, in generale, non richiedono cooperazione con altri agenti.

Mobile Agents

Gli **Agenti Mobili** [2] sono processi software in grado di muoversi all'interno di **WAN** (**Wide Area Networks**) oppure nel **WWW** (**World Wide Web**); sono capaci di interagire con altri hosts, di ottenere eventuali informazioni e di ritornare "alla base" dopo aver portato a termine i loro compiti.

Tali compiti possono variare, per esempio, dalla prenotazione di un volo alla gestione di una rete di telecomunicazioni. Comunque, la mobilità non è né una condizione necessaria né sufficiente per essere considerati agenti.

Gli Agenti Mobili sono agenti in quanto godono delle proprietà di autonomia e cooperazione, sebbene in maniera differente dai Collaborative Agents. Per esempio possono cooperare e comunicare con altri agenti rendendo loro noti la locazione di qualche oggetto interno oppure i loro metodi; nel fare ciò un agente scambia dei dati o qualche informazione con altri senza necessariamente fornire tutte le sue informazioni.

L'ipotesi base di questo tipo di agenti è che essi non devono essere stazionari: ciò, in certe applicazioni, può portare dei benefici di tipo non funzionale rispetto all'utilizzo di agenti statici.

Per esempio [21], si consideri il caso in cui si voglia scrivere un'applicazione che permetta da casa di poter prenotare un volo accedendo ai diversi database di sistema delle compagnie aeree nei quali si gestisce tale operazione. L'utente dovrebbe immettere dati quali: classe, fascia oraria di partenza, città di partenza e arrivo, numero di scali; un agente di tipo statico, per fare ciò, potrebbe richiedere ai vari database un elenco di tutti i voli compresi nella fascia oraria impostata e questo potrebbe costare in termini di kilobytes.

Tale agente potrebbe richiedere ancora altre informazioni quali una lista di tutti i collegamenti tra le due città impostate e procedere sino a restringere la ricerca. Tutte queste azioni coinvolgono una serie di operazioni spesso inutili che non fanno altro che affollare la rete, rendere più lunga l'operazione con eventuali costi di collegamento alla rete che aumentano.

Lo scenario opposto è quello che prevede l'uso di Agenti Mobili. L'utente potrebbe incapsulare, in uno stile orientato agli oggetti, l'intera applicazione in un agente che occuperebbe probabilmente meno di 2 kilobytes. Tale agente provvede a muoversi attraverso i sistemi di prenotazione dei voli delle varie compagnie, interroga localmente i loro database e ritorna all'utente una semplice scheda con il volo scelto che l'utente può accettare o rifiutare.

Si capisce come ciò possa ridurre i costi di collegamento e i kilobytes di informazioni inutili scaricati localmente. In sostanza, tali agenti offrono una serie di vantaggi di tipo pratico, sebbene non funzionale che li distinguono dalle loro controparti

di tipo statico. Tali vantaggi possono essere:

- Riduzione dei costi di comunicazione dovuta all'eliminazione di informazioni inutili da scaricare localmente. L'agente restituisce all'host locale solo i dati necessari selezionandoli direttamente sull'host remoto. Ciò è un grosso vantaggio nel caso la connessione alla rete venga pagata in base al tempo di utilizzo
- Possibilità di realizzare operazioni in modo asincrono: mentre l'agente lavora, l'utente può fare altre cose o, addirittura, spegnere il terminale ricevendo i risultati in un secondo momento nella propria mailbox
- Possibilità di superare i limiti dovuti a risorse locali limitate; infatti la capacità di calcolo e di immagazzinamento di una macchina può essere molto limitata ma ciò può essere superato tramite l'uso degli Agenti Mobili
- Possibilità di realizzare architetture distribuite molto potenti

Un'applicazione di tali agenti mobili è l'architettura **Telescript** [21][22] mostrata in figura 2.5.

Telescript è un linguaggio di programmazione interpretato, object-oriented e remoto per la scrittura di applicazioni distribuite. L'ambiente di sviluppo è rappresentato dal Telescript engine integrato tramite API al sistema operativo.

La primitiva più importante è rappresentata dalla "go"; essa permette la comunicazione tra i processi: due o più processi-agente possono comunicare per usufruire dei servizi dell'altro instaurando un canale di comunicazione.

La primitiva richiede uno spazio di destinazione ed il Telescript engine incapsula l'agente con tutti i suoi dati e lo invia; la destinazione può anche trovarsi su una

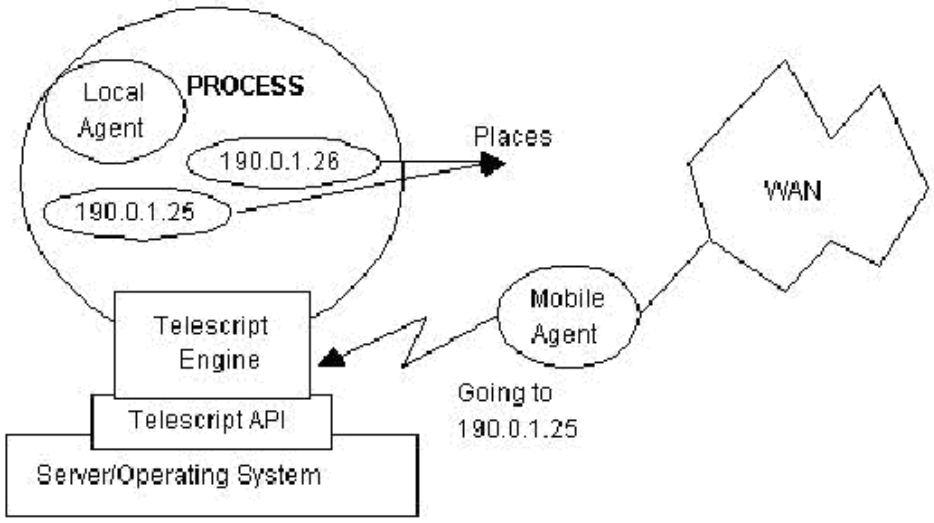

Figure 2.5: Architettura Telescript

WAN (come mostrato in figura). A destinazione, l'altro Telescript engine autentica l'agente permettendogli di eseguire i suoi task e al termine l'agente ritorna sull'host di partenza.

Concludiamo il paragrafo indicando quelli che sono i principali oggetti della ricerca nel campo dei Mobile Agents [23]:

- Stabilire dei metodi affidabili di autenticazione per stabilire se un agente che intende entrare in un sistema è davvero chi dice di essere
- Stabilire dei metodi sicuri per gestire la segretezza dei dati che viaggiano tramite tali agenti
- Stabilire dei metodi per la sicurezza di un sistema in cui accede un agente per evitare che questo vada in loop consumando cicli di CPU
- Stabilire degli standard in modo tale che un agente scritto in un certo linguaggio

di programmazione possa funzionare su un motore scritto in un altro

Si evince che uno degli argomenti di ricerca più importante è rappresentato dalla sicurezza. In molte aziende sono installati opportuni firewall che impediscono l'accesso a tali tipi di agenti, indipendentemente dal fatto che essi siano o meno pericolosi.

Altra tecnica per aumentare la sicurezza di chi deve accettare un agente mobile nel proprio sistema, è quella di impedire la realizzazione di agenti che possano scrivere nella memoria o sui dischi del sistema stesso.

Information/Internet Agents

Gli **Information Agents** [2] sono nati a causa della forte domanda di applicazioni per la gestione della sempre crescente mole di dati presente oggi in rete. Essi hanno il compito di gestire, manipolare e collezionare informazioni da sorgenti distribuite.

L'ipotesi alla base degli Information Agents è che la rete è sovraccaricata da dati e informazioni e, quindi, è richiesta una migliore e più efficiente gestione di essi.

Tom Henry, vice presidente dell'azienda SandPoint [24], ha asserito che *una delle sfide più affascinanti dell'informatica è quella di realizzare un sistema dove la richiesta e l'ottenimento di informazioni utili usando tali agenti sia naturale come usare il telefono o leggere un giornale.*

In teoria, usando tali agenti ed addestrandoli opportunamente, un utente potrebbe ricevere sul proprio terminale il suo giornale preferito all'ora stabilita oppure raccogliere in maniera personalizzata tutta una serie di informazioni utili all'interno di un dominio.

Sono due i principali motivi che spingono allo sviluppo di Information/Internet Agents; il primo, come già sottolineato, è che a causa della crescente mole di dati

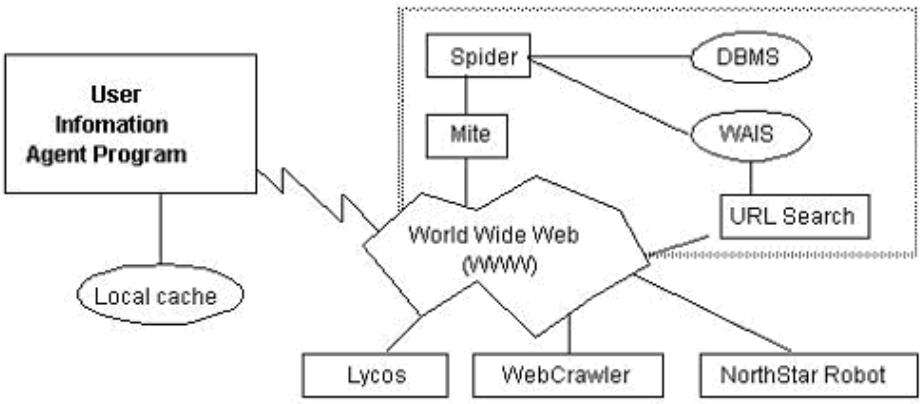

Figure 2.6: Information Agent

presenti in rete, si sente l'esigenza di avere applicazioni per gestire efficacemente questa crescente quantità.

Il secondo motivo è di natura economica: si ritiene che chi riuscirà a realizzare un sistema ad agenti di tipo proattivo, dinamico, adattativo e cooperativo per la gestione delle informazioni sul WWW, facilmente potrà arricchirsi poichè sempre maggiori sono le spese che vengono effettuate sul WEB. Soffermandoci sulle caratteristiche che tali agenti debbono possedere, si può affermare che essi in genere sono di tipo mobile, possono essere non cooperativi e possono o non apprendere.

Esistono anche casi di Information Agent di tipo non mobile, come evidenziato nella figura 2.6

L'Information Agent è integrato nel browser [24] e si appoggia a sistemi di spidering o a motori di ricerca per ottenere le informazioni desiderate.

Uno Spider è un sistema in grado di muoversi nel WEB e di memorizzare nel database di sistema la topologia dei siti visitati. L'agente, al fine di ottenere informazioni su un certo argomento, si interfaccia con i motori di ricerca o lo Spider

(oppure si interfaccia con la cache locale se una certa richiesta era già stata effettuata ed i risultati memorizzati).

Si possono citare diversi esempi di Information Agents sviluppati di recente. Etzioni & Weld [25] nel 1994 hanno sviluppato un sistema detto **Internet Softbot**: si tratta di un agente che permette di effettuare query complesse (con congiunzioni, disgiunzioni, quantificazioni) e che è capace di muoversi nel WWW al fine di inferire conoscenza e di soddisfare le richieste. È in grado di tollerare ambiguità, omissioni ed errori da parte dell'utente. È capace, inoltre, non solo di inferire conoscenza, ma anche di filtrare e-mail, schedulare appuntamenti e realizzare task di gestione del sistema su cui opera.

Un tale tipo di agente è classificato come Information Agent e non come Interface Agent poichè l'apprendimento non è un fattore cruciale, sebbene possa implementare tecniche di apprendimento elementari. Per esempio, è in grado di capire, impostata una query, se una simile è già stata sottoposta. Altro esempio è l'agente sviluppato da Davies e Weeks nel 1995 presso i BT Labs: il **Jasper Agent (Joint Access to Stored Pages with Easy Reatrieval)** [26]. Esso è in grado di immagazzinare, ritrovare e indicizzare pagine Web sulla base delle query dell'utente. In più, è capace di interagire con altri agenti presenti sul WEB al fine di migliorare le ricerche. Offre la possibilità di estrarre delle keywords da ogni pagina Web in modo tale da restituire queste pagine nel caso in cui un utente effettui delle ricerche usando le keywords estratte.

Per quanto concerne la ricerca sugli Information Agents, si può affermare che gli oggetti principali sono simili a quelli già trattati quando si è parlato degli Interface

Agents o dei Mobile Agents; infatti se l'agente è di tipo statico, ad esso si può applicare tutto quello già visto per gli Interface Agents (tecniche di apprendimento, negoziazione), mentre se è di tipo mobile si può applicare quello visto per i Mobile Agents (autenticazione, sicurezza, segretezza). Si tornerà in seguito su questi agenti, quando si parlerà più approfonditamente degli **Intelligent Search Agents (ISA)**.

Reactive Agents

Gli **Agenti Reattivi** [2] rappresentano una particolare categoria di agenti caratterizzati dal fatto che non posseggono al loro interno modelli simbolici del loro ambiente; al contrario, essi agiscono in uno stile sollecitazione/risposta allo stato presente dell'ambiente in cui si trovano. Si tratta di agenti in genere relativamente semplici ed in grado di interagire con altri in maniera elementare.

Tre sono le ipotesi principali alla base di tali agenti [27]; la prima è la *semplicità*, in quanto non c'è nessuna specificazione a priori del comportamento di un insieme di agenti reattivi.

La seconda è la possibilità di decomporre in task un'applicazione; infatti un agente reattivo è visto come una collezione di moduli ognuno operante in maniera autonoma e responsabile di specifici task (rilevazioni, computazioni). La comunicazione tra i moduli è ridotta al minimo e di basso livello.

La terza è che tali agenti lavorano su semplici modelli dell'ambiente circostante che sono quelli forniti, per esempio, da sensori in contrasto con modelli simbolici di alto livello adottati negli altri agenti discussi in precedenza.

L'elemento chiave del successo di tali agenti, è proprio questo nuovo modo di vedere l'ambiente circostante; infatti, i principali teorici nel campo dell'intelligenza

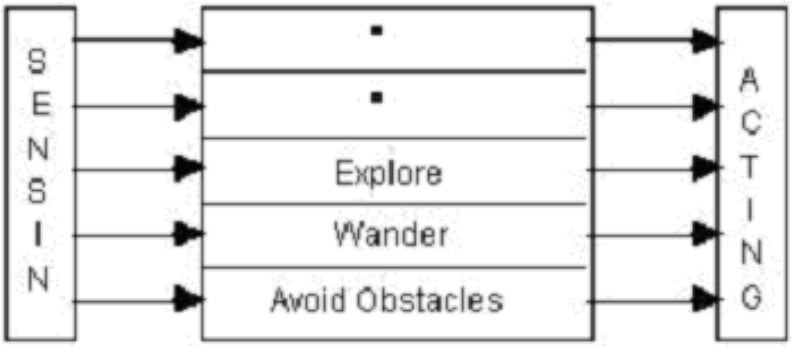

Figure 2.7: Brooks Subsumption Architecture

artificiale sostenevano che per costruire un sistema intelligente si dovesse fornire una rappresentazione simbolica del mondo, mentre ora, questa viene bandita insieme ai modelli di ragionamento simbolico [28]. L'ambiente, ora, viene visto "per quello che è" tramite una serie di sensori ed attuatori. In tal modo gli agenti reattivi si dimostrano semplici, facili da comprendere e con costi di memorizzazione ridotta dato che devono "ricordare" poco. Essi non programmano nulla in anticipo, ma le loro azioni dipendono da quello che accade in quel momento [6].

I benefici che spingono ad uno sviluppo di sistemi basati su agenti reattivi, oltre a quelli già menzionati, sono:

- Maggiore robustezza e tolleranza ai guasti: un agente può danneggiarsi senza effetti catastrofici

- Maggiore flessibilità e adattabilità, in contrasto alla rigidezza, agli alti tempi di risposta e all'instabilità dei classici sistemi di intelligenza artificiale

Un architettura basata su agenti di tipo reattivo è la **Brooks Subsumption Architecture** [29] raffigurata in figura 2.7.

Si tratta di un'architettura usata per costruire robot meccanici: essa è formata da una serie di moduli descritti come una macchina a stati finiti. Un modulo entra in azione se il suo segnale di input supera una soglia fissata. Tali moduli sono le uniche unità di processazione nell'architettura, in quanto non ci sono simboli come quelli usati nei classici sistemi di Intelligenza Artificiale.

I moduli sono organizzati a livelli e ciascun modulo ha uno scopo ben preciso (per esempio, evitare ostacoli o abilitare e controllare il movimento).

Un'applicazione interessante è quella della simulazione artificiale. Ferber [6] nel 1994 realizzò un'applicazione basata su agenti reattivi in grado di simulare colonie di formiche; ogni formica era modellata tramite un agente ed egli riuscì a simulare un piccolo ecosistema. Altro simulatore è quello realizzato da Nwana [30] nel 1993 dove si simulava un gruppo di bambini in un campo da gioco.

Perciò gli agenti reattivi possono far diventare il computer un "laboratorio virtuale" dove i ricercatori possono far variare molteplici parametri e validare i più svariati modelli.

Un altro campo di applicazione è rappresentato dall'industria dei giochi e dell'intrattenimento, come dimostrano le ricerche effettuate negli ultimi anni nei laboratori della Philips [31].

Concludiamo il paragrafo indicando quelli che sono gli oggetti principali di ricerca nel campo degli agenti reattivi:

- Cercare di ampliare la tipologia ed il numero di applicazioni basate su tali agenti in quanto attualmente esse sono concentrate principalmente ai simulatori e ai

giochi

- Sviluppare un'adeguata metodologia di progettazione delle applicazioni e, quindi, teorie, architetture e linguaggi
- Affrontare in maniera ingegneristica i problemi di scalabilità, estendibilità e valutazione delle performance

Comunque, a discapito di queste osservazioni finali, c'è da dire che ci si aspetta uno sviluppo notevole nei prossimi anni di applicazioni basate su questo tipo di agenti.

Hybrid Agents

Sino ad ora si sono analizzati cinque tipi di agenti: collaborativi, di interfaccia, mobili, di Internet e i reattivi; ampio è il dibattito su quale tra essi sia il migliore. Poichè ciascuno di essi ha i suoi pregi ed i suoi difetti, si è cercato di massimizzare i primi e ridurre i secondi adattando al meglio ciascuno di essi ai propri scopi.

Un modo per massimizzare i propri obiettivi è quello che comunemente viene denominato approccio ibrido [2] che cerca di unire i pregi sia degli agenti basati sul paradigma deliberativo (e quindi i collaborativi, quelli di interfaccia, i mobili e di internet) sia di quelli basati sul paradigma reattivo. Perciò, con il termine **Agente Ibrido** ci si riferisce ad agenti che fondono caratteristiche di due o più tipi di agenti in uno solo.

L'ipotesi alla base di tali agenti è che spesso, in diverse applicazioni, si possono ottenere performance migliori combinando diverse filosofie in un unico agente piuttosto che affidarsi ad un unico agente che si basi su una singola filosofia tra quelle già viste. Ovviamente, i benefici derivanti dall'uso di tale tipo di agenti nascono dall'unione dei benefici dei diversi tipi di agenti adottati. Per esempio, si supponga di realizzare un agente che unisca le caratteristiche di collaborazione (e, quindi, un agente di tipo deliberativo) e di reazione; la componente reattiva, che potrebbe avere la precedenza sull'altra, potrebbe portare i seguenti benefici: robustezza, adattabilità e tempi di risposta minori. La componente deliberativa potrebbe occuparsi degli scopi a lungo termine, aumentando la flessibilità del sistema. In genere i sistemi che si basano su un

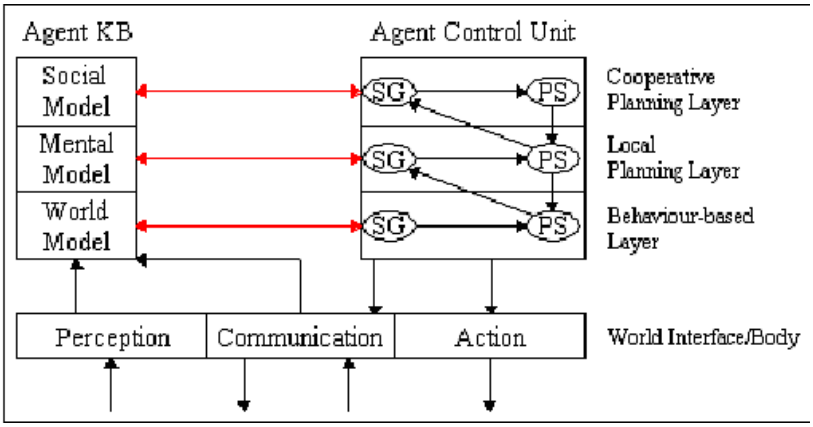

Figure 2.8: The InteRRap Hybrid Architecture

approccio ibrido presentano un'architettura a livelli; un esempio è il sistema sviluppato presso il Centro di Ricerca Tedesco sull'Intelligenza: **The InteRRaP Hybrid Architecture** [32][33]. Esso è rappresentato in figura 2.8. Si tratta di una struttura a livelli per la realizzazione di agenti e robot autonomi e che sposa la filosofia reattiva con quella deliberativa.

La parte reattiva (*Behaviour-based Layer - BBL*) del sistema è basata su agenti con regole di tipo stimolo/reazione e che contribuiscono alle proprietà di efficienza e robustezza. Il livello intermedio (*Local Planning Layer - LPL*) è costituito da agenti con comportamenti di tipo goal driven mentre l'ultimo livello (*Cooperative Planning Layer - CPL*) è costituito da agenti che permettono la cooperazione con altri in modo da realizzare una struttura multi-agente.

Un'altra architettura è la **Touring Machine** [34] sviluppata da Ferguson nel 1992; si tratta ancora di un sistema a livelli: un livello *reattivo*, uno di *pianificazione* ed un altro di *modellazione*.

Si tratta di un'architettura di tipo orizzontale, al contrario di quella precedentemente esposta che è di tipo verticale; infatti nella prima tutti i livelli hanno accesso ai sensori e possono contribuire all'azione, mentre nell'applicazione vista in precedenza solo il livello più basso riceve gli stimoli dai sensori e agisce.

Nonostante i molti vantaggi che può portare un'architettura basata su agenti ibridi, sono ancora relativamente poche quelle realmente progettate; infatti due delle critiche mosse è che un approccio di tal genere può portare a sistemi troppo specifici e poco generali e che l'approccio teorico e ingegneristico non è ben specificato in letteratura. Perciò, gli obiettivi dei ricercatori in questo campo sono molto simili a quelli dei ricercatori nel campo degli agenti di tipo reattivo.

Inoltre ci si aspetta di vedere sistemi ibridi che non uniscano solo le due filosofie deliberativa/reattiva ma anche altre; per esempio combinare in un unico agente, le caratteristiche degli Interface Agents e dei Mobile Agents, entrambi appartenenti alla categoria degli agenti di tipo deliberativo.

Heterogeneous Agents System

I sistemi basati su Agenti Eterogenei [2], a differenza dei sistemi ibridi descritti in precedenza, sono dei sistemi integrati di due o più tipi di agenti tra quelli discussi nel capitolo.

Mentre gli agenti ibridi combinavano filosofie diverse in un'unica struttura, ora vengono combinati direttamente diverse tipologie di agenti.

Uno dei principali motivi che hanno portato allo sviluppo di tali sistemi è l'ampia raccolta di prodotti software ciascuno dei quali offrono una vasta quantità di servizi [35]. Sebbene essi lavorino in modalita stand alone, c'è una crescente richiesta di

interoperabilità tra di essi nella speranza di ottenere un valore aggiunto dalla loro collaborazione. Al fine di ingegnerizzare e standardizzare l'interoperabilità tra diversi agenti software è nata una nuova branca dell'informatica: l'*Agent-Based Software Engineering*.

Uno dei suoi principali obiettivi è stato quello di avere un linguaggio di comunicazione tra agenti (*Agent Communication Language - ACL*) comune in modo tale che agenti software differenti potessero comunicare in una piattaforma comune.

