ALMA MATER STUDIORUM – UNIVERSITA' DI BOLOGNA
SEDE DI CESENA

FACOLTA' DI SCIENZE MATEMATICHE, FISICHE E NATURALI
CORSO DI LAUREA IN SCIENZE DELL'INFORMAZIONE

UN APPROCCIO PER LA CONCETTUALIZZAZIONE DI INSIEMI DI DOCUMENTI

Relazione finale in:
Programmazione

Relatore:
Chiar.ma Prof.ssa Antonella Carbonaro

Presentata da:
Marco Angelini

Sessione I
Anno Accademico 2012/2013

Dedico questa tesi alle persone più importanti
e vicine a me in questo momento il cui sostegno
non è mai venuto a mancare.
Ringrazio inoltre la professoressa Carbonaro per
la disponibilità offerta per poter conseguire lo
svolgimento del lavoro di questo progetto.

"Internet è talmente sconfinata, potente e priva di scopo
che per alcuni diventa un sostituto della vita"
Andrew Brown

INDICE

Introduzione

Nell'era della crescente disponibilità di informazioni dettata dall'esponenziale crescita di documenti presenti nel web, un numero sempre in aumento di utenti utilizza le risorse della rete per reperire le informazioni di cui necessita. E' quindi essenziale fornire agli utenti validi metodi per comprenderne facilmente ed intuitivamente gli svariati contenuti disponibili riguardanti ogni tipologia di disciplina, categoria ed attività umana.

Per rendere possibile e meno difficoltosa la ricerca da parte degli utenti, si è reso necessario lo sviluppo di software in grado di aiutare la ricerca di determinati documenti, tali sistemi sono conosciuti a tutt'oggi come motori di ricerca.

Questi motori però, non sempre soddisfano pienamente i criteri di ricerca, ovvero, durante una di queste, essi possono restituire oltre a risultati precisi e coerenti anche altri contenenti tipologie ed argomenti del tutto diversi ed incoerenti. Ciò accade perché molti effettuano dei controlli sintattici sui documenti web analizzandone titolo e contenuto senza però interpretarne il significato ed il contesto, causando spesso ambiguità ed imprecisioni; sono state studiate e sviluppate così nel corso degli anni diverse soluzioni in grado di offrire un contributo migliore per la ricerca sul web, ad esempio offrendo la possibilità di personalizzare le ricerche oppure di condividere risultati e ricerche ottenute con altri utenti. Quindi, bisogna considerare la modalità di visualizzazione dei risultati, la quale obbliga spesso a dover leggere tutte le informazioni contenute all'interno del documento determinando in seguito se i concetti presenti possono essere più o meno coerenti con i criteri di ricerca traducendo il tutto in una notevole perdita di tempo, considerando in particolare il caso in cui tale documento risultasse irrilevante.

Questo limite può essere in parte superato affidandosi all'utilizzo della sommarizzazione automatica di pagine web, utilizzando diverse tecniche come il raggruppamento di documenti e di visualizzazione delle informazioni, applicazione di misure di similarità semantica facendo uso di risorse ontologiche, quali possono

essere ad esempio WordNet o MeSH, e non, sfruttando il concetto di web semantico, termine coniato dal suo inventore Tim Berners-Lee.

Lo scopo principale della sommarizzazione delle pagine web è quello di estrarre le informazioni ed i concetti più importanti da un documento, offrendo una panoramica più chiara, rapida e concisa dei contenuti presenti all'interno di esso.

Le misure di similarità semantica risultano importanti in svariati ambiti di elaborazioni del linguaggio naturale (NLP), dalla modellazione del linguaggio alla comprensione di un vocabolo, dall'induzione grammaticale di un vocabolo al riconoscimento del senso di parola disambigua.

Il continuo evolversi del web e del mondo che lo circonda implica un costante impegno nello studio di nuovi metodi di ricerca o di ottimizzazione di quelli esistenti, considerando che i motori di ricerca e più in generale le documentazioni presenti sul web sono divenute le risorse maggiormente utilizzate dagli utenti.

L'obiettivo di questo progetto è lo sviluppo di un sistema che, oltre a svolgere le normali funzioni offerte da un motore di ricerca (attualmente la ricerca è ottimizzata per la lingua inglese), sia in grado, integrando metodi di sommarizzazione e funzioni di tipo semantico-lessicali, di raggruppare i documenti comuni in base ai loro concetti chiave e visualizzarli agli utenti, inoltre di generare un albero che rappresenti la gerarchia ontologica dei concetti chiave, come un albero genealogico; più precisamente, le funzioni offerte sono le seguenti:

- *estrazione dei concetti chiave*: attraverso una procedura di sommarizzazione integrata con funzioni semantico-lessicali viene prodotto un elenco contenente i concetti chiave ricavati dai documenti restituiti dalla ricerca;
- *visualizzazione dei documenti inerenti ai concetti*: l'utente può visualizzare i documenti in base ai concetti chiave estratti, il tutto tramite una procedura di ricerca interna ai documenti stessi, dove, questi ultimi vengono collegati tramite delle liste ai corrispondenti concetti chiave;
- *visualizzazione albero gerarchico delle ontologie*: in base ai concetti chiave estratti, viene generato un albero contenente le relazioni ontologiche tra i

concetti stessi, inoltre l'utente è in grado di visualizzare anche tramite l'albero i documenti con testi inerenti ai concetti chiave.

L'applicazione presenta diversi punti di sviluppo e si è cercato di produrre una struttura il più modulare possibile che permettesse successive e facili espansioni, miglioramento delle funzioni o aggiunta di nuove funzionalità.

Il lavoro svolto e la struttura di questa tesi è organizzata attraverso i seguenti capitoli il cui contenuto verrà ora brevemente illustrato:

- *il capitolo 1* si pone come obiettivo quello di esporre la problematica dell'information retrieval (IR) e dei motori di ricerca in generale, illustrandone i vari aspetti e funzionamenti;
- *il capitolo 2* offre una panoramica generale sulla tematica riguardante il web semantico, i processi e metodi di sommarizzazione, e le misure per i calcoli della similarità semantica;
- *il capitolo 3* illustra alcune applicazioni web che trattano sia la sommarizzazione, che le misure di calcolo di similarità, attuando i processi di gerarchie ontologiche;
- *il capitolo 4* ha lo scopo di descrivere in via completa il progetto in ogni suo aspetto. In primis, vengono descritte le diverse tecnologie utilizzate e i relativi strumenti di supporto utilizzati; successivamente, sono illustrate e spiegate, le diverse funzionalità del sistema, utilizzando schermate di esempio e diagrammi UML per i casi d'uso; infine, si mostrano come suddette funzioni vengono implementate facendo vedere frammenti di codice e pseudo-codice significativi;

1. WWW, i motori di ricerca, il web semantico e l'information retrieval

1.1 Il World Wide Web (WWW)

Il World Wide Web, spesso abbreviato in Web, è un servizio di Internet che permette di navigare ed usufruire di un insieme vastissimo di contenuti multimediali e di ulteriori servizi accessibili a tutti o ad una parte selezionata degli utenti di Internet.

Caratteristica principale del Web è che i suoi contenuti sono tra loro collegati (formando un ipertesto, tramite i cosiddetti *link*, collegamenti). Per quanto riguarda i contenuti, quindi, il Web possiede la straordinaria peculiarità di offrire a chiunque la possibilità di diventare editore e non solo.

La nascita del Web risale al 6 agosto 1991, giorno in cui Berners-Lee mise *on-line* su Internet il primo sito Web. Inizialmente utilizzato solo dalla comunità scientifica, il 30 aprile 1993 il CERN decide di rendere pubblica la tecnologia alla base del *Web*. A tale decisione fa seguito un immediato e ampio successo del *Web* in virtù della possibilità offerta a chiunque di diventare editore, della sua efficienza e, non ultima, della sua semplicità. Con il successo del Web ha inizio la crescita esponenziale e inarrestabile di Internet ancora oggi in atto, nonché la cosiddetta "era del *Web*" [WIK-WWW].

1.2 I motori di ricerca

Grazie al successo ottenuto dal Web, ed al suo continuo evolversi, in rete attualmente sono presenti milioni di documenti riguardanti qualsiasi argomento e materia; la necessità di ogni utente, sia esperto che neofita, è quella di reperire in

maniera efficace ed efficiente le informazioni di cui ha bisogno, considerando il fatto che esisteranno sicuramente diversi documenti che contengono l'informazione corretta.

Per permettere agli utenti di Internet di recuperare tali risorse all'interno della vasta mole di dati presenti nel Web, sono state create delle applicazioni, dette motori di ricerca, cioè dei sistemi automatici che analizzano un insieme di dati, spesso raccolti, restituendo un indice dei contenuti disponibili classificandoli in base a formule di tipo statistico-matematiche che ne indichino il grado di rilevanza data una determinata chiave di ricerca [WIK-MDR].

Attualmente il successo e la diffusione di tali motori risulta indispensabile per qualsiasi utente che intende cercare qualsiasi tipo di informazione presente nel Web. Per essere efficienti, tali motori di ricerca devono essere costantemente aggiornati.

Le fasi di lavoro dei motori di ricerca si dividono principalmente in:

- Analisi del campo d'azione (raccolta);
- Catalogazione del materiale ottenuto (Indicizzazione);
- Risposta alle richieste dell'utente (Ordinamento).

1.2.1 Analisi

L'analisi del campo (detta anche raccolta di informazioni), viene eseguita dai *crawler* (o *spyder*), che si occupano di raccogliere i documenti web e di scaricarli sul server locale. Considerato che non esiste un elenco completo delle pagine Web, l'unico metodo in grado di poter effettuare delle ricerche su di esse è quello di sfruttare i collegamenti tra di esse. I crawler si occupano di visitare automaticamente degli URI iniziali (definiti da una lista) e a seguire i successivi URI che trovano all'interno dei documenti analizzati, evitando i link già visitati ed inserendo di volta in volta nel database tutte le informazioni della pagina (contenuto testuale, informazioni su di essa tra cui la data di aggiornamento, e tanto altro ancora). I crawler sono in grado di effettuare visite sia in ampiezza che in

profondità con la possibilità di impostare gli opportuni criteri di limite per ciascun tipo di visita che si vuole effettuare [CR-05].

1.2.2 Catalogazione

In seguito si considera l'analisi delle pagine, a seconda di criteri che variano da motore a motore, alcune di esse vengono inserite nel database e nell'indice del motore di ricerca.

La parte testuale archiviata durante la fase di analisi verrà in seguito analizzata per fornire le risposte alle ricerche degli utenti.

L'analisi consiste nel segmentare il testo in singole parole ignorando la punteggiatura e tenendo conto anche dei caratteri alfanumerici; a questo punto, la lista di parole ottenute può essere ulteriormente filtrata utilizzando una "stop-list" (cioè eliminando termini come articoli, preposizioni e pronomi) e lemmatizzando le parole. Le parole estratte, vengono poi inserite in un file indice, il quale associa a ciascuna parola le pagine in cui essa compare. Ai fini delle prestazioni del motore di ricerca, un file indice è implementato mediante l'utilizzo di strutture dati che consentono un accesso veloce a ciascuna parola, ad esempio una tavola hash.

Nel file indice comparirà, per ogni singola parola, oltre alla lista delle relative pagine in cui compare, anche un peso della parola in ciascuna pagina; tale valore servirà poi nella fase successiva di lavoro del motore [CR-05].

1.2.3 Risposta

Considerato che anche una ricerca più specifica o particolare può restituire migliaia di risultati, è necessario ordinare i risultati di ricerca per pertinenza ed importanza; per questo motivo, viene assegnato un peso ad ogni associazione pagina-parola.

Il peso di un termine in un documento dipende da quanto il termine caratterizza il documento in oggetto, da quanto lo discrimina rispetto agli altri documenti e dalla lunghezza del documento stesso. Quindi le pagine con i pesi maggiori rispetto ai termini ricercati compariranno nelle prime posizioni rispetto alle altre pagine [CR-05].

Un secondo criterio fondamentale per l'ordinamento dei risultati, più efficace del metodo appena descritto, è basato sulla struttura del Web, a prescindere dal contenuto delle pagine; questo approccio è alla base dell'algoritmo di PageRank utilizzato da Google.

Il PageRank è un algoritmo di analisi che assegna un peso numerico ad ogni elemento di un collegamento ipertestuale di un insieme di documenti, con lo scopo di quantificare la sua importanza relativa all'interno della serie [WIK-PR]. Quindi si parte dal presupposto che certi siti siano più importanti e popolari di altri e che un indice significativo della loro popolarità sia dato dal numero di pagine che puntano ad essi. In realtà, per misurare la popolarità di un sito, contano sia la quantità dei siti che puntano ad esso sia la popolarità di questi ultimi.

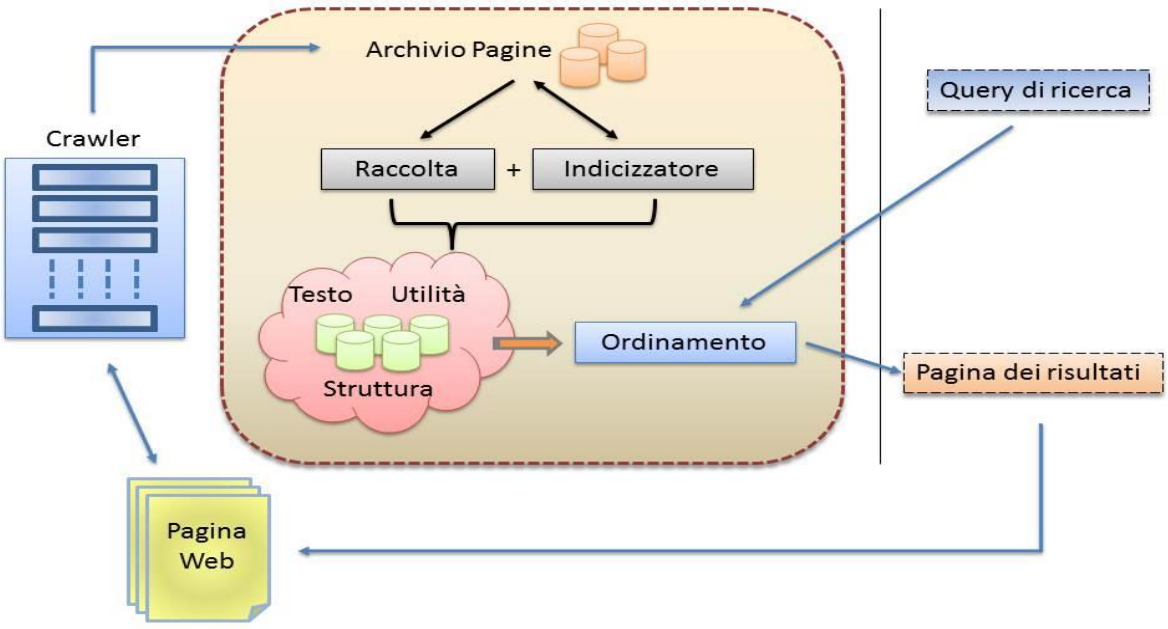

Figura 1.1 - Schema di funzionamento di un motore di ricerca

1.2.4 I limiti dei motori di ricerca

I motori di ricerca nonostante ciò, presentano comunque dei limiti che ne influenzano la qualità delle ricerche; il primo limite è quello che essi indicizzano solo una parte del Web. Tutta la parte non indicizzata la si può definire come Web

nascosto, costituito da tutte quelle pagine che non sono collegate ad altre e che i crawler non possono accedervi fisicamente; tra di esse sono presenti pagine create dinamicamente e pagine create tramite compilazione di form HTML.

Un altro limite è riconducibile alla loro logica di funzionamento: una volta caricate le pagine, essi recuperano quelle che contengono esattamente le parole specificate nell'interrogazione, risulta così una forte limitazione data l'ambiguità del linguaggio naturale, ad esempio una pagina che contiene lo stesso concetto con termini differenti non viene recuperata.

Altra ulteriore difficoltà viene data dalla modalità di visualizzazione dell'output, la quale costringe l'utente a dover scansionare i documenti uno ad uno per capirne i contenuti, e spesso per motivi di tempo e di costo tipicamente ci si sofferma solamente sui primi risultati offerti dalla prima pagina, con una forte sottoutilizzazione delle capacità del sistema [CR-05].

Infine, oltre ai problemi sopra elencati, bisogna considerare il fattore spam, ovvero il tentativo di influenzare il posizionamento delle pagine utilizzando metodi scorretti come ad esempio inserire parole "civetta" nascoste all'interno della pagina.

1.2.5 Ottimizzazione dei motori di ricerca

Per cercare di superare gli ostacoli sopra esposti, sono state realizzate diverse tipologie di motori di ricerca che cambiano o ampliano la logica di funzionamento di un normale motore di ricerca.

L'introduzione della tecnica di analisi definita *clicktrough* ha migliorato la qualità dei risultati di ricerca. Tale metodo si basa tenendo conto del numero di volte che un documento web viene visitato dagli utenti.

Nonostante questo valore non fornisca dati esatti, esso restituisce comunque preziose informazioni circa l'interesse e le intenzioni degli utenti, infatti se una pagina viene visitata è probabile che questa fornisca contenuti utili in base alla ricerca effettuata.

Grazie alla continua evoluzione sono stati sviluppati diverse tipologie di motori di ricerca:

- *Motori di ricerca sociali*: sono una tipologia di motori che determina la rilevanza dei risultati considerando le interazioni ed i contributi degli altri utenti, dove la condivisione di informazioni può avvenire in svariati modi, dalla condivisione di segnalibri all'assegnazione di parole chiave ai documenti [WIK-MRS]
- *Motori di ricerca collaborativi*: questa tipologia di motori è emergente per la ricerca web all'interno di reti intranet aziendali. Permettono agli utenti più esperti di guidare le persone meno esperte nelle ricerche, fornendo termini di ricerca o link, essi possono essere classificati in base alla tipologia di collaborazione che offrono; *implicita,esplicita,mediata* [WIK-MRC].
- *Motori di ricerca personalizzati*: in questa tipologia, ogni utente (studente, ricercatore, ecc) che esegue la ricerca possiede un profilo contenente informazioni personali e tale profilo è in grado di influenzare l'ordine dei risultati, grazie ad una serie di combinazioni totalmente personalizzabili.
- *Motori di ricerca multimediali*: sul Web oltre ai classici documenti testuali, sono presenti migliaia di documenti multimediali, come immagini, file audio e video, ecc. La tecnica più nota per la ricerca di immagini è quella basata sul testo che avviene utilizzando il contenuto circostante all'immagine. Il reperimento di immagini può anche essere eseguito calcolando la similarità visuale tra un esempio fornito dall'utente e le immagini da selezionare. Inoltre esistono anche tecniche avanzate per il riconoscimento del parlato, infatti è possibile estrarre le parole da una traccia audio ed utilizzare per accedere direttamente al segmento che le contiene, in risposta ad una interrogazione effettuata.
- *Motori di risposta*: questo tipo di motori, non si limitano ad indicare un insieme di pagine che potrebbero contenere la risposta alla ricerca effettuata, ma sono in grado di fornire direttamente la risposta. Infatti rappresentano una funzionalità aggiuntiva che può essere integrata nei motori di ricerca già esistenti, come, ad esempio se in Google cerchiamo il meteo di una determinata città, è possibile vedere la risposta alla ricerca direttamente in prima posizione senza dover visitare nessun sito specifico.