Diversi sono i benefici che si possono ottenere dalla realizzazione di tali sistemi [35]:

- Le applicazioni di tipo stand alone possono aumentare il numero dei servizi offerti inserendole in un contesto eterogeneo
- I legacy system possono vedere ridotti i costi dovuti alla reigegnerizzazione del software se operanti in collaborazione con altri sistemi
- Nuovi orizzonti nella ricerca della progettazione software e nella sua implementazione e manutenzione

Da quanto detto sino ad ora, si evince che uno dei problemi da affrontare per avere sistemi ad agenti eterogenei è rappresentato dal protocollo di comunicazione. Sono stati studiati diversi meccanismi al fine di facilitare ciò:

- Affiancare ad ogni agente un trasduttore il quale riceve i messaggi dagli altri agenti e li traduce in un linguaggio comprensibile al suo agente
- Utilizzare opportuni wrappers che, come affermano Genesereth e Ketchpel [35], "iniettino codice in un programma per consentirgli di comunicare". In tal modo

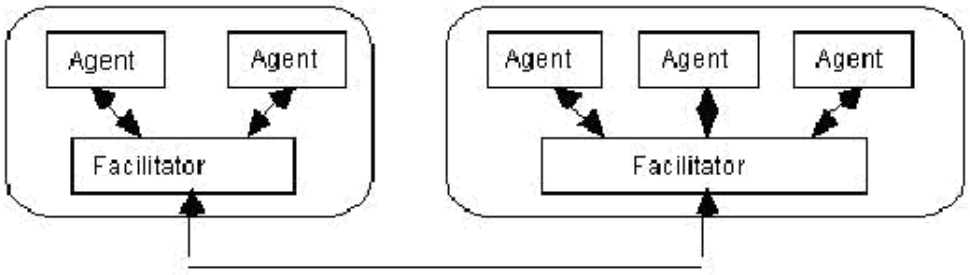

Figure 2.9: Sistema ad agenti eterogenei

ogni agente ha associato un pezzo di codice che è espresso in un comune linguaggio usato dagli altri agenti

- Riscrivere da zero il codice degli agenti; tale alternativa si dimostra essere molto dispendiosa

- Usare, come proposto da Wiederhold [36], un apposito mediatore con il quale si interfacciano tutti gli agenti del sistema e provvede allo smistamento dei messaggi

Nella figura 2.9 è mostrato un esempio di sistema basato su agenti eterogenei [35].

Nel sistema sono presenti cinque agenti distribuiti su due macchine, una con due agenti e l'altra con tre.

Gli agenti non comunicano tra di loro in maniera diretta, bensì tramite dei facilitator che fungono da opportuni mediatori. Gli agenti cedono una parte della loro autonomia ai mediatori i quali sono in grado di mettersi in contatto con gli altri agenti nella rete per richiederne i servizi. I mediatori si occupano anche di stabilire le connessioni nella rete e di assicurare una corretta comunicazione tra gli agenti.

Un prototipo che si basa sull'architettura presentata nella precedente figura è il **PACT**: *Palo Alto Collaborative Testbed* [37]. Si tratta di un sistema per integrare quattro legacy system in un'architettura comune. Sono coinvolti trentuno agenti che girano su quindici workstations e microcomputers. Gli agenti sono organizzati gerarchicamente e comunicano tramite opportuni mediatori.

Il lavoro di sviluppo di sistemi basati su agenti eterogenei è in continua evoluzione ed è molto sentita dai ricercatori l'esigenza di metodologie, tools, tecniche e standards al fine di raggiungere gli obiettivi di interoperabilità tra fonti di informazioni eterogenee. Gli oggetti principali di ricerca sono:

- Fissare un comune linguaggio di comunicazione tra gli agenti
- Stabilire in che modo gli agenti possono comunicare sfruttando un eventuale linguaggio comune
- Fissare degli standards con i quali progettare architetture che permettano l'interoperabilità tra gli agenti

Nel seguito si riprenderanno alcuni dei concetti ora esposti al fine di spiegare nel dettaglio cosa si intende per **Agente di Ricerca Intelligente** (**Intelligent Search Agent - ISA**) e verranno presentati gli agenti progettati nella nostra architettura[2].

[2]Non si è descritto il tipo di agente ombreggiato in figura 2.2 (*Smart Agents*) poiché esso rappresenta un'aspirazione dei ricercatori piuttosto che una realtà. Ricordiamo che un tale tipo di agente è uno che gode contemporaneamente degli attributi di Autonomia, Apprendimento e Cooperazione.

2.2 Ontologia

Un'ontologia è la specificazione esplicita di una concettualizzazione [55]. La parola "ontologia" sembra generare controversie nelle discussioni riguardanti l'intelligenza artificiale. È possibile definire un'ontologia come una descrizione dei concetti e delle relazioni che intercorrono tra di essi. Essa permette di lavorare con un insieme strutturato di concetti, dove le relazioni risultino chiare e, soprattutto, significative a livello semantico. Una ontologia può essere specificata informalmente mediante proposizioni in linguaggio naturale. Una ontologia formale è invece specificata da una raccolta di nomi per i concetti che si vogliono specificare e dalle relazioni che intercorrono tra di loro.

Le ontologie sono spesso considerate alla stregua delle gerarchie tassonomiche di classi, ma le prime, nella visione di T. Gruber [55] (studioso dell'università di Stanford), non si limitano a dare definizioni conservative, ovvero definizioni nel senso tradizionale di introdurre terminologia senza aggiungere conoscenza a proposito del mondo. Infatti, per mantenere un sistema "aperto", l'ontologia non è basata su una gerarchia fissa di categorie, ma su un insieme di distinzioni, a partire dalle quali la gerarchia è generata automaticamente. È importante considerare gli obiettivi di un'ontologia. Se ne progetta una per permettere la condivisione della conoscenza e il suo riutilizzo. Pragmaticamente, si sceglie di scrivere un'ontologia come un insieme strutturato di definizioni di un vocabolario formale. Sebbene questo non sia l'unico modo di specificare una concettualizzazione, esso ha l'interessante proprietà di permettere la condivisione della conoscenza tra software di intelligenza artificiale. È sostanzialmente un accordo sull'uso di un certo vocabolario (ovvero effettuare interrogazioni e fare affermazioni) in modo che esso sia consistente (ma non completo)

rispetto alla teoria specificata dall'ontologia. A partire dalle ontologie, si possono costruire agenti che fanno affidamento su di esse. Le ontologie sono quindi progettare allo scopo di condividere la conoscenza con e tra gli agenti. Non è richiesto comunque che un tale agente risponda a tutte le domande che possono essere formulate nel vocabolario condiviso. Infatti, un accordo sull'uso di una ontologia comune è una garanzia di consistenza ma non di completezza rispetto alle interrogazioni e alle affermazioni che possono essere fatte usando il vocabolario definito dall'ontologia stessa.

2.2.1 Semantic Network

Dati un insieme di concetti[3] in qualche modo correlati tra loro, nasce la necessità di definire una **Semantic Network - Rete Semantica**, per poter effettuare una analisi semantica soddisfacente.

Definition 2.2.1 (Semantic Network). *Dati un insieme di n concetti* $C_1, .., C_n$ *si definisce* Semantic Network *il grafo pesato i cui nodi rappresentano i concetti* C_i *e gli archi non orientati le relazioni tra essi. I pesi associati agli archi rappresentano il* ***grado di correlazione*** *tra i concetti interessati.*

Le relazioni rappresentate dagli archi possono essere di similitudine, di generalizzazione/specializzazione o di semplice attinenza.

I pesi associati agli archi costituiscono una misura del grado di parentela tra i concetti collegati dall'arco stesso.

[3]Una definizione di concetto sarà data successivamente quando verrà illustrato il modello formale al quale il sistema fa riferimento

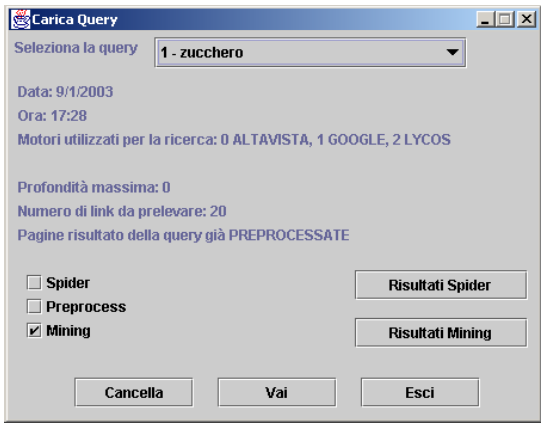

Figure 2.10: Legame tra due concetti

Più precisamente, con riferimento alla figura 2.10, considerati due concetti C_1 e C_2 tali che esistano due archi diretti rispettivamente da C_1 a C_2 e da C_2 a C_1:

- P_{12} è la probabilità che il concetto C_2 possa soddisfare una richiesta del concetto C_1
- P_{21} è la probabilità che il concetto C_1 possa soddisfare una richiesta del concetto C_2

Definiti i pesi sugli archi, possiamo dare, con riferimento alla figura 2.11, la definizione di distanza probabilistica tra due concetti.

Definition 2.2.1 (Distanza probabilistica). Dato un insieme di n concetti C_1,...,C_n e considerati due di tali *concetti* C_i *e* C_j, collegati da uno o più percorsi, si definisce **distanza probabilistica tra i concetti** C_i **e** C_j la quantità

$$\mathbf{DP(C_i, C_j)} = \max_{k \in 1..n} (\prod_{t=1}^{na} P_{k,t}) \tag{2.2.1}$$

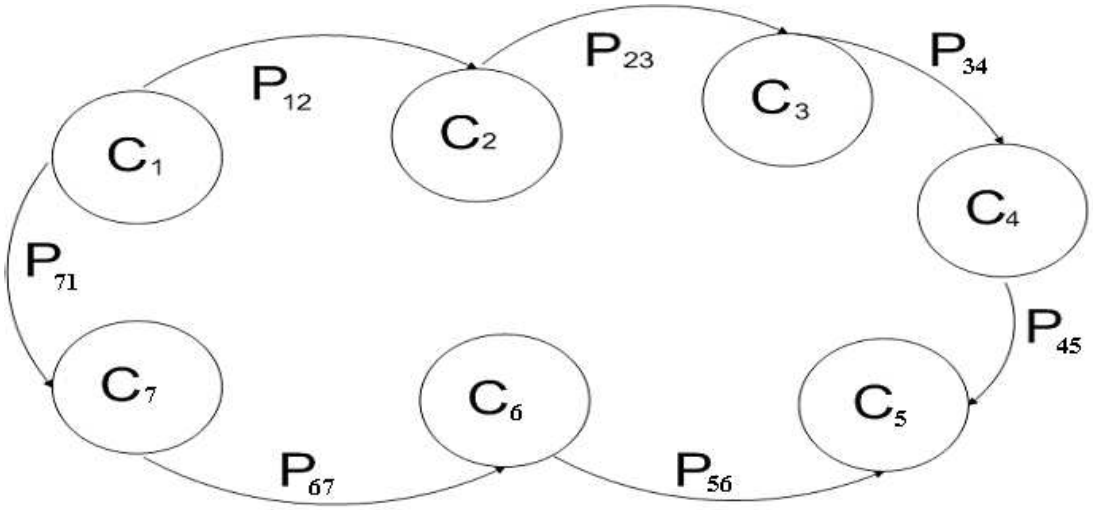

Figure 2.11: Distanza probabilistica tra due concetti

dove n è il numero di percorsi esistenti tra i concetti C_i e C_j, na è il numero di archi del k-simo percorso e $P_{k,t}$ è il peso associato al t-simo arco del k-simo percorso.

Si noti che *DP(C_i, C_j)* può ancora essere interpretata come la probabilità che il concetto C_j possa soddisfare una richiesta del concetto C_i e coincide con P_{ij} se esiste un arco diretto da C_i a C_j e non esistono tra gli stessi concetti percorsi migliori di quello diretto. Due concetti sono tanto più vicini dal punto di vista semantico quanto più la loro distanza probabilistica è prossima ad 1.

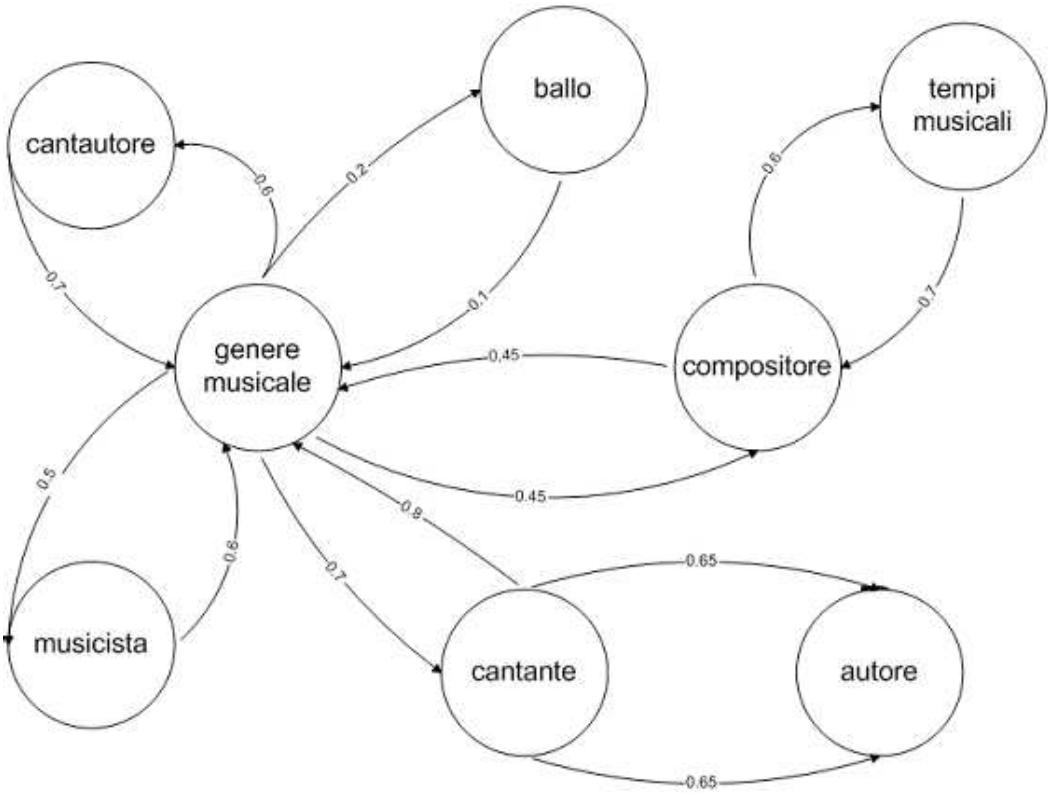

Figure 2.12: Una rete semantica per i concetti del dominio musicale

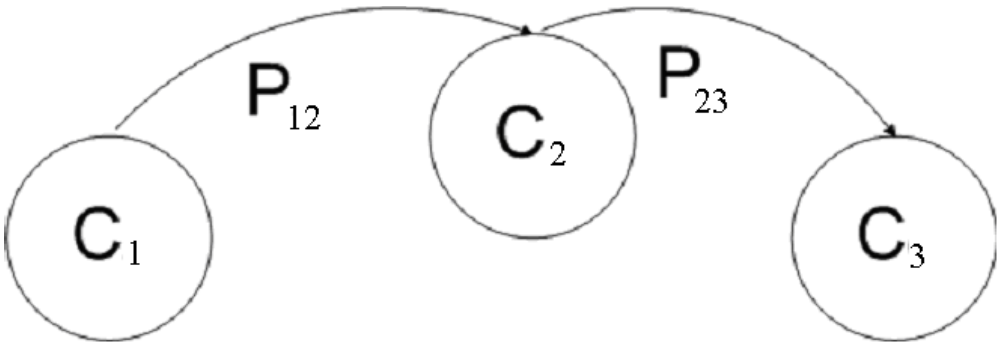

Figure 2.13: Distanza probabilistica lungo un percorso

Example 2.2.1. *Con riferimento alla figura 2.12, che rappresenta una possibile rete semantica specializzata sul dominio musicale, alcune delle distanze probabilistiche individuabili tra i concetti sono:*

- *DP(*genere musicale*,* cantante*) = 0.8*
- *DP(*cantante*,* genere musicale*) = 0.7*
- *DP(*cantautore*,* compositore*) = 0.7 · 0.45 = 0.315*
- *DP(*tempi musicali*,* musicista*) = 0.7 · 0.45 · 0.5 = 0.1575*
- *DP(*musicista*,* tempi musicali*) = 0.6 · 0.45 · 0.6 = 0.162*

Facciamo ora alcune considerazioni per giustificare la distanza introdotta. Motiviamo prima di tutto la presenza del simbolo di produttoria. Con riferimento al semplice esempio di figura 2.13, la probabilità che il concetto C_3 possa soddisfare una richiesta del concetto C_1 è pari alla probabilità dell'evento *"C_3 soddisfa una richiesta del concetto C_2 e C_2 soddisfa una richiesta del concetto C_1"*, ovvero alla probabilità

dell'intersezione di due eventi statisticamente indipendenti. Com'è noto dalla teoria della probabilità, dati due eventi statisticamente indipendenti A e B risulta:

$$P(A \cdot B) = P(A) \cdot P(B)$$

Inoltre bisogna chiarire perché si assume come distanza tra due concetti la probabilità associata al percorso probabilisticamente più corto, ossia il percorso cui è associata la distanza probabilistica più prossima ad 1. Se tra due concetti esiste più di un percorso, il percorso probabilisticamente più corto è chiaramente quello che esprime le relazioni più esplicite tra i concetti in questione e risulta dunque naturale assumere la probabilità ad esso associata come misura di similitudine tra gli stessi.

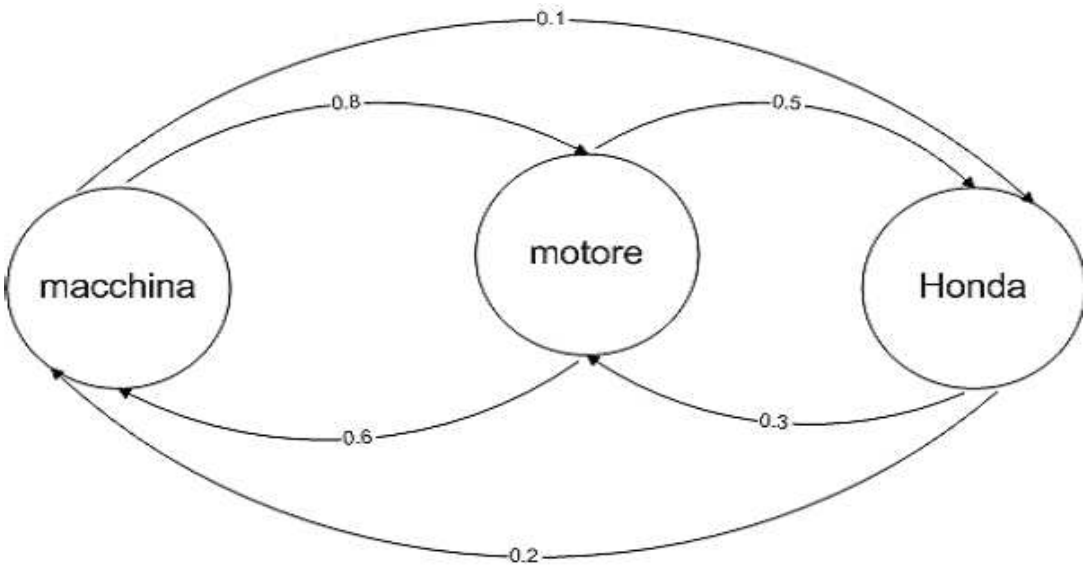

Figure 2.14: Legami tra alcuni concetti del dominio automobilistico

Example 2.2.2. *Con riferimento alla figura 2.14, che rappresenta i legami tra alcuni concetti del dominio automobilistico, si ha che*

- *DP(*macchina*,* Honda*) = 0.8 · 0.5 = 0.4*
- *DP(*Honda*,* macchina*) = 0.2*

Si noti come nel primo caso il percorso diretto macchina-Honda *non sia quello probabilisticamente più corto. Nel secondo caso accade, invece, l'esatto contrario.*

Per concludere, osserviamo che la *distanza probabilistica* soddisfa le seguenti proprietà:

- $DP(C_i, C_i) = 1 \qquad \forall\ Ci \in$ *insieme dei concetti*
- $DP(C_i, C_j) \neq DP(C_j, C_i)$
- $DP(C_i, C_j) \neq DP(C_i, C_k) \cdot DP(C_k, C_j) \quad \forall\ C_i, C_j, C_k \in$ *insieme dei concetti*
- $DP(C_i, C_j) = 0 \qquad \iff$ *non esiste percorso tra* C_i *e* C_j

2.3 Modello formale

Consideriamo un insieme di elementi A: nella *teoria classica degli insiemi* l'appartenenza di ogni elemento x ad A è indicata in maniera certa. Ci sono però dei casi in cui la teoria classica non riesce a classificare in maniera adeguata gli elementi di un sottoinsieme dell'universo del discorso. In questi casi può essere opportuno non

indicare la semplice appartenenza di un elemento all'insieme, ma la probabilità percentuale che l'elemento appartenga o meno all'insieme: questa probabilità percentuale di appartenenza sarà un valore compreso nell'intervallo [0,1] dei numeri reali ed ovviamente l'appartenenza certa si avrà con la probabilità percentuale pari ad 1, mentre in corrispondenza della probabilità percentuale pari a 0 si avrà la non appartenenza all'insieme.

2.3.1 Domini, concetti e parole

Iniziamo con il definire cosa sono un vocabolo ed una lingua.

Definition 2.3.1 (Vocabolo). *Si definisce* vocabolo *un insieme di caratteri isolato, colto fuori da un contesto e da ogni legame grammaticale o logico, nella sua individualità lessicale [62].*

Example 2.3.1. *Alcuni vocaboli sono, ad esempio:*

Riserva, House, Maman, Pfad, Elicóptero,...

I vocaboli sono la parte fondamentale di una lingua.

Definition 2.3.2 (Lingua). *Si definisce una* lingua *come l'insieme dei vocaboli utilizzati da un popolo, insieme alle regole che li governano [62].*

Example 2.3.2. *Alcune possibili lingue possono essere:*

ITALIANO, TEDESCO, INGLESE, SPAGNOLO,...

Definition 2.3.3 (Regole sintattiche). *Si definiscono* regole sintattiche *(o* sintassi*) l'insieme delle norme che studiano le relazioni che i vocaboli hanno in una frase (gruppo di vocaboli) [62].*

Definition 2.3.4 (Regole semantiche). *Si definiscono* regole semantiche *(o* semantica*) l'insieme delle norme che si occupano del significato dei vocaboli e dei cambiamenti di essi [62].*

Ci occuperemo di seguito della lingua Italiana e delle sue regole sintattiche e semantiche. Definiamo ora cosa è un dominio ed un concetto.

Definition 2.3.5 (Dominio). *Detto S l'universo del discorso si definisce* dominio D *un sottoinsieme di S, ovvero*

$$D \subseteq S.$$

Example 2.3.3. *Possibili esempi di domini, nella lingua Italiana, potrebbero essere il dominio della* MUSICA*, il dominio della* PITTURA*, ecc.*

Definition 2.3.6 (Concetto). *Per concetto linguistico si intende una parola, frase, acronimo o nome ricco di significato, che è stato estratto da componenti non strutturati del testo, incluso blocchi isolati di testo, sommari, intestazioni, paragrafi etc. Esso è definito anche come termine linguistico, mentre ogni documento si riferisce a un gruppo di concetti o termini [61].*

Dato un dominio D e detti C_i i concetti in esso definiti si ha si ha che l'unione di tutti i concetti è una copertura per il dominio stesso, ovvero

$$\mathbf{D} = \bigcup_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{C_i}$$

Example 2.3.4. *Se consideriamo il dominio della MUSICA, possibili esempi di concetti di tale dominio potrebbero essere il concetto* STRUMENTI*, il concetto* CANTANTI*, ecc.*

Ogni concetto è costituito da **vocaboli** della lingua alla quale si ci riferisce. Possiamo, però considerare che i vocaboli non appartengano ad un concetto necessariamente con una probabilità pari ad 1. Soprattutto nei casi in cui si vuole rappresentare un dominio con un insieme sintetico di concetti, può accedere che alcuni vocaboli facenti parte del dominio, siano inseriti in concetti ai quali non appartengono pienamente. Inoltre un vocabolo può appartenere a più concetti e pare opportuno, in questi casi, immaginare che non appartenga a pieno ad entrambi i concetti.

Example 2.3.5. *Consideriamo ad esempio i due vocaboli della lingua italiana:* basso

e fisarmonica. *Possiamo immaginare che fisarmonica appartenga al concetto* strumento *del dominio musicale con una probabilità maggiore rispetto a basso, in quanto quest'ultimo crea maggiori ambiguità.*

Per tale motivo introduciamo il concetto di centralità di un vocabolo rispetto ad un concetto.

Definition 2.3.7 (Centralità). *Definiamo* centralità *la probabilità percentuale di appartenenza del vocabolo lessicale ad un* concetto.

Appare evidente che più il valore della centralità è prossimo ad 1 e più è probabile che il vocabolo appartenga al concetto associato.

Definition 2.3.8 (Peso). *Definiamo* peso *il grado con cui un vocabolo lessicale rappresenta il* dominio *al quale è associato.*

Anche il peso è un valore appartenente all'intervallo [0, 1] e più è grande, maggiore è il livello di rappresentazione, da parte del vocabolo, del dominio associato.

Definition 2.3.9 (Parola). *Definiamo* parola *un vocabolo di una lngua, a cui è associato un* concetto C*, il* dominio D *a cui il concetto appartiene e due indici* centralità *e* peso*, quindi una parola è la quadrupla*

<vocabolo lessicale, concetto, dominio, centralità, peso>

Example 2.3.6 (Parole). *Possibili esempi di parole possono essere i seguenti.*

Vocabolo	***Concetto***	***Dominio***	***Centralità***	***Peso***
Chitarra	*Strumenti*	*Musica*	*0.8*	*0.6*
Olio	*Ricambi*	*Motori*	*0.6*	*0.55*
Aereo	*Mezzi di trasporto*	*Turismo*	*0.9*	*0.7*
Presentatore	*Televisione*	*Spettacolo*	*0.85*	*0.6*
Bacio	*Amore*	*Sentimenti*	*0.85*	*0.9*

Indicato con S sempre l'universo del discorso, è possibile anche definire una **funzione F** tale che data una *parola* x ed un concetto C appartenente ad un dominio D, il valore

$$F_C(x) \in [0, 1]$$

rappresenta la centralità di x rispetto al concetto C, con la convenzione che se il vocabolo inserito nella parola x non appartenga al concetto C la funzione F restituisca 0 [4].

In maniera analoga alla centralità è possibile definire una **funzione P** tale che dato una *parola* x ed un dominio D, il valore

$$P_D(x) \in [0, 1]$$

rappresenta il peso di x rispetto al Dominio D.

[4] In questo caso è come dire che il vocabolo associato alla parola x appartiene al concetto con una probabilità nulla, cioé non vi appartiene.

2.3.2 Dominio unico

Consideriamo innanzitutto l'esistenza di un solo **dominio** e diamo in tal caso la definizione di insieme monoconcetto e multiconcetto.

Definition 2.3.10 (Insieme monoconcetto). *Dato il* ***dominio D*** *ed un suo* ***concetto C****, l'insieme di parole A si definisce* ***monoconcetto*** *se*

$$\forall x \in \mathcal{A} \rightarrow F_C(x) \neq 0. \tag{2.3.1}$$

Example 2.3.7. *Un possibile esempio di insieme monoconcetto A, riferito al concetto Repulsione appartenente al dominio Sentimenti è illustrato nella figura 2.15*

$$A = < \begin{array}{lllll} \textit{Detestare} & \textit{Repulsione} & \textit{Sentimenti} & \textit{0.9} & \textit{0.9} \\ \textit{Disprezzo} & \textit{Repulsione} & \textit{Sentimenti} & \textit{0.6} & \textit{0.7} \\ \textit{Avversione} & \textit{Repulsione} & \textit{Sentimenti} & \textit{0.9} & \textit{0.5} \\ \textit{Scontentezza} & \textit{Repulsione} & \textit{Sentimenti} & \textit{0.3} & \textit{0.3} \\ \textit{Grasso} & \textit{Repulsione} & \textit{Sentimenti} & \textit{0.2} & \textit{0.1} \end{array} >$$

Figure 2.15: Esempio di insieme monoconcetto

Definition 2.3.11 (Insieme multiconcetto). *Dato il* dominio D *ed i* concetti C_1,...,C_n *di D, l'insieme di parole* A *è un insieme* ***multiconcetto*** *se*

$$\forall x \in \mathcal{A} \rightarrow \ \exists C_i, con\ i = 1..n : F_{C_i}(x) \neq 0. \tag{2.3.2}$$

dove la funzione $F_{C_i}(x)$, come detto in precedenza, restituisce la centralità della parola x rispetto al concetto C_i.

Example 2.3.8. *Un possibile esempio di insieme multiconcetto è illustrato nella figura 2.16*

$$A = < \begin{array}{lllll} \textit{Disprezzo} & \textit{Repulsione} & \textit{Sentimenti} & \textit{0,6} & \textit{0,7} \\ \textit{Indebolire} & \textit{Violenza} & \textit{Sentimenti} & \textit{0,3} & \textit{0,2} \\ \textit{Vendicare} & \textit{Violenza} & \textit{Sentimenti} & \textit{0,8} & \textit{0,5} \\ \textit{Attaccare} & \textit{Violenza} & \textit{Sentimenti} & \textit{0,9} & \textit{0,8} \\ \textit{Attaccare} & \textit{Conflitto} & \textit{Sentimenti} & \textit{0,8} & \textit{0,7} \end{array} >$$

Figure 2.16: Esempio di insieme multiconcetto

Si noti che uno stesso vocabolo lessicale può appartenere all'insieme in due *parole* differenti, ognuna delle quali si riferisce ad un *concetto*, come nel caso del vocabolo **attaccare**.

Sia nel caso di insieme monoconcetto, che in quello di insieme multiconcetto, è possibile dare una definizione di **centralità dell'insieme**; ovviamente nel caso di insieme monoconcetto la centralità la si potrà calcolare rispetto al concetto al quale si riferisce (negli altri casi appare evidente che la centralità sarà nulla), mentre nel caso di insieme multiconcetto potremo calcolare la centralità rispetto ad uno qualsiasi dei concetti al quale si riferisce.

Definition 2.3.12 (Centralità di un insieme monoconcetto). *Dato un insieme monoconcetto A riferito al concetto C, si definisce* ***centralità di A rispetto a C*** *la media delle centralità delle parole appartenenti all'insieme A.*

$$CEN_C(A) = \frac{\sum_{x \in \mathcal{A}} (F_C(x))}{N}. \tag{2.3.3}$$

dove N è il numero di parole appartenenti ad A

Example 2.3.9. *Consideriamo ad esempio l'insieme monoconcetto A nella figura 2.15, in tal caso avremo che la sua centralità di rispetto alla* ***Repulsione****, sarà 0,58; ovvero*

$$CEN_{REPULSIONE}(A) = 0,58$$

Per un insieme multiconcetto è, come detto, è possibile definire la centralità che esso ha nei vari concetti al quale fa riferimento.

Definition 2.3.13 (Centralità di un insieme multiconcetto). *Dato un insieme multiconcetto A riferito agli n concetti* C_1,...,C_n *si definisce* ***centralità di A rispetto a*** $\mathbf{C_i}$ *la media delle centralità delle parole appartenenti all'insieme A ed associate al concetto* C_i*, ovvero*

$$CEN_{C_i}(A) = \frac{\sum_{x \in \mathcal{A}} (F_{C_i}(x))}{N_i}. \tag{2.3.4}$$

dove N_i *è il numero di parole appartenenti ad A ed associate al concetto* C_i*.*

Example 2.3.10. *Consideriamo ad esempio l'insieme multiconcetto A della figura 2.16, in tal caso la centralità di quest'insieme rispetto al concetto* ***Repulsione*** *è 0.6, rispetto al concetto* ***Violenza*** *è 0.66, mentre rispetto al concetto* ***Conflitto*** *è 0.8, ovvero*

$$CEN_{REPULSIONE}(A) = 0.6$$

$$CEN_{VIOLENZA}(A) = 0.66$$

$$CEN_{CONFLITTO}(A) = 0.8$$

È possibile anche definire delle operazioni tra gli insiemi monoconcetto o multiconcetto.

Definition 2.3.14 (Unione di insiemi). *Detti A e B due insiemi (monoconcetto o multiconcetto), l'insieme* ***C*** *si definisce* ***unione tra A e B****, e si indica nel seguente modo*

$$C = Unione\ (A, B)$$

se esso è formato dalle parole *appartenenti ad A e B. Nel caso in cui una* parola *fosse contenuta sia in A che in B questa viene inserita una solo volta in C.*

$$C = \{x \in insieme\ delle\ parole \mid x \in A\ oppure\ x \in B\} \tag{2.3.5}$$

In relazione all'operazione di unione valgono le seguenti **proprietà**:

- se A e B sono due insiemi monoconcetto che si riferiscono allo stesso concetto, allora la loro unione sarà ancora un insieme monoconcetto.
- se A e B sono due insiemi monoconcetto che si riferiscono a due concetti differenti, allora la loro unione sarà un insieme multiconcetto.
- se A e B sono due insiemi multiconcetto, allora la loro unione sarà comunque un insieme multiconcetto.

Dal punto di vista analitico è ovvio che

Unione(A, A) = A

Unione(A, B) = Unione(B, A)

Unione(Unione (A, B), C) = Unione(A, Unione(B, C))

Definition 2.3.15 (Dimensione di un insieme). *Dato un insieme A (monoconcetto o multiconcetto) definiamo la* ***dimensione di A*** *e la indichiamo con* ***DIM(A)*** *il numero di PAROLE contenute all'interno dell'insieme.*

Example 2.3.11. *Consideriamo sempre l'insieme multiconcetto A di figura 2.16: si avrá che la dimensione di A è pari a cinque, cioé*

$$\boldsymbol{DIM(A)} = 5$$

Definition 2.3.16 (Peso di un insieme). *Dato un* dominio D *composto dai* concetti C_1,...,C_n *ed un* insieme A *(monoconcetto o multiconcetto), si definisce* ***peso di A*** *il valore ottenuto facendo la media dei valori dei pesi delle parole dell'insieme A.*

$$\boldsymbol{Peso(A)} = \sum_{j=1}^{DIM(A)} (\frac{P_D(x_i)}{DIM(A)}) \tag{2.3.6}$$

Example 2.3.12. *Considerato sempre l'insieme multiconcetto A precedentemente illustrato in figura 2.16 si avrà che il peso di A sarà 2.9/5.*

Si noti che poiché il peso è una caratteristica che riguarda il dominio, non ci sono differenza tra insiemi monoconcetto o multiconcetto per quanto riguarda il calcolo del loro peso. In relazione al peso vale la seguente disuguaglianza: dati due insiemi (monoconcetto o multiconcetto) A e B si ha che

$$\textbf{Peso(Unione(A, B))} \neq \frac{Peso(A) + Peso(B)}{2} \tag{2.3.7}$$

Dato un dominio D costituito da un insieme di concetti C_1,...,C_n è possibile definire una grado di associazione tra i concetti C_i e C_j, con i, j = 1..n

$$< \mathbf{C_i}, \mathbf{C_j}, \mathbf{gradodiassociazione} >$$

Si può costruire in questo modo una **Semantic Network** tra i concetti, in cui i nodi rappresentino i concetti e l'arco dal nodo i al nodo j il grado di associazione tra C_i e C_j.

Definition 2.3.17 (Voto di un insieme monodominio). *Dato un* dominio D *composto dai concetti C1,...,Cn ed un* insieme A *si definisce* ***Voto di A*** *la somma di due contributi uno pari al peso di A, e l'altro funzione della* ***distanza probabilistica DP*** *tra tutti i concetti interessati da A.*

$$\boldsymbol{Voto(A) = Peso(A) + f(DP)} \tag{2.3.8}$$

In particolare la f(DP) è la somma delle distanze probabilistiche tra i concetti di tutte le coppie di parole relative ad A moltiplicate per la media delle centralità delle parole nei relativi concetti, che però superano un valore limite.

2.3.3 Domini indipendenti distinti

Consideriamo adesso l'esistenza di m *domini* D_1,..,D_m indipendenti.

Definition 2.3.18 (Insieme multidominio). *Dati m* domini D_1,..,D_m *ognuno dei quali composto dai concetti* $C_{d1}, .., C_{dn_d}$*, dove d è l'indice dei domini mentre* n_d *è il numero di concetti del dominio d, (si noti che può capitare che uno stesso concetto C appartenga a due domini diversi) si definisce* ***insieme multidominio*** *quell'insieme di* Parole *i cui elementi appartengono a domini differenti.*

Un insieme multidominio A può essere visto come l'unione degli insiemi (monoconcetto o multiconcetto) ottenuti facendo la proiezione di A sui domini D_1,..,D_m. Indichiamo la proiezione di A sul dominio Di con

$$\mathbf{A|D_i} = \textit{proiezione di A sul dominio } D_i$$

È possibile in questo modo estendere le definizioni di centralità e peso ad un insieme multidominio.

Definition 2.3.19 (Centralità di un insieme multidominio). *La* centralità di un insieme multidominio A *rispetto ad uno dei concetti* C_{ij} *apparteneti al dominio* D_i*, si definisce come la centralità dell'insieme (monoconcetto o multiconcetto)* $A|D_i$ *rispetto al concetto* C_{ij}*.*

$$\mathbf{CEN_{C_{ij}}(A)} = \mathbf{CEN_{C_{ij}}(A|D_i)} \qquad \textit{con } C_{ij} \in D_i$$

Definition 2.3.20 (Peso di un insieme multidominio rispetto ad un dominio $\mathbf{D_i}$). *Dati m* domini D_1,..,D_m *ognuno dei quali composto dai concetti* $C_{d1}, .., C_{dn_d}$, *dove d è l'indice dei domini mentre* n_d *è il numero di concetti del dominio d, e considerato un insieme multidomio A si definisce* ***peso di un insieme multidominio A*** *rispetto ad un dei domini* D_i, *la sommatoria dei pesi delle parole appartenenti all'insieme dell'insieme (monoconcetto o multiconcetto) A|Di diviso però per la dimensione di A.*

$$\mathbf{Peso_{D_i}(A)} = \sum_{j=1}^{DIM(A|D_i)} (\frac{P_{D_i}(x_i)}{DIM(A)}) \tag{2.3.9}$$

Come detto in precedenza, avremo una *mappa dei concetti* per ogni dominio, ma poiché è anche possibile che stessi concetti appartengano a domini differenti, potrebbe capitare che esista una *distanza probabilistica* tra i concetti C_k e C_t in due o più domini; indicata con $DP_{D_i}(C_k, C_t)$ la distanza probabilistica tra Ck e Ct in relazione al dominio Di, consideriamo la seguente situazione:

$$\begin{gathered} DP_{D_1}(C_k, C_t) \\ DP_{D_2}(C_k, C_t) \\ \vdots \\ DP_{Dm}(Ck, Ct) \end{gathered}$$

in questo caso la distanza probabilistica tra C_k e C_j può essere calcolata con la seguente formula

$$\mathbf{DP(C_k, C_j)} = \frac{1}{m} \sum_{i=1}^{m} (DP_{D_i}(C_k, C_j)) \tag{2.3.10}$$

La motivazione della precedente formula è la seguente: dalla teoria della probabilità è noto che dato un evento E ed una partizione D_1,..,D_m dell'evento certo S per la **legge della probabilità totale**

$$\mathbf{P}(\mathbf{E}) = \sum_{i=1}^{m} (P(D_i) \cdot P(E|D_i))$$

Nel caso in esame i domini D_1,..,D_m costituiscono una partizione dell'universo lessicale (evento certo) e sono equiprobabili con probabilità $\frac{1}{m}$, inoltre la distanza pobabilistica $DP_{D_i}(C_k, C_t)$ può essere considerata come la probabilità condizionata dato il dominio D_i.

Nel caso di m domini indipendenti il voto di un insieme multidominio rispetto ad uno degli m domini può essere calcolato con una formula analoga a quella definita nel caso di insiemi monodominio.

$$\mathbf{Voto}(\mathbf{A}) = \mathbf{Peso}(\mathbf{A}) + \mathbf{f}(\mathbf{DP}) \tag{2.3.11}$$

2.3.4 Domini distinti non indipendenti

Consideriamo infine l'esistenza di m *domini* D_1,..,D_m non indipendenti tra loro. In questo caso è possibile definire una **mappa dei domini**.

Definition 2.3.21 (mappa dei domini). *Dati m* domini D_1,..,D_m *ognuno dei quali composto dai concetti* $C_{d1}, .., C_{dn_d}$*, dove d è l'indice dei domini mentre* n_d *è il numero di concetti del dominio d, si definisce mappa dei domini il grafo pesato i cui nodi rappresentano i domini* D_i *e gli archi non orientati le relazioni tra essi. I pesi associati agli archi rappresentano il* ***grado di correlazione*** *tra i domini interessati: si indica con* $\boldsymbol{GA(D_j, D_j)}$ *il grado di associazione tra* D_i *e* D_j*. Si assume GA(*D_i*,* D_j*) = 0 se non esiste un arco diretto tra* D_i *e* D_j*.*

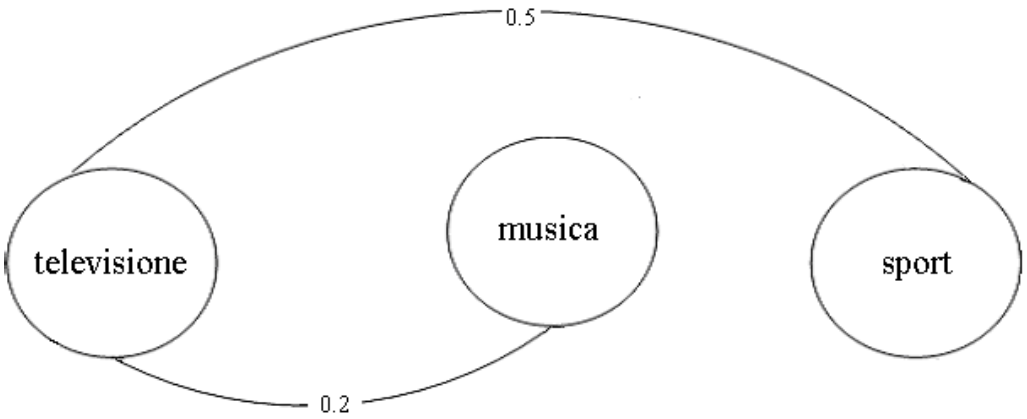

Figure 2.17: Esempio di mappa dei domini

Anche nel caso di m domini non indipendenti è possibile calcolare il voto di un insieme multidominio rispetto ad uno degli m domini utilizzando la definizione data. In questo caso ovviamente la funzione **f(DP)** sarà più complessa dovendo tener conto delle relazioni tra i domini.

2.4 La metrica

Per poter discriminare le pagine di interesse per uno degli m domini esistenti, da quelle che non lo sono, occorre definire una metrica che consenta al sistema di dare un voto alle pagine a seconda del loro contenuto sintattico e semantico.

La metrica qui proposta prende in considerazione cinque informazioni che fanno riferimento alla pagina a cui vogliamo dare un giudizio, due delle quali legate alla sintattica e tre alla semantica delle pagine:

- *Profondità della pagina rispetto al motore di ricerca*
- *Posizione mediata della pagina*
- *Peso dei vocaboli nella pagina*
- *Premio dell'accoppiata di vocaboli che si riferiscono a concetti correlati tra loro, pesato con la distanza dei vocaboli stessi*
- *Premio dovuto all'associazione tra domini correlati*

2.4.1 GPrS e posizione mediata

Consideriamo Π l'insieme delle pagine WEB. Dati n motori di ricerca, viene assegnato ad ogni motore un peso w_i, compreso tra 0 e 1, tale che:

$$\sum_{i=1}^{n}(w_i) = 1 \tag{2.4.1}$$

Definition 2.4.1 (Funzione posizione). *Definiamo* ***funzione di posizione***

$$p_i : \Pi \rightarrow N$$

come quella funzione intera che ad ogni pagina WEB x appartenenti all'insieme Π *associa la sua posizione all'interno dell'elenco dei risultati del motore di ricerca i, se la pagina viene restituita dal motore stesso, 0 nel caso in cui la pagina non fosse trovata dal motore.*

Figure 2.18: Esempio di risultato di ricerca con Google

Example 2.4.1. *Considerato l'esempio di figura 2.18, si ha che la posizione della pagina www.spagnacontemporanea.it/ ha una posizione pari a 3, rispetto al motore di ricerca Google.*

Definition 2.4.2 (GPrS). *Data una pagina WEB x appartenente all'insieme* Π *e siano A l'insieme di numeri naturali* i *tali che il motore di ricerca* i-esimo *restituisca la pagina x, e B l'insieme di numeri naturali* j *tali che il motore di ricerca* j-esimo *non restituisca la pagina x. Sia inoltre* a=card(A)*, si definisce* ***GPrS*** *della pagina x la quantità:*

$$\boldsymbol{GPrS(x)} = \frac{\sum_{i \in A}(w_i)}{\sum_{i \in A} p_i(x) \cdot (w_i + \frac{\sum_{j \in B} w_j}{a})} \tag{2.4.2}$$

ovviamente risulta $GPrS(x) \in]0, 1]$.

La situazione migliore si ha quando GPrS(x) = 1, infatti in tal caso tutti i motori di ricerca hanno trovato la pagina x nella posizione numero 1, man mano che la pagina viene trovata in posizioni maggiori o da meno motori di ricerca, GPrS(x) si avvicina sempre più a 0.

Definition 2.4.3 (Posizione mediata). *Data una pagina WEB x appartenente all'insieme* Π*, si definisce* posizione mediata *di x, e la si indica con* $\mu(\mathbf{x})$*, la quantità inversa di GPrS (x), cioé*

$$\mu(\mathbf{x}) = \frac{1}{GPrS(x)} \tag{2.4.3}$$

ovviamente risulta $\mu(x) \in [1, \infty]$.

Si noti che se tutti i motori di ricerca restituissero la pagina x, la posizione mediata sarebbe la media pesata delle posizioni nei singoli motori, con i pesi attribuiti ai motori di ricerca.

2.4.2 Contributo sintattico

Definition 2.4.4 (Quantità q(x)). *Data una pagina WEB x appartenente all'insieme* Π*, e detto L il numero di link prelevati da ogni motore di ricerca, si definisce* ***q(x)*** *la quantità*

$$q(x) = \frac{L}{(L + GPrS(x) - 1)} \tag{2.4.4}$$

ovviamente si ha che $q(x) \in]0, 1]$.

Si noti che la situazione migliore si ha con q(x) = 1: tale valore si ottiene nel caso in cui GPrS(x)=1 ovvero quando il valore di L tende ad ∞.

Definition 2.4.5 (Contributo sintattico). *Detta b la profondità della pagina x, si definisce contributo sintattico, e lo si indica con Sin(x), la quantità*

$$Sin(x) = \frac{q(x)}{1 + b \cdot K_b} \tag{2.4.5}$$

dove K_b *è una costante da trovare sperimentalmente per definire quanto la profundità di una pagina debba incidere sul suo contributo sintattico. Ovviamente si ha che* $Sin(x) \in]0, 1]$ *e la situazione migliore si ha con Sin(x) = 1.*

Se consideriamo una pagina WEB x appartenente all'insieme Π, è possibile che al suo interno siano presenti dei link ad altre pagine WEB sempre appartenenti all'insieme Π.

Definition 2.4.6 (Profondità di una pagina). *Data una pagina WEB x appartenente all'insieme* Π *e detta y una pagina WEB appartenente a* Π *il cui link si trova all'interno di x, definiamo la profondità della pagina y come la profondità della pagina x aumentata di 1. Si definisce inoltre che la profondità delle pagine restituite da qualche motore di ricerca sia pari a 0*[5]*.*

Definition 2.4.7 (Contributo sintattico). *Data una pagina WEB x appartenente all'insieme* Π*, detta b la sua profondità e* K_b *un valore reale, si definisce* contributo sintattico*, e lo si indica con* ***Sin(x)****, la quantità*

$$Sin(x) = \frac{q(x)}{1 + b \cdot K_b} \tag{2.4.6}$$

ovviamente si ha che $Sin(x) \in]0, 1]$*.*

Si noti che anche per il contributo sintattico la situazione migliore si ha quando il suo valore è pari ad 1, ciò avviene per le pagine che hanno q(x) uguale ad 1 ed appartengono all'insieme delle pagine restituite da qualche motore di ricerca.

Si noti ancora che il valore della costante della profondità K_b deve essere trovato

[5] *Il motivo della definizione della profondità verra meglio chiarita in seguito nella spiegazione del funzionamento del sistema*

sperimentalmente; tale valore indica quanto la profundità di una pagina debba incidere sul suo contributo sintattico; nel caso in cui K_b fosse pari a 0 si avrebbe che la profondità non influenza il contributo sintattico.

2.4.3 Contributo semantco

Il contributo semantico, che possiamo indicare con V, è somma di tre quantità:

- *Peso dei vocaboli trovati in uno dei dizionari presenti nel sistema*
- *Contributo funzione delle distanze probabilistiche dei concetti trovati*
- *Peso dovuto all'associazione tra domini correlati*

Si noti che d'ora in poi faremo riferimento in generale ad una pagina anche se tutto può essere riferito separatamente al contenuto, al titolo, alla descrizione o alle keywords della pagina stessa.

2.4.4 Peso dei vocaboli di una pagina

Definition 2.4.8 (Peso di una pagina in un dominio D_i). *Data una pagina x appartenente all'insieme* Π *e m domini D_1,..,D_m, detto inoltre N_i il numero di vocaboli presenti in x ed associati al dominio D_i e NT il numero di vocaboli presenti in x ed appartenenti ad uno degli m domini,cioè*

$$NT = \sum_{i=1}^{m} N_i$$

si definisce peso di una pagina nel dominio D_i, *e lo si indica con $Peso_{D_i}(x)$, la somma di tutti i pesi $t_{i,j}$, relativi al dominio D_i, associati agli N vocaboli presenti in*

una pagina (ovviamente se un vocabolo non è presente nel dominio D_i il suo peso vale 0).

$$\mathbf{Peso_{D_i}(x)} = \frac{\sum_{j=1}^{N} t_{i,j}}{NT} \tag{2.4.7}$$

Si noti che il *Peso di una pagina* porta in se anche un significato sintattico in quanto il peso viene dato in base alla forma del vocabolo (e quindi in base alla sua sintassi).