- *Meta-motori di ricerca*: questi sono strumenti di ricerca che inviano delle interrogazioni ad altri motori di ricerca e aggregano tutti i risultati ottenuti in una singola lista.
- *Motori di ricerca con clustering*: essi consentono di avere una panoramica sui temi e sugli argomenti disponibili, visualizzando gruppi e sottogruppi, si possono trovare risultati interessanti, inoltre i risultati simili sono raggruppati insieme piuttosto che essere sparsi in un elenco.
- *Motori di ricerca per il web semantico*: un passo avanti nello sviluppo dei motori di ricerca è stato effettuato con l'invenzione del Web Semantico. Questi motori sono in grado di generare ricerche sfruttando le informazioni contenute attraverso degli schemi, utilizzando linguaggi come RDF per il contenuto dei dati e le ontologie formali per specificare concetti e regole, quindi descrivere la giusta destinazione di un collegamento, trovando così risposte collegate *logicamente* alle interrogazioni effettuate.

1.3 Il Web Semantico

Internet, come già detto, è un insieme di testi, un insieme molto vasto di documenti che descrivono dei contenuti, collegati tra di loro tramite l'elemento chiave che l'HTML ha saputo proporre, il *link*.

L'utente si orienta nel Web principalmente grazie a due fattori, la propria esperienza di navigazione e la capacità di evocazione che possono avere parole o espressioni chiave di ricerca, utilizzando principalmente come strumento, i motori di ricerca.

Tuttavia, la capacità espressiva di un collegamento automatico dipende dalla applicazione che lo gestisce. Interrogando un qualsiasi motore di ricerca, un utente che inserisce una qualsiasi espressione è in grado di individuare il contenuto cercato tra i vari risultati di risposta offerti dal motore stesso stabilendo quale espressione si adatti meglio come contenuto di ricerca, basandosi sul titolo o sul testo proposto. Comunque si deve considerare il fatto che, una qualsiasi query attivata tramite un motore è sempre soggetta al rischio della ambiguità. Ad

esempio, cercando la parola "albero" si possono trovare contenuti legati all'informatica, alla nautica e alla botanica.

La morale di tutto ciò, è che Internet è si una miniera di testi ed informazioni collegate tra di loro, ma proprio questi collegamenti spesso risultano deboli, nel senso che sono troppo generici o vaghi.

I collegamenti intesi, non sono quelli di tipo sintattico, cioè legati al funzionamento di un programma, quanto piuttosto quelli legati alla capacità di descrivere il significato di un collegamento, il termine corretto per descrivere questa funzione, è *capacità semantica*.

Prestando attenzione, abitualmente si usano strutture con valore semantico, anche se minimo. Utilizzando come esempio un qualsiasi archivio, si può facilmente organizzare un DataBase (DB) nel quale inserire campi con diversi elementi. A questo punto si può permettere all'utente di effettuare svariati tipi di ricerche all'interno del DB basandosi su uno schema prestabilito in fase di realizzazione del progetto, il quale indica il metodo utilizzato per archiviare le informazioni.

Lo schema (si pensi ad uno schema XML) è un insieme di regole che stabiliscono il modo in cui i dati devono essere organizzati, e, dato che uno schema definisce anche le relazioni fra i dati, ecco che, si è in grado di esprimere i vincoli che legano o oppongono due classi di dati.

L'idea del Web Semantico nasce semplicemente estendendo l'idea di utilizzare schemi per descrivere domini di informazione.

1.3.1 La nascita

Quando si parla di Web Semantico, si intende una proposta di Web che possieda strutture di collegamenti più espressive di quelle attuali.

Il termine *Semantic Web*, coniato dal suo ideatore, Tim Berners-Lee e proposto per la prima volta nel 2001, è stato associato all'idea di un Web nel quale agiscano "agenti intelligenti". Si intende quindi la trasformazione del Web in un ambiente dove i documenti pubblicati (pagine HTML, file, immagini, e così via) siano associati ad informazioni e dati (metadati) che ne specifichino il contesto

semantico in un formato adatto all'interrogazione, all'interpretazione e, più in generale, all'elaborazione automatica.

Con tale interpretazione, saranno possibili ricerche molto più evolute di quelle attuali, basate sulla presenza nel documento di parole chiave, e altre operazioni specialistiche come la costruzione di reti di relazioni e connessioni tra documenti secondo logiche più elaborate del semplice collegamento ipertestuale. Questa evoluzione del nuovo Web, inizia con la definizione, da parte del W3C, dello standard *Resource Description Framework* (RDF), una particolare applicazione XML che standardizza la definizione di relazioni tra informazioni ispirandosi alla logica dei predicati e ricorrendo agli strumenti tipici del Web (ad es. URI) e dell'XML (namespace) [WIK-WS].

1.3.2 Gestione e prospettive

La ri-formulazione in modo più efficace di strutture e dati esistenti, non è che il primo e più semplice vantaggio offerto dall'introduzione del Web Semantico. Infatti, grazie all'introduzione delle ontologie, si ha la possibilità di esprimere direttamente la struttura della nostra conoscenza e di permettere alle macchine di elaborare automaticamente la conoscenza stessa, non più solo le semplici informazioni che erano disponibili in precedenza. In questo modo, si aggiunge alla semplice informatica, una elaborazione automatica di conoscenza.

Analizzando questi aspetti, il web come si presenta oggi, richiede sempre maggiormente l'utilizzo di strumenti più progrediti, in grado di facilitare e velocizzare la navigazione attraverso le migliaia di documenti forniti dalla pubblicazione multimediale. L'obietto proprio del Web Semantico è quello di fornire in futuro, un senso vero e proprio alle pagine web ed ai suoi collegamenti, dando la possibilità di cercare solo ciò che realmente si richiede.

Sfruttando questa metodologia, si può aggiungere un senso compiuto alle pagine create, quindi ai testi presenti, ed affiancando queste regole al motore di ricerca, si ha la possibilità di individuare realmente ciò che si sta cercando, perché i risultati che non soddisfano la richiesta saranno scartati a priori.

Il Web Semantico per funzionare deve poter disporre di una informazione strutturata e di regole di deduzione per gestirla, in modo da accostare quelle informazioni che un'interrogazione ha richiesto. Bisogna sottolineare, che uno degli elementi fondamentali del web semantico sarà la compresenza di più ontologie [WIK-WS].

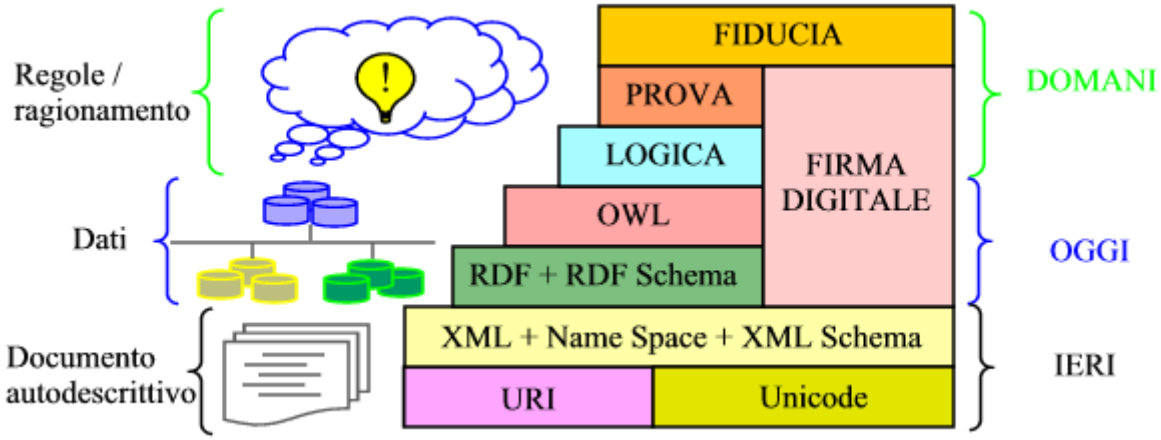

Figura 1.2 - Architettura web semantico

1.3.3 Ontologia informatica

In informatica una *ontologia* è una rappresentazione formale, condivisa ed esplicita di una concettualizzazione di un dominio di interesse. In dettaglio, si tratta di una teoria assiomatica del primo ordine esprimibile in una logica descrittiva.

Il termine ontologia è entrato in uso nel campo dell'intelligenza artificiale e della rappresentazione della conoscenza, per descrivere il modo in cui diversi schemi vengono combinati in una struttura dati contenente tutte le entità rilevanti e le loro relazioni di dominio. I programmi informatici possono poi usare l'ontologia per una varietà di scopi, tra cui il ragionamento induttivo, la classificazione, e svariate tecniche per la risoluzione dei problemi [WIK-ON].

Nonostante essa sia di tradizione eminentemente teorica, l'uso e lo studio delle ontologie applicate al settore informatico, in particolare alle problematiche relative al sovraccarico informativo ed altre moderne tecnologie (guerra di informazione, dipendenza da Internet, ricerca) è sempre maggiore.

Con l'introduzione del Web Semantico, l'uso dell'ontologia si è reso piuttosto popolare, infatti , si è iniziato ad usare il termine per definire generici modelli di dati. Nel settore informatico, si consolida l'idea, che il termine *ontologia* si debba riferire al tentativo di formulare una concettualizzazione esaustiva e rigorosa nell'ambito di un dato dominio; generalmente si tratta di una struttura dati gerarchica che contiene tutte le entità rilevanti, le relazioni esistenti tra esse, le regole, gli assiomi ed i vincoli specifici del dominio.

Un database liberamente disponibile, basato e progettato come se fosse una rete semantica, è "WordNet", attualmente visto anche come un dizionario grazie all'aggiunta di definizioni ai propri termini. Tratteremo WordNet nei particolari nei capitoli successivi.

1.4 Information Retrieval

Citando motori di ricerca e web semantico, non si può non introdurre un argomento più vasto e generico, come l'*information retrieval* (IR). Con tale termine si definiscono l'insieme delle tecniche utilizzate per il recupero mirato dell'informazione in formato elettronico. Per "informazione" si intendono tutti i documenti, i metadati, i file presenti all'interno di banche dati o nel Web in generale. Il termine, è stato coniato allla fine degli anni '40 da Calvin Mooers, ma attualmente viene usato quasi esclusivamente in ambito informatico [WIK-IR].

L'IR nasce dall'incrocio di diverse discipline, quindi viene definito come un campo interdisciplinare, infatti coinvolge la psicologia cognitiva, l'architettura informatica, la filosofia (ontologie), il design, la linguistica, la scienza dell'informazione e l'informatica.

Il recupero di informazione dei sistemi di IR, si basa su linguaggi di interrogazione basati su comandi testuali; i concetti fondamentali sono:

- *Query*: le interrogazioni sono stringhe di parole-chiave che rappresentano l'informazione richiesta (es. motori di ricerca);
- *Oggetto*: rappresenta un'entità che mantiene o racchiude informazioni in una banca dati (es. un documento di testo è un oggetto di dati).

I motori di ricerca presenti nel Web sono l'esempio di applicazioni più note ed ovvie delle teorie dell'Information Retrieval.

1.4.1 Tecniche di IR

Come detto, l'information retrieval adopera tutte le tecniche disponibili per il recupero di dati, una di quelle che stanno alla base dell' IR, consiste nell'effettuare la differenza della ricerca delle parole tra una pagina web e la ricerca delle parole all'interno di parti di testo più brevi e create appositamente dall'autore perché il documento possa essere riconoscibile da tutti.

Un'altra tecnica molto citata nell'IR è il rapporto tra il "richiamo" dei documenti che citano quella frase nel database e la "precisione" di estrarre solo quelli più pertinenti:

- $richiamo = \frac{documenti\ pertinenti\ recuperati\ nella\ ricerca}{documenti\ (pertinenti\ o\ no) recuperati\ nella\ ricerca}$
- $precisione = \frac{documenti\ pertinenti\ recuperati\ nella\ ricerca}{documenti\ pertinenti\ esistenti\ nella\ banca\ dati}$

Le strategie di *richiamo* e *precisione* dei vari motori possono variare da caso a caso e da chiave a chiave. È chiaro che una ricerca che fornisca un richiamo ed una precisione totale è quasi impossibile da ottenere, come è evidente che se si aumenta il richiamo, diminuisce la precisione.

Risulta molto conveniente, in termini di costo di ricerca, offrire un richiamo molto ampio senza cercare molti raffinamenti quando la ricerca è in corso. Tra i raffinamenti in possesso dell'utente ci può essere l'operatore Booleano ovvero:

- operatore *AND*: utilizzato per rintracciare tutti i documenti che contengono tutti i termini, ad esempio " x AND y" si otterranno tutti i record che contengono sia la parola "x" che quella "y";
- operatore *OR*: utilizzato per rintracciare i documenti che contengono almeno un termine, ad esempio specificando "x OR y" si otterranno tutti i

documenti presenti nel database che contengono il termine "x", tutti quelli che contengono il termine "y", e tutti quelli che le contengono entrambe;

- operatore *NOT*: utilizzato per rintracciare i documenti di una determinata chiave escludendo quella che però ne soddisfa un'altra, ad esempio specificando "x NOT y" si otterranno tutti i documenti che contengono il termine "x" tranne quelli che contengono anche il termine "y";
- operatore *XOR*: utilizzato per rintracciare i documenti che conterranno solo una chiave di quelle indicate, ad esempio specificando "x XOR y" si otterranno tutti i documenti che conterranno il solo termine "x" e tutti quelli che conterranno il termine "y", ma non quelli che le conterranno entrambe.

Un ulteriore tecnica per la classificazione, viene indicato dalla *F-measure*, in pratica si definisce la media armonica pesata fra precisione e recupero. La versione tradizionale, detta anche *bilanciata*, è data da:

- $F\text{-}measure = \frac{2 \text{ x } precisione \text{ x } recupero}{precisione + recupero}$

Questa misura, è detta anche F_1 dato che sia la precisione che il recupero hanno peso 1. In generale, la formula è:

- $F\text{-}measure = \frac{(1+N^2) \text{ x } precisione \text{ x } recupero}{((N^2 \text{ x } precisione) + recupero)}$

Per concludere con successo una ricerca di informazioni, è necessario rappresentare i documenti. Essi possono essere classificati secondo due criteri, uno matematico e l'altro in base alle proprietà del modello.

I modelli matematici si possono classificare in:

- *Modelli Set-theoretic*: rappresentano i documenti mediante insiemi. Le somiglianze derivano in genere da operazioni teoriche su questi insiemi. I modelli più comuni sono quello *Booleano standard ed esteso* e il *recupero fuzzy*;
- *Modelli Algebrici*: rappresentano i documenti e le query con vettori, matrici o tuple, che, utilizzando un numero finito di operazioni algebriche, vengono

trasformati in una misura numerica, la quale esprime il grado di somiglianza dei documenti con la query;

- *Modelli Probabilistici*: trattano il processo di recupero dei documenti come un esperimento aleatorio multi-livello. Le somiglianze sono quindi rappresentate come probabilità. Sono spesso usati i teoremi probabilistici come quello di Bayes;

I modelli in base alle proprietà si possono classificare in:

- *Modelli senza interdipendenza dei termini*: trattano diversi termini/parole come non interdipendenti. Ciò viene rappresentato spesso nei modelli a spazi vettoriali affermando che i vettori dei termini siano ortogonali, o nei modelli probabilistici affermando che le variabili dei termini siano indipendenti;
- *Modelli con interdipendenza dei termini intrinseca*: consentono una rappresentazione diretta delle interdipendenze tra termini. Il grado di interdipendenza tra due termini è definito dal modello stesso, generalmente, esso è direttamente o indirettamente derivato dalla co-occorrenza di questi termini nell'intero insieme di documenti;
- *Modelli con interdipendenza dei termini trascendente*: consentono una rappresentazione diretta delle interdipendenze tra termini, ma essi non riportano come l'interdipendenza tra due termini sia definita. Si riferiscono ad una fonte esterna per stabilire il grado di interdipendenza tra due termini (ad esempio, una persona fisica o ad algoritmi sofisticati).

2. SOMMARIZZAZIONE E SIMILARITA' SEMANTICA NEL WEB

2.1 Introduzione

Nel capitolo appena concluso si è descritto lo scopo, il funzionamento di un motore di ricerca ed in particolare, i limiti che tali motori presentano, dalla capacità di limitare l'ambiguità della ricerca e dei risultati, all'aspetto più pratico rappresentato dalla modalità di visualizzazione dei risultati di ricerca.

Infatti, eseguita una ricerca, all'utente viene restituita una pagina di output contenente tutti i risultati, ed analizzando questo aspetto, bisogna considerare da un lato la "lessicalità" di tali motori, e dall'altro il quantitativo enorme di documenti che sono presenti nel Web. L'unione di questi due fattori restituirà all'utente migliaia di risultati, ma con molta probabilità la maggior parte di essi saranno di dubbia utilità.

Effettuata una ricerca, quindi all'utente non resta che la sola scelta di esplorare i documenti restituiti, basandosi su poche righe di testo presenti nel titolo e nell'anteprima del documento, ed inoltre affidandosi alla propria esperienza senza aver comunque nessuna garanzia di trovare ciò di cui si ha veramente bisogno.

Una soluzione possibile per superare questo limite è stata vista nell'applicazione della sommarizzazione automatica delle pagine Web e dal calcolo di similarità tra i termini presenti all'interno di tali documenti.

Tali approcci applicati al mondo del Web sono paragonati oggi alla punta del progresso, infatti, partendo dal presupposto fondamentale che la somiglianza di contesto implica somiglianza di significato, questi metodi applicati ai motori di ricerca attuali consentono di individuare i documenti pertinenti tra di loro e di sfruttare poi le informazioni contestuali recuperate tramite le parole di interesse maggiore per l'elaborazione dei dati a disposizione.

2.2 Definizione

Il reperimento di numerose informazioni e l'elaborazione di applicazioni legate al linguaggio naturale richiedono oggi la conoscenza della similarità semantica tra parole o termini e della sommarizzazione di pagine. Inoltre, le misure di similarità semantica risultano importanti per molti compiti di elaborazione del linguaggio naturale (NLP), come la modellazione del linguaggio, l'induzione di grammatica, il senso di parola disambigua, la comprensione del parlato e dei sistemi di dialogo del parlato.

In generale la sommarizzazione delle pagine Web si pone come obiettivo, quello di creare un insieme di termini (detto "sommario") in grado di descrivere quali siano gli argomenti principali contenuti nel documento; lo sviluppo di questo metodo risale agli anni '50 con la sommarizzazione di file di testo puro tramite l'algoritmo di Luhn, ed a tutt'oggi tale approccio appartiene al ramo dell'*Information Retrieval* [LUHN-58].

Come anticipato, i primi metodi di sommarizzazione furono applicati a documenti di puro testo, successivamente si è pensato di poter applicare tali metodi alle pagine Web, dove però si è vista la necessità di dover effettuare lievi adattamenti in base alla loro logica di funzionamento considerata la complessa struttura che rappresenta un documento Web rispetto ad un semplice documento di testo.

Attuando una procedura di sommarizzazione automatica, essa deve essere in grado di produrre un sommario contenente una panoramica generale dei principali argomenti presenti nel documento, senza però esporre più volte lo stesso concetto, evitando così di creare ridondanza.

Una prima classificazione dei metodi di sommarizzazione la si può effettuare osservando il rapporto con il dominio applicativo:

- *Generale*: viene generato un sommario contenente un resoconto di tutti gli argomenti principali trattati nei documenti;
- *Query-based*: viene generato un sommario contenente le informazioni più rilevanti presenti nel documento rispetto ad una chiave di ricerca, tale

metodologia viene applicata specialmente dai motori di ricerca in ambito di indicizzazione per produrre sommari delle pagine Web.

Una ulteriore suddivisione può essere fatta analizzando la logica di funzionamento ovvero, si possono classificare in:

- *Supervisionato*: vengono generati sommari attraverso un'analisi semantica del testo esaminato utilizzando un insieme di dati, quindi, la qualità del risultato dipende molto dalla qualità dell'insieme di dati utilizzato;
- *Non supervisionato*: vengono generati sommari riferendosi a misure statistiche presenti nel documento (come la frequenza di parole, lunghezza del testo …) e dei parametri imposti dal programmatore (come la frequenza minima e massima entro cui un termine può essere considerato).

Un'ultima classificazione può essere fatta basandosi sulla tipologia di sommario che un metodo produce, ovvero può risultare:

- *Estrattivo*: viene generato un sommario contenente frasi e termini presenti nel documento che sono stati considerati significativi, tale tipologia è attualmente la più utilizzata in quanto produce buoni risultati in un tempo accettabile;
- *Astrattivo*: viene generato un sommario che necessariamente non contiene termini o frasi presenti nel documento, ma vocaboli che ne descrivono al meglio il contenuto ed il senso dei principali argomenti, generalmente questa tipologia genera sommari con qualità migliore ma richiedono un tempo di elaborazione elevato a causa dell'analisi semantica che si deve eseguire all'interno del documento.

Definito il metodo di sommarizzazione in generale, bisogna anche evidenziare l'importanza che le misure di calcolo di similarità semantica hanno saputo dare a favore dei metodi di sommarizzazione.

Generalmente, gli approcci utilizzati per definire la somiglianza o non dei documenti web, utilizzando i risultati restituiti da uno o più motori di ricerca, che utilizzano una o più query per estrarre i risultati, si possono suddividere nelle seguenti categorie di misura di somiglianza :

- *Misure che si basano solo sul numero di risultati restituiti;*
- *Misure che scaricano una serie di documenti di alto livello (top-ranked) e che applicano poi tecniche di elaborazione dei testi su di essi;*
- *Misure che combinano i due approcci precedenti.*

Gli algoritmi utilizzati per il calcolo di similarità tra documenti Web, sono utilizzati in una vasta gamma di applicazioni, quali l'annotazione automatica di pagine Web [CHS-04], costruzione di social networks [MIKA-05], classificazioni di qualsiasi genere ed ottimizzazione dei motori di ricerca.

Tuttavia, nella maggior parte dei casi, la forma delle query e/o il processo caratteristico di estrazione dei termini o dati, è relativo all'applicazione, per esempio, se si è interessati alla classificazione dei generi di film, sarà utile includere il termine "film" nella query utilizzata.

I risultati di valutazione forniscono, inoltre, la comprensione del testo nel processo di acquisizione della lingua e nella cognizione umana, risultando (a livello semantico) principalmente un metodo non supervisionato.

Diversamente, nell'utilizzo dei metodi supervisionati, spesso si fa riferimento a database ontologici quali WordNet o database specifici per il settore medico come il MeSH.

2.3 Metodi di sommarizzazione

Esistono diverse tecniche di base per creare sommari automatici, solitamente tali tecniche non vengono utilizzate mai singolarmente, ma vengono combinate con altre a seconda del tipo di risultato che si vuole ottenere; di seguito si presenteranno alcune tra quelle più utilizzate.

2.3.1 Metodo di Luhn per le pagine Web

Il metodo di Luhn [LUHN-58], è stato uno dei primi metodi di sommarizzazione automatica appartenente alla classe dei metodi estrattivi non supervisionati.

E' stato ideato per sviluppare sommari automatici di documenti di testo, e successivamente adattato per creare sommari di pagine Web.

In generale il metodo, crea il sommario automatico sulla base di supposizioni comuni a tutti i documenti di testo.

Innanzitutto è facile dedurre che i termini più utilizzati rappresenteranno anche i termini più rilevanti per la costruzione del sommario, mentre in secondo luogo l'importanza di una frase dipende dal numero di termini rilevanti presenti al suo interno e dalla rispettiva posizione che occupa all'interno del documento. Sulla base di questi valori statistici, il metodo di Luhn elabora il testo cercando le frasi ed i termini con maggior rilievo nelle seguenti modalità:

- inizialmente viene analizzato tutto il testo e calcolato il numero di occorrenze di ogni singola parola;
- da tale lista vengono poi scartati i termini più comuni, quali, articoli, congiunzioni, avverbi ecc…, generalmente tale lista viene creata dal programmatore, in più possono essere scartati anche termini che presentano un numero di occorrenze o troppo elevato, o poco. In genere per poter effettuare questo passo si definiscono due parametri (scelti dal programmatore) che rappresentano un intervallo di valori per cui un termine possa essere considerato o meno;
- per migliorare i risultati si possono considerare uguali quei termini che risultano simili tra di loro, sommando le rispettive occorrenze;
- finita la scansione di ciascun termine, è possibile analizzare l'importanza di una frase, la quale viene valutata in base al numero di termini significativi presenti in essa. Per far ciò, si deve quindi determinare il numero di termini significativi e calcolarne la distanza che li separa, misurati in termini di parole non significative. Il programmatore quindi deve anche impostare una

variabile che rappresenti la distanza massima, e come in precedenza, tali parametri avranno una certa influenza sulla qualità del risultato;

- fatto ciò, viene considerata una determinata frase del documento e presa in atto una sua sottoparte compresa tra due termini significativi, la cui distanza non superi il parametro impostato;
- successivamente viene contato il numero di parole chiave all'interno della sottoparte, ed il quadrato di questo valore viene poi diviso per il numero totale di termini presenti nella sottofrase, ottenendo così il valore numerico che rappresenta il *fattore di significatività* della frase:

$$F = \frac{TS * TS}{N}$$

dove TS rappresenta il valore dei termini chiave ed N rappresenta il valore totale dei termini presenti.

Quindi, per utilizzare tale metodo all'interno delle pagine Web, è necessario apportare lievi modifiche al metodo in questione.

Considerato che, la sommarizzazione di pagine Web serve soprattutto agli utenti per facilitarne la navigazione, le ricerche e quant'altro, si ha come primo obiettivo di trasformare tale metodo, in un metodo query-based, cioè in grado di restituire dei sommari in base alla chiave di ricerca.

Per l'associazione delle query di ricerca ai termini del documento, viene assegnato un valore di significatività *w* tramite la seguente formula:

$$W_i = (1 - \alpha) * \frac{tf_i^p}{\max(tf_i^p)} + \alpha * \frac{tf_i^q}{\max(tf_i^p)}$$

Dove:

- tf_i^p è la frequenza con cui compare l'i-esimo termine nella pagina web
- tf_i^q è la frequenza con cui compare l'i-esimo termine nell'insieme delle parole chiave delle query

Il simbolo α è un parametro il cui valore è compreso tra 0 e 1. Viene utilizzato per bilanciare i pesi delle due misure.

Dopo aver calcolato la significatività di ciascun termine all'interno del testo, vengono selezionati i termini la cui significatività è maggiore, ed utilizzando il normale metodo di Luhn, viene calcolata la significatività delle singole frasi per la creazione del sommario.

2.3.2 DOM-Based Summarization

L'algoritmo DOM-Based, è stato realizzato sfruttando la struttura delle pagine Web per effettuarne la somarizzazione.

Le pagine Web, realizzate grazie al linguaggio di markup HTML (HyperText Markup Language), prevedono una struttura ad albero, dove ogni elemento è racchiuso all'interno di specifiche marcature dette "*tag*".

Questo tipo di algoritmo, sfruttando proprio la particolare struttura offerta, esegue la sommarizzazione sul documento, eliminando gli elementi indesiderati e di disturbo, come possono essere i pop-up, link o immagini che non contribuiscono a descrivere il contenuto informativo del documento. In particolare, viene utilizzato un procedimento di estrazione dei contenuti per eliminare gli elementi di intralcio, ed i risultati di questa operazione possono essere utilizzati per risolvere problemi legati al Natural Language Processing (NLP) ed all'Information Retrieval, inclusa la sommarizzazione automatica.

Il primo step di questo procedimento consiste nell'analizzare tramite un parser HTML la pagina web, creando un DOM-Tree rappresentante una struttura dati utilizzata per memorizzare gli elementi della pagina in maniera gerarchica.

Figura 2.1 - Struttura ad albero di un documento HTML

Considerando l'esempio di pagina HTML presente in figura (2.1) nella pagina precedente, viene generata la struttura così come segue:

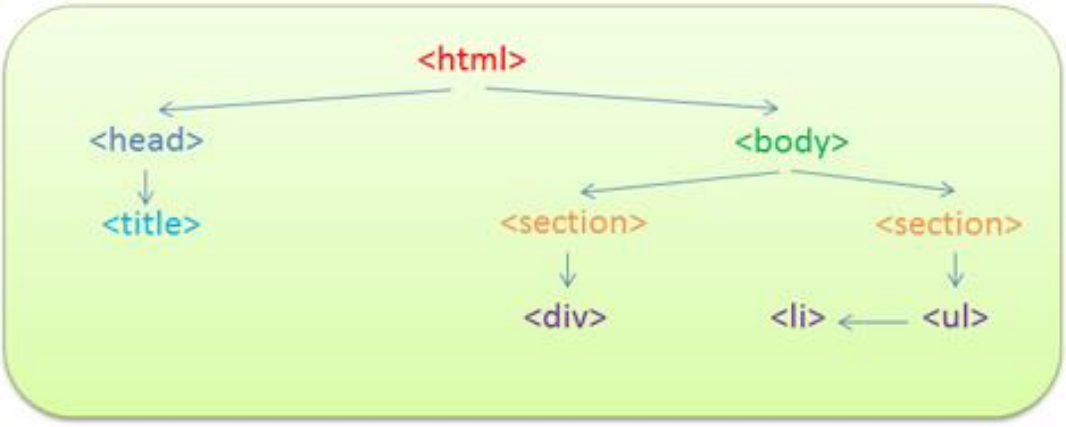

Figura 2.2 - Struttura DOM-Tree

Nella struttura creata, assieme alle informazioni rilevanti, vengono estratte anche quelle irrilevanti, quindi, è necessario dover eliminare tali informazioni, e per far ciò viene effettuata la scansione dell'albero più volte, applicando differenti filtri a seconda delle necessità.

I filtri che possono essere utlilizzati sono di tipo:

- *Advertisement Remover*: ovvero, durante la scansione verifica se i tag *<herf>* e gli attributi *src* fanno riferimento ad indirizzi di server presenti su una black list (per esempio quelli che gestiscono gli annunci pubblicitari), e se così fosse, tutti gli elementi che ne fanno riferimento vengono eliminati dalla struttura; una lista nera di tali server può essere reperita all'indirizzo [ACCS-09];
- *Link List Remover*: durante la scansione, in presenza di tabelle vengono analizzate le celle *<td>* presenti e per ognuna di esse viene calcolato il rapporto tra i numeri di termini presenti nel link ed il numero di termini che non sono presenti, se il valore ricavato supera una determinata soglia imposta a priori, il contenuto della cella viene rimosso;
- *Empty Table Remover*: considerata una estensione del precedente filtro, durante la scansione vengono rimosse le tabelle vuote o con scarso contenuto informativo, in pratica, una tabella viene considerata poco rilevante quando non contiene un numero minimo di termini al suo interno (tale valore è sempre impostato a priori);
- *Removed Link Retainer*: ultimo filtro applicabile, in quanto memorizza tutti gli elementi che sono stati eliminati dai filtri precedenti reinserendoli in testa al documento in modo che nessun contenuto informativo sia perso nel caso in cui uno dei filtri precedenti abbia considerato erroneamente una informazione corretta come se fosse errata, ciò permette di creare una sorta di cuscinetto di salvataggio per la costruzione del sommario.

Applicati i filtri, la struttura ottenuta contiene esclusivamente le informazioni rilevanti, le quali possono poi essere visualizzate sotto forma di pagina HTML o eliminando i tag HTML sotto forma di documento di testo, al quale è poi possibile applicare un qualsiasi metodo di sommarizzazione automatica.

2.4 Metodi di calcolo per la similarità semantica

Applicati ai modelli di sommarizzazione automatica delle pagine web, sono stati studiati ed elaborati diversi metodi per rilevare la similarità semantica tra termini o parole.

Tali metriche che misurano la similarità semantica, si possono classificare nel seguente modo:

- *Metrica di sorveglianza delle risorse base*: questa metrica consulta solamente le risorse di conoscenza umane, come le ontologie;
- *Metrica di conoscenza approfondita del testo estratto*: rappresenta quella metrica che esegue l'estrazione del testo facendo riferimento anche alle risorse di conoscenza;
- *Metrica non supervisionata di co-occorrenza*: rappresenta quella metrica non supervisionata che presuppone che la similarità semantica tra parole o termini possa essere espressa da un rapporto di associazione alla funzione in base al loro co-avvenimento;
- *Metrica di base su testo non supervisionato*: rappresenta la metrica che si basa completamente sul testo del documento e sfrutta la prossimità delle parole o dei termini per computarne la similarità semantica.

Le ultime due categorie di metrica descritte non usano alcuna risorsa di lingua o di conoscenza approfondita, entrambe dipendono esclusivamente dai risultati restituiti dai motori di ricerca. In questo senso, questi approcci si riferiscono ai metodi "non-supervisionati" in quanto nessun dato semanticamente identificato, o nessun concetto di ontologia, è richiesto per computare la distanza semantica tra i termini in questione.

Tuttavia, le risorse di base e la conoscenza approfondita dell'estrazione del testo, sfruttano le ontologie, quindi fanno riferimento ai concetti dei metodi "sorvegliati". Molti metodi basati sull'estrazione delle risorse di base sono stati proposti all'interno della letteratura, per effettuare il calcolo della similarità

semantica, facendo uso di database che sfruttano concetti ontologici, come WordNet o MeSH.

Tipicamente, le metriche che si basano sul testo dei documenti, usano caratteristiche contestuali per computarne la similarità. Il contesto base opera sotto il presupposto che termini con simili contesti, abbiano anche simile significato.

Uno dei primi studi su questa ipotesi, è data dagli studi di Rubenstein e di Goodenough dove dichiarano che "le parole che hanno simile significato, si presentano in simili contesti" [RG-65]. Facendo uso di tale asserzione, il presupposto di similarità semantica tra due termini può essere misurato stimando la differenza tra le distribuzioni di probabilità delle loro caratteristiche contestuali.

Svariati contesti di metriche sono proposti dalla letteratura, dalle sopracitate distribuzioni di capacità contestuali alla rappresentazione dell'ambiente contestuale di un termine come il modello "*bag-of-words*" [LN-98], dove secondo questo modello le caratteristiche contestuali di un termine formano gli elementi di un vettore. Per cui assumendo l'indipendenza tra le caratteristiche, la similarità di due termini viene computata come il prodotto della caratteristica dei loro vettori (definita similarità di coseno) [ITPFP-06].

Recentemente, la metrica di similarità dei contesti è stata applicata per costruire collezioni di documenti, tramite l'interrogazione di motori di ricerca e di download di documenti top-ranked computandone la similarità semantica.

2.4.1 Metrica di similarità in base al conteggio di pagine

La metrica che si basa su conteggio di pagine usa i rapporti di associazione tra i termini in questione sfruttando la loro frequenza di co-avvenimento all'interno dei documenti. Il presupposto base dettato da questo metodo, è l'alto rapporto indicato dalle relazioni semantiche tra i termini.

Di seguito vengono definite per i documenti indicizzati dai motori di ricerca le seguenti notazioni:

NOTAZIONE	DESCRIZIONE
{ D }	Set of all documents indexed by search engine
| D |	Number of documents in {D}
w	A word or term
{ D | w }	Subset of {D}, documents indexed by *w*
{ D | $\boldsymbol{w_1}$, $\boldsymbol{w_2}$ }	Subset of {D}, documents indexed by $\boldsymbol{w_1}$, $\boldsymbol{w_2}$
| D | w |	Fraction of documents in {D} indexed by *w*
| D | $\boldsymbol{w_1}$, $\boldsymbol{w_2}$ |	Fraction of documents in {D} indexed by $\boldsymbol{w_1}$ and $\boldsymbol{w_2}$

Figura 2.3 - Tabella di definizione per gli insiemi di documenti indicizzati dai motori di ricerca

Alcune dei metodi di co-avvenimento dei termini per computare la similarità semantica tra le coppie di termini che sono valutati tramite l'indicizzazione dei documenti a cura dei motori di ricerca sono il calcolo del coefficiente di Jaccard, di Dice, il reperimento di informazioni reciproche quali la Mutual Information ed il Google-based Semantic Relatedness.