Contributo funzione delle distanze probabilistiche dei concetti presenti nella pagina

Abbiamo visto che il peso di una pagina dipende dai vocaboli trovati al suo interno. Poiché ad ogni vocabolo è associato un concetto e la relativa centralità possiamo parlare anche di concetti, e centraità associate, trovati in una pagina.

Definition 2.4.9 (Distanza tra due concetti). *Data una pagina x appartenente all'insieme* Π *ed una coppia di concetti C_i e C_j presente all'interno della pagina si definisce* distanza tra i concetti*, e la si indica con DIST(C_i, C_j) la differenza tra le posizioni dei concetti nella pagina.*

$$\mathbf{DIST(C_i, C_j)} = |Posizione(C_i) - Posizione(C_j)| \tag{2.4.8}$$

Definition 2.4.10 (Centralità media tra due concetti). *Data una pagina x appartenente all'insieme* Π *ed una coppia di concetti* C_i *e* C_j *presente all'interno della pagina si definisce* Centralità media tra due concetti*, e la si indica con CENM*(C_i, C_j) *la media delle centralità associate ai concetti* C_i *e* C_j*.*

$$\mathbf{CENM(C_i, C_j)} = \frac{CEN(C_i) + CEN(C_j)}{2} \tag{2.4.9}$$

Definition 2.4.11 (Distanza probabilistica normalizzata). *Data una pagina x appartenente all'insieme* Π *ed una coppia di concetti* C_i *e* C_j *trovata all'interno della pagina si definisce* distanza probabilistica normalizzata rispetto al dominio D_i*, e la si indica con* $DPN_{D_i}(C_i, C_j)$ *la distanza probabilistica che i due concetti hanno nel dominio* D_i *diviso la distanza tra i due concetti nella pagina.*

$$\mathbf{DPN_{D_i}(C_i, C_j)} = \frac{DP_{D_i}(C_i, C_j)}{DIST(C_i, C_j)} \tag{2.4.10}$$

Definition 2.4.12 (Contributo diretto delle DP). *Dati m* domini D_1,..,D_m *ognuno dei quali composto dai concetti* $C_{d1}, .., C_{dn_d}$*, dove d è l'indice dei domini mentre* n_d *è il numero di concetti del dominio d, ed una pagina x appartenente all'insieme* Π *si definisce* contributo diretto delle DP *relativo al dominio* D_i*, e lo si indica con* $CdDP_{D_i}(x)$*, la somma delle* $DPN_{D_i}(C_{ik}, C_{ih})$ *che superano una certa soglia, tra tutte le coppie di concetti trovati in x ed associati al dominio* D_i*, moltiplicate per la centralità media CENM*(C_{ik}, C_{ih})*.*

$$\mathbf{CdDP_{D_i}(x)} = \sum_{h=1}^{NT} \sum_{k=h+1}^{NT} (DPN_{D_i}(C_{ik}, C_{ih}) \cdot CENM(C_{ik}, C_{ih})) \cdot X_{k,h} \qquad (2.4.11)$$

dove $X_{k,h} = 1$ *se* $DPN_{D_i}(C_{ik}, C_{ih}) \geq$ ***soglia*** *mentre vale 0 negli altri casi.*

Definition 2.4.13 (Contributo indiretto delle DP). *Dati* m domini D_1,..,D_m *ognuno dei quali composto dai concetti* $C_{d1}, .., C_{dn_d}$*, dove* d *è l'indice dei domini mentre* n_d *è il numero di concetti del dominio* d*, ed una pagina* x *appartenente all'insieme* Π *si definisce* contributo indiretto delle DP *reativo al dominio* D_i*, e lo si indica con* $CiDP_{D_i}(x)$*, la somma delle* $DPN_{D_i}(C_{jh}, C_{jk})$ *che superano una certa soglia, tra tutte le coppie di concetti trovati in* x *e associati a qualche dominio* D_j *(con* $j \neq i$*) moltiplicate per la centralità media CENM*(C_{jh}, C_{jk}) *ed il grado di correlazione, GC(*D_i*,* D_i*), tra il dominio* D_i *e il dominio* D_j*.*

$$\mathbf{CiDP_{D_i}(x)} = \sum_{k=1}^{NT} \sum_{h=k+1}^{NT} DPN_{D_i}(C_{jk}, C_{jh}) \cdot CENM(C_{jk}, C_{jh}) \cdot GC(D_i, D_j) \cdot X_{k,h} \qquad (2.4.12)$$

dove $X_{k,h} = 1$ *se* $DPN_{D_i}(C_{jk}, C_{jh}) \geq$ ***soglia*** *mentre vale 0 negli altri casi.*

Peso dovuto all'associazione tra domini correlati

Definition 2.4.14 (Peso indiretto). *Dati m* domini D_1,..,D_m *ognuno dei quali composto dai concetti* $C_{d1}, .., C_{dn_d}$*, dove d è l'indice dei domini mentre* n_d *è il numero di concetti del dominio d, ed una pagina x appartenente all'insieme* Π *si definisce* Peso indiretto di x *reativo al dominio* D_i*, e lo si indica con* $PesoI_{D_i}(x)$*, la somma dei pesi* $Peso_{D_j}(x)$ *della pagina x nei domini* D_j *con* $j \neq i$ *per il grado di correlazione,* $GC(D_i, D_i)$*, tra* D_i *e* D_j*.*

$$\mathbf{PesoI_{D_i}(x)} = \sum_{k=1}^{m} (Peso_{D_k}(x) \cdot GC(D_i, D_j)) \quad con\ j \neq i \tag{2.4.13}$$

Contributo semantico relativo al dominio $\mathbf{D_i}$

Il contributo semantico di una pagina è dipende dal **titolo**, dal **contenuto** della pagina, dalle **keywords** e della **descrizione**. Indichiamo le precedenti parti di interesse rispettivamente con i pedici **t**, **c**, **k** e **d**.

Definition 2.4.15 (Contributo semantico al dominio D_i). *Dati m* domini D_1,..,D_m *ognuno dei quali composto dai concetti* $C_{d1}, .., C_{dn_d}$*, dove d è l'indice dei domini mentre* n_d *è il numero di concetti del dominio d, ed una pagina x appartenente all'insieme* Π *si definisce* Contributo semantico relativo al dominio D_i della pagina x*, e lo si indica con* V_{sem}*, la somma pesata dei diversi contributi diretti e indiretti delle DP relativa a titolo, contenuto, descrizione e keywords.*

$$\mathbf{V_{sem}} = \sum_{p \in (t,c,d,k)} w_p \cdot (CdDP_{D_i}(x) + CiDP_{D_i}(x))_p \tag{2.4.14}$$

dove w_c, w_d, w_t, w_k *sono rispettivamente i pesi del contenuto, della descrizione, del titolo e delle keyword della pagina x, tali che*

$$w_c + w_d + w_t + w_k = 1 \tag{2.4.15}$$

Contributo semantico-sintattico relativo al dominio D_i

Anche il contributo semantico di una pagina dipende dal **titolo**, dal **contenuto** della pagina, dalle **keywords** e della **descrizione**.

Definition 2.4.16 (Contributo semantico-sintattico al dominio D_i**).** *Dati m* domini D_1,..,D_m *ognuno dei quali composto dai concetti* $C_{d1}, .., C_{dn_d}$*, dove d è l'indice dei domini mentre* n_d *è il numero di concetti del dominio d, ed una pagina x appartenente all'insieme* Π *si definisce* Contributo semantico-sintattico relativo al dominio D_i della pagina x*, e lo si indica con* V_{ss}*, la somma pesata dei diversi pesi diretti e indiretti del titolo, contenuto, descrizione e keywords.*

$$\mathbf{V_{ss}} = \sum_{p \in (t,c,d,k)} w_p \cdot (Peso_{D_i}(x) + PesoI_{D_i}(x))_p \tag{2.4.16}$$

dove w_c, w_d, w_t, w_k *sono sempre i pesi del contenuto, della descrizione, del titolo e delle keyword della pagina x.*

2.4.5 Voto complessivo

Definition 2.4.17 (Voto complessivo di una pagina). *Dati m* domini $D_1, .., D_m$ *ognuno dei quali composto dai concetti* $C_{d1}, .., C_{dn_d}$*, dove d è l'indice dei domini mentre* n_d *è il numero di concetti del dominio d, ed una pagina x appartenente all'insieme* Π *si definisce* Voto complessivo di x *relativo al dominio* D_i*, e lo si indica con* $Voto_{Di}(x)$*, la quantità*

$$\mathbf{Voto_{D_i}(x)} = \frac{K_{sint} \cdot Sin(x) + K_{sem} \cdot V_{sem} + K_{ss} \cdot V_{ss}}{K_{sint} + K_{sem} + K_{ss}} \tag{2.4.17}$$

dove $K_{sint}, K_{sem}, K_{ss}$ *sono tre costanti chiamate rispettivamente* ***costante sintattica***, ***costante semantica*** *e* ***costante semantico-sintattica***.

Chapter 3

Il sistema

Nel corso del capitolo verrà presentata l'architettura del sistema realizzato

Il sistema SSA è stato progettato secondo il paradigma degli "agenti". Verrà quindi definito cos'è un agente software e verranno descritti gli agenti progettati mediante il Service Description Language.

3.1 Intelligent Search Agent

Gli **Intelligent Search Agents** sono programmi che filtrano le informazioni ritrovate sul WEB in base agli interessi dell'utente, memorizzandone le scelte e costruendone il profilo. Ad essi si stanno interessando molte tra le più importanti aziende del settore informatico, da Netscape a Microsoft, da IBM a AT&T.

Di norma, un buon agente dovrebbe permettere di utilizzare meno tempo per il reperimento delle informazioni e più tempo per l'analisi di quelle già reperite. Il nome di questi strumenti software richiede un piccolo chiarimento [3]:

- **Intelligent**: sono chiamati così in quanto non si limitano solamente a restituire pagine che contengono le keywords immesse dall'utente (come fanno i motori di

ricerca), ma anche pagine con parole di significato simile alle keywords assicurando che siano utilizzate nel giusto contesto evitando di restituire link inutili;

- **Search**: chiamati così in quanto realizzano, anche se in modo diverso, lo stesso task dei comuni motori di ricerca ossia il reperimento di pagine sul WEB;

- **Agent**: viene chiamato così qualcosa (nel nostro caso, un software) che agisce in base agli interessi di qualcuno. L'idea alla base di questo concetto è che un utente potrebbe fornire all'agente un set di istruzioni ed esso potrebbe eseguirle e riportare in seguito i risultati.

La maggior parte degli strumenti di Information Retrieval su Web non prescindono da un utente estremamente "impegnato" nell'attività di ricerca dell'informazione utile e presuppongono una ricerca in qualche modo "occasionale", non fissata a priori nei tempi. Nel momento specifico in cui una determinata informazione serve, ci si mette a cercarla e la ricerca richiede che vengano compiuti, ogni volta ed in prima persona, un certo numero di azioni alcune delle quali senz'altro ripetitive: per esempio, collegarsi alla pagina di Google (www.google.com) oppure a quella di Altavista (www.altavista.com), impostare i parametri della ricerca, restare collegato mentre si aspettano i risultati ed infine filtrare le risposte del motore. Sarebbe molto comodo, in questi casi, disporre di una sorta di *"segreteria intelligente"* che conosca in linea di massima i propri interessi, sappia prevedere ed anticipare le richieste, ci sostituisca, prendendo autonomamente le decisioni più opportune, sia per le azioni più ripetitive che per quelle meno ripetitive.

L'idea di *Agente di Ricerca Intelligente* cerca di avvicinarsi a questo ideale: in

sostanza, si tratta di incaricare un programma di svolgere, anche a intervalli prefissati, determinate ricerche, chiedendogli di reagire autonomamente ai risultati della ricerca stessa (magari "filtrandoli" attraverso l'uso di criteri che potrebbero essere difficili se non impossibili da impostare direttamente su un motore di ricerca eøraggruppandoli in base al dominio di appartenenza).

Addirittura, se l'agente non è fisicamente localizzato sul proprio computer poiché situato su un server remoto o distribuito fra più server remoti, gli si potrebbero affidare ricerche e compiti da svolgere anche mentre si è scollegati dalla rete. Per esempio, un utente potrebbe chiedere all'agente di cercare informazioni su un certo argomento ed appartenenti ad un certo dominio, e l'agente potrebbe ricercare le pagine Web più aggiornate restituendo un report con le risorse disponibili sul WWW. Tali azioni potrebbero richiedere un giorno intero e l'utente potrebbe, una volta sottomessa la query, continuare i suoi lavori e ottenere i risultati in seguito. A molti, ciò potrebbe risultare sconveniente rispetto ad un motore di ricerca, dove i risultati vengono forniti in brevissimo tempo: in effetti, per veloci ricerche niente è meglio dei motori di ricerca, ma gli agenti di ricerca intelligenti offrono altri vantaggi. Per esempio, un utente potrebbe volere tutte le informazioni più recenti su diversi argomenti e potrebbe volerle ogni mattina sul proprio terminale, alla stregua dei servizi di news personalizzati. Esse verrebbero presentate esattamente come l'utente vuole e l'agente addirittura potrebbe apprendere nel tempo quali sono le sue preferenze in modo da migliorare il servizio.

Per certi aspetti, alcuni strumenti oggi disponibili sul WEB possiedono già, almeno in parte, qualcuna delle caratteristiche esposte: è il caso, ad esempio, dei servizi

di net filtering che permettono di impostare una ricerca da ripetere ad intervalli regolari comunicando i risultati via posta elettronica. Oppure programmi client per ricevere informazioni attraverso meccanismi di *information pushing*: una volta impostati, saranno loro ad occuparsi di collegarsi al server o ai server remoti e a scaricare dati. Il problema, in questi casi, è la carenza, o meglio l'assenza, della capacità di prendere decisioni realmente autonome, reagendo dinamicamente alle caratteristiche dell'universo informativo nel quale si muovono.

La realizzazione [49] di un agente di questo tipo richiede di includere meccanismi per poter effettuare ricerche sulla base di keywords sia multiple sia in combinazione tra di esse, per poter gestire l'esclusione di informazioni non rilevanti rispetto ai domini conosciuti dal sistema.

Data una query, l'agente lavora per soddisfarla considerando più percorsi possibili sul WEB. In genere si parte da una locazione di partenza (location seed) fornita dall'utente e l'agente esplora tutte le altre locazioni collegate alla radice che possono soddisfare la richiesta. Diversi sono i parametri che si possono impostare: la query (per esempio, delle keywords o una frase), da dove partire, quante iterazioni effettuare, sino a che profondità spingersi, quanto tempo impiegare e che algoritmi di ricerca impiegare nella navigazione sul WEB. Sono due le principali classi di algoritmi di ricerca [50]:

- di tipo informato
- di tipo non informato

I primi, o *algoritmi di tipo euristico*, si basano su alcune informazioni quali il

costo dei cammini effettuati o il numero di passi da effettuare a partire dal punto di partenza. Tali algoritmi sono fondati su tali informazioni per potare l'albero o il grafo di ricerca eliminando opportunamente diversi rami. I secondi, invece, non sfruttano nessuna di queste informazioni e si limitano a esplorare esaustivamente l'albero di ricerca o il grafo di ricerca sino a raggiungere l'obiettivo. In genere gli algoritmi di tipo euristico permettono di ottenere performance migliori ed è per questo che vengono sfruttati dalla maggior parte degli agenti di ricerca.

La progettazione si basa su quattro fasi principali:

- *Inizializzazione*
- *Percezione*
- *Azione*
- *Effetto*.

Nella fase di *inizializzazione* vengono fissate le variabili, le strutture e tutti i dati necessari per effettuare la ricerca. Vengono impostate la query, gli obiettivi della ricerca, il numero massimo di pagine Web da visitare, una locazione di partenza ed un metodo di ricerca se ne sono disponibili più di uno.

La fase di *percezione* sfrutta i dati forniti dall'utente per contattare un sito e prelevare da esso le informazioni. In tale fase l'agente è in grado di capire se la pagina soddisfa la query e di identificare percorsi verso altri siti Web (per esempio link ad altre pagine).

Nella fase di *azione* l'agente raccoglie tutte le informazioni disponibili e determina

se l'obiettivo è stato raggiunto. Se non è stato raggiunto, l'agente decide dove proseguire la ricerca: proprio in questo sta l'intelligenza dell'agente e il metodo di ricerca usato indica quanto esso sia abile.

Nella fase di *effetto* viene presentata all'utente una lista delle migliori locazioni trovate.

L'unione di tali fasi ed una connessione ad Internet permettono di realizzare un Agente di Ricerca Intelligente, o, come viene spesso chiamato in letteratura, un **Web Hunter** [51].

Durante la fase di inizializzazione vengono create apposite strutture dati per gestire le informazioni sui siti che verranno visitati: in genere è una lista a puntatori o un array. Ogni elemento o nodo contiene informazioni sull'URL della pagina e la sua locazione e un valore che rappresenta il costo per raggiungerla a partire dalla locazione di partenza (se si usano algoritmi di ricerca euristici). Il nodo radice contiene un valore detto seme che rappresenta la locazione da cui si parte nella ricerca: esso può essere un indirizzo IP messo a caso ma in genere è consigliabile partire dall'URL di un motore di ricerca o dall'indirizzo IP di una pagina attinente all'argomento della ricerca.

Effettuata l'inizializzazione, il Web Hunter entra nella fase di percezione aprendo una socket sulla porta HTTP (la porta 80) verso l'URL specificato. Stabilita la connessione, si sfrutta il comando HTTP "GET" per richiedere al server la pagina [**?**]; essa viene sottoposta a parsing al fine di trovare i link in essa contenuti e per determinare se essa è attinente alla query dell'utente. Il parser solitamente si basa sulla ricerca dei tag HTML "HREF=" e "SRC=" che indicano la presenza di un link nella pagina. È da notare che il parser può essere reso flessibile impostando diversi

metodi di ricerca all'interno della pagina: si possono, per esempio, ottenere file di grafica semplicemente cercando file con estensione "JPG" oppure "GIF". Le pagine Web possono contenere un numero molto elevato di link ad altre: per questo può essere utile, se l'agente lo consente, impostare un upper bound sul numero di link da prelevare da ogni pagina.

Una volta catturata la pagina ed estratte le informazioni più importanti, si entra nella fase in cui il Web Hunter manifesta la sua intelligenza; infatti, se l'obiettivo è stato raggiunto (confrontando, ad esempio, il numero di siti Web visitati con quello impostato dall'utente), il suo lavoro è terminato con successo. Se ciò non si è verificato, l'agente decide qual è la successiva azione da intraprendere: il cuore della decisione è rappresentato da quale algoritmo di ricerca utilizzare; infatti, i link estratti da una pagina e che, quindi, devono ancora essere visitati, vengono aggiunti alla lista spiegata in precedenza assieme al loro costo. La modalità di inserimento e, soprattutto, di prelievo è funzione dell'algoritmo di ricerca. Più il metodo è informato, maggiori sono le probabilità di raggiungere gli obiettivi prefissati. Ci sono diversi elementi per valutare il costo di una pagina, oltre a quelli già esposti in precedenza:

- Valutazione del suo round-trip-time tramite il comando "ping". Quanto più è basso, tanto più sarà basso il costo della pagina rispetto alla radice

- Valutazione della nazionalità della pagina. In una ricerca che si basi esclusivamente su siti, per esempio, americani, si potrebbe dare a pagine che non abbiano estensione compresa tra "COM", "NET", "EDU", "ORG", "GOV" o "MIL" (che si suppone essere pagine americane) un valore di costo elevato per penalizzarle

Se lo spazio di ricerca si esaurisce (non ci sono altri nodi da esplorare), l'agente

può richiedere un nuovo seme e ripartire oppure semplicemente terminare, entrando nella fase di effetto dove viene presentata all'utente una lista dei risultati ottenuti.

Il Web Hunter lavora muovendosi dalla fase di inizializzazione a quella di effetto, passando per quelle di percezione e azione. Tutto ciò sino a quando l'obiettivo non viene raggiunto. Diverse sono le applicazioni in cui può trovare uso un agente di questo tipo; infatti, dotandolo di un opportuno dizionario, può diventare un potente motore di ricerca domain-dependant oppure può essere modificato per trasformarsi da Web Hunter in Document Hunter in modo tale da cercare file di documenti in locale piuttosto che pagine Web.

Quando si testa un software di questo tipo è preferibile farlo localmente sino a quando non sia stabilita con certezza la sua stabilità [51]. Inoltre è preferibile che tali agenti si identifichino sul WEB utilizzando il campo HTTP "User-agent" e si registrino presso il *"Web Robots Database"* [53] in modo tale che possa essere identificato su un server protetto nel caso in cui debba effettuare su di esso diverse operazioni. Un'ultima nota che deve essere tenuta presente durante lo sviluppo e l'uso di agenti di questo tipo è la presenza su diversi siti Web di un file denominato *robot.txt*. Gli agenti di ricerca intelligente dovrebbero testare sempre la sua presenza, in quanto contiene informazioni rilevanti quali l'accettazione o meno da parte del sito di questo tipo di agenti oppure la politica di utilizzo delle informazioni in esso contenute. Altre pagine possono contenere tag HTML quali *<Meta Name="Robots" Content="Nofollw">* che indicano che è vietato il parsing della pagina stessa.

Per concludere il paragrafo, c'è da dire che di programmi che rispecchiano fedelmente quanto scritto ne esistono ben pochi, almeno in versioni "avanzate". È prevedibile, però, che il settore degli *Agenti di Ricerca Intelligente* conoscerà nei prossimi

anni un'evoluzione tale da far sembrare i primi strumenti di oggi solo rozze e primitive approssimazioni di applicazioni assai più sofisticate, potenti ed autonome.

3.1.1 La struttura ad agenti

Il sistema SSA è stato progettato secondo il paradigma degli "agenti". Abbiamo già illustrato cosa è un agente software e nel seguito di questo paragrafo verranno descritti gli agenti progettati mediante il *Service Description Language*.

Il Service Description Language (SDL)

Gli agenti progettati nel nostro sistema verranno di seguito descritti tramite uno specicifico linguaggio di descrizione: il **Service Description Language** (**SDL**).

Si tratta di un linguaggio che permette di descrivere gli agenti dal punto di vista dei servizi offerti. La descrizione di un servizio si basa sulla specifica dei seguenti elementi [11][15]:

- **Nome del servizio**: è un'espressione costituita da un verbo e da un nome (nella forma verbo: nome) che specifica il servizio offerto
- **Input**: è un elenco degli ingressi richiesti dal servizio
- **Output**: è un elenco delle uscite fornite dal servizio
- **Attributi**: alcuni servizi possono avere degli attributi associati. Per esempio, costo del servizio e tempo medio di esecuzione

Quando si devono elencare gli input e gli output, è necessario specificare le variabili ed i tipi ad esse associati. Ciò viene fatto tramite un apposito formalismo:

Definition 3.1.1 (item s: τ). *Se* s *è una variabile avente valori nell'insieme* τ*, allora l'espressione* s: τ *è detta* ***item****.*

Ogni servizio può richiedere zero, uno o più input. Alcuni di essi sono obbligatori (il servizio non può essere portato a termine senza di essi), mentre altri sono facoltativi (non sono necessari, ma possono, se forniti, aumentare l'efficienza e la qualità del servizio offerto).

Prima di passare alla definizione dei servizi offerti dagli agenti progettati, è necessario fornire qualche definizione.

Definition 3.1.2 (item atomici). *Se s:* τ *è un item, allora l'espressione*

$$< I > s : \ \tau < \backslash I > (rispettivamente < MI > s : \ \tau < \backslash MI >)$$

indica un item di ingresso atomico *(rispettivamente* item di ingresso obbligatorio atomico*), mentre l'espressione*

$$< O > s : \ \tau < \emptyset >$$

indica un item di uscita atomica.

Definition 3.1.3 (lista di item). *Se* s_1*:* τ_1*,...,* s_n*:* τ_n *sono degli item, allora l'espressione*

$$< I > s_1 : \tau_1, ..., s_n : \tau_n < \backslash I > \ ^{1}$$

[1] *abbreviazione di* $< I > s_1 : \tau_1 < \backslash I > .... < I > s_n : \tau_n < \backslash I >$

indica una lista di item di ingresso, mentre le espressioni

$$< MI > s_1 : t_1, ..., s_n : t_n < \backslash MI > e < O > s_1 : t_1, ..., s_n : t_n < \backslash O > \ ^2$$

indicano una lista item di ingressi obbligatori ed una lista di uscite.

Definition 3.1.4 (Service Description). *Siano s_n il nome di un servizio, i_1,....,i_n item di ingresso atomici, mi_1,....,mi_n item di ingresso obbligatori e o_1,....,o_n item di uscita atomici. Viene chiamata* ***Descrizione del Servizio (Service Description)****, la seguente espressione:*

$< S >$ s_n

$mi_1 ... mi_k$

$i_1 ... i_n$

$o_1 ... o_r$

$< \backslash S >$

Tramite questo formalismo, è possibile presentare gli agenti dal punto di vista dei servizi da essi offerti.