2.4.1.1 Coefficiente di Jaccard e Dice

Il coefficiente di Jaccard è una misura che calcola la similarità (o diversità) fra gli insiemi e viene definito come:

$$J(w_1, w_2) = \frac{|\, D\,|w_1,w_2\,|}{|D|w_1|+|D|w_2|-|D|w_1,w_2|} \quad (1)$$

In termini probabilistici, l'equazione (1) restituisce la stima di probabilità massima del rapporto tra la probabilità di individuazione del documento dove le parole w_1 e w_2 co-accadono e la probabilità di individuazione del documento dove w_1 e w_2 si verificano. Se i termini in esame risultassero la medesima parola, il coefficiente di Jaccard risulterebbe uguale ad 1 (similarità semantica assoluta). Al contrario, se due

termini non coincidessero mai in un documento, il coefficiente risulterebbe uguale a 0.

Il coefficiente di Dice è collegato a quello di Jaccard, e viene computato come:

$$C(w_1, w_2) = \frac{2|\ D\ |w_1,w_2\ |}{|D|w_1|+|D|w_2|} \quad (2)$$

Analogamente al coefficiente Jaccard, quello di Dice rispecchia la completa identicità tra i due termini se il valore risulta uguale ad 1, mentre se uguale a 0 indica la totale discordanza tra i termini.

2.4.1.2 Mutual Information

Le informazioni reciproche, Mutual Information (MI), considerano che l'evento dei termini w_1 e w_2 siano variabili "x" ed "y", rispettivamente, che misurino la dipendenza reciproca tra l'aspetto dei due termini [CH-90].

La stima di probabilità massima di MI è data dalla seguente formula:

$$I(w_1, w_2) = log\frac{\frac{|D|w_1,w_2|}{|D|}}{\frac{|D|w_1|}{|D|}*\frac{|D|w_2|}{|D|}} \quad (3)$$

La MI misura parte delle informazioni delle variabili "x" ed "y". Questo quantifica come, la conoscenza di una variabile, riduca l'incertezza dell'altra. Ad esempio, se la "x" e la "y" sono indipendenti, conoscere una delle due non fornirà alcuna informazione utile, ed il valore ottenuto sarà pari a 0. Viceversa, se "x" ed "y" fossero uguali, la conoscenza di una darà il valore dell'altra con certezza, ed il valore ottenuto sarà pari ad 1.

Da notare che le frazioni dei documenti vengono normalizzate dal numero di documenti indicizzati da un motore di ricerca, |D|, restituendo una stima di probabilità massima di trovare almeno un documento nel Web che contenga tale termine.

2.4.1.3 Google-based Semantic Relatedness

La distanza normalizzata di Google (Normalized Google Distance) [CV-07], da sottolineare che il termine "*Google*" viene usato per indicare la sua applicazione

ai motori di ricerca, è stata motivata ad essere sviluppata secondo la "teoria della complessità algoritmica" introdotta da Kolmogorov, la quale si occupa dello studio della complessità descrittiva degli algoritmi e non delle risorse computazionali (memoria occupata e tempi di calcolo) necessarie ad eseguirli, quindi completamente diversa dalla "teoria della complessità computazionale" [WIK-CA].

La Google Normalized Distance viene calcolata tramite la seguente formula:

$$\mathrm{G}(w_1, w_2) = \frac{\max\{A\} - \log|D|w_1, w_2|}{\log|D| - \min\{A\}} \quad (4)$$

Dove "A" viene definito come: $A = \{\log|D|w_1|, \log|D|w_2|\}$.
Quindi, come la similarità semantica tra due termini aumenta, la distanza computata dall'equazione (4) diminuisce. Questa metrica può essere intesa come il calcolo di una misura di diversità, infatti, essa può variare da 0 ad ∞.

Per definire la misura di similarità tra termini, successivamente è stata introdotta una variazione della distanza normalizzata, chiamata *Google-based Semantic Relatedness*, ovvero:

$$G'^{(w_1, w_2)} = e^{-2G(w_1, w_2)} \quad (5)$$

dove $\mathrm{G}(w_1, w_2)$ viene computato secondo la formula (4). Il valore restituito dall'equazione (5) è compreso tra i valori 0 e 1.

2.4.2 Metrica di similarità basata sul testo

Una famiglia della metrica di similarità basata sul testo è rappresentata dalla similarità di coseno tra le caratteristiche dei vettori creati dall'estrazione del termine o dai contesti del vocabolo, quindi un modello di contesto del tipo "*bag-of-words*".

Il presupposto base di tale metrica è che la similarità di contesto implica la similarità del significato, cioè, i termini che compaiono in ambienti lessicali simili presentino una relazione semantica [ACCS-09].

Il modello su cui si basa questa metrica, impiega una finestra del contesto di "misura di parole" stabilita da "*k*" termini per estrarre la caratteristica inerente. Nello specifico, per ciascuna occorrenza del termine di interesse "*w*" nel

documento, saranno considerati i contesti di destra e di sinistra di lunghezza "*k*" (stabiliti in precedenza), per esempio $[v_{K,L} \dots v_{2,L}\ v_{1,L}]\ w\ [v_{1,R}\ v_{2,R} \dots v_{K,R}]$ dove $v_{i,L}$ e $v_{i,R}$ rappresentato rispettivamente la i-esima parola a sinistra (L) e a destra (R) del termine in esame.

Il vettore della caratteristica di ciascun termine "*w*" viene definito come $T_{w,K} = (t_{w,1}, t_{w,2}, \dots, t_{w,N})$ dove $t_{w,i}$ è un numero intero non negativo (che può essere concordato a priori secondo schemi che considerano la frequenza di avvenimento di una data caratteristica) e "*K*" è la dimensione di contesto della finestra da esaminare. Da notare, che la dimensione del vettore della caratteristica è uguale alla dimensione "*N*" del vocabolario, in pratica, si ha una caratteristica per ogni singolo termine presente nel vocabolario "*V*". L'i-esimo valore della caratteristica $t_{w,i}$ riflette l'occorrenza (la frequenza) del termine nel vocabolario v_i all'interno della finestra "*K*" di contesto destro o sinistro del termine "*W*".

Selezionato lo schema di ponderazione della caratteristica viene selezionato il "*bag-of-words*" di base S^K calcolando la similarità tra due termini, w_1 e w_2, come fosse la similarità di coseno delle loro corrispondenti caratteristiche dei vettori $T_{w1,K}$ e $T_{w2,K}$ nel seguente modo [ACCS-09] :

$$S^K(w_1, w_2) = \frac{\sum_{i=1}^{N} t_{w1,i} t_{w2,i}}{\sqrt{\sum_{i=1}^{N} (t_{w1,i})^2} \sqrt{\sum_{i=1}^{N} (t_{w2,i})^2}} \quad (6)$$

La similarità di coseno, assegna un punteggio compreso tra 0 e 1, dove 0 indica i termini completamente dissimili ed 1 quelli identici.

Di seguito saranno elencati i vari schemi di ponderazione delle caratteristiche per la computazione della $t_{w,i}$; tali schemi possono essere classificati in frequenze di base binarie. Si assegna il peso $t_{w,i} = 1$ quando la parola i-esima nel vocabolario esiste almeno in un caso nel contesto destro o sinistro del termine "*w*" e 0 altrimenti.

SCHEMA	VALORE DI $t_{w,i}$ PER $c(v_i) > 0$
Binary (B)	1
Term Frequency (TF)	$\frac{c(v_i)}{c(w)}$
Log of TF (LTF)	$\frac{\log(c(v_i))}{\log(c(w))}$
Add-one LTF (LTF1)	$\frac{\log(c(v_i)+1)}{\log(c(w)+\alpha)}$
TF-inverse document frequency (TFIDF)	$\frac{c(v_i)}{c(w)}\log\frac{\lvert D\rvert}{\lvert D\rvert v_i\rvert}$
Log of TFIDF (LTFIDF)	$\frac{\log(c(v_i))}{\log(c(w))}\log\frac{\lvert D\rvert}{\lvert D\rvert v_i\rvert}$
Add-one LTFIDF (LTF1IDF)	$\frac{\log(c(v_i)+1)}{\log(c(w)+\alpha)}\log\frac{\lvert D\rvert}{\lvert D\rvert v_i\rvert}$

Figura 2.4 - Schemi di ponderazione della caratteristica di contesto

Obiettivo base degli schemi di ponderazione è quello di computare la frequenza (normalizzata) del numero di occorrenze di un termine all'interno del contesto stesso. Svariati metodi di calcolo delle frequenze sono utilizzati nell'ambito di elaborazione del linguaggio naturale ed all'interno delle applicazioni web, in particolare, la frequenza dei termini (TF), la frequenza logaritmica dei termini (LTF), la frequenza inversa del documento di frequenza dei termini (TFIDF) ed il logaritmo del TFIDF.

2.4.3 Metrica di similarità basata sul corpus

Riassumendo, nelle precedenti sezioni si è parlato dei metodi inerenti al conteggio basato su pagine ed alla similarità basata sul testo. Questi due metodi,

spesso applicati a progetti di integrazione ai motori di ricerca, hanno contribuito a migliorarne la qualità di visualizzazione in categorie di contenuti e concetti chiave in modo da poter servire all'utente finale un output per la ricerca dei documenti eccellente. Utilizzando tali metodi, ed ottimizzando le ricerche sfruttando la potenza degli operatori logici (vedi cap. 1.4.1), viene reso noto che i risultati restituiti dai motori utilizzando query con "*AND*" superano significativamente quelle con "*OR*" per il calcolo semantico di similarità basato sul contesto dei documenti.

Una volta scaricati i documenti, vengono esaminati i contesti destri e sinistri di tutti gli avvicinamenti dei due termini considerati, e successivamente vengono costruiti i vettori corrispondenti alle caratteristiche dei singoli termini secondo i parametri sperimentali indicati di seguito:

- Il numero dei documenti web |D| : ovvero quanti documenti sono stati usati;
- La dimensione contestuale della finestra "*K*" : ovvero vengono esaminati i contesti destri e sinistri dei termini secondo la dimensione contestuale della finestra; la dimensione della finestra è applicata entro i limiti della frase;
- Il filtraggio delle parole di terminazione (si/no): se i termini di arresto devono essere considerati o no nei vettori delle caratteristiche;
- Il tipo di schema di ponderazione: i valori delle caratteristiche dei vettori vengono fissati secondo uno schema di ponderazione scelto in precedenza.

Infine la similarità semantica tra i termini viene computata con il calcolo della similarità di coseno tra i vettori delle caratteristiche come indicato nell'equazione (6)

3. STATO DELL'ARTE

3.1 Applicazioni Web basate su risorse ontologiche

Nell'era della costante crescita di disponibilità di informazioni, è fondamentale fornire agli utenti metodi convenienti per comprendere facilmente le informazioni. Per raggiungere questo traguardo, ad oggi sono state sviluppate già diverse applicazioni web e non, tra cui anche motori di ricerca, che eseguono il clustering di documenti o la visualizzazioni delle informazioni inerenti.

Comunque, la maggior parte dei metodi implementati fino ad ora forniscono solamente una via più semplice agli utenti di accedere alle informazioni, ma non aiutano direttamente a catturare il concetto chiave e le relazioni legate ad esso all'interno del documento.

I termini chiave, sono la rappresentazione migliore per descrivere i concetti inerenti ad un insieme di documenti. Infatti, la comprensione dei concetti chiave risulta fondamentale per catturare la struttura concettuale di un gruppo di documenti, evitando il sovraccarico di informazioni all'utente.

Dal momento che una ontologia è definita come una descrizione dei concetti e delle relazioni che sono in grado di rappresentare la conoscenza di un dominio, alcuni progetti sviluppati negli ultimi anni ne fanno sicuramente un uso consistente, proponendo così all'utente di visualizzare durante le proprie ricerche, non solo i concetti chiave, ma anche le relazioni semantiche esistenti tra i documenti analizzati, creando gruppi e quant'altro.

Di seguito saranno analizzate alcune applicazioni web e mostrato il loro utilizzo.

3.1.1 SenseBot

SenseBot è un motore di ricerca semantico che genera una sintesi del testo di pagine Web restituite in base al tema inserito nella query di ricerca [SB-07].

SenseBot viene inserito nella categoria dei meta-motori di ricerca, in quanto restituisce risultati interrogando a sua volta altri motori di ricerca.

La ricerca di documenti/informazioni è molto semplice, infatti, questo meta-motore, offre in fase di ricerca, la possibilità di scegliere, il motore di ricerca di proprio interesse, potendo scegliere tra Google, Yahoo! e Bing, oppure tentare di effettuare una ricerca all'interno di SenseBot stesso. Inoltre offre la possibilità di scegliere la lingua per la query tra l'inglese, il tedesco, il francese e lo spagnolo.

Figura 3.1 - Interfaccia iniziale per la ricerca del motore SenseBot

Diversamente da altri meta-motori, offre la possibilità di eseguire un processo di sommarizzazione sui risultati ottenuti producendo un sommario.

Inoltre, utilizza anche algoritmi di text-mining per eseguire le elaborazioni sui significati semantici dei concetti chiave estratti.

Questa funzione può risultare utile nel caso in cui si cerchino argomenti nuovi, o temi poco conosciuti, perché essa permette di trovare immediatamente

informazioni maggiori che permetteranno all'utente di avere una panoramica generale dei contenuti e delle relazioni tra gli argomenti estratti.

SenseBot, in risposta alla fase di ricerca, genera due tipi di sommari:

- *Sommario generale*: inserito in un tag-cloud, al suo interno vengono inseriti tutti i termini più rilevanti estratti, ed in base alla loro frequenza ed importanza rispetto alla query di ricerca vengono più o meno evidenziati;
- *Sommario specifico*: su ciascun termine ricavato, viene eseguita una sommarizzazione e generato un sommario, che, a differenza di quello generale, contiene solo termini separati, rappresentando così una frase di senso compiuto che descrive brevemente il contenuto del documento.

Inoltre, è possibile selezionare quali risultati includere nell'insieme dei link che contribuiranno alla creazione del sommario generale e di salvare ogni suo singolo sommario in un file di testo.

Figura 3.2 - Interfaccia di risultati di ricerca di SenseBot

3.1.2 Carrot2

Carrot2 è un progetto "*Open Source*" che può essere inserito nella categoria dei motori di ricerca semantica con cluster [CAR-02]. Il progetto mette a disposizione degli utenti, inoltre, API sviluppate tramite tecnologia Java e C#/.NET da poter inserire in altri progetti. Questo software, è in grado di organizzare automaticamente sommari rappresentanti piccole collezioni di documenti, restituiti dai risultati della ricerca, suddivisi per categoria tematica.

Come il motore di ricerca presentato nel paragrafo precedente, per estrarre i propri risultati di ricerca Carrot2 si affida ad altri motori, quali Google, Yahoo! e Bing, e sfrutta la potenza di alcune librerie e tool riguardanti le meta-ricerche come Lucene, Solr, Google Desktop ed altro ancora.

L'interfaccia web del motore Carrot, si presenta relativamente semplice. E' possibile effettuare le proprie ricerche scegliendo tramite le apposite schede il motore di ricerca preferito, tra Google, Bing e Yahoo (come sopra citato), oppure scegliere ricerche nella piattaforma "Wiki", cercare immagini, notizie e blog.

Offre anche una scheda di opzioni avanzate dove l'utente ha la possibilità di scegliere il numero massimo di documenti scaricabili, di scegliere il metodo "cluster" che si vuole utilizzare tra *Lingo*, *STC*, *K-means*, oppure di raggruppare i documenti nei metodi "by URL" o "by Source". Inoltre, è possibile impostare la ricerca per paese (tutti, Austria, Francia, Germania, Gran Bretagna, Italia, Liechtenstein, Spagna e Svizzera) e per lingua (tutte, inglese, francese, tedesco, italiano e spagnolo).

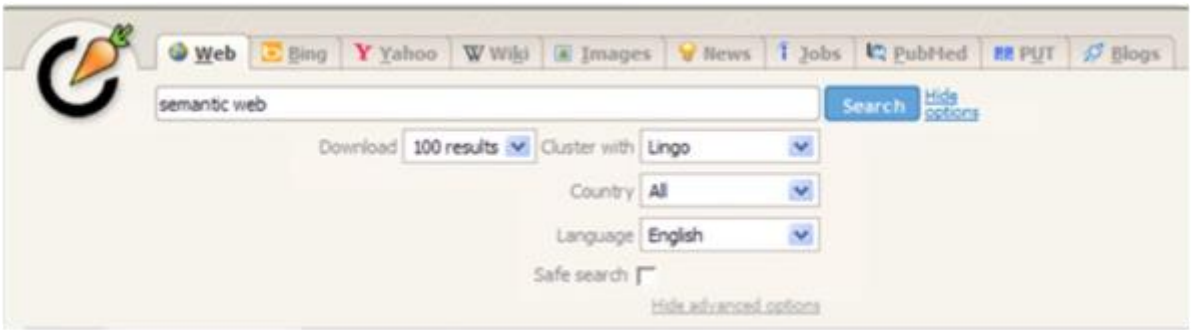

Figura 3.3 - Interfaccia web per la ricerca di Carrot2

Impostati i parametri di ricerca, ed ottenuti i risultati, Carrot2 genera una lista di concetti chiave, i quali specificano l'area di interesse dei testi, e collegati alle suddette liste, genera liste specifiche contenenti i temi legati ai concetti chiave generati nel sommario.

I concetti chiave possono essere visualizzati in tre modi; uno tramite una lista di termini chiave ordinata in modo decrescente, un altro da un grafico a torta che evidenzia i temi principali di ricerca rispetto alla query, ed infine un albero a schiuma contenente anch'esso i temi principali dei documenti.

Scegliendo uno qualsiasi dei metodi di visualizzazione dei concetti chiave, a lato di tali schemi, è possibile visualizzare i documenti inerenti al concetto preso in considerazione dall'utente cliccando sul concetto stesso.

Grazie ai vari tipi di visualizzazione, è possibile per l'utente trovare gli argomenti legati ad un determinato concetto o tema di interesse in via più semplice, inoltre, affrontando nuove tematiche, si è in grado di avere maggiori informazioni sugli argomenti stessi inerenti alla query di ricerca inizale.