[2] *abbreviazioni, rispettivamente, di* $< MI > s_1 : \ \tau_1 < \backslash MI > .... < MI > \ s_n : \ \tau_n < \backslash MI > e < O > \ s_1 : \ \tau_1 < \backslash O > .... < O > \ s_n : \ \tau_n < \backslash O >$

Struttura ad agenti del sistema progettato

Il **sistema SSA** (**Semantic Search Agent**) progettato può essere visto come costituito da due agenti principali:

- ***Spider Agent***
- ***Mining Agent***

Spider Agent

È l'agente che si occupa di reperire le pagine Web sulla base delle query immesse dall'utente e di memorizzarle nel sistema. I servizi offerti, descritti tramite il *Service Description Language*, sono:

$< S >$ Adapt: *query*
$< MI >$ *Query*: String $< \backslash MI >$
$< O >$ *AdaptedQuery*: String $< \backslash O >$
$< \backslash S >$

Questo servizio accetta come input la *query* sottoposta dall'utente (costituita da una stringa contenente singole keywords o intere frasi racchiuse tra virgolette). Compito del servizio è adattare la query immessa dall'utente alla sintassi adottata dai diversi motori di ricerca interrogati.

$< S >$ Submit: *query*
$< MI >$ *AdaptedQuery*: String $< \backslash MI >$
$< MI >$ *MaxRes*: Integer $< \backslash MI >$
$< O >$ *Results*: WebPage $< \backslash O >$
$< \backslash S >$

Questo servizio accetta come input la *query adattata*, insieme al massimo numero di link che ciascun motore deve prelevare dal WEB, e la sottopone al motore di ricerca corrispondente.

L'output sarà costituito dalla pagina restituita dal motore di ricerca contenente i link trovati.

$< S >$ ResultsParse: *Results*
$< MI >$ *Results*: WebPage $< \backslash MI >$
$< O >$ *Links*: Strings $< \backslash O >$
$< \backslash S >$

Questo servizio accetta come input la pagina Web restituita dal motore di ricerca e fornisce come output i link in essa contenuti.

$< S >$ WebCatch: *Links*
$< MI >$ *Link*: String $< \backslash MI >$
$< MI >$ *MaxProf*: Integer $< \backslash MI >$
$< O >$ *Page*: WebPage $< \backslash O >$
$< O >$ *Average Position*: Float $< \backslash O >$
$< O >$ *Depth*: Integer $< \backslash O >$
$< \backslash S >$

Questo servizio accetta come input uno dei link contenuti all'interno delle pagine Web, insieme alla profondità massima che ogni pagina Web deve avere rispetto al motore di ricerca affinchè essa venga prelevata dal sistema, e fornisce come output la corrispondente WebPage. Vengono inoltre restituiti, di ogni pagina Web, la posizione mediata e la profondità rispetto al motore di ricerca (parametri necessari ad alcuni servizi offerti dal Mining Agent). I concetti di profondità e posizione mediata verranno

ampiamente spiegati nel corso del capitolo successivo.

$< S >$ DocumentParse: *WebPage*
$< MI >$ *Document*: WebPage $< \backslash MI >$
$< O >$ *Links*: Strings $< \backslash O >$
$< \backslash S >$

Questo servizio accetta come input una delle pagine prelevate dal WEB e restituisce in output i link in esse contenuti.

$< S >$ Construct: *Repository*
$< MI >$ *Document*: WebPage $< \backslash MI >$
$< MI >$ *Link*: String $< \backslash MI >$
$< O >$ *Web Repository*: DataBase $< \backslash O >$
$< \backslash S >$

Questo servizio accetta come input i link trovati ed i vari documenti prelevati dal WEB e costruisce il Web Repository di sistema.

Mining Agent

È l'agente che si occupa di analizzare le pagine Web prelevate dallo Spider, al fine di trovare gli elementi necessari all'attribuzione del voto semantico, un valore che misura l'attinenza ad un certo dominio delle pagine reperite. Si occupa, inoltre, dell'attribuzione del voto stesso. I servizi offerti, sono:

$< S >$ Get: *Content*
$< MI >$ *Document*: WebPage $< \backslash MI >$
$< O >$ *Content*: Text $< \backslash O >$
$< \backslash S >$

Questo servizio accetta come input le varie pagine Web prelevate dal sistema e fornisce come uscita il loro contenuto semantico, ossia le pagine senza i tag HTML.

$< S >$ Preprocess: *Content*
$< MI >$ *Content*: Text $< \backslash MI >$
$< MI >$ *Knowledge Base*: Database $< \backslash MI >$
$< O >$ *Adapted SemanticContent*: Text $< \backslash O >$
$< \backslash S >$

Questo servizio permette di preprocessare il contenuto semantico estrapolato da ogni pagina Web al fine di sostituire eventuali caratteri di escape con i corrispondenti caratteri ASCII. I caratteri di escape sono preceduti dal simbolo & e servono per codificare in HTML speciali caratteri (per esempio, in HTML la "e commerciale" o &, viene codificata con la sequenza &).

Tale servizio, inoltre, consente di eliminare, all'interno del contenuto semantico estrapolato, eventuali stop-words, ossia quelle parole che non danno contributo alla semantica del documento. Per effettuare queste operazioni, è necessario accedere ad una base di conoscenza (Knowledge Base) ove è contenuta una lista dei caratteri di escape noti (con le rispettive corrispondenze) ed una lista delle stop-words.

$< S >$ Get: *Title*
$< MI >$ *Document*: WebPage $< \backslash MI >$
$< O >$ *WebTitle*: Text $< \backslash O >$
$< \backslash S >$

Questo servizio accetta come input le varie pagine Web prelevate dal sistema e fornisce come uscita il contenuto del tag HTML Title.

$< S >$ Get: *Meta Description*
$< MI >$ *Document*: WebPage $< \backslash MI >$
$< O >$ *Description*: Text $< \backslash O >$
$< \backslash S >$

Questo servizio accetta come input le varie pagine Web prelevate dal sistema e fornisce come uscita il contenuto del Meta Tag Description.

$< S >$ Get: *Meta Keywords*
$< MI >$ *Document*: WebPage $< \backslash MI >$
$< O >$ *Keywords*: Text $< \backslash O >$
$< \backslash S >$

Questo servizio accetta come input le varie pagine Web prelevate dal sistema e fornisce come uscita il contenuto del Meta Tag Keywords.

$< S >$ Get: *Semantic Vote*
$< MI >$ *MaxRes*: Integer $< \backslash MI >$
$< MI >$ *Average Position*: Float $< \backslash MI >$
$< MI >$ *Depth*: Integer $< \backslash MI >$
$< MI >$ *Adapted Content*: Text $< \backslash MI >$
$< MI >$ *WebTitle*: Text $< \backslash MI >$
$< MI >$ *Description*: Text $< \backslash MI >$
$< MI >$ *Keywords*: Text $< \backslash MI >$
$< MI >$ *Knowledge Base*: Database $< \backslash MI >$
$< O >$ *Semantic Vote*: Floats $< \backslash O >$
$< \backslash S >$

Questo servizio accetta come input il massimo numero di link che l'utente ha chiesto allo Spider Agent di prelevare all'atto della sottomissione della query, la posizione mediata di una pagina Web, la sua profondità rispetto al motore di ricerca, il suo contenuto semantico, il suo titolo ed il contenuto dei Meta Tag Description e Keywords e restituisce in uscita il voto semantico. Per effettuare questa operazione, è necessario accedere ad una base di conoscenza (**Knowledge Base**) ove è contenuta una lista delle parole e dei concetti conosciuti dal sistema.

3.2 Architettura

L'architettura del sistema **SSA** (**Semantic Search Agent**) è rappresentata in figura 3.1; illustriamo dettagliatamente i componenti che ne fanno parte.

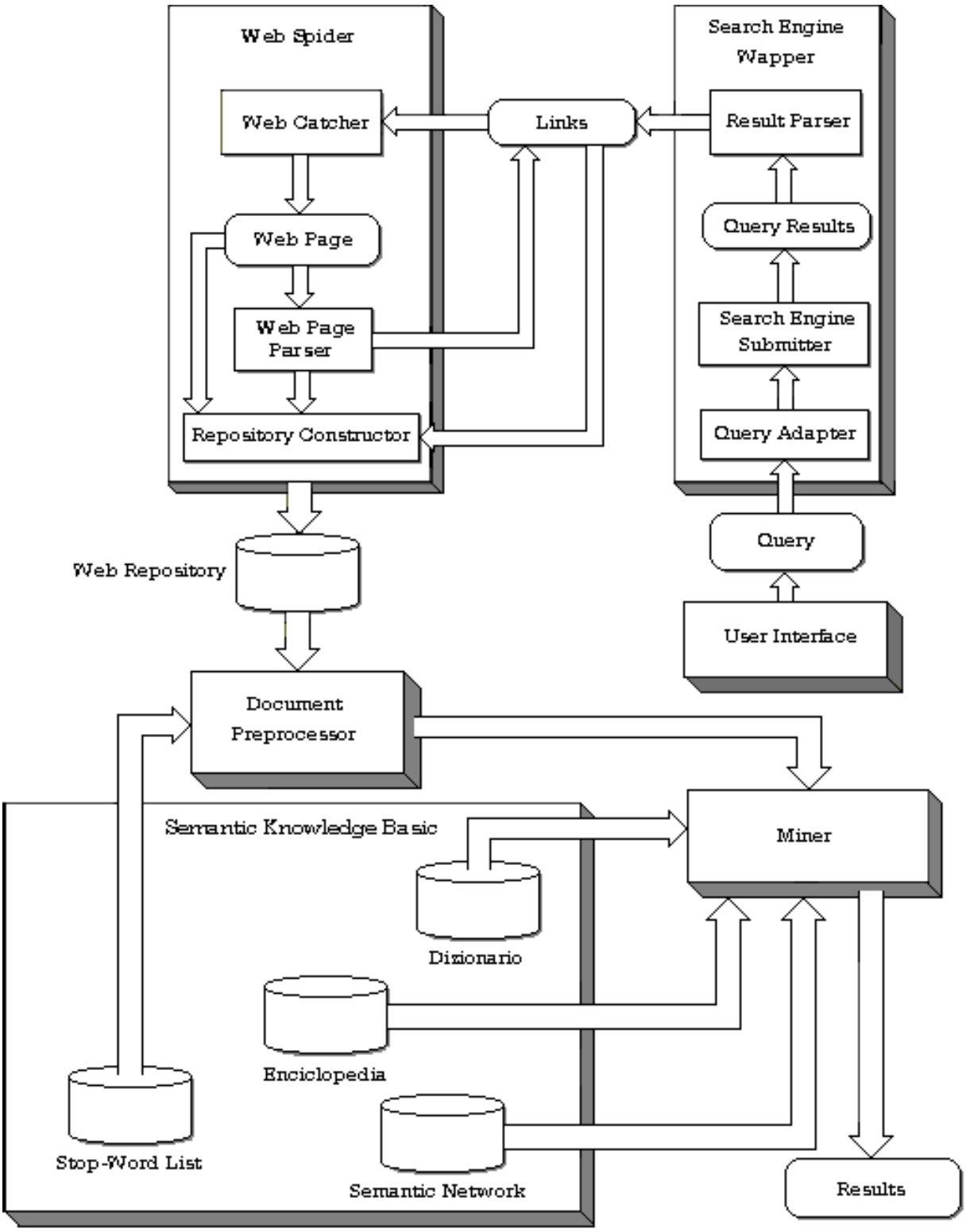

Figure 3.1: Architettura del sistema

3.2.1 User Interface

La **User Interface - Interfaccia Utente** è delegata a chiedere all'utente, tra le altre cose, la query con le parole chiave che egli vuole cercare sul WEB.

Figure 3.2: Menu Iniziale

Essa si presenta come in figura 3.2 ed i suoi pulsanti sono:

- *Gestione Motori* per inserire o eliminare uno dei motori di ricerca utilizzati dal sistema
- *Nuova Query* per la gestione di una nuova query da sottoporre al sistema
- *Carica Query* per la gestione delle query già sottoposte al sistema

- *Gestione Rete Semantica* per la gestione della Rete Semantica usata dal sistema per la valutazione dell'attinenza delle pagine ai domini
- *Gestione Dizionario* per la gestione della lista delle parole di interesse di tutti i domini
- *Impostazioni* per la gestione dei parametri dell'applicazione
- *Esci* per terminare l'applicazione

Tali funzionalità verranno spiegate in maniera dettagliata successivamente.

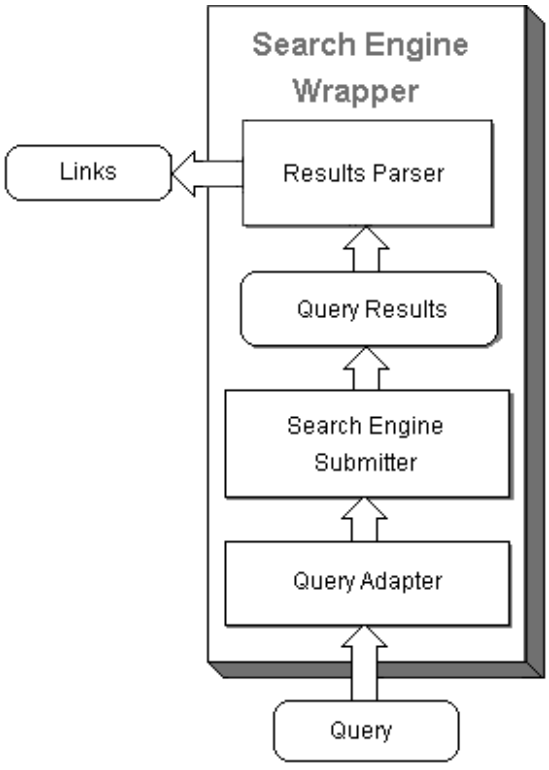

Figure 3.3: Search Engine Wrapper

3.2.2 Search Engine Wrapper

Il modulo **Search Engine Wrapper** è rappresentato in figura 3.3

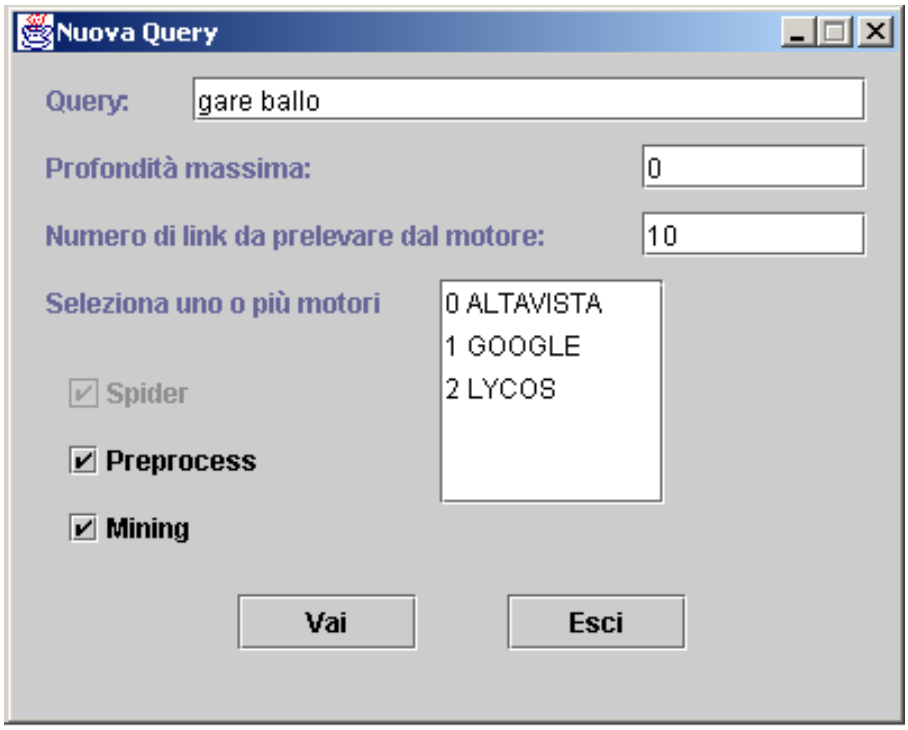

Figure 3.4: Esempio di Query

Questo modulo prende la query inserita mediante l'interfaccia utente, mostrata in figura 3.4, e la adattata alle specifiche sintassi dei motori di ricerca, mediante il *Query Adapter*, che, partendo dalla query effettuata nella sintassi del sistema, crea la stringa di interrogazione per i singoli motori di ricerca da interrogare. Siccome i motori di ricerca utilizzano regole differenti per l'inserimento della query, viene definita una sintassi unica a cui dovrà attenersi l'utente del sistema. Tale sintassi è simile a quella utilizzata dal motore di ricerca Google [56][57] ed è definita come segue

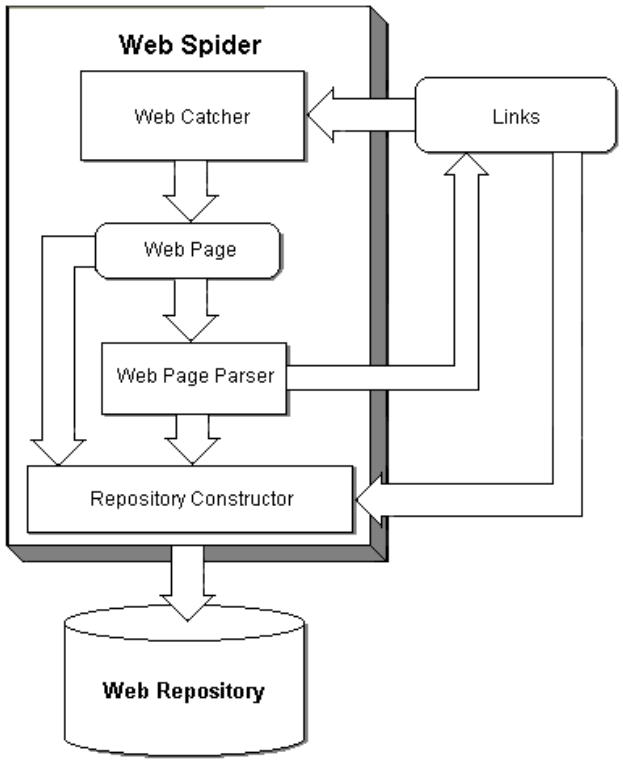

Figure 3.5: Web Spider

Definition 3.2.1 (Sintassi della query). *Le parole immesse sono tutte in AND tra loro, ossia devono essere tutte presenti nel testo della pagina affinché essa venga restituita dal motore di ricerca, ed inoltre è possibile immettere query contenenti intere frasi racchiudendo tali frasi tra virgolette*

Allo scopo di rendere del tutto trasparente per l'utente l'interrogazione dei motori di ricerca, il *Search Engine Wrapper* si occupa della sottomissione delle query adattate dal *Query Adapter* ai motori di ricerca e di prelevarne i risultati. Il *Search Engine Submitter* sottopone la query ad un motore di ricerca ottenendo la pagina con i

risultati da esso restituiti ed il *Results Parser* analizza tale pagina allo scopo di prelevarne i link in essa contenuti.

3.2.3 Web Spider

Il modulo **Web Spider** è rappresentato in figura 3.5

Le pagine corrispondenti ai link restituiti dai motori di ricerca vengono prelevate e memorizzate nel repository locale dal *Web Spider*. Di ogni pagina viene effettuato il prelievo dal *Web Catcher* ed un parsing finalizzato alla ricerca dei link contenuti nella pagina da parte del *Web Page Parser*. In tal modo è possibile seguire la struttura di un sito Web fino alla profondità scelta dall'utente. Le pagine prelevate ed i link trovati vengono inseriti nel *Web Repository* che contiene le informazioni necessarie per ricostruire localmente le pagine prelevate e la struttura del sito. La costruzione del *Web Repository* è delegata al *Repository Constructor*, che provvede all'inserimento delle pagine Web e dei link trovati nel database di sistema.

3.2.4 Document Preprocessor

Il modulo **Document Preprocessor**è rappresentato in figura 3.6.

Le pagine Web sono costituite da tre tipi di informazioni:

- **Tag HTML**, il cui significato riguarda essenzialmente la formattazione del testo nella pagina
- **Meta Tag HTML**, in cui l'autore della pagina può inserire una descrizione

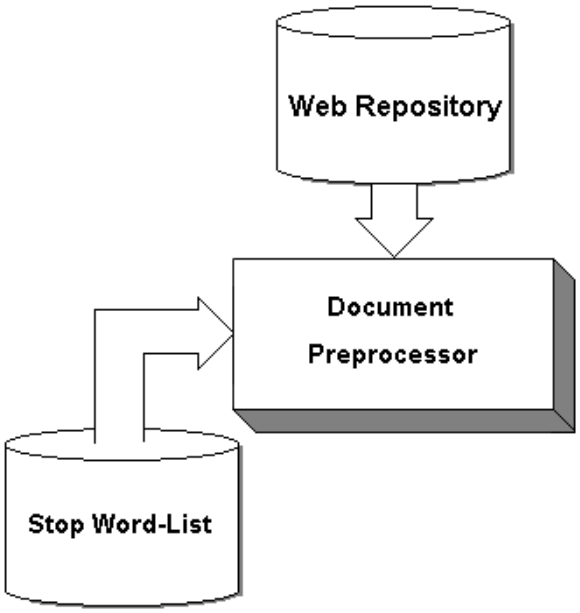

Figure 3.6: Document Preprocessor

sintetica del sito, le parole chiave che lo caratterizzano, l'autore del sito ed, eventualmente, altre informazioni. La presenza dei Meta Tag non è obbligatoria ed è cura dell'autore inserirli. Peraltro, l'utente è spinto ad immettere un contenuto coerente in tali campi in quanto alcuni motori di ricerca ne fanno uso per semplificare l'indicizzazione delle pagine e, talvolta, premiano le pagine in cui i Meta Tag sono particolarmente significativi

- **Testo**

Il contenuto semantico di una pagina è chiaramente concentrato nel testo, ma anche i Meta Tag hanno la loro importanza, in quanto sintetizzano il contenuto semantico della pagina.

Nella metrica utilizzata dal sistema realizzato, che verrà ripresa successivamente,

vengono considerati i seguenti campi della pagina HTML:

- *Testo*
- *Titolo*
- *Meta Tag "Description"*
- *Meta Tag "Keywords"*

Per estrarre tali informazioni dalla pagina è necessario analizzarne i tag HTML. Tale operazione viene effettuata dal *Document Preprocessor*, che estrapola il contenuto semantico dalla pagina HTML, lo separa nelle quattro componenti succitate e le memorizza nel database di sistema. Inoltre il *Document Preprocessor* provvederà anche ad una pulizia del documento dalle Stop Words e alla sostituzione dalle stringhe che servono per la formattazione del documento in HTML con le relative combinazioni di caratteri.

3.2.5 Semantic Knowledge Base

Il cuore dell'agente di *Mining* è costituito dalla **Semantic Knowledge Base**, rappresentato in figura 3.7

In essa sono contenute le informazioni necessarie per analizzare il contenuto semantico di una pagina e misurarne l'attinenza con un certo dominio. A tal fine occorre valutare una metrica che consente di attribuire ad ogni pagina un voto che misura il grado di appartenenza della pagina ai diversi domini in esame.

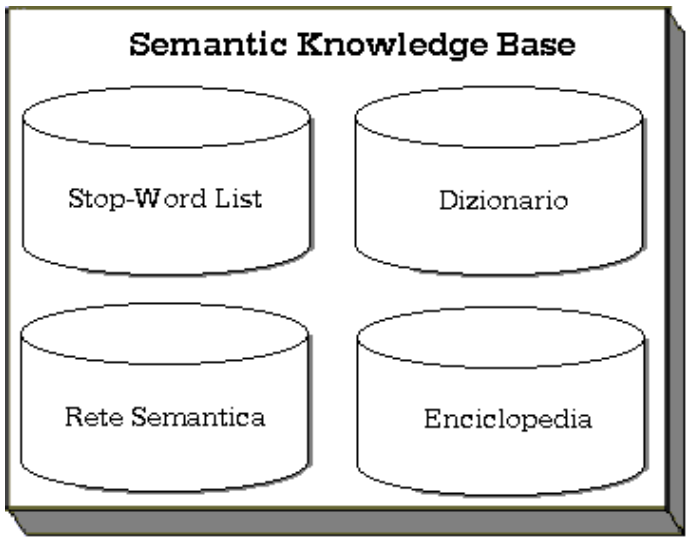

Figure 3.7: Semantic Knowledge Base

La metrica usata dal sistema realizzato, come detto in precedenza, considera cinque tipi di informazioni:

- Profondità della pagina rispetto al motore di ricerca
- Posizione mediata della pagina
- Peso dei vocaboli presenti
- Premio dell'accoppiata di vocaboli che si riferiscono a concetti tra loro correlati, pesato con la distanza dei vocaboli
- Premio dovuto a domini con un grado di correlazione diverso da zero

I primi due contributi vengono valutati dal *Repository Constructor* all'atto della costruzione del *Web Repository*.