Di seguito verranno mostrati i vari metodi di ricerca del motore Carrot2:

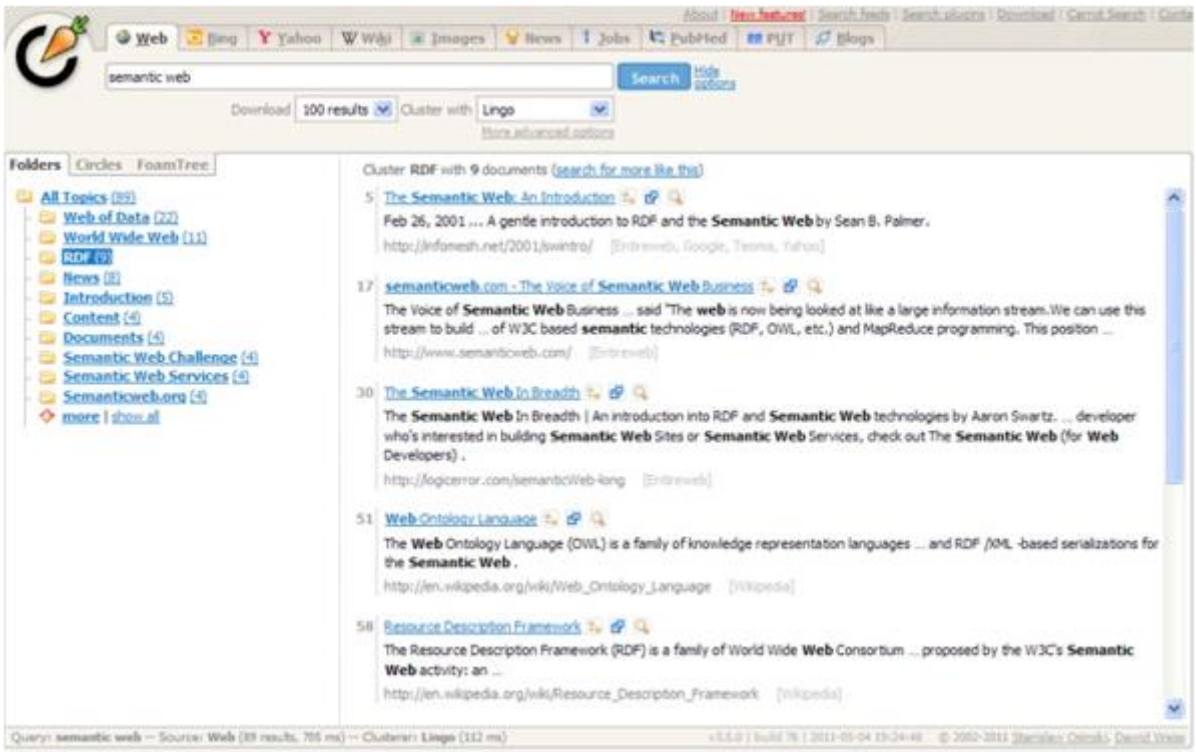

Figura 3.4 - Risultati di ricerca visualizzati con "Folder"

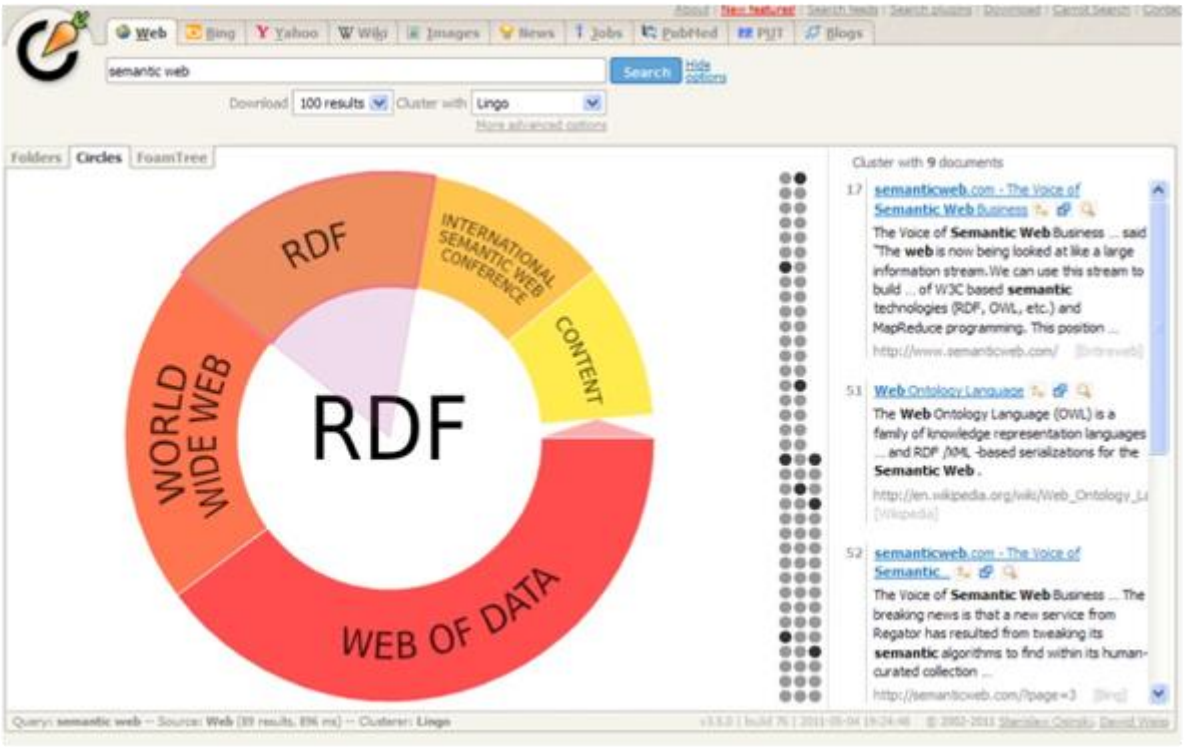

Figura 3.5 - Risultati di ricerca visualizzati con "Circles"

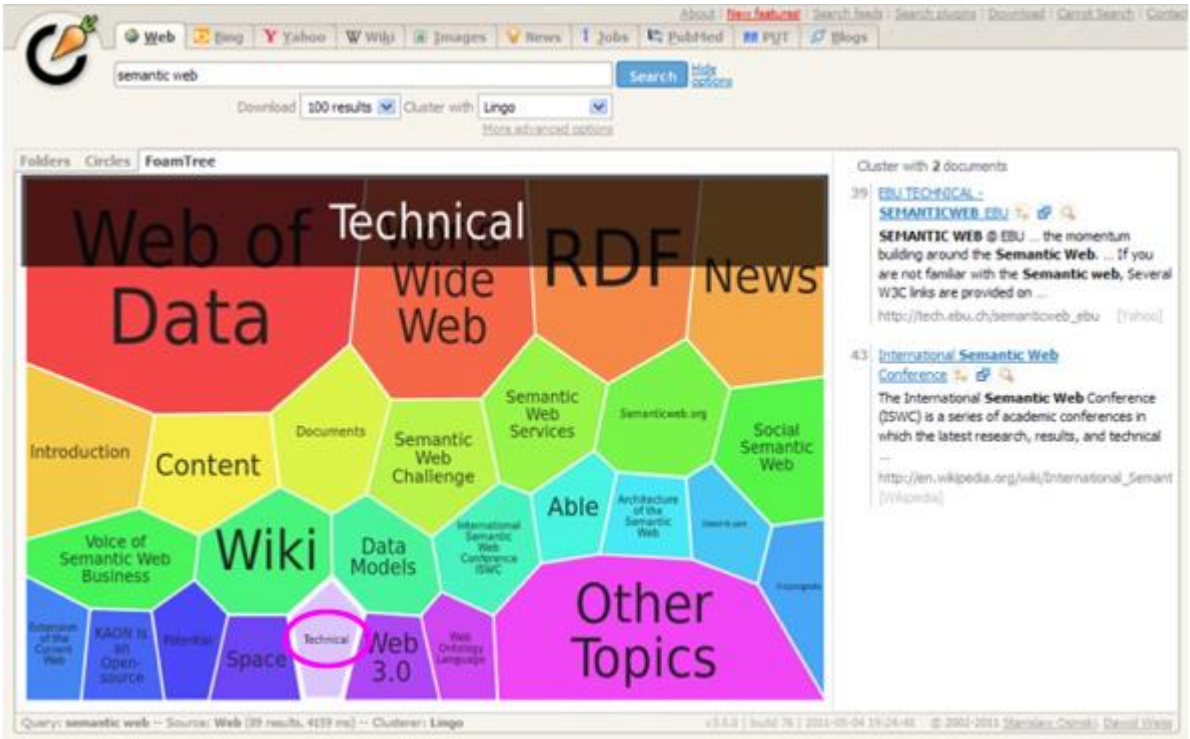

Figura 3.6 - Risultati di ricerca visualizzati con "FoamTree"

4. IL PROGETTO

4.1 SemaWeb: introduzione

SemaWeb è un'applicazione Web in grado di compiere ricerche di documenti ricreando in parte un ambiente familiare di un qualsiasi motore di ricerca con l'aggiunta di alcune funzionalità di tipo semantico-lessicali e di estrazione dei termini chiave in base ai risultati di ricerca.

Nei capitoli precedenti, sono stati descritti le logiche con cui i motori di ricerca reperiscono informazioni, del ruolo fondamentale che hanno oggi in qualsiasi attività di ricerca sul web, ma anche dei loro limiti funzionali e delle soluzioni studiate ed attuate sino ad oggi per migliorarne il loro funzionamento.

Il progetto SemaWeb pone come obiettivo principale quello di limitare l'ambiguità tra i documenti restituiti da una ricerca, di facilitare la comprensione dei documenti in base ai concetti chiave estratti e di rappresentare tramite una struttura ontologica ad albero, le relazioni presenti tra i vari concetti chiave, quindi tra i documenti esaminati.

In particolare, effettuata una ricerca, è possibile scegliere ogni singolo concetto dalla lista dei concetti chiave restituita, e con un semplice click, visualizzare a lato i documenti inerenti al concetto stesso, come è possibile, visionare il percorso di un singolo concetto ripercorrendo le iperonimie (cioè quando un termine possiede un significato che include un altro termine più specifico), o visualizzare l'albero di dipendenza ontologica tra i vari concetti chiave.

L'applicazione, realizzata interamente in html5 e css3 per la grafica di front-end presenta diverse prospettive di sviluppo che saranno descritte in seguito. SemaWeb è stata sviluppata in ambiente Java, utilizzando la tecnologia Java/JSP e Servlet, con il supporto di tecnologia di scripting quale Javascript. Per eseguire le ricerche sul Web sono state utilizzate le API di Google, mentre le funzioni semantico-lessicali, sono state realizzate sfruttando il database semantico-lessicale per la lingua inglese WordNet. In seguito verranno descritte tutte le tecnologie utilizzate ed il processo di elaborazione dei dati.

4.2 Tecnologie utilizzate

4.2.1 Java

Il progetto è realizzato attraverso l'utilizzo del linguaggio di programmazione Java ed in particolare della tecnologia Java/JSP.

Concepito e sviluppato nel 1995 da James Gosling e il suo team di Sun Microsystem, java è un linguaggio interpretato ed orientato agli oggetti [WIK-JA].

La caratteristica principale che ha reso possibile la diffusione di questo linguaggio è stata la portabilità, a differenza di altri linguaggi molto diffusi all'epoca, come il C e il C++, comunque, per facilitare il passaggio a Java per i programmatori "*old-fashioned*", la sintassi di base (strutture di controllo, operatori e quant'altro) è stata mantenuta pressoché identica a quella del C++. Tuttavia, non sono state introdotte caratteristiche ritenute fonti di una complessità non necessaria a livello di linguaggio e che favoriscono l'introduzione di determinati *bug* durante la programmazione, come l'aritmetica dei puntatori, l'ereditarietà multipla delle classi e l'istruzione *goto*. Per le caratteristiche *"object-oriented*" del linguaggio, ci si è ispirati al C++ ed in particolare all'Objective C.

A differenza di Java, la maggioranza dei linguaggi di programmazione, sono stati progettati per essere compilati da una macchina specifica. Un esempio, lo si può avere osservando la compilazione di un programma scritto in C++; questo lo si può compilare per qualsiasi tipo di CPU, ma richiede un compilatore C++ specifico per la CPU di destinazione.

Questa problematica è divenuta più evidente con l'avvento del Web. Infatti il Web è costituito da diversi tipi di computer, CPU e sistemi operativi, dove però ciascun utente connesso deve essere in grado di eseguire qualsiasi applicazione.

Java , è stato sviluppato per abbattere questa barriera. L'output di un compilatore Java non è codice eseguibile, ma *byte code,* cioè set di istruzioni altamente ottimizzato e progettato per essere interpretato ed eseguito dalla JVM (Java Virtual Machine). A questo punto per ogni tipo di piattaforma deve essere implementata solo la JVM [HS-07].

Quindi, i programmi scritti in Java saranno destinati all'esecuzione sulla piattaforma Java, ovvero, saranno lanciati su una JVM e, a tempo di esecuzione, avranno accesso alle API della libreria standard.

In linea di principio si dovrebbe essere in grado di scrivere programmi una sola volta e di poterli eseguire ovunque, e proprio da questa supposizione nasce il famoso slogan di Sun: "*write once, run everywhere*".

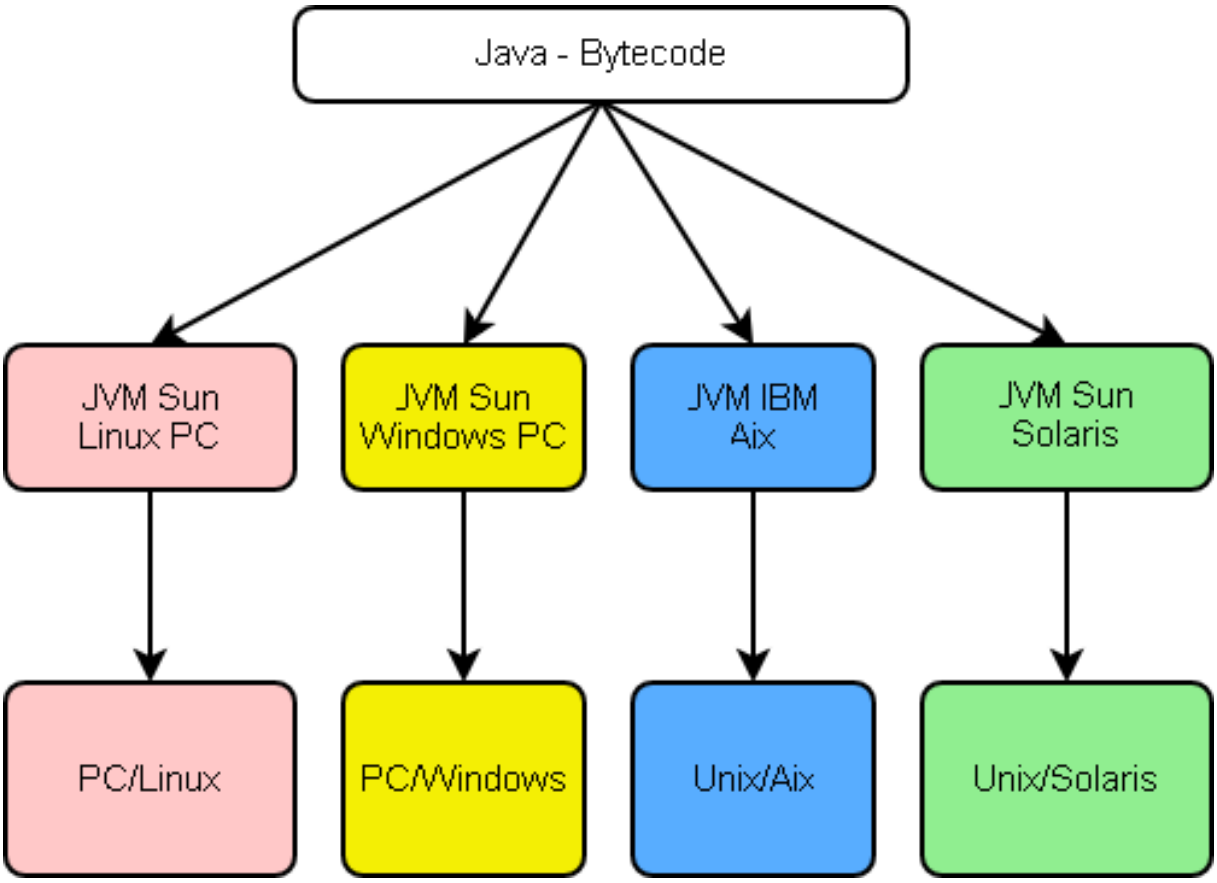

Figura 4.1 - Schema di funzionamento del byte code su diverse piattaforme

La scelta di sviluppare l'applicazione in Java, oltre ai motivi citati in precedenza, è dovuta dal fatto che è un linguaggio ad oggetti, quindi è possibile godere dei vantaggi offerti dal paradigma ad oggetti, permettendo a chiunque e in qualsiasi momento, l'ampliamento del progetto sviluppando nuove funzionalità, aggiungere codice senza modificare l'implementazione esistente, inoltre,

l'interfaccia creata per estrarre i legami semantici dal database WordNet è costituita da classi Java.

Per lo sviluppo delle classi Java e la loro gestione è stato utilizzato l'ambiente di sviluppo IDE NetBeans 6.9.1 e per la compilazione è stata utilizzata la versione 1.6 dell'SDK Java; per l'esecuzione delle classi si è fatto uso del server web Apache Tomcat 6.0.

4.2.2 Servlet/JSP

Per lo sviluppo dell'applicazione web è stata utilizzata la tecnologia server-side Servlet/JSP.

Le servlet sono oggetti che estendono le funzionalità di un web server permettendogli di generare pagine web dinamicamente.

Tipicamente una servlet è rappresentata da una classe che implementa l'interfaccia *Servlet* o *HttpServlet* che a sua volta implementa funzionalità specifiche del protocollo HTTP.

Esse vengono istanziate ed eseguite nel momento in cui il web server riceve una richiesta http da un qualsiasi browser; considerando il nostro caso, l'esecuzione di tali oggetti consistono nell'elaborazione di dati ricevuti nella richiesta, e nell'inviare i risultati di elaborazione in fase di scrittura di una pagina HTML dinamica.

I documenti dinamici creati sono costituiti da due parti:

- *Presentation*: contiene il codice HTML statico utilizzato per descrivere la struttura della pagina;
- *Business*: contiene il codice che genera il contenuto dinamico della pagina.

In fase di costruzione di Servlet, talvolta risulta molto oneroso in termini di tempistiche e di visibilità generale del codice, dover scrivere la parte di P*resentation*. Per abbattere questo ostacolo, è stata sviluppata un'ulteriore tecnologia: le JSP (*JavaServer Pages*).

Tale tecnologia consente di scrivere codice HTML puro integrato a codice Java. In questo modo è possibile separare completamente la parte di *Presentation* da quella di *Business*, permettendo una migliore gestibilità di codice sorgente.

Le pagine JSP, possono essere viste come un'astrazione a livello superiore rispetto alle Servlet, in quanto, quando essa viene richiesta per la prima volta, viene tradotta automaticamente da un compilatore JSP in una normale Servlet che sarà poi eseguita.

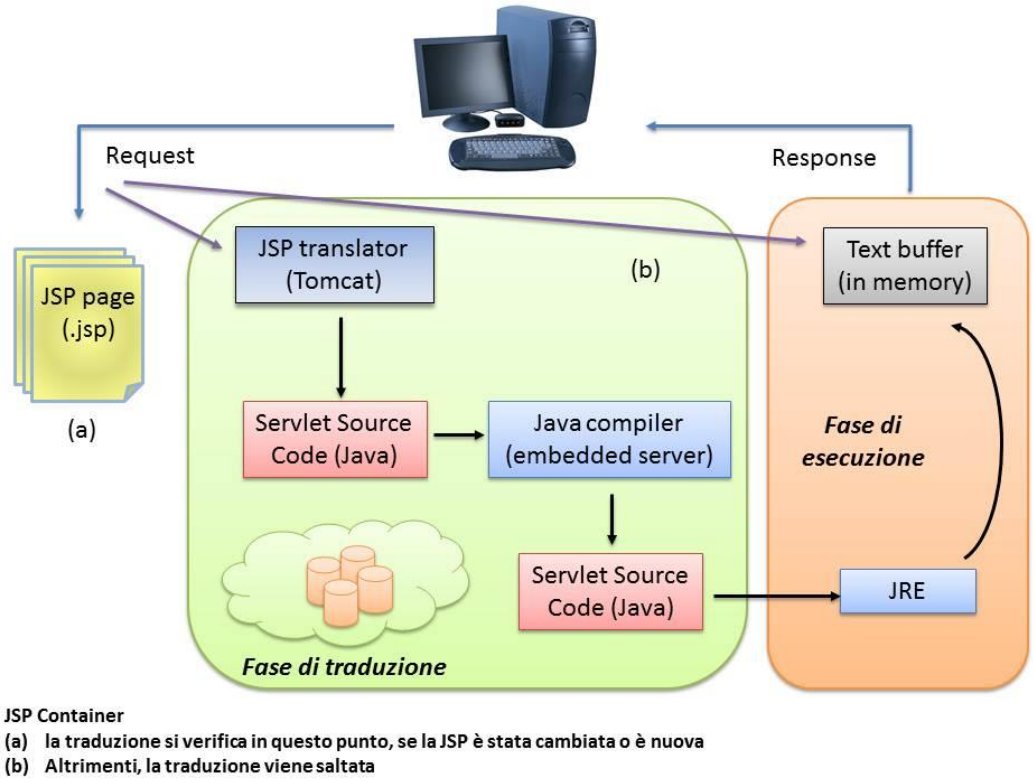

Figura 4.2 - Schema di vita di un file JSP

4.2.3 JavaScript

Javascript è un linguaggio di scripting orientato agli oggetti, comunemente utilizzato nei siti web. Fu sviluppato da Brendan Eich nel 1996 con il nome di Mocha, e successivamente di LiveScript. In seguito fu rinominato "JavaScript" in quanto formalizzato con una sintassi più vicina a quella del linguaggio Java di Sun,

e standardizzato tra il 1997 e 1999 con il nome di ECMAScript. Attualmente la versione di JavaScript corrisponde alla 1.5, ed è anche uno standard ISO [WIK-JS].

La principale caratteristica di JavaScript, è quella di essere un linguaggio interpretato (generalmente, l'interprete è incluso nel browser di utilizzo), inoltre la sintassi si presenta relativamente simile a quella del C,C++ e Java. Questo linguaggio inoltre, definisce le funzionalità tipiche dei linguaggi di programmazione di alto livello dotati quindi dei principali costruttori di controllo e consente anche l'utilizzo del paradigma object oriented.

Inoltre sono comprese tra le principali caratteristiche di JavaScript anche:

- *Loose typing*: ovvero, la tipizzazione debole dei dati non consente di dichiarare i tipi di variabili, i quali vengono convertiti in automatico dall'interprete durante l'esecuzione dello script;
- *Linguaggio debolmente orientato agli oggetti*: ovvero, gli oggetti possono essere considerati come contenitori generici;
- *Esecuzione lato client*: ovvero, lo script viene eseguito dopo che il server ha restituito la pagina richiesta al browser, quindi senza sollecitare il server, con la possibilità di eseguire uno script particolarmente "pesante" senza sovraccaricare il server.

Un frequente uso del linguaggio JavaScript nel Web, è la scrittura di funzioni integrate all'interno di codice HTML, che interagendo con il browser permettono di compiere azioni non possibili con il solo HTML statico, come ad esempio, il controllo di valori all'interno di campi di tipo input, oppure visualizzare o nascondere determinati elementi.

Il collegamento con il browser avviene tramite interfacce di tipo DOM (Document Object Model). Il DOM è una forma di rappresentazione dei documenti strutturati come modello orientato agli oggetti.

Uno dei limiti di JavaScript, risiede proprio nella sua principale caratteristica, cioè di essere una tecnologia client-side, in quanto ogni browser possiede uno specifico motore JavaScript, ognuno con specifiche ed eccezioni proprie, rendendo così impossibile la certezza del funzionamento cross-browser degli script realizzati.

Altro limite si presenta nel caso in cui sia necessario, a tempo di esecuzione, l'utilizzo di dati presenti in un database ad esempio, dove è necessario eseguire una richiesta esplicita al server per ottenere i dati e quindi dover ricaricare la pagina ad ogni richiesta, limitando l'interattività ed aumentando i tempi di risposta. Limite superato oggi grazie all'avvento di AJAX (Asynchronous JavaScript and XML).

4.3 Strumenti utilizzati

4.3.1 Google Ajax Search API

Per eseguire le ricerche di documenti sul web, sono stati utilizzati dei servizi offerti dal motore di ricerca Google, in specifico, le Google Ajax Search API. Tali API rappresentano una libreria JavaScript che consente agli sviluppatori di poter integrare nelle proprie pagine web l'ambiente di ricerca offerto da Google. E' possibile integrare qualsiasi tipo di ricerca, da quella web a video, news, blog, ecc… [GASAPI-09].

Nel progetto verrà utilizzata solamente la funzionalità che permette di ottenere risultati di ricerca data una query; tale funzionalità è accessibile eseguendo una richiesta http alla seguente Servlet:

```
http://ajax.googleapis.com/ajax/services/search/web
```

Alla richiesta è necessario accorpare una serie di parametri necessari per personalizzare la ricerca:

- *q*: parametro che contiene la query di ricerca;
- *v*: parametro che contiene la versione del servizio da richiamare, al momento presente solo la versione 1.0;
- *rsz*: parametro opzionale che rappresenta il numero di risultati che si vogliono ottenere. Attualmente sono ammissibili solo due valori: *large* che restituisce 8 risultati e *small* che ne restituisce 4. Se non specificato di default viene impostato *small*;

- *hl*: parametro opzionale che rappresenta la lingua con cui deve essere eseguita la ricerca e se non specificato, di default viene impostata la lingua inglese (*en*);
- *start*: parametro opzionale che rappresenta l'indice di partenza del primo risultato visualizzato, ad esempio, se una ricerca restituisce 50 risultati, ponendo start a 11, saranno scartati i primi 10 risultati di ricerca e visualizzati gli altri 40.

Analizzando i parametri che impostano i criteri di ricerca, si può notare che viene imposto il limite di poter ottenere al più 8 risultati per ricerca. Tale limite è superabile eseguendo all'interno di un ciclo la richiesta http cambiando di volta in volta il valore del parametro di partenza *start*.

Eseguita una ricerca, viene fornita una risposta in formato JSON (JavaScript Object Notation). Tale formato è adatto per lo scambio di dati in applicazioni client-server [WIK-JSON]. E' basato sul linguaggio JavaScript, ma ne è indipendente, e talvolta viene utilizzato in alternativa a XML/XSLT, data la sua semplicità di eseguirne il parsing.

Il formato JSON è costruito su due strutture [JSON-99]:

- una collezione di coppie nome/valore, intese come *object*, record, strutture, dizionari, tabelle hash;
- una lista ordinata di valori, realizzata ad esempio tramite *array*, vettori, liste o sequenze.

Questo formato inoltre supporta i seguenti tipi di dato:

- valore *null*, valori booleani (*true* e *false*);
- interi, reali, virgola mobile;
- stringhe racchiuse da doppi apici (");
- array (separati da virgole e racchiusi in parentesi quadre [ ]), array associativi (sequenze di coppie chiave-valore separate da virgole racchiuse in parentesi graffe { });

Il formato standard di ogni risposta in formato JSON è il seguente:

```
{
   "responseData" :{
      "results" : [],
      "cursor" : {}
   },
   "responseDetails" : null | string-on-error,
   "responseStatus" : 200 | error-code
}
```

Analizzando i risultati di risposta ottenuti dal file JSON di Google, ed in particolare il contenuto "responseData" si ha la seguente struttura:

```
"results" : [
   {
    "GsearchResultClass" : "GwebSearch",
    "unescapedUrl" : "",
    "url" : "",
    "visibleUrl" : "",
    "cacheUrl" : "",
    "title" : "",
    "titleNoFormatting" : "",
    "content" : ""
   },
   {}]
```

Dove:

- *title* e *titleNoFormatting*: rappresentano il titolo del risultato con e senza formattazione HTML;
- *content*: rappresenta l'anteprima del contenuto del documento;
- *url*, *visibleUrl*, *cacheUrl*, *unescapedUrl*: rappresentano i link del risultato con o senza formattazione;
- *cache Url*: se presente rappresenta il link del risultato nella cache di Google.

4.3.2 JSON-lib

JSON-lib è una libreria Java per la trasformazione di beans, mappe, collezioni, array e formato XML in formato JSON e viceversa.

Offre quindi la possibilità di costruire, manipolare e lavorare con vari tipi di elementi rendendolo ideale per lo scambio di dati e non solo [JLIB-06].

Quì verranno illustrate brevemente le funzioni principali della libreria:

4.3.2.1 Come utilizzare JSON-lib

Di seguito verranno elencate le varie classi presenti nella libreria:

- *JSONObject*: è una collezione non ordinata di coppie nome/valore, che può essere di diverso tipo: boolean, JsonArray, JsonObject, numero e stringa, o null. È possibile utilizzare al suo interno i comandi *get()*, *opt()* e *put()*;
- *JSONArray*: è una sequenza ordinata di valori i cui tipi e comandi sono analoghi a quelli di JSONObject;
- *JSONStringer*: è uno strumento per la produzione rapida di testo JSON;
- *JSONWriter*: è uno strumento per la scrittura rapida del testo JSON;
- *JSONTokener*: può essere utilizzato dai costruttori JsonObject e JsonArray per analizzare le stringhe JSON;
- *JSONString*: è una interfaccia che permette alle classi di attuare la serializzazione di JSON.

JSON inoltre può anche essere utilizzato come formato di interscambio dati tra diversi formati.

4.3.2.2 Principali operazioni con JSON

La classe JSONSerializer è in grado di trasformare qualsiasi oggetto Java in notazione JSON e viceversa, sfruttando i costruttori di *JsonObject* e *JsonArray*.

Di seguito verranno elencate le principali operazioni che si possono effettuare:

- *operazioni con Array e Collection*: il modo più semplice per creare un JsonArray da un array Java o da una collection è utilizzare i metodi *JsonArray.* e *JSONArray.fromObject()* che verificherà la natura del parametro ed effettuerà la chiamata corretta al costruttore;
- *operazioni con JavaBean e Maps*: il modo più semplice per creare un JsonObject da una Maps o da un Bean è utilizzare i metodi *JsonObject.* e *JSONObject.fromObject()* che verificherà la natura del parametro;
- *operazioni con XML*: tramite *XMLSerializer* è possibile convertire formati XML in JSON e viceversa, inoltre se si vuole convertire un JSON in XML, oltre a richiamare la semplice funzione *XMLSerializer.write()*, è possibile configurare svariate opzioni per un migliore controllo dell'output XML come modificare ad esempio i nomi predefiniti per l'elemento radice, per un oggetto, per un array o per l'elemento.

4.3.3 JQuery

Talvolta, la scrittura di codice JavaScript può risultare onerosa e per questo motivo grazie all'introduzione massiccia di tale linguaggio all'interno delle attuali applicazioni Web sono stati sviluppati diversi framework JavaScript grazie ai quali è possibile aggiungere dinamicità, effetti, interazioni asincrone e tanto altro con poche righe di codice.

Tali framework sono stati progettati inoltre in modo tale da garantire il funzionamento cross-browser degli script.

Per integrare alcune funzioni in questo progetto è stato scelto di utilizzare il framework JQuery.

Sviluppato nel 2005 da John Resig, JQuery è una libreria di funzioni per pagine web, codificata in JavaScript, che si propone come obiettivo quello di astrarre ad un livello più alto la programmazione lato client del comportamento di ogni singola pagina HTML [WIK-JQ].

Il "*core*" di JQuery fornisce i costruttori per l'utilizzo della libreria stessa, i metodi e le proprietà per accedere agli elementi contenuti in un oggetto JQuery, i

metodi per creare e utilizzare liste e code ed i metodi per estendere il framework tramite plug-in.

In particolare la gestione avviene attraverso l'utilizzo di:

- *selettori*: utilizzati per ottenere gli elementi della pagina utilizzando la stessa sintassi dei selettori CSS (ad esempio selezionare un elemento in base al suo "id" o "classe" di appartenenza);
- *attributi*: gestiti da un insieme di metodi che verificano gli attributi degli elementi;
- *DOM traversing*: insieme di metodi che permettono di attraversare e scorrere il DOM di un documento;
- *manipolatori DOM*: insieme di metodi in grado di gestire gli elementi del DOM (ad esempio aggiungere o eleminare elementi alla pagina);
- *CSS*: insieme di metodi che permettono il controllo dello stile degli elementi;
- *eventi*: insieme di metodi per la gestione degli eventi interattivi;
- *AJAX*: insieme di metodi per la gestione delle richieste asincrone e del caricamento dinamico dei contenuti.

4.3.4 EJML

Efficient Java Matrix Library (EJML) è una libreria di algebra lineare per Java in grado di effettuare operazioni su matrici semplici e dense ed altro ancora in tempi ottimizzati[EJML-11].

Si è scelto di utilizzare tale libreria all'interno del progetto per integrare ed ottimizzare alcuni calcoli applicati ai metodi di SVD (Singular Value Decomposition) necessari per determinare l'estrazione dei termini chiave (argomento che sarà trattato successivamente).

EJML offre le seguenti funzionalità:

- *operazioni di base*: addizione, sottrazione, ecc..;
- *Matrix Manipulation*: estrazione, inserimento, unione, ecc…;
- *Linear Solvers*: algebra lineare, minimi quadrati, incrementali, ecc…;

- *Decomposition*: tecniche di decomposizione come LU, QR, SVD, ecc…;
- *Matrix Features*: calcolo del rango, matrici simmetriche, ecc…;
- *Different Internal Formats*;
- *Unit Testing*.

In EJML si possono trovare varie strutture di matrici, ma in generale, la maggior parte degli utenti ne utilizza solamente due.

L'interfaccia *DenseMatrix64F* che viene utilizzata da operatori che richiedono un controllo ristretto sulle performance e che sono preoccupati di scrivere codice ottimizzato, mentre l'interfaccia *SimpleMatrix* (che è un wrapper di DenseMatrix64F) molto più semplice da utilizzare, offre funzionalità più limitate ma semplicizza ulteriormente la grande mole di lavoro con le matrici. Inoltre utilizzandole entrambe non si ha l'obbligo di dover sceglierne una in particolare, in quanto è facile effettuare il passaggio tra le due classi.

4.3.5 Jenkov-PrizeTags

La libreria PrizeTags realizzata da Jakob Jenkov in Java, giunta alla versione stabile 3.4.0 nel 2007, è una collezione di "tag" e funzioni per creare e gestire sia dinamicamente tramite AJAX, o staticamente pagine HTML o JSP [JEN-07].

Tale libreria permette tramite appositi costruttori la possibilità di creare,manipolare e gestire alberi in Java, ed inoltre contiene una collezione di tags che permettono di effettuare svariate operazioni all'interno di pagine HTML e JSP come citato in precedenza. I principali tags sono:

- *AJAX Tag*: rendono semplice l'aggiunta di funzionalità AJAX all'applicazione creata, come reindirizzare collegamenti, invio di moduli di elementi HTML anziché dover aggiornare la pagina, e tanto altro;
- *Tree Tag*: con tali tag è possibile visualizzare dinamicamente i contenuti di un dato albero. Dotati di listener in grado di effettuare espansioni e selezioni sia lato client che server, permettono una personalizzazione totale in fase di visualizzazione dei contenuti;

- *Tabbed Pane Tag*: tag che rendono facili la creazione di schede all'interno di pagine, tali riquadri possono includere ad esempio altre pagine come contenuti, oppure memorizzare la scheda selezionata nella sessione o gestire riquardri a schede multiple o annidati nella stessa pagina;
- *Calendar Tag*: dato un elenco di eventi di oggetti di un calendario, una data di inizio ed una di fine, questi tag sono in grado di visualizzare gli eventi corretti nel periodo corrente. Questo tipo di tag può risultare utile in applicazioni che necessitano di visualizzare gli elenchi degli eventi in un calendario;
- *Alternate Tag*: tag utili che alternano il colore dello sfondo in maniera diversa delle righe orizzontali nelle tavole ad esempio. Questo tag, può essere integrato all'interno dei TreeTag e CalendarTag;
- *Icon Tag*: tag che rendono semplice la visualizzazione di immagini che cambiano con lo spostamento del mouse o con un click su di essa in applicazioni web, utile quando si realizzano pulsanti icona (link immagine);
- *Template Tag*: tag che rendono semplice la creazione di un modello di pagina dove altre pagine possono essere incluse dinamicamente, permettono così la semplificazione della creazione di tonnellate di pagine con lo stesso layout.