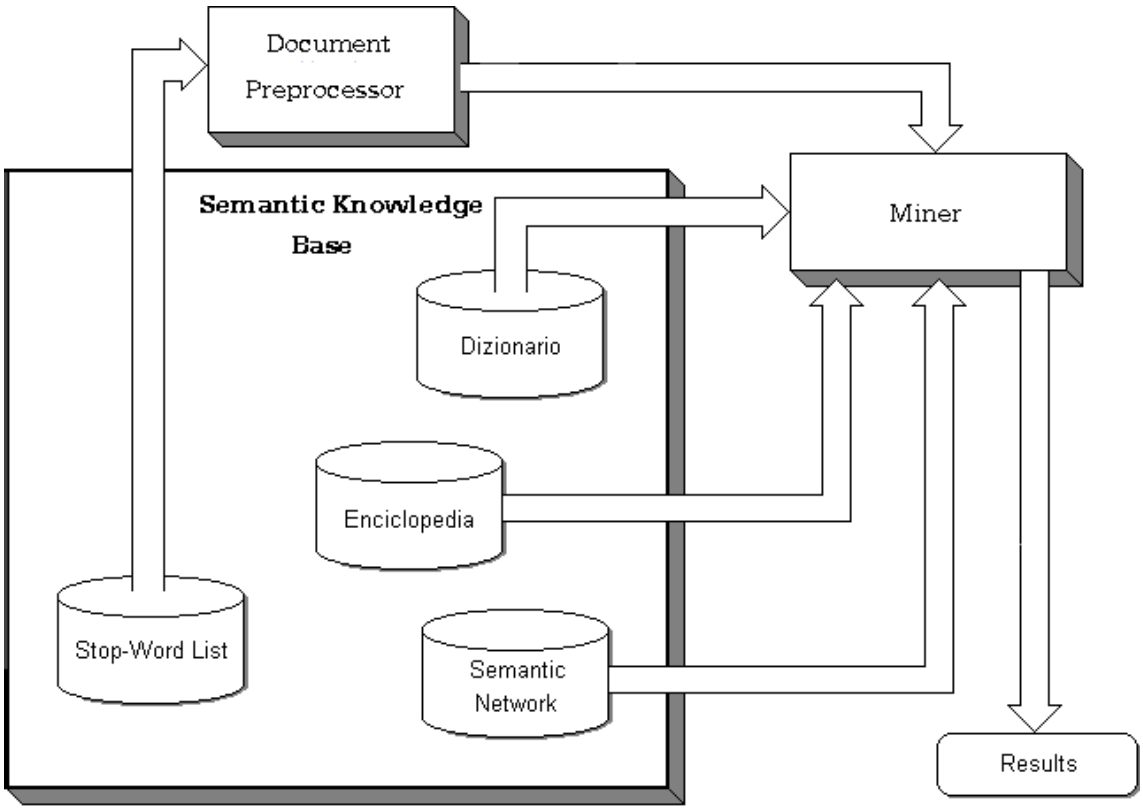

Figure 3.8: Esempio di Query

Gli ultimi tre contributi, invece, devono essere valutati dal *Mining Agent*. A tal fine, la prima operazione che esso deve effettuare è l'eliminazione delle stop-words, ossia di quelle parole che non danno contributo alla semantica del documento. Tali parole sono tipiche di ogni lingua e, poiché nel nostro sistema viene considerata esclusivamente la lingua italiana, la lista delle stop-words della lingua italiana sono memorizzate nel Database di sistema nella *Stop-Words List*.

Ottenuto il documento depurato dalle stop-words, occorre valutare i tre contributi relativi alla presenza dei vocaboli, all'accoppiata di vocaboli relativi a concetti legati in qualche modo tra loro e al grado di associazione tra i domini.

Il primo contributo è valutato confrontando i vocaboli contenuti nella pagina con le parole presenti nel dizionario, il quale contiene le parole appartenenti a tutti i domini conosciuti dal sistema sistema.

Il secondo contributo, invece, viene valutato a partire da una tabella nella quale sono presenti il concetto associato ad ogni parola trovata e la posizione del vocabolo all'interno del documento. A partire dalla tabella, si vanno a considerare le accoppiate dei vocaboli corrispondenti a concetti correlati tra loro e se ne calcola la distanza probabilistica. Tale distanza è valutata a partire dalla *Semantic Network*, che è una struttura a grafo che descrive tutte le possibili relazioni (di similitudine, di generalizzazione, specializzazione, etc.) esistenti tra concetti noti.

Infine, l'ultimo contributo è valutato tenendo presente il peso di una pagina rispetto ad un dominio, ed assegnando, in base a questo, un premio ai domini associati. Al fine di assegnare questi ultimi tre contributi la Semantic Knowledge Base contiene quattro strutture in cui sono memorizzate in maniera permanente le stopwords, il Dizionario, la rete semantica tra i concetti e la rete delle associazioni tra i domini.

3.2.6 Miner

Il **Miner**, consultando la *Semantic Knowledge Base*, valuta la metrica di attinenza delle pagine ai diversi domini conosciuti dal sistema, fornendo i risultati all'utente. Esso è rappresentato in figura 3.8

3.3 Struttura del Web Repository

Il **Web Repository** è la struttura dati che contiene le informazioni relative alle pagine Web prelevate dallo *Spider* e preprocessate dal *Document Preprocessor*. Il diagramma Entità-Relazioni è mostrato in figura 3.9.

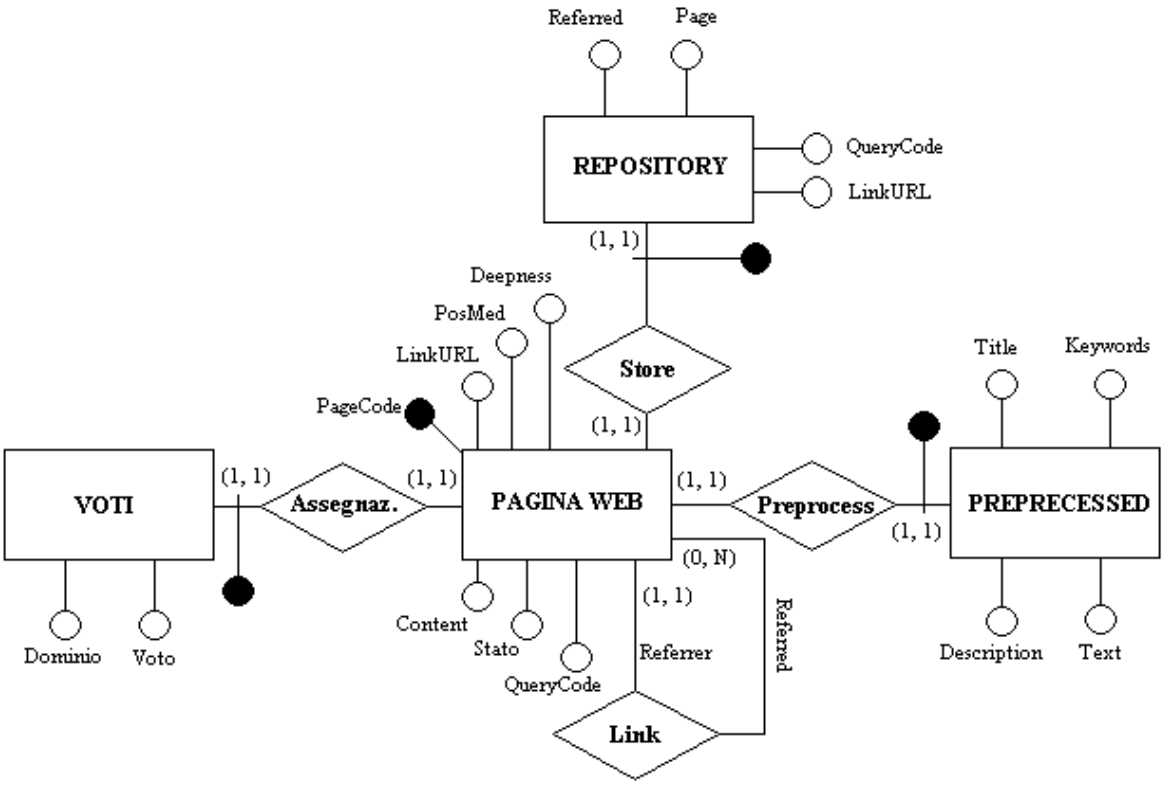

Figure 3.9: Modello E-R del Web Repository

A tale modello concettuale corrisponde il seguente schema di Database:

WEBPAGE(**PageCode**, LinkURL, Referrer, Deepness, PosMed, Content, Stato)

PREPROCESSED(**PageCode**, Title, Keywords, Description, Text)

REPOSITORY(**PageCode**, LinkURL, QueryCode, Page, Referrer)

3.4 Struttura della Semantic Knowledge Base

La **Semantic Knowledge Base** è composta dalla **Stop-Words List**, dal **Dizionario**, dalla **Semantic Network** e dalla **mappa dei domini**. Per quanto riguarda la lista delle stop-words, essa verrà immagazzinata in una tabella apposita con un unico campo atto a contenere le parole considerate stop-words. Per costruire lo schema delle tabelle atte a contenere il *Dizionario*, la *Semantic Network* e la *mappa dei domini* è di aiuto il diagramma Entità-Relazioni mostrato in figura 3.10

PAROLA(**Name**, **ConceptCode**, **DomCode**, Peso, centralità)

CONCETTO(**ConceptCode**, Name)

DOMINIO(**DomCode**, Nome)

LINK(**Source**,**Target**, costo, dominio)

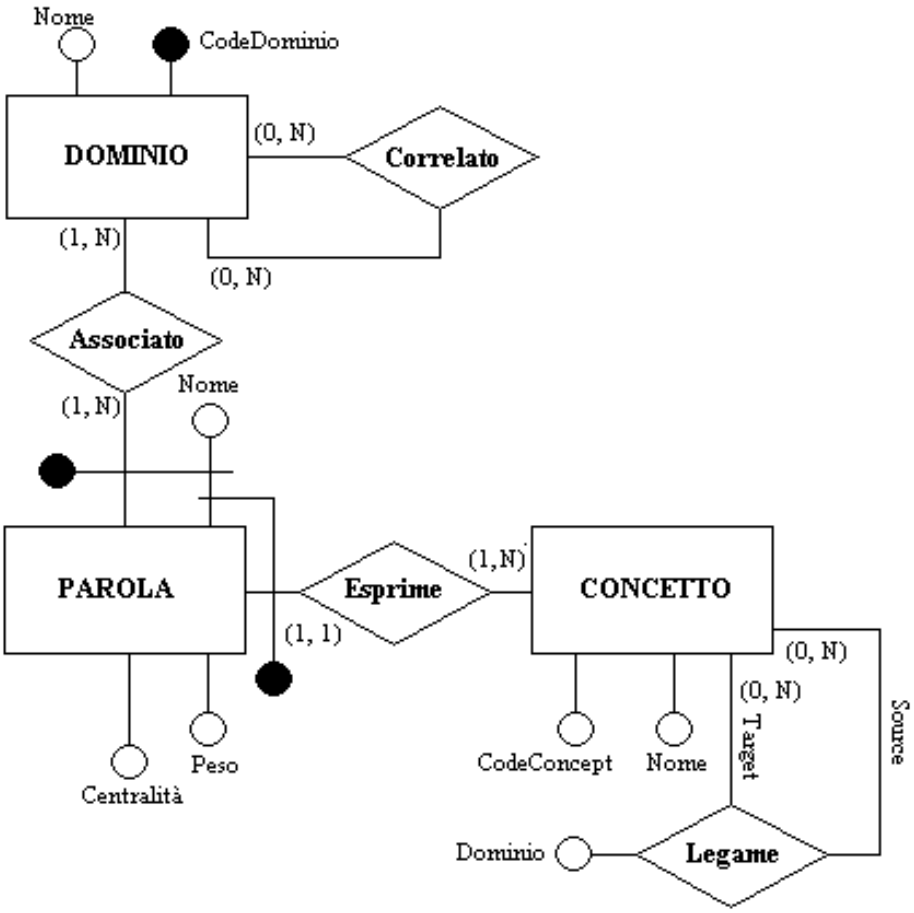

Figure 3.10: Modello E-R del Semantic Knowledge Base

Chapter 4

Sperimentazione

In questo capitolo viene mostrata una prova sperimentale effettuata sul sistema progettato. La prova è stata effettuata utilizzando la seguente configurazione della macchina:

- Processore Intel Pentium II 400 MHz
- 512 MB di memoria
- S.O. Microsoft Windows 2000 Professional
- Oracle Enterprise Edition 8.1

4.1 Parametri del sistema

Come illustrato nel capitolo relativo agli aspetti teorici, le formule che permettono di attribuire il voto semantico ad una pagina web sono le seguenti:

1. $\mathbf{Sin(x)} = \frac{q(x)}{1+b \cdot K_b}$

2. $\mathbf{V_{ss}} = \sum_{p\in(t,c,d,k)} w_p \cdot (Peso_{D_i}(x) + PesoI_{D_i}(x))_p$

3. $\mathbf{V_{sem}} = \sum_{p\in(t,c,d,k)} w_p \cdot (CdDP_{Di}(x) + CiDP_{Di}(x))_p$

4. $\mathbf{Voto_{D_i}(x)} = \frac{K_{sint}\cdot Sin(x)+K_{sem}\cdot V_{sem}+K_{ss}\cdot V_{ss}}{K_{sint}+K_{sem}+K_{ss}}$

I parametri che l'utente può inserire per gestire la modalità di votazione delle pagine sono i seguenti:

- **Costante sintattica $\mathbf{k_{sint}}$**, che misura l'importanza data al contributo sintatttico nella valutazione della pagina
- **Costante semantica $\mathbf{k_{sem}}$**, che misura l'importanza data al contributo semantico nella valutazione della pagina
- **Costante sintattico-sintattica $\mathbf{k_{ss}}$**, che misura l'importanza data al contributo sintatttico-sintattico nella valutazione della pagina
- **Soglia sematica T**, che rappresenta il minimo valore di attinenza di una coppia di token ritrovata nel testo affinché la loro accoppiata possa essere considerata significativa
- **Peso del titolo $\mathbf{w_t}$**, che misura l'importanza data al titolo di una pagina web nel calcolo del voto semantico
- **Peso della descrizione $\mathbf{w_d}$**, che misura l'importanza data al contenuto del Meta Tag description di una pagina web nel calcolo del voto semantico
- **Peso delle keywords $\mathbf{w_k}$**, che misura l'importanza data al contenuto del Meta Tag keywords di una pagina web nel calcolo del voto semantico

- **Peso del contenuto $\mathbf{w_c}$**, fattore che misura l'importanza data al contenuto semantico di una pagina web nel calcolo del voto semantico
- **Costante di penalizzazione della profondità $\mathbf{k_b}$**, fattore che controlla la penalizzazione delle pagine dovuta alla loro profondità
- **Peso dei motori di ricerca**, che misurano l'importanza dei risultati restituiti da un motore di ricerca

È da notare che la somma dei pesi dati ai motori di ricerca deve essere necessariamente pari ad uno e che tutti i pesi devono essere maggiori di zero.

4.2 Ipotesi preliminari

I motori di ricerca inseriti nel sistema sono stati Google [56], Altavista [58]e Lycos [59] [1]. I loro pesi sono stati scelti in base alla qualità dei motori; nella comunità web si ritiene che la qualità dei risultati restituiti da Google sia superiore di quella dei risultati restituiti da Altavista e Lycos. Pertanto, nella prova successiva, sono stati utilizzati i seguenti valori:

- Peso di Google = 0.4
- Peso di Altavista = 0.3
- Peso di Lycos = 0.3

[1] Si ricordi che è comunque possbile aggiungere o eliminare motori di ricerca

La costante di penalizzazione della profondità k_b è stata fissata pari a 0.25. Tale parametro è di difficile calibrazione poiché le query a profondità maggiore di 0 sono particolarmente onerose in termini di tempi di elaborazione. Pertanto si è supposto di penalizzare il voto di ogni pagina a profondità b decrementandolo del $b \cdot 25\%$.

Inoltre si è considerato che volendo fare una analisi semantica, bisognava dare maggior peso al contenuto di una pagina e eliminare, quanto più possibile, il contributo semantico delle keywords[2]. In aggiunta è parso opportuno assegnare un peso maggiore al titolo nei confronti della descrizione, in quanto spesso il titolo è una sintetica frase che però descrive il contenuto della pagina. Per tale motivo sono state assegnati i seguenti pesi[3]

- $w_c = 0.5$
- $w_t = 0.2$
- $w_d = 0.15$
- $w_k = 0.15$

Si è infine ipotizzato di ritenere non attinenti al dominio considerato quei link la cui votazione non superasse il valore 0.025.

[2]Essendo le keywords un insieme di parole e non una frase, dovrebbero avere un contributo semantico pressocché nullo

[3]Anche tenendo presente i precedenti sviuppi del sistema

Dopo una attività di calibrazione volta all'ottimizzazione di un fattore di merito costituito dal rapporto tra numero di link restituiti dal sistema e numero di link giudicati rilevanti, si sono ottenuti i seguenti parametri ottimali:

- $K_{sint} = 1$
- $K_{sem} = 70$
- $K_{ss} = 5$
- $T = 0.1$

4.3 La prove effettuate

La prove effettuate sono state scelte utlizzando query quanto più ambigue. Per tale motivo sono state scelte effettuare le seguenti interrogazioni:

1. **Zucchero**
2. **Morandi**
3. **Bongiorno**

4.3.1 Query: Zucchero

La query zucchero è stata fatta a profondità 0, prendendo 20 link per ogni motore e utilizzando tutti i motori del sistema, come mostrato nella figura 4.1

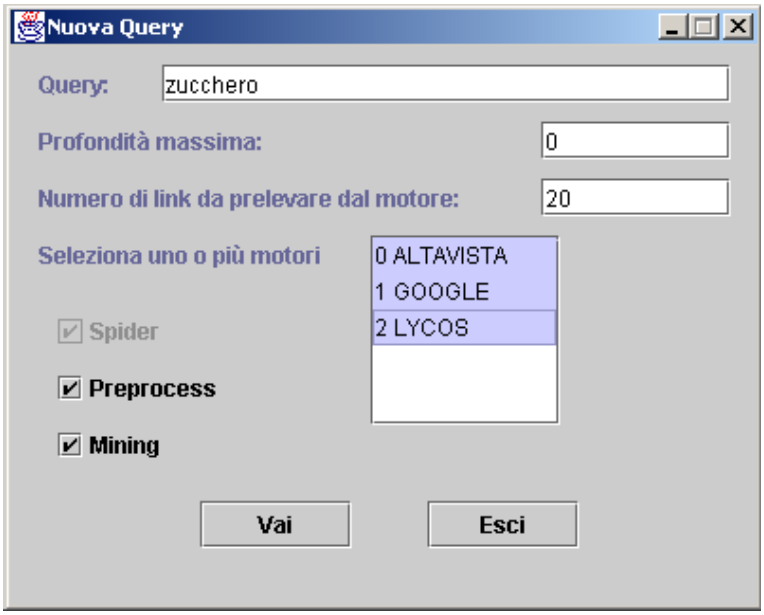

Figure 4.1: Query sottoposta al sistema: zucchero

Il numero dei link restituiti dal sistema (ossia quelli che hanno ottenuto un voto maggiore di 0.25), quello dei link giudicati rilevanti tra quelli restituiti e quello dei link rilevanti ma non restituiti sono riportati in tabella 4.1.

Nella figura 4.2 viene riportato il risultato del sistema alla query effettuata.

Per apprezzare maggiormente il risultato ottenuto si riportano, nelle figure 4.3, 4.4 e 4.5, le prime pagine di risultati della query, ottenuti dai tre motori di ricerca.

Query	**Link restituiti**	**Link rilevanti**	**Link non restituiti**
Zucchero	17	16	0

Table 4.1: Link restituiti e link giudicati rilevanti nella query Zucchero nel dominio Musica

Figure 4.2: Risultato del sistema alla query: zucchero

Figure 4.3: Risultato di Google alla query: zucchero

Figure 4.4: Risultato di Altavista alla query: zucchero

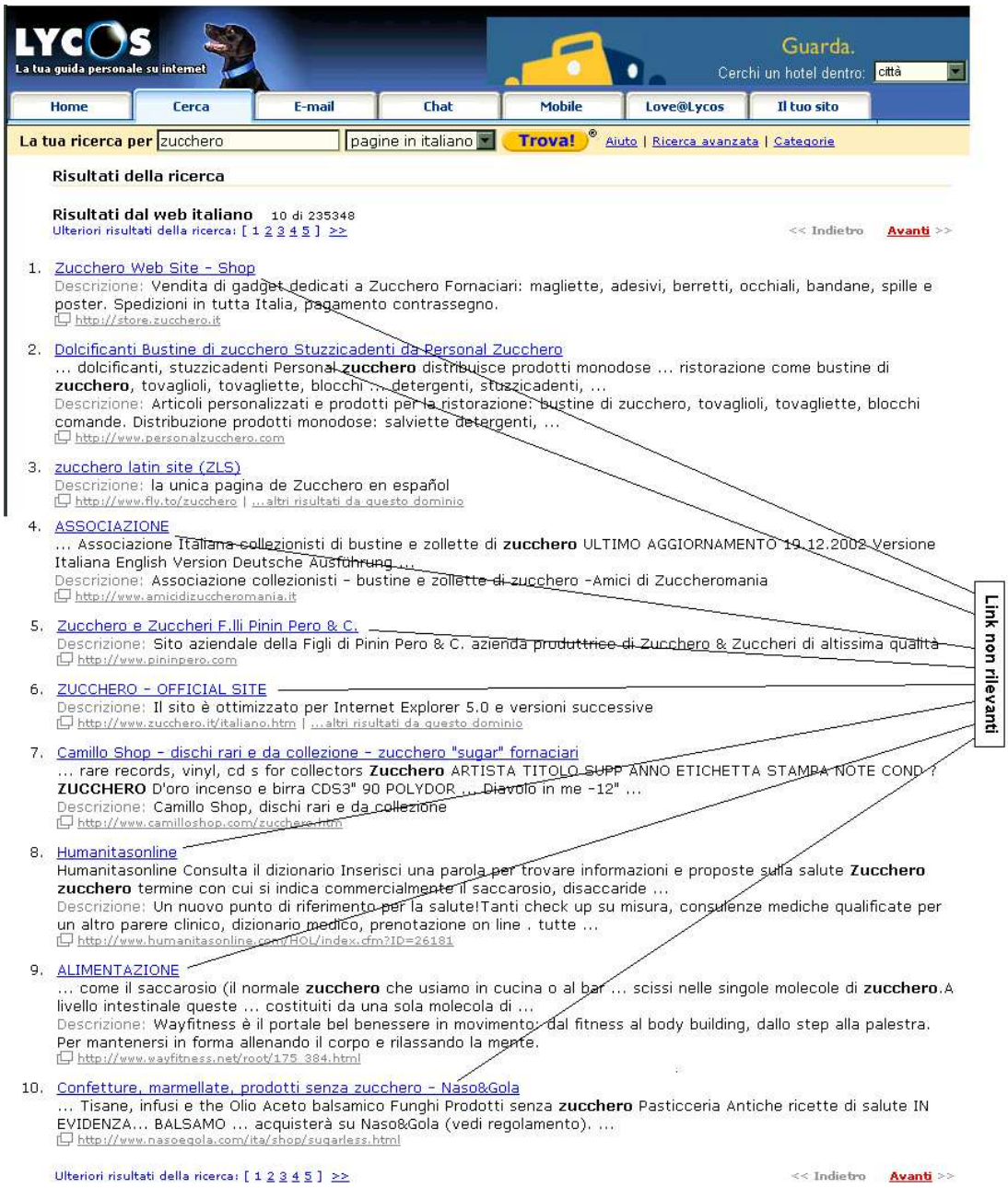

Figure 4.5: Risultato di Lycos alla query: zucchero

4.3.2 Query: Morandi

La query morandi è stata fatta a profondità 0, prendendo 20 link per ogni motore e utilizzando tutti i motori del sistema.

Il numero dei link restituiti dal sistema, quello dei link giudicati rilevanti e quello dei link rilevanti ma non restituiti dal sistema sono riportati in tabella 4.2

Nella figura 4.6 viene riportato il risultato del sistema alla query effettuata [4].

Query	**Link restituiti**	**Link rilevanti**	**Link non restituiti**
Morandi	5	5	2

Table 4.2: Link restituiti e link giudicati rilevanti nella query Morandi nel dominio Musica

Per apprezzare maggiormente il risultato ottenuto si riportano, anche in questo caso,, nelle figure 4.7, 4.8 e 4.9, le prime pagine di risultati della query, ottenuti dai tre moroti di ricerca.