La scelta di tale libreria per il progetto è stata effettuata per poter gestire la creazione dell'albero delle ontologie all'interno della Servlet, ed all'interno delle JSP per la gestione e visualizzazione dell'albero ontologico e dei suoi contenuti.

4.3.6 WordNet

Il database lessicale WordNet (versione 2.1) è stato utilizzato per implementare le funzioni semantico lessicali del progetto.

WordNet è un database semantico-lessicale per la lingua inglese elaborato dal linguista George A. Miller presso l'Università di Princeton il cui obiettivo è quello di proporre una organizzazione come la descrizione e la definizione di

concetti espressi dai vocaboli, ed in particolare nomi, verbi, aggettivi e avverbi [WIK-WN].

Attualmente nella versione utilizzata sono presenti oltre 200.000 vocaboli. Del database WordNet, esistono anche versioni in altre lingue, tra cui quella italiana, ma attualmente, sono poveri di contenuti e meno supportate della versione principale inglese.

All'interno del database, l'organizzazione del lessico si avvale di raggruppamenti di termini (*set*) di sinonimi (synonym) chiamati *synset*. Ciascun synset descrive un concetto ben preciso ed i vocaboli in essi contenuti rappresentano quel concetto.

I synset sono collegati tra di loro attraverso una rete di puntatori che ne descrivono le relazioni semantiche e lessicali che li legano, ottenendo così la rete di collegamenti e di relazioni tra i vari concetti e vocaboli [WNPU-09].

Sapendo che il linguaggio naturale risulta soggetto ad ambiguità, in particolare un termine può assumere vari significati, nell'esempio seguente si può notare come, effettuando una ricerca di un dato termine (ad esempio "web") all'interno del database WordNet, questo restituisce la rappresentazione dei suoi diversi significati, potendo così visualizzare in quali differenti synset tale termine compare.

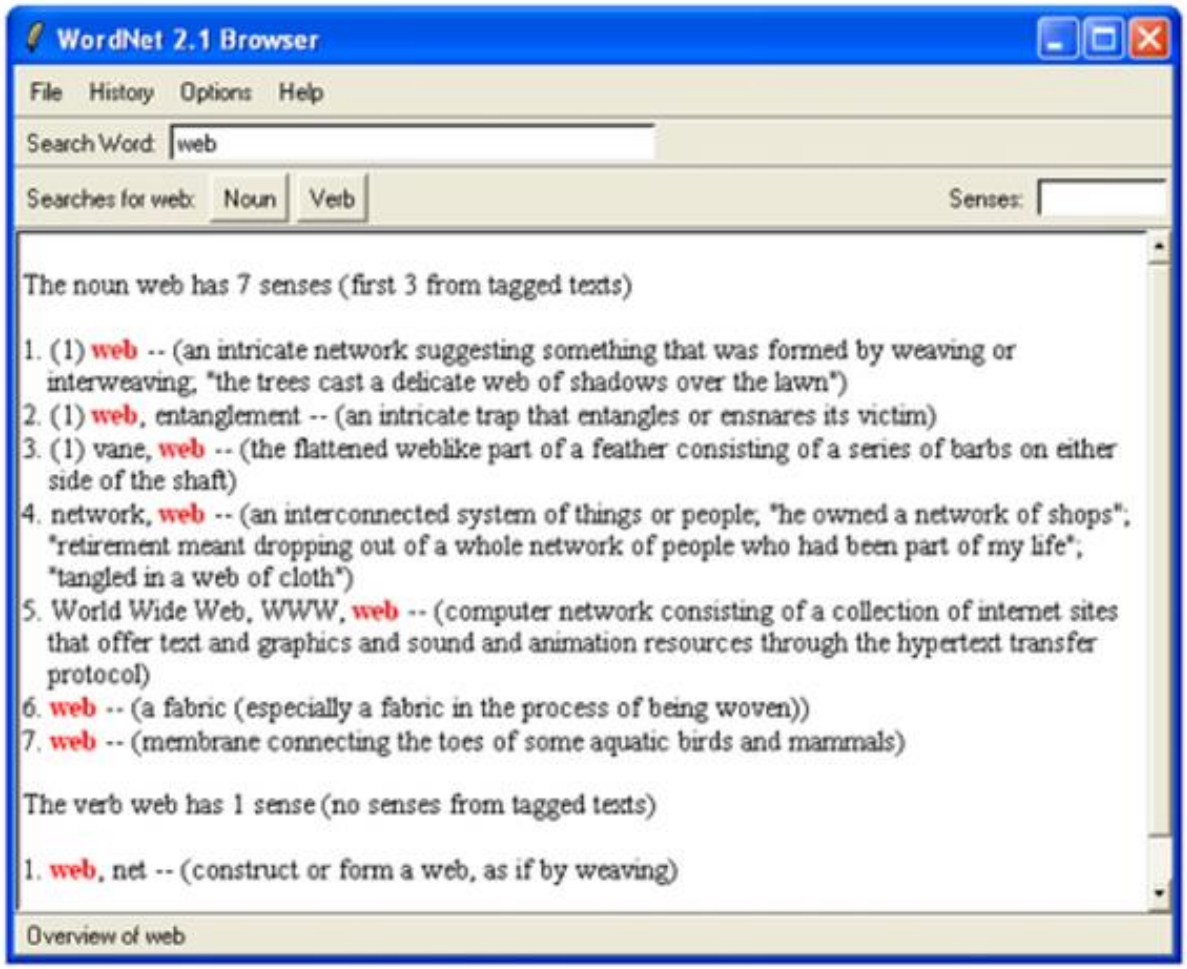

Figura 4.3 - Esempio di ricerca di un termine in WordNet

4.3.6.1 Le relazioni semantico-lessicali

All'interno di WordNet esistono diversi tipi di relazioni semantiche e lessicali tra i vari synset, dipendenti anche dal genere di analisi del termine che si vuole effettuare.

Per i sostantivi valgono i seguenti tipi di relazione:

- *Iperonimia* (hyperonyms): quando un termine possiede un significato che include altre parole dal significato più ristretto, ad esempio, il termine "albero" è iperonimo di "abete", ovvero la relazione è vista come, Y iperonimo di X se ogni Y (è specie) di X;

- *Iponimia* (hyponyms): è l'inverso di una iperonimia, ovvero si verifica quando il significato di un termine è incluso all'interno di un termine dal significato più esteso, ovvero la relazione è vista come, Y è iponimo di X se ogni Y è (una specie di) X;
- *Olonimia* (holonym): è una relazione semantica tra due vocaboli dove il primo rappresenta l'intero ed il secondo ne rappresenta una parte o un membro, ad esempio, "albero" è olonimia di "tronco", ovvero la relazione è vista come, Y è olonimo di X se X è parte di Y;
- *Meronimia* (meronym): è l'inverso di una olonimia, ad esempio, "tronco" è meronimia di "albero", ovvero la relazione è vista come, Y è meronimo di X se Y è parte di X.

Per i verbi valgono i seguenti tipi di relazione:

- *Iperonimia*: stessa relazione vista per i sostantivi, ad esempio "viaggio" è iperonimo di "movimento", ovvero la relazione è vista come, il verbo Y è iperonimo del verbo X se l'attività X è (una specie) di Y;
- *Troponimia* (troponyms): un verbo è troponimo di un altro verbo se durante la prima azione, viene eseguita anche la seconda, ad esempio, "mormorare" è troponimo di "parlare", ovvero la relazione è vista come, il verbo Y è troponimo del verbo X se nel fare l'attività X si fa anche la Y;
- *Implicazione* (entailment): un verbo è implicazione di un altro verbo se durante lo svolgimento di quest'ultimo deve essere eseguito anche il primo, ad esempio "dormire" è una implicazione di "russare", ovvero la relazione viene vista come, il verbo Y è implicazione del verbo X se nel fare X uno deve per forza fare Y.

Per gli aggettivi valgono i seguenti tipi di relazione:

- *Nomi relativi*
- *Similitudini*
- *Participi dei verbi*

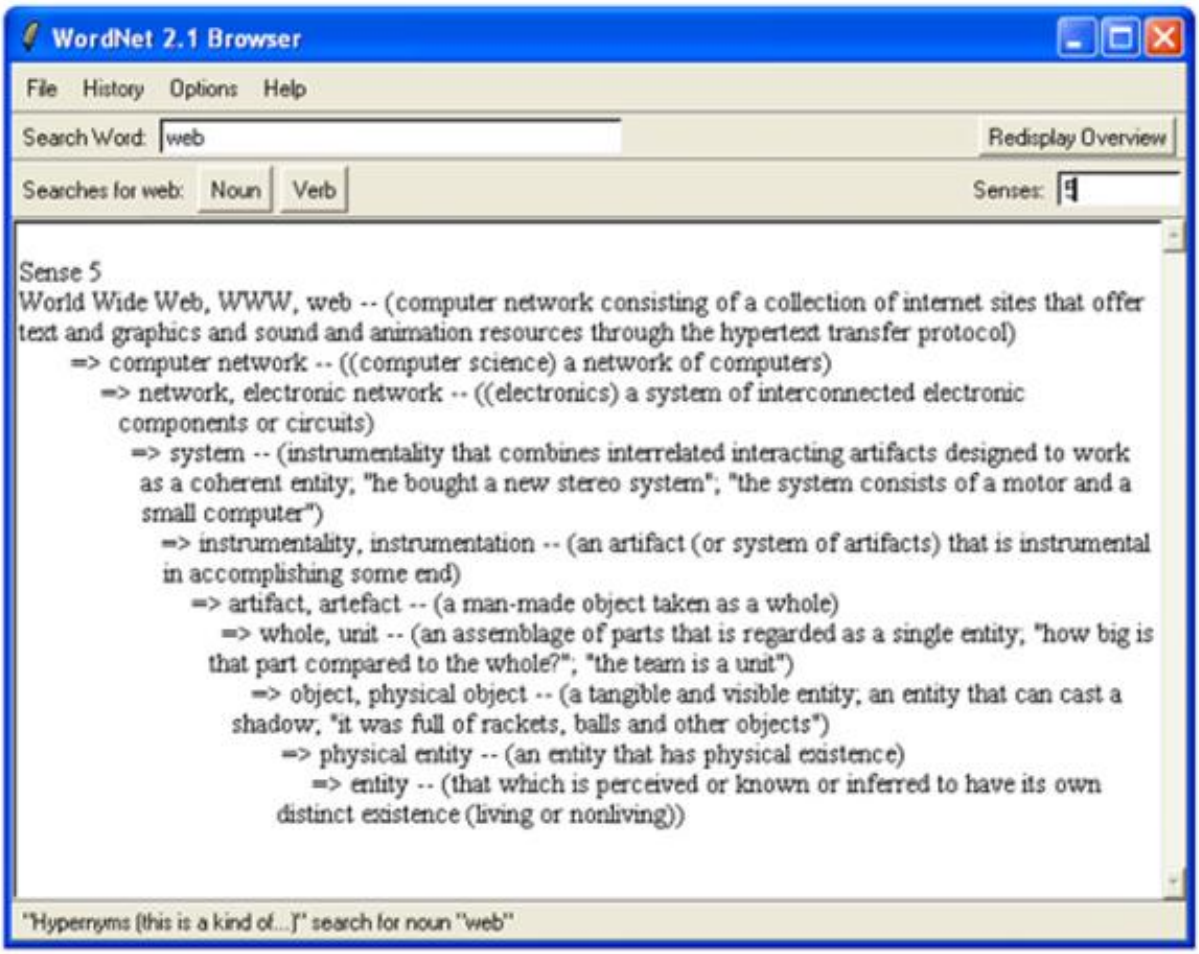

Figura 4.4 - Esempio di ricerca dei termini in relazione "iperonimia" con il sostantivo "web"

4.3.6.2 Struttura del batabase WordNet

All'interno del database lessicale WordNet, per ciascuna categoria sintattica, sono necessari due file per rappresentare il contenuto:

- index.*pos*
- data.*pos*

Dove "*pos*" può essere sostituito con *noun* (sostantivi), *verb* (verbi), *adj* (aggettivi) e *adv* (avverbi).

Ogni file indice si presenta come una lista ordinata alfabeticamente e contenente tutti i termini che si possono trovare all'interno di WordNet in corrispondenza di una determinata categoria sintattica.

Ogni riga nel file indice viene scritta nel seguente formato:

lemma pos synset_cnt p_cnt [ptr_symbol ...] sense_cnt tagsense_cnt synset_offset [synset_offset ...]

dove:

- *lemma*: corrisponde al testo del vocabolo scritto in caratteri ASCII minuscoli, e nel caso in cui il termine sia composto da più vocaboli, essi sono separati dal carattere "_";
- *pos*: rappresenta la categoria sintattica (*n* - noun, *v* - verb, *a* - adjective, *r* - adverb).

I restanti campi sono in relazione a *lemma* e *pos*:

- *synset_cnt*: indica il numero di synset in cui il termine è presente, cioè corrisponde al numero di significati del termine memorizzati all'interno di WordNet;
- *p_cnt*: indica il numero totale dei puntatori che partono dal termine verso tutti i *synset* in cui esso è contenuto;
- *[ptr_symbol ...]*: rappresenta una lista di simboli ognuno dei quali rappresenta una tipologia di puntatore che ha origine dal termine;
- *sense_cnt*: indica il numero di significati memorizzati in WordNet, è un dato ridondante rispetto a *synset_cnt*, ma è necessario per motivi di compatibilità con le versioni precedenti di WordNet;
- *tagsense_cnt*: indica il numero di synset in cui il termine è presente come parte di un termine comprendente più vocaboli;
- *synset_offset*: rappresenta un offset di 8 interi decimali che corrispondono al numero di byte dall'inizio del file data per trovare un synset contenente il

termine, quindi, per ogni synset del termine è presente una sequenza di 8 byte.

Nel file data ogni riga corrisponde ad un synset, posizionati in base all'offset di byte definito nel file indice. I puntatori sono creati tramite una gestione di offset che permette di muoversi da un synset ad un altro.

Ciascuna riga nel file data viene rappresentata dal seguente formato:

synset_offset ss_type w_cnt word[word ...] p_cnt [ptr ...] | gloss

Dove:

- *synset_offset*: rappresenta l'offset di 8 decimali interi corrispondenti alla distanza in byte dall'inizio del file;
- *ss_type*: rappresenta la forma sintattica dei termini contenuti nel synset;
- *w_cnt*: rappresenta il valore numerico formato da due cifre esadecimale che indicano il numero di termini presenti nel synset;
- *p_cnt*: rappresenta il valore numerico formato da tre cifre decimali che indicano il numero di puntatori che partono dal synset verso gli altri synset;
- *word*: rappresenta l'elenco dei termini presenti nel synset;
- *ptr*: rappresenta un puntatore verso un altro synset, dove tali puntatori hanno inoltre la seguente struttura: *pointer_symbol synset_offset pos source/target* dove:
    - *pos*: indica il suffisso del file indice in cui andare a cercare i termini;
    - *source/target*: usato per individuare il termine da cui parte la relazione, nel primo synset, e il termine che si trova al termine oggetto del puntatore nel secondo synset. È' rappresentato da un campo formato da quattro cifre esadecimale dove, le prime due cifre rappresentano la posizione del termine all'interno del synset sorgente, mentre le ultime due cifre indicano la posizione del termine all'interno del synset di destinazione. Se il valore di tale campo corrisponde a "0000", significa che esiste una relazione semantica tra i due synset;

- *gloss*: ciascun synset termina con tale campo, il quale, inizia con il carattere "|" seguito da una stringa di testo che continua fino al termine della riga. Essa può contenere una definizione del concetto semantico che vuole rappresentare il synset, una serie di esempi di frasi che aiutino a comprenderne il significato oppure entrambi i casi.

4.3.7 JWordNet

Per potersi interfacciare con il database WordNet, si è scelto di utilizzare le API Java del progetto JWordNet [JWN-09]. Utilizzando le potenzialità offerte da questa libreria, tramite piccoli insiemi di metodi, si ha la possibilità di percorrere l'albero delle relazioni ed eseguire analisi morfologiche su determinati termini, vocaboli, all'interno di metodi Java.

Per realizzare tali metodi, si è scelto di utilizzare la versione 1.4 di tali API. Per poter utilizzare le funzionalità offerte, è necessario eseguire inizialmente una breve procedura di configurazione:

- in primis, è necessario aver installato il database WordNet;
- successivamente, è necessario impostare alcuni parametri all'interno di un file XML di configurazione chiamato "*file_properties.xml*". Tra i vari parametri, solamente uno risulta obbligatorio da impostare, ed è il "*dictionary_path*", il cui valore deve corrispondere alla path assoluta in cui è stato installato il database WordNet.

Una volta eseguita la configurazione è possibile utilizzare tutti i metodi proposti dalle API JWordNet. Di seguito verranno illustrate le classe ed i metodi principali che si possono utilizzare.

4.3.7.1 JWNL

Classe statica utilizzata per contenere le informazioni di configurazione, ed in particolare, per poter utilizzare il database è necessario invocare il metodo *initialize()*, specificando la path del file "*file_properties.xml*"

```
JWNL.initialize(FileInputStream);
```

4.3.7.2 POS

Classe di enumerazione contenente le categorie sintattiche gestite dal database, definite in: *NOUN, VERB, ADJECTIVE, ADVERB*.

4.3.7.3 Dictionary

Classe grazie alla quale è possibile effettuare la ricerca di un termine all'interno del dizionario, ed in particolare, vengono utilizzati i seguenti metodi:

- `getIstance()`: è il metodo utilizzato per creare un oggetto Dictionary;
- `getMorphologicalProcessor()`: è il metodo utilizzato per creare un oggetto di tipo MorphologicalProcessor in grado di eseguire un'analisi morfologica di un termine come ad esempio la lemmatizzazione;
- `getIndexWord(POS pos, String lemma)`: individua all'interno del database il termine *lemma* passato come parametro dandogli una interpretazione di tipo POS, quindi, se la corrispondenza viene trovata nel database, restituirà un oggetto *IndexWord*, altrimenti *null*;
- `lookupAllIndexWord(String lemma)`: effettua la ricerca all'interno del database di tutte le occorrenze del termine *lemma* in ogni sua forma sintattica restituendo poi un oggetto di tipo *IndexWordSet*.