[4]Si noti che una delle due pagine non restituite ha una pagina di redirect.

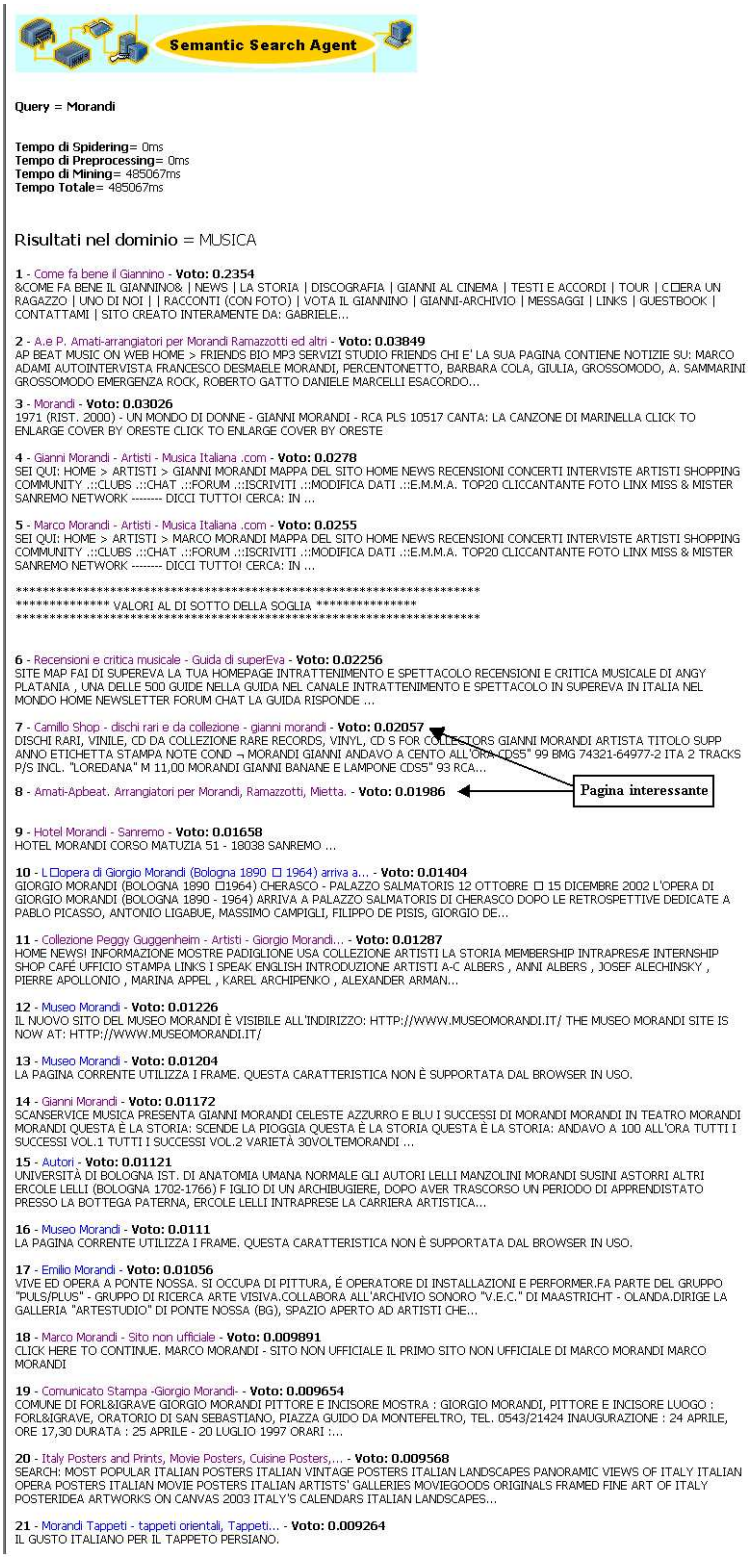

Figure 4.6: Risultato del sistema alla query: morandi

Figure 4.7: Risultato di Google alla query: morandi

Figure 4.8: Risultato di Altavista alla query: morandi

Figure 4.9: Risultato di Lycos alla query: morandi

4.3.3 Query: Bongiorno

La query bongiorno è stata fatta a profondità 0, prendendo 10 link per ogni motore e utilizzando tutti i motori del sistema.

Il numero dei link restituiti dal sistema, quello dei link giudicati rilevanti e quello dei link non restituiti anche se giudicati rilevanti sono riportati in tabella 4.3

Query	**Link restituiti**	**Link rilevanti**	**Link non restituiti**
Bongiorno 2	2	0	

Table 4.3: Link restituiti e link giudicati rilevanti nella query Bongiorno nel dominio Televisione

Chapter 5

Interfaccia utente del sistema SSA

In questo capitolo viene presentata l'interfaccia utente del sistema e vengono spiegati singolarmente la funzione dei vari pulsanti in essa contenuta.

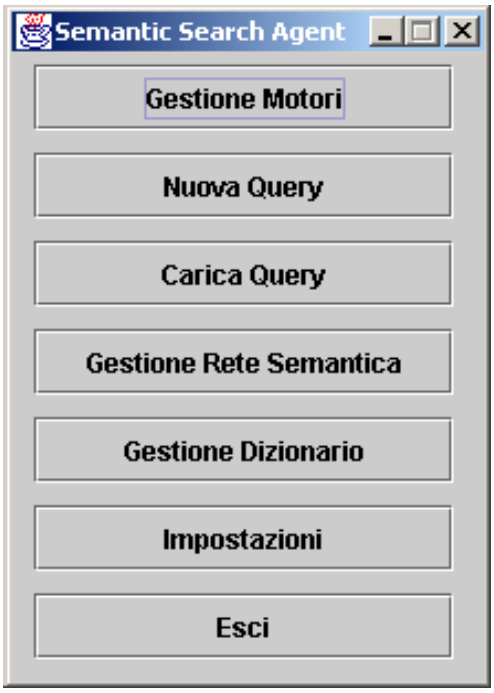

Figure 5.1: Interfaccia principale del sistema SSA

La suddetta interfaccia si presente come nella figura 5.1.

In essa sono presenti i seguenti pulsanti:

- **Gestione Motori**: per la gestione dei motori di ricerca utilizzati dal sistema
- **Nuova Query**: per la gestione di una nuova query da sottoporre al sistema
- **Carica Query**: per la gestione delle query già effettuate e memorizzate nel repository
- **Gestione Rete Semantica**: per la gestione delle reti semantiche utilizzata dal sistema
- **Gestione Dizionario**: per la gestione del dizionario multi-dominio inserito nel sistema
- **Impostazioni**: per la gestione dei parametri dell'applicazione
- **Esci**: per terminare l'applicazione

5.1 Gestione Motori

La pressione del pulsante Gestione Motori provoca l'apertura della finestra di dialogo illustrata nella figura 5.2.

Nella parte in alto della finestra di dialogo è presente l'elenco dei motori di ricerca conosciuti dal sistema. Selezionandone uno è cliccando sul pulsante *Cancella motore* è possibile eliminare un motore di ricerca dal sistema.

Nella parte inferiore è possibile inserire le informazioni per l'aggiunta di un motore di ricerca tra quelli conosciuti dal sistema; tali informazioni sono le seguenti:

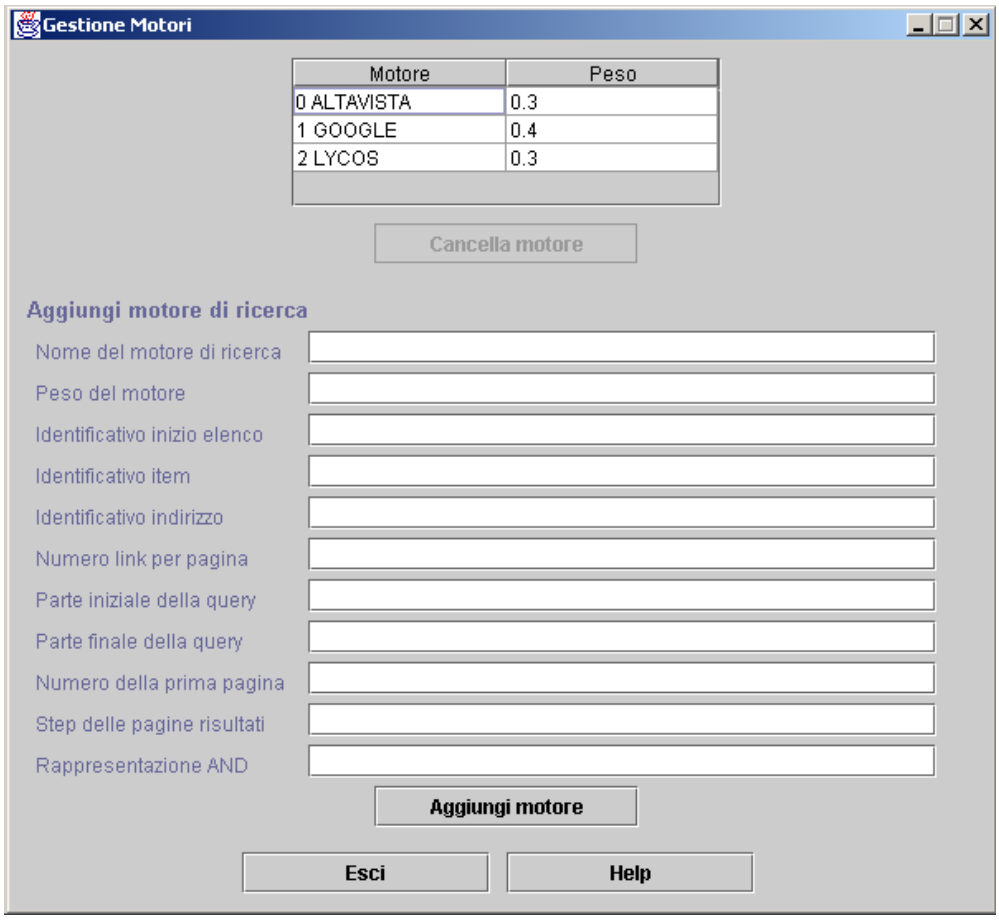

Figure 5.2: Finestra di dialogo per la gestione dei motori di ricerca

- **Nome del motore di ricerca**: tale informazione potrebbe essere anche un identificativo simbolico e non il nome reale del motore di ricerca.

- **Peso del motore**: tale valore rappresenta il peso da dare ai risultati della ricerca del motore. Una volta inserito il peso bisogna cambiare quello degli altri motori affinché la somma sia 1[1].

- **Identificativo inizio elenco**: stringa presente nella pagina HTML, risultato della ricerca del motore, che ci permetta di individuare in modo univoco la parte in cui inizia l'elenco delle pagine trovate

- **Identificativo item**: stringa presente nella pagina HTML, risultato della ricerca del motore, che ci permetta di individuare in modo univoco un punto elenco delle pagine trovate

- **Identificativo indirizzo**: stringa presente nella pagina HTML, risultato della ricerca del motore, che ci permetta di individuare in modo univoco un indirizzo dell'elenco delle pagine trovate

- **Numero di link per pagina**: numero di link trovati per ogni pagina risultato del motore. Ogni motore restituisce una pagina con un certo numero di risultati e poi un link alla successiva pagina con altri risultati

- **Parte iniziale della query**: stringa che rappresenta la query di ricerca fino al punto in cui ci sono le parole cercate adattate.

[1]Si veda anche la sezione relativa alle Impostazioni.

- **Parte finale della query**: stringa che rappresenta la query di ricerca dal punto in cui ci sono le parole cercate adattate, fino alla posizione in cui inserire il numero della pagina contenente i risultati
- **Numero della prima pagina**: ogni motore identifica la prima pagina contenente i risultati della ricerca con un intero. Tale intero è solitamente 0
- **Step delle pagine risultato**: passo che intercorre tra gli identificativi (numeri interi) di due pagine contenenti i risultati, restituite dal motore
- **Rappresentazione AND**: stringa che indica come il motore rappresenta l'AND nella query adattata (solitamente +)

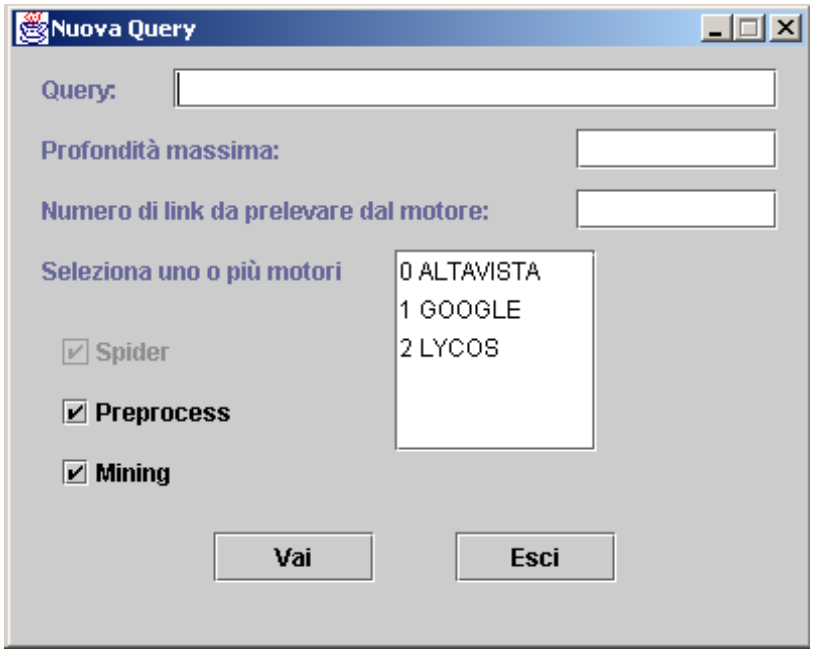

Figure 5.3: Interfaccia Nuova Query

5.2 Nuova Query

La pressione del pulsante Nuova Query provoca l'apertura della finestra di dialogo illustrata nella figura 5.3.

- **Query**: vengono inserite le keywords che verranno utilizzate dai motori di ricerca per il reperimento delle pagina Web. La sintassi adottata è quella di Google[57,60] che prevede che tutte le keywords immesse siano in AND tra loro, ossia devono essere presenti tutte in una pagina affinché la stessa venga restituita dal motore. È anche possibile immettere query contenenti frasi (o gruppi di parole) racchiuse tra virgolette; la stringa di caratteri racchiuse tra virgolette deve essere presente nella pagina affinché venga restituita dal motore.

- **Profondità massima**: deve essere inserita la profondità massima che una pagina deve avere, rispetto al motore di ricerca, affincé questa venga presa in considerazione. Si usa la convenzione che le pagine restituite dal motore abbiano profondità 0, mentre le pagine ottenute perchè linkate da qualche altra pagina abbiano profondità pari alla profondità della pagina linkante aumentata di 1. Il concetto di profondità è illustrato nella figura 5.4 e nella figura 5.5[2].

- **Numero di link da prelevare da ogni motore di ricerca**: deve essere inserito il numero massimo di link che si desidera considerare tra quelli restituiti dai singoli motori di ricerca utilizzati.

- **Motori di ricerca**: è necessario infine selezionare almeno un motore tra quelli conosciuti dal sistema. È possibile anche selezionare più motori di ricerca.

[2]Si noti che la pagina di figura 5.5 è presente tra i link di livello 0 della figura 5.4

Figure 5.4: Esempio di link di livello 0

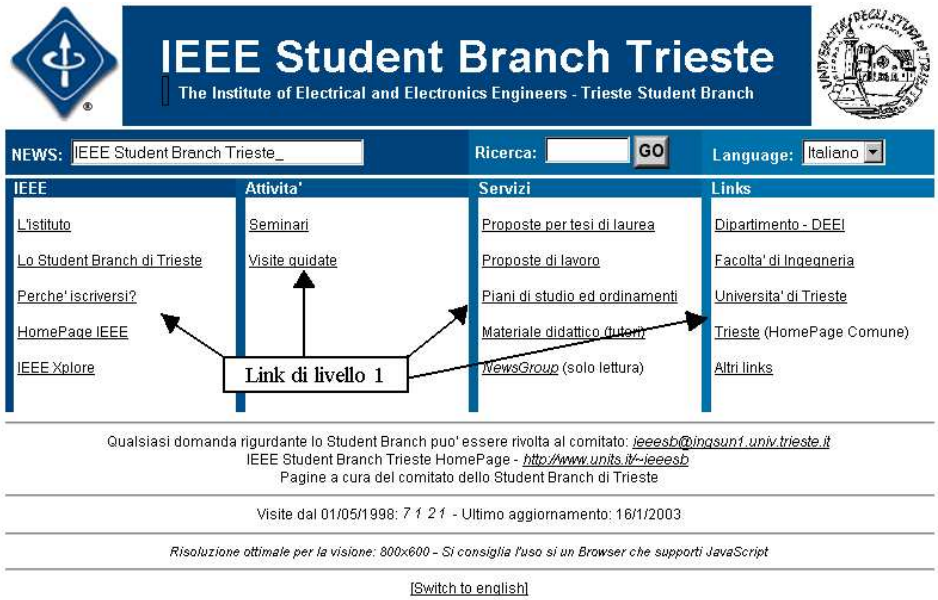

Figure 5.5: Esempio di link di livello 1

Una volta immessi i dati necessari alla ricerca e selezionati i motori da utilizzare, è possibile scegliere quali operazioni far compiere al sistema. Poiché si sta sottoponendo una nuova query, la casella di Spidering risulta già spuntata e non modificabile. Tale situazione è giustificata dal fatto che non è possibile effettuare le altre operazioni previste senza che le pagine siano state prelevate e memorizzate nel sistema (che è compito dello Spider).

Le operazioni che si possono compiere (oltre allo Spidering) sono le seguenti:

- **Preprocess**: per l'analisi delle pagine web al fine di ottenerne il contenuto semantico (ossia la pagina senza i tag HTML), il contenuto del campo Title e il contenuto dei Meta Tag Description e Keywords
- **Mining**: per l'analisi semantica delle pagine web al fine di ricavarne il voto semantico che rappresenta il grado di attinenza della pagina al dominio considerato

È da notare che l'operazione di *Mining* può avvenire solo a valle dell'operazione di *Preprocess* in quanto la prima utilizza, per la valutazione delle pagine web, i risultati della seconda. Pertanto, non è consentito effettuare il Mining senza che si sia anche scelto di effettuare il *Preprocess* delle pagine.

Alla pressione del **tasto Vai**, se non si è scelto di effettuare né il *Preprocess* né il *Mining*, viene restituita una pagina HTML dove viene presentato l'elenco dei link trovati ordinati in base alla profondità e alla posizione. Quest'ultimo valore dipende dal tipo di link: se è uno di livello zero, rappresenta la posizione mediata all'interno dei motori selezionati (come già spiegato); se è di livello maggiore, rappresenta la posizione mediata del suo migliore link referrer.

Nel caso in cui venga selezionata anche l'operazione di *Mining*, viene restituita una diversa pagina HTML contenente, stavolta, i link ordinati in base al voto attribuito dal sistema alle pagine web corrispondenti. Di ogni pagina web si riporta il titolo (se presente) e parte del testo. All'interno della pagina HTML costruita, si riportano anche i tempi (sempre in millisecondi) impiegati per effettuare le operazioni di Spidering, Preprocess e Mining.

Se si sceglie solo l'operazione di *Preprocess*, viene restituita la stessa pagina HTML costruita nel caso in cui si opti solo per lo Spidering e, in aggiunta, viene effettuato il preprocess delle pagine web (come già spiegato) andando a riempire le specifiche tabelle del database di sistema.

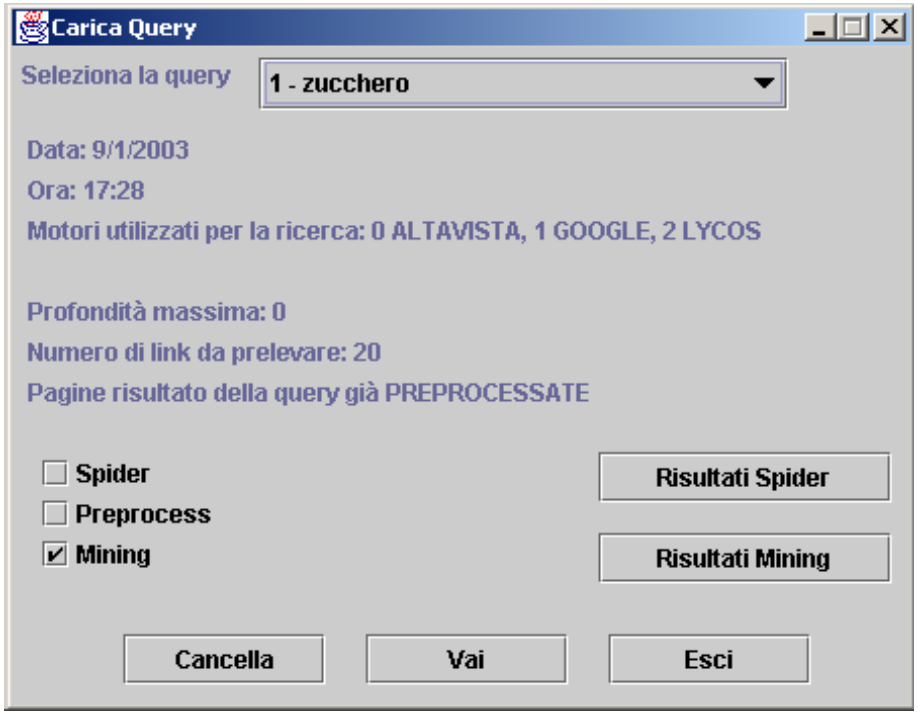

Figure 5.6: Interfaccia Carica Query

5.3 Carica Query

La funzione Carica Query è preposta alla gestione delle query già sottoposte al sistema. Essa si presenta come illustrato nella figura 5.6

Viene presentato, tramite una casella a scorrimento, l'elenco delle query già effettuate.

Selezionata una di esse, vengono restituite le seguenti informazioni:

- Data e ora dell'ultimo Spidering effettuato
- I motori di ricerca utilizzati
- La massima profondità di ricerca
- Il numero massimo di link da prelevare
- Se le pagine web associate alla query sono state o meno preprocessate

È possibile decidere se effettuare nuovamente lo *Spidering* oppure il *Preprocess* o il *Mining*. La prima operazione risulta utile nel caso in cui si ritenga che la query selezionata sia datata e, quindi, necessiti di un nuovo Spidering; la seconda e la terza permettono rispettivamente di preprocessare e di effettuare l'analisi semantica delle pagine web nel caso in cui ciò non era stato fatto quando la query veniva sottomessa per la prima volta al sistema. L'opzione di Mining risulta utile anche nel caso in cui si siano apportate delle modifiche ai parametri utilizzati dal sistema per l'attribuzione del voto alle pagine oppure modifiche alla Rete Semantica o alla Token List e, quindi, è richiesta una nuova analisi semantica delle pagine. È da notare che, nel caso in cui si voglia effettuare lo Spidering modificando i motori utilizzati oppure modificando il

numero di link da prelevare o la profondità di ricerca, ciò deve essere fatto andando ad impostare una nuova query tramite la funzionalità già descritta.

Allorquando si sceglie di effettuare un nuovo Spidering della query selezionata, tutti i vecchi dati ad essa associati vengono cancellati dal sistema e aggiornati. È da notare ancora che la casella Preprocessing risulta già spuntata nel caso in cui le pagine associate alla query siano ancora da preprocessare e che non è consentito effettuare il Mining se a monte non è stato prima effettuato il Preprocess delle pagine.

Come esposto in Nuova Query, alla pressione del tasto Vai, a seconda se si è scelto di effettuare solo lo Spidering oppure anche il Mining, viene costruita una specifica pagina HTML con i risultati dell'operazione. All'interno della finestra sono presenti i tre pulsanti: Risultati Spider, Risultati Miner e Cancella.

Il primo consente di vedere la pagina HTML che riporta i risultati dell'ultimo Spidering effettuato. Il secondo è attivo soltanto se le pagine restituite dalla query risultano già analizzate semanticamente attraverso la rete semantica e la token list e, quindi, presentano un voto associato. Alla pressione di tale tasto viene presentata la pagina HTML che riporta i risultati dell'ultima operazione di mining effettuata. Il terzo pulsante consente di cancellare tutte le informazioni memorizzate nel sistema associate alla query selezionata.

5.4 Gestione Rete Semantica

Alla pressione del pulsante Gestione Rete Semantica viene presentata l'interfaccia mostrata nella figura 5.7 in cui sono presenti i seguenti pulsanti:

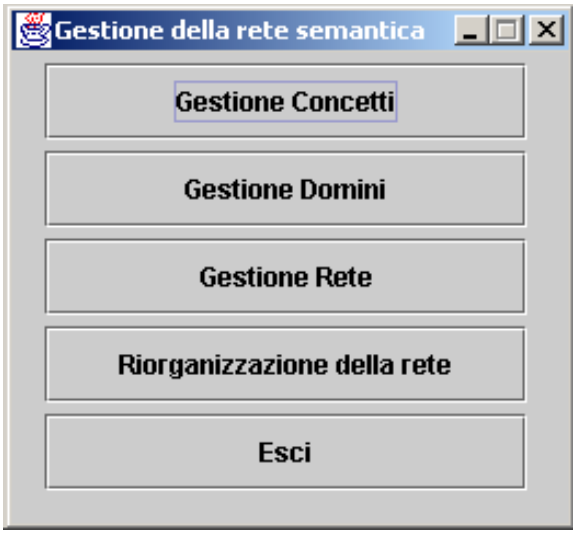

Figure 5.7: Interfaccia della Gestione della Rete Semantica

- **Gestione Concetti**: per la gestione dei concetti (eliminazione o aggiunta)
- **Gestione Domini**: per la gestione dei domini (eliminazione, aggiunta o definizione della loro correlazione)
- **Gestione Rete**: per la gestione della Semantic Network dei concetti.
- **Riorganizzazione della rete**: per riorganizzare la Semantic Network e la mappa dei domini dopo una modifica
- **Esci**: per uscire dal menù.[3]

[3]Si noti che se si tenta di uscire senza aver fatto la riorganizzazione viene comunque chiesto se si desidera farla.

5.4.1 Gestione Concetti

La funzionalità per la gestione dei concetti si presenta con la finestra di dialogo mostrata nella figura 5.8.

Nella parte alta della finestra è presente un elenco dei concetti presenti. Da tale elenco è possibile selezionarne uno e, premendo il pulsante *Cancella concetto*, eliminarlo.

Figure 5.8: Finestra di dialogo per la gestione dei concetti

Nella parte bassa è possibile scrivere il nome di un nuovo concetto e, premendo il pulsante *Aggiungi*, inserirlo nel sistema.

5.4.2 Gestione Domini

La funzionalità per la gestione dei domini si presenta con la finestra di dialogo mostrata nella figura 5.9.

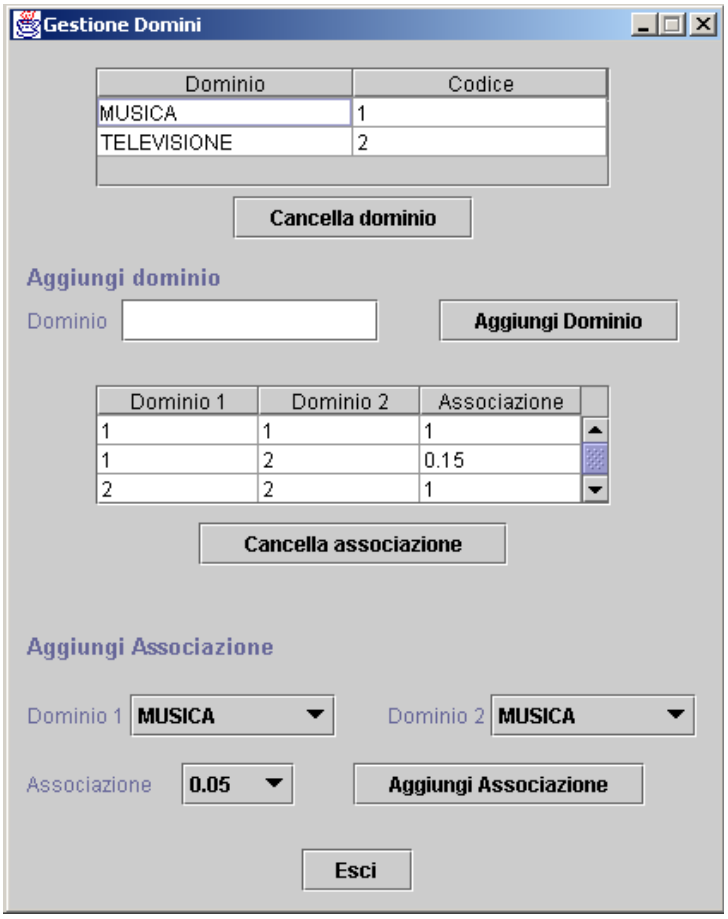

Figure 5.9: Finestra di dialogo per la gestione dei domini

Nella parte alta della finestra è presente un elenco dei domini presenti. Da tale elenco è possibile selezionarne uno e, premendo il pulsante *Cancella dominio*, eliminarlo.

Immediatamente sotto è possibile scrivere il nome di un nuovo dominio e, premendo il pulsante *Aggiungi dominio*, inserirlo nel sistema.

Ancora più in basso vi è un elenco delle associazioni tra i concetti. Da tale elenco è possibile selezionare una associazione e, premendo il pulsante *Cancella associazione*, eliminarla.

Infine, nella parte bassa della finestra c'è la possibilità di aggiungere una nuova associazione tra i domini scegliendo i due domini interessati, il grado di associazione e premendo il pulsante *Aggiungi associazione*[4].

5.4.3 Gestione Rete

La finestra per la gestione della Semantic Network tra i concetti si presenta come in figura 5.10.

Innanzi tutto è necessario scegliere il dominio della Semantic Network che si vuole gestire[5].

Nella parte centrale c'è l'elenco degli archi della rete. Da tale elenco è possibile selezionare un arco ed eliminarlo mediante la pressione del pulsante *Cancella arco*.

Nella parte inferiore della finestra è possibile scegliere due concetti, da altrettanti elenchi a discesa, e un peso per inserire un nuovo arco nella rete semantica del dominio selezionato.

---

[4]Si ricordi che la mappa dei domini è un grafo i cui archi non sono orientati quindi non può essere inserito un arco dal dominio D_i al dominio D_j se esiste giè l'arco da D_j a D_i

[5]Si noti che anche se nel Database le reti semantiche vengono gestite come una unica rete i cui archi pesati portano anche l'informazione del dominio in cui hanno senso, dal punto di vista dell'utente è comodo pensare che ci siano diverse reti semantiche una per ogni dominio.

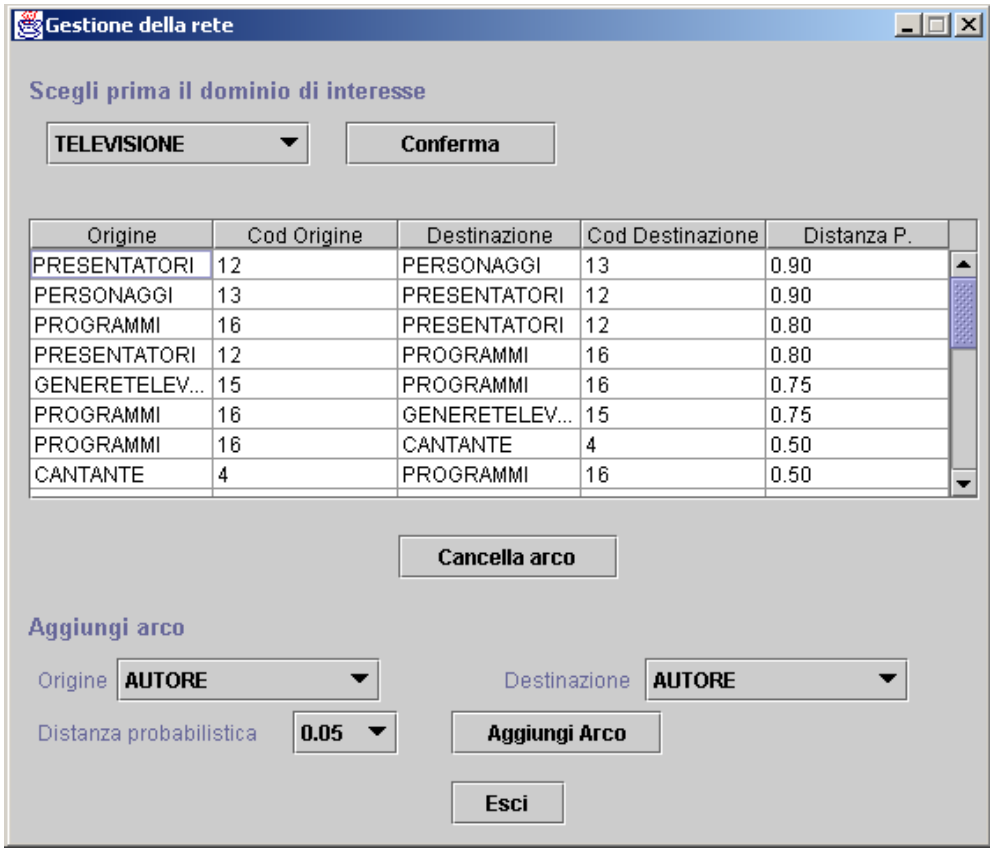

Figure 5.10: Finestra di dialogo per la gestione della Semantic Network

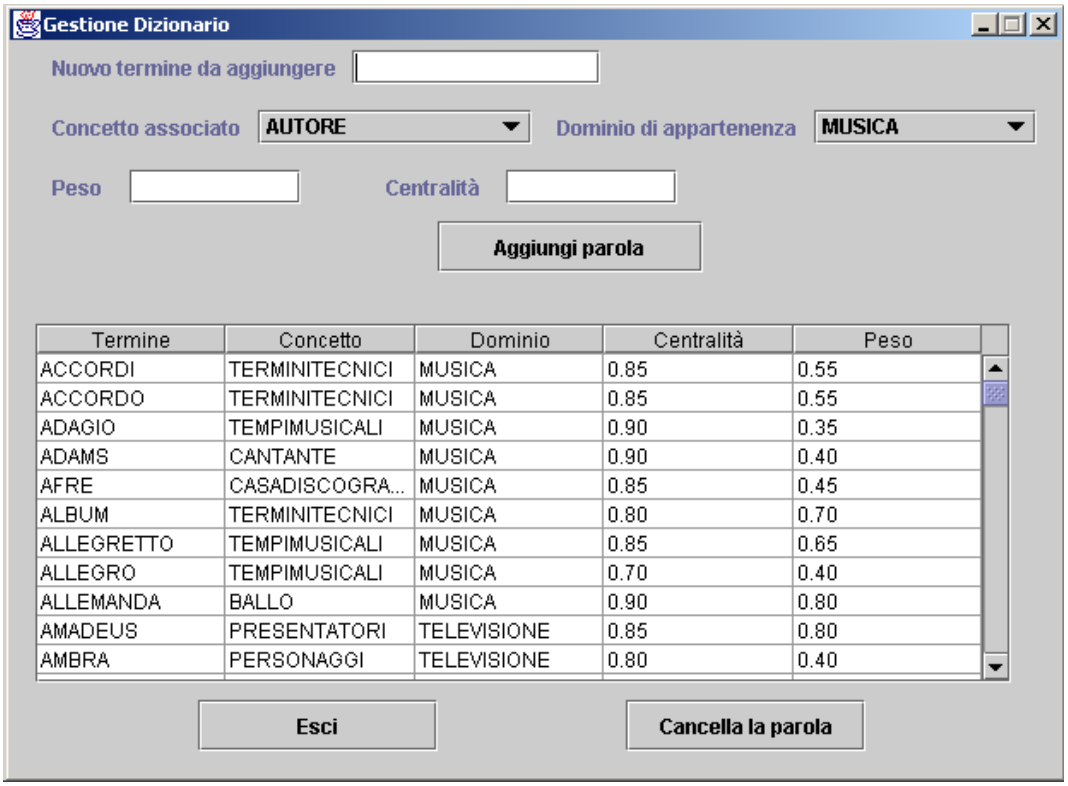

Figure 5.11: Finestra di dialogo per la gestione del Dizionario contenente le parole conosciute dal sistema

5.5 Gestione Dizionario

La finestra per la gestione del Dizionario si presenta come in figura 5.11.

Nella parte alta della finestra c'è la sezione mediante la quale è possibile aggiungere una nuova parola nel dizionario. Per fare tale operazione è necessario inserire il vocabolo, il concetto associato, il dominio di appartenenza, la centralità del vocabolo rispetto al concetto e il peso del vocabolo nel dominio. Una volta inseriti le precedenti informazioni la nuova parola può essere aggiunta al Dizionario premendo il pulsante *Aggiungi parola*.

Nella parte inferiore della finestra vi è l'elenco di tutte le parole presenti nel dizionario. È possibile selezionarne una e cancellarla mediante la pressione del pulsante *Cancella Parola*.

5.6 Impostazioni

La funzionalità *Impostazioni* è preposta alla gestione dei parametri dell'applicazione. La funzione si presenta come nella figura 5.12:

La finestra mostra le ultime impostazioni salvate dall'utente. Si possono impostare i seguenti parametri:

- **Pesi dei motori di ricerca**: la somma di tali pesi deve necessariamente essere 1. Tali pesi misurano l'importanza dei risultati restituiti dai singoli motori.
- **Peso del contenuto**: fattore che misura l'importanza data al testo di una pagina web nel calcolo del voto semantico

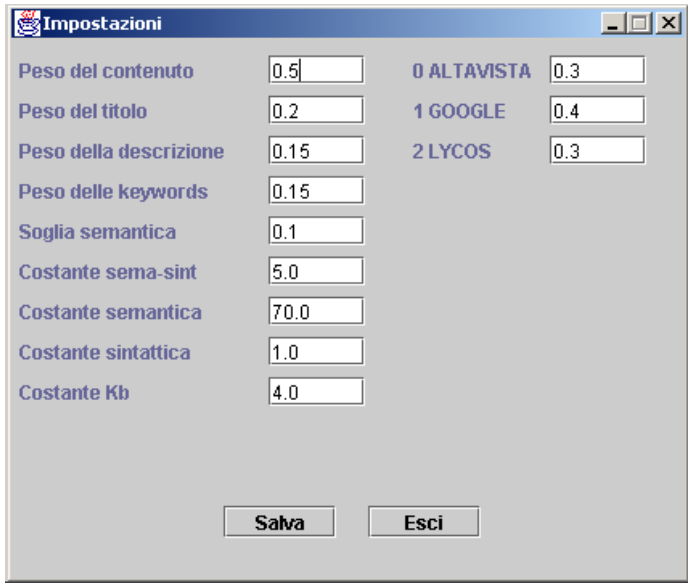

Figure 5.12: Finestra di dialogo la modifica delle impostazioni

- **Peso del titolo**: fattore che misura l'importanza data al titolo di una pagina web nel calcolo del voto semantico
- **Peso della descrizione**: fattore che misura l'importanza data al contenuto del Meta Tag description di una pagina web nel calcolo del voto semantico
- **Peso delle keywords**: fattore che misura l'importanza data al contenuto del Meta Tag keywords di una pagina web nel calcolo del voto semantico
- **Soglia semantica**: minimo valore di attinenza di una coppia di vocaboli ritrovata nel testo affinché la loro accoppiata possa essere considerata significativa.

Si noti che la somma dei pesi relativi al *contenuto*, *titolo*, *descrizione* e *keywords* deve essere necessariamente pari ad uno; inoltre tutti i pesi devono essere maggiori di zero.

Alla pressione del tasto Salva tutti i valori inseriti verranno memorizzati all'interno del sistema.

Bibliography

[1] A. Castellucci, G. Ianni, D. Vasile, S. Costa - *"Searching and surfing the web using a semi-adaptative meta-engine"* - IEEE, 2001

[2] H. S. Nwana - *"Software Agents: An Overview"* - Cambridge University, 1996. - `http://www.sce.carleton.ca/netmanage/docs/AgentsOverview/ao.html`

[3] Darren Greaves - *"Intelligent Search Agents"*- `http://krapplets.org/report/report.html`

[4] R. Cooley, B. Mobasher, J. Srivastava - *"Web Mining: Information and Pattern Discovery on the World Wide Web"* - IEEE, 1997 - `http://maya.cs.depaul.edu/ mobasher/webminer/survey/survey.html`

[5] V. S. Subrahmanian, Piero Bonatti, Jurgen Dix, Thomas Eiter, Sarit Kraus, Fatma Ozcan, Robert Ross - *"Heterogeneous Agent System"* - The MIT Press Cambridge, Massachussets, London, England.

[6] Ferber, J. - *"Simulating with Reactive Agents"* - In Hillebrand, E. & Stender, J. (Eds.), Many Agent Simulation and Artificial Life, Amsterdam: IOS Press, 8-28, 1994

[7] M. Wooldridge, N. Jennings - *"Intelligent Agents: Theory and Practice"* - The Knowledge Engineering Review 10 (2), 115-152, 1995

[8] M. N. Huhns - M. P. Singh - *"Distributed Artificial Intelligence for Information Systems"* - CKBS-94 Tutorial, June 15, University of Keele, UK, 1994

[9] E. H. Durfee - V. R. Lesser - D. Corkill - *"Coherent Cooperation among Communicating Problem Solvers"* - IEEE Transactions on Computers C-36(11), 1275-1291, 1987

[10] `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-5/www/pleiades.html`

[11] K. Sycara - *"Intelligent Agents and the Information Revolution"* - UNICOM Seminar on Intelligent Agents and their Business Applications, 8-9 November 1995, London, 143-159,

[12] M. Georgeff - *"Agents with Motivation: Essential Technology for Real World Applications"* - The First International Conference on the Practical Applications of Intelligent Agents and Multi-Agent Technology, London, UK, 24th April 1996

[13] R. Smith - Personal Communication - 1996

[14] H. S. Nwana, L. Lee, N. R. Jennings - *"Coordination in Software Agent Systems"* - British Telecommunications Technology Journal 14 (4), October 1996

[15] P. Maes - *"Agents that Reduce Work and Information Overload"* - Communications of the ACM 37 (7), 31-40, 1994

[16] P. Maes - *"Intelligent Software"* - Scientific American 273 (3), September 1995

[17] R. Kozierok, P. Maes - *"A Learning Interface Agent for Scheduling Meetings"* - Proceedings of the ACM-SIGCHI International Workshop on Intelligent User Interfaces, Florida, 81-93, 1993

[18] H. Lieberman - *"Letizia: An Agent that Assists Web Browsing"* - In Proceedings of IJCAI 95, AAAI Press, 1995

[19] B. J. Rhodes, T. Starner - *"Remembrance Agent: A Continuously Automated Information Retrieval System"* - In Proceedings the First International Conference on the Practical Application of Intelligent Agents and Mullti-Agent Technology (PAAM 96), London, April 1996

[20] A. Chavez, P. Maes - *"Kasbah: An Agent Marketplace for Buying and Selling Goods"* - In Proceedings the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 96), London, April 1996

[21] P. Wayner - *"Agents Unleashed: A Public Domain Look at Agent Technology"* - Boston, MA: AP Professional, 1995

[22] `http://www.genmagic.com`

[23] P. Wayner - *"Free Agents"* - Byte, March, 1995

[24] K. Indermaur - *"Baby Steps"* - Byte, March, 1995

[25] O. Etzioni, D. Weld - *"A Softbot-Based Interface to the Internet"* - Communications of the ACM 37 (7), 72-76, 1994

[26] N. J. Davies, R. Weeks - *"Jasper: Communicating Information Agents"* - in Proceedings of the 4th International Conference on the World Wide Web, Boston, USA, December 1995

[27] P. Maes - *"Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back"* - London: The MIT press, 1991

[28] R. A. Brooks - *"Intelligence without Representation"* - Artificial Intelligence 47, 139-159, 1991

[29] R. A. Brooks - *"A Robust Layered Control System for a Mobile Robot"* - IEEE Journal of Robotics and Automation 2 (1), 14-23, 1986

[30] H. S Nwana - *"Simulating a Childrens Playground in ABLE"* - Working Report, Department of Computer Science, Keele University, UK, 1993

[31] P. Wavish, M. Graham - *"A Situated Action Approach to Implementing Characters in Computer Games"* - Applied AI Journal, 1995

[32] K. Fisher, J. P. Muller, M. Pischel - *"Unifying Control in a Layered Agent Architecture"* - Technical Report TM-94-05, German Research Center for AI - (DFKI GmbH), 1996

[33] J. P. Muller, M. Pishel, M. Thiel - *"Modelling Reactive Behaviour in Vertically Layered Agent Architectures"* - In Wooldridge, M. & Jennings, N. (eds.) (1995b), Intelligent Agents, Lecture Notes in Artificial Intelligence 890, Heidelberg: Springer Verlag, 261-276, 1995

[34] I. A. Ferguson - *"TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents"* - PhD Thesis, Computer Laboratory, University of Cambridge, UK, 1992

[35] M. R. Genesereth, S. P. Ketchpel - *"Software Agents"* - Communications of the ACM 37 (7), 48-53, 1994

[36] G. Wiederhold - *"Mediators in the Architecture of Future Information Systems"* - IEEE Computer 25 (3), 38-49, 1992

[37] M. R. Cutkosky, R. S. Engelmore, R. E. Fikes, M. R. Genesereth, T. R. Gruber, J. M. Tenenbaum, J. C. Weber - *"PACT: An Experiment in Integrating Concurrent Engineering Systems"* - IEEE Computer 1, January, 28-37, 1993

[38] Wang Jicheng, Huang Yuan, Wu Gangshan, Zhang Fuyan - *"Web Mining: Knowledge Discovery on the Web"* - Department of Computer Science and Technology, Nanjing University

[39] H. Kawano - *"Overview of Mondou web search engine using text mining and information visualizing technologies"* - IEEE, 2001

[40] P. Atzeni, S. Ceri, S. Paraboschi, R. Torlone - *"Basi di dati - Seconda Edizione"* - McGraw-Hill, 1999

[41] C. M. Brown, B. B. Danzig, D. Hardy, U. Manber, M. F. Schwartz - *"The Harvest information discovery and access system"* - Proc. 2nd International World Wide Web Conference, 1994

[42] K. Hammond, R. Burke, C. Martin, S. Lytinen - *"Faq-finder: A case-based approach to knowledge navigation"* - In Working Notes of the AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments. AAAI Press, 1995.

[43] T. Kirk, A. Y. Levy, Y. Sagiv, D. Sristava - *"The information manifold"* - In Working Notes of the AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments. AAAI Press, 1995.

[44] R.B.Doorenbos, O.Etzioni, D.S.Weld - *"A scalable comparison shopping agent for world wide web"* - Technical Report 96-01-03, University of Washington, Dept. Of Computer Science and Engineering, 1996

[45] R. Weiss, B. Velez, M. A. Sheldon, C. Namprempre, P. Szlagyi, A. Duda, D. K. Gifford - *"Hypursuit: a hierarchical network search engine that exploits content-link hypertext clustering"* - In Hypertext'96: The Seventh ACM Conference on Hypertext, 1996

[46] R. Armstrong, d. Freitag, T. Joachims, T. Mitchell - *"Webwatcher: A learning apprentice for the world wide web"* - In Proc. AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments, 1995.

[47] Konopnicki, O. Shmueli - *"W3qs: A query system for the world wide web"* - In Proc. Of the 21th VLDB Conference, pagine 54-65, Zurigo, 1995

[48] B. Mobobasher, N. Jain, E. Han, J. Srivastava - *" Web mining: Pattern Discovery from world wide web transactions"* - Technical Report TR 96-050, University of Minnesota, Dept. of Computer Science, Minneapolis, 1996

[49] G. M. Youngblood *"Web Hunting: Designe of a Simple Intelligent Web Search Agent"* - 1999 - `http://www.acm.org/crossroads/xrds5-4/webhunting.html`

[50] Russell, Stuart, P. Norvig - *"Artificial Intelligence A Modern Approach"* - Prentice Hall, Upper Saddle River, N.J., 1995 `http://www.amazon.com/exec/obidos/ISBN=0131038052/acmcrossroadsstu`

[51] Koster, Martijn - *"Guidelines for Robot Writers"* `http://info.webcrawler.com/mak/projects/robots/guidelines.html`, 30 Nov. 1998.

[52] Berners-Lee, R. Fielding, H. Frystyk - *"Hypertext Transfer Protocol – HTTP/1.0"* - RFC:1945, May 1996 `http://info.internet.isi.edu:80/in-notes/rfc/files/rfc1945.txt`

[53] Koster, Martijn - *"The Web Robots Database"* - 30 Nov. 1998 - `http://info.webcrawler.com/mak/projects/robots/active.html`

[54] V. S. Subrahmanian, T. Eiter, T. Lukasiewicz, J. J. Lu - *"Probabilistic Object Bases"* - Infsys Research Report 1843-99-11, November 1999 & April 2001

[55] T. Gruber - *"What is an ontology?"* - `http://ksl-web.stanford.edu/kst/what-is-an-ontology.html`

[56] *Google web site* - `www.google.com`

[57] *Sintassi del motore di ricerca Google* - `http://www.google.it/intl/it/help/basics.html`

[58] *Altavista web site* - `www.altavista.com`

[59] *Lycos web site* - `www.lycos.it`

[60] P. Merialdo, P. Atzeni, G. Mecca - *"Semistructured and structured data in the web: Going back and forth"* - in Proceedings of the Workshop on the Management of Semistructured Data, 1997

[61] C. Latiri, S. Ben Yahia - *Generating implicit association rules from Texual Data* - IEEE

[62] *Dizionario della lingua italiana* - Centri Garzanti, 1980