4.3.7.4 MorphologicalProcessor

Utilizzando una istanza offerta da questa classe è possibile trasformare la forma flessa di un termine nella sua forma base. Grazie al metodo *lookupBaseForm()* viene eseguita la lemmatizzazione dei termini

```
MorphologicaProcessor.lookupBaseForm(POS, String term)
```

4.3.7.5 IndexWord

Utilizzando una istanza di questa classe è possibile rappresentare una riga di un file indice. A partire da questo oggetto è poi possibile esplorare il database WordNet, ed in particolare, viene sfruttato il seguente metodo per individuare tutti i significati di un termine:

- `getSenses()`: il quale restituisce un array di oggetti *Synset*, praticamente restituisce tutti i synset che contengono il termini in analisi.

4.3.7.6 Synset

Un oggetto di questo tipo rappresenta l'astrazione di un synset, ovvero, attraverso esso è possibile richiamare metodi per trovare relazioni semantiche con altri synset, in particolare:

- `getPointers(PointerType)`: il quale restituisce degli oggetti di tipo *Pointer* che rappresentano tutte le relazioni semantiche di tipo *PointerType* che il synset ha con altri synset. La classe PointerType è una classe di enumerazione utilizzata per contenere i tipi di relazioni semantiche gestite da WordNet;
- `getWords()`: il quale restituisce un array di oggetti di tipo Word, ciascuno dei quali rappresenta un vocabolo contenuto nel synset.

4.3.7.7 Pointer

Un oggetto di questo tipo rappresenta il collegamento semantico tra due synset. Utilizzando tale metodo è possibile ottenere il synset di destinazione:

- `getTarget()`

4.4 SemaWeb: il metodo

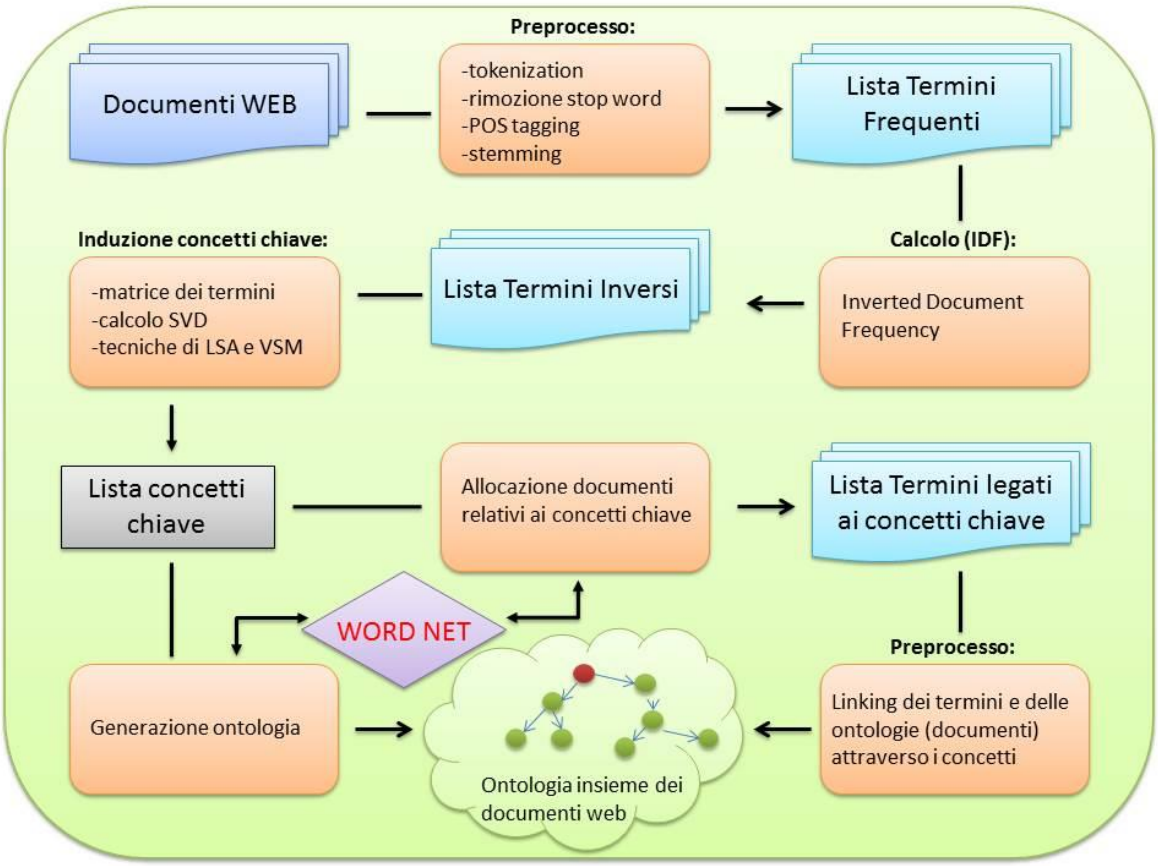

Figura 4.5 - Schema di funzionamento di SemaWeb

La figura 4.4 mostra una panoramica del metodo utilizzato per realizzare i processi di elaborazione e sviluppare le induzioni ontologiche in SemaWeb.

Il primo step è rappresentato dalla pre-elaborazione dei documenti scaricati dalla ricerca web effettuata, dove, ciascun singolo risultato viene rappresentato da una lista di termini di frequenza.

Il secondo step, è rappresentato dal calcolo della frequenza inversa dei termini del documento, dove tale peso ricavato, viene calcolato moltiplicando la frequenza del termine e la frequenza inversa del documento. Per ciascun termine viene ricavato il suo peso, e basandosi su questi ultimi, viene poi costruita una matrice dei termini del documento.

Il terzo step, è rappresentato dall'induzione dei concetti chiave basati sui termini frequenti ricavati in precedenza, in particolare, vengono utilizzate tecniche di LSA (Latent Semantic Analysis), per creare la lista dei concetti chiave dei documenti scaricati.

Il quarto step, è rappresentato dalla realizzazione delle generazioni ontologiche grazie all'utilizzo del database WordNet. In particolare, tale database è stato usato per poter ricavare in particolare per ciascun concetto chiave, i relativi termini sinonimi, iponimi ed iperonimi, ed altre funzioni ontologiche, rendendo possibile tramite l'utilizzo di metodi semantici, la creazione delle ontologie legate ai documenti restituiti dal motore di ricerca, più precisamente, la creazione di un sommario dei termini frequenti in grado di offrire all'utente una lista dei documenti inerenti, e la creazione di un albero ontologico basato sui concetti chiave e sulle dipendenze tra i vari iperonimi comuni a ciascun concetto.

4.4.1 Tecnica di LSA (Latent Semantic Analysis)

Latent Semantic Analysis (LSA) è una tecnica utilizzata nella elaborazione del linguaggio naturale, in particolare nell'ambito della semantica vettoriale, permettendo di analizzare le relazioni tra un insieme di documenti e termini che contengono, una serie di concetti relativi ai documenti stessi [WIK-LSA].

All'interno del progetto, per estrarre i concetti chiave inerenti ai documenti elaborati, viene utilizzata questa tecnica, in particolare, si utilizza la tecnica di VSM (Vector Space Model) e di SVD (Singular Value Decomposition).

VSM è un metodo di recupero delle informazioni basato sull'utilizzo dell'algebra lineare e di operazioni di confronto su dati testuali, che associa un singolo vettore multidimensionale a ciascun documento, e ogni componente di tale vettore, riflette una determinata parola chiave o termine correlato al documento.

SVD è un metodo di fattorizzazione di una matrice reale o complessa, utilizzata in algebra lineare, che permette di effettuare elaborazioni e statistiche [WIK-SVD]. Formalmente, la decomposizione in valori singolari di una m x n matrice M reale o complessa è una fattorizzazione della forma $M = U\Sigma V^*$, dove U è un m x m matrice unitaria reale o complessa, Σ è un m x n matrice diagonale con

numeri reali non negativi sulla diagonale, e *V** (il trasposto coniugato di *V*) è un *n* x *n* matrice unitaria reale o complessa.

In conclusione, una volta estratti i termini frequenti di maggior rilievo, per creare le ontologie, sono stati utilizzati i sinonimi dei concetti stessi, in quanto importanti anch'essi tanto quanto i concetti stessi, inoltre, sono stati valutati anche gli iponimi di ciascun concetto per poter cogliere così significati più specifici dei concetti stessi.

4.4.2 Pre-elaborazione

Nella fase di pre-elaborazione, al cui termine verrà restituita una lista contenente i termini di maggior frequenza, come primo passo viene effettuata la tokenizzazione e la rimozione della punteggiatura di ciascun documento, e basandosi su una lista di "*stop word*", vengono rimossi quei termini che non hanno significato importante, come gli articoli ad esempio.

Successivamente, viene effettuato il POS tagging (Part of Speech tagging), ovvero, vengono eliminati quei termini che corrispondo a verbi, avverbi e aggettivi, e la lemmatizzazione della parola stessa.

Inoltre, per costruire la lista dei termini frequenti, vengono scartati tutti quei termini, la cui frequenza risulta minore o uguale a due.

4.4.3 Matrice dei termini del documento e induzione dei concetti chiave

Per la costruzione della matrice dei termini del documento, viene applicata la tecnica di *tfidf* (term frequency - inverted document frequency) in grado di ricavare il peso dei termini.

In particolare, nel modello di spazio vettoriale (VSM) un documento *d* viene

$$\vec{d} = (tf_{t1}, \dots, tf_{ti})$$

dove tf_t restituisce la frequenza assoluta del termine $t \in T$ nel documento $d \in D$, dove *D*, è il corpus dei documenti e $T = (t_i, \dots, t_i)$ è l'insieme dei termini diversi che si verificano in *D*. Per poter ricavare il peso della frequenza di un termine in un

documento con un fattore che sconti la sua importanza quando appare in quasi tutti i termini, l'*idf* del termine *t* nel documento *d* è stato calcolato tramite la seguente formula proposta da Salton [SWY-75]:

$$w_t = tf_t \; x \; (\log_2 n - \log_2 df_t + 1)$$

dove df_t corrisponde alla frequenza nel documento del termine *t*, il quale conta quante volte tale termine appare all'interno del documento stesso.

Per condurre l'induzione dei concetti chiave, la tecnica di LSA viene applicata per poter elaborare la matrice dei termini dei documenti, eseguendo il calcolo dell'SVD sulla matrice ottenuta.

L'SVD, spezza la matrice ricavata in tre matrici (*U*, *S* e *V*) in modo tale che la matrice "*M*" sia: $M = USV^T$.

Selezionando poi i *k* valori singolari più grandi da *S*, e i loro corrispondenti vettori singolari da *U*, si otterà un'approssimazione di *M* con il minor margine di errore assumendo *k* come rango.

In aggiunta, l'SVD traduce i termini del vettore e dei documenti in un concetto di spazio. Quindi, le prime *r* colonne di *U* (dove *r* corrisponde al rango di *M*), formeranno una base ortogonale per lo spazio dei termini inerenti alla matrice dei termini dei documenti. Quindi i vettori base che corrispondono ai vettori di colonna della matrice *U* sono una rappresentazione vettoriale dei termini astratti dei documenti.

Considerando il tutto, se si prendessero tutti gli *r* vettori base come termini astratti, otterremmo uno spropositato ed inappropriato numero di concetti chiave. All'interno del progetto, per evitare tutto ciò, si è scelto di impostare un valore di soglia settato ad 1.0 per determinare quante colonne di *U* si dovranno considerare.

Di conseguenza, si otterrà U_k, consistente nelle prime *k* colonne di *U*. Utilizzando la matrice ottenuta, ed effettuandone la "trasposta", si otterrà la matrice U_k^T, e, da tale matrice, il valore più alto di ogni riga denoterà la corrispondente importanza del termine.

Infine, estraendo i valori massimi da ciascuna riga, come sopra citato, e trovando i rispettivi concetti all'interno della lista dei termini, si otterrà la lista dei concetti chiave.

4.4.4 Allocazione dei documenti e generazione ontologica

Per effettuare l'allocazione dei documenti in base ai concetti chiave estratti, si è attuato un processo simile a quello dell'estrazione dei documenti in base ad una query in un modello di information retrieval espandendo la ricerca dei termini all'interno dei documenti, ed includendo oltre ai relativi concetti chiave, i loro termini sinonimi ed iponimi.

Conseguentemente, i documenti saranno allocati tramite liste ai loro relativi concetti, basandosi sul calcolo della loro similarità di coseno, tra il documento in analisi e l'insieme dei termini che includono il concetto chiave, i suoi sinonimi ed i suoi iponimi. Per ciascun concetto, se il valore della similarità di coseno tra il documento e i concetti risulta maggiore del valore di soglia impostato, tale documento sarà allocato alla lista del concetto chiave.

In aggiunta alla creazione della lista dei documenti legati ai concetti chiave, nel progetto è stato realizzato un ulteriore metodo in grado di creare un albero delle ontologie utilizzando la potenzialità di WordNet . Grazie ad alcune funzioni offerte dalle API JWordNet, vengono estratti per ciascun concetto chiave i propri iperonimi diretti. Per ragioni di ottimizzazione si è scelto di estrarre l'albero degli iperonimi per ciascun concetto selezionando il primo senso all'interno del database, ovvero, quello con ricorrenze di significato maggiore nell'uso quotidiano della lingua. Utilizzando tali concetti estratti, viene costruito un albero rappresentante i concetti iperonimi comuni legati ai concetti chiave in questione, potendo visualizzare così le dipendenze comuni tra i vari termini.

Tale metodo è stato sviluppato tramite il seguente schema algoritmico**:**

Input:
α: lista vuota {utilizzata per tener traccia la lista dei termini iperonimi di un singolo concetto chiave}
β: lista contenente i concetti chiave

δ: lista vuota { è una lista di nodi (albero) utilizzata per costruire le relazioni ontologiche tra i concetti chiave}

Algoritmo - Generazione Ontologia

FOR each concept C_i in β **DO**
    ADD HypernymList of C_i to α
    CREATE TreeNode e_i that represent C_i
        FOR each hypernym h_i in α **DO**
            CREATE TreeNode p_j that represent h_i
            IF p_j exist in δ
                CONTINUE
            ELSE
                ADD p_j to δ as subnode
            END IF
        END FOR
    ADD e_i to the correct direct hypernym
END FOR

Output:
δ -> una lista di nodi che rappresenta l'albero delle ontologie, dove a ciascun concetto chiave è collegata la relativa lista dei documenti di interesse

Realizzato l'albero delle dipendenze ontologiche, i documenti vengono allocati ai loro rispettivi concetti chiave.

Quando un utente scorre l'albero, e visualizza all'interno della lista un concetto, questo automaticamente mostrerà nell'apposita schermata i documenti inerenti.

Con questo tipo di visualizzazione ontologica dei concetti, qualsiasi utente può essere in grado di visualizzare e carpire la struttura delle dipendenze tra i vari termini, ed essere in grado di semplificare le proprie ricerche di interesse, in quanto, un termine

chiave se ad esempio, inserito ad un livello superiore, molto probabilmente conterrà oltre ai propri documenti inerenti, anche quelli di concetti che si possono trovare in sottolivelli del termine stesso.

4.5 SemaWeb: funzionalità

Verranno illustrate di seguito le funzionalità principali dell'applicazione in modo dettagliato servendosi anche di alcuni diagrammi UML e schermate appartenenti all'interfaccia del progetto stesso, in modo da poter illustrare i vari casi d'uso che intercorrono tra l'utente ed il sistema.

Nel diagramma seguente, è possibile notare come il fulcro centrale dell'attività elaborativa sia dettato dalla "ricerca web" e dai risultati restituiti da essa.

Quindi, i risultati di elaborazione dipenderanno molto dall'esecuzione della query, se generata con termini di ricerca generici, oppure con termini più specifici.

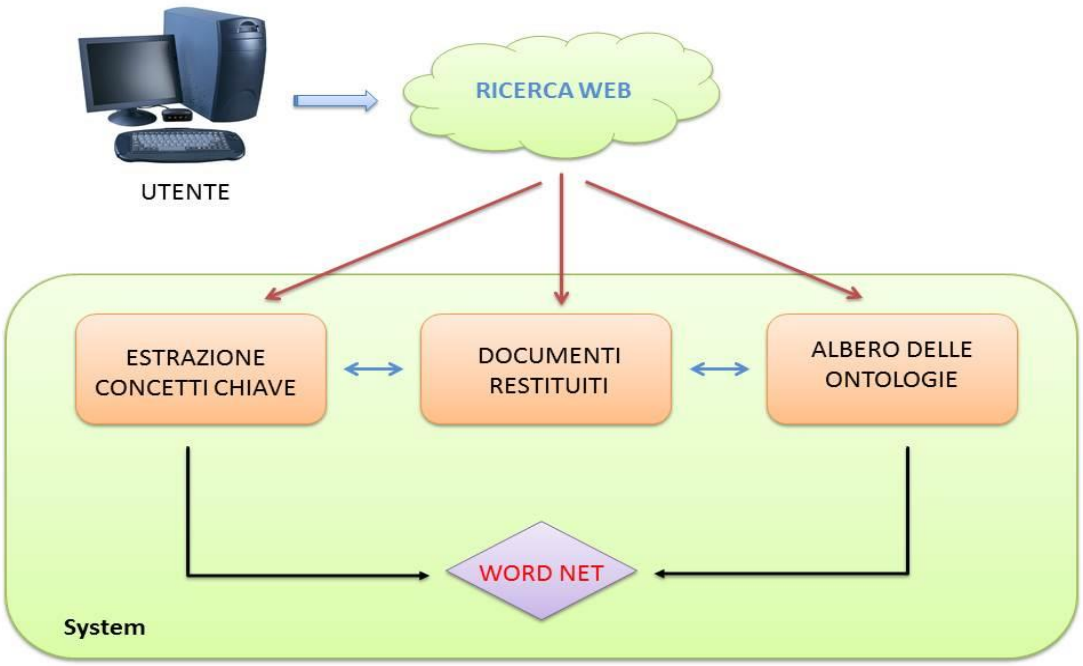

Figura 4.6 - Diagramma d'uso del progetto

4.5.1 Ricerca Web e documenti restituiti

La ricerca web dei documenti avviene come in un qualsiasi altro motore di ricerca, attraverso una form. I risultati restituiti dalla query di ricerca saranno poi visualizzati a video mostrando il titolo, una anteprima del testo del documento ed il link per raggiungere tale articolo.

Figura 4.7 - Schermata di ricerca di Semaweb

Attualmente causa motivi di risorse, e limiti imposti dalle API di Google per la ricerca, è possibile cercare ed estrarre solamente i primi 64 risultati.

Per facilitare la visualizzazione, inizialmente, l'intero set di risultati viene suddiviso in pagine, contenente ognuna 10 risultati.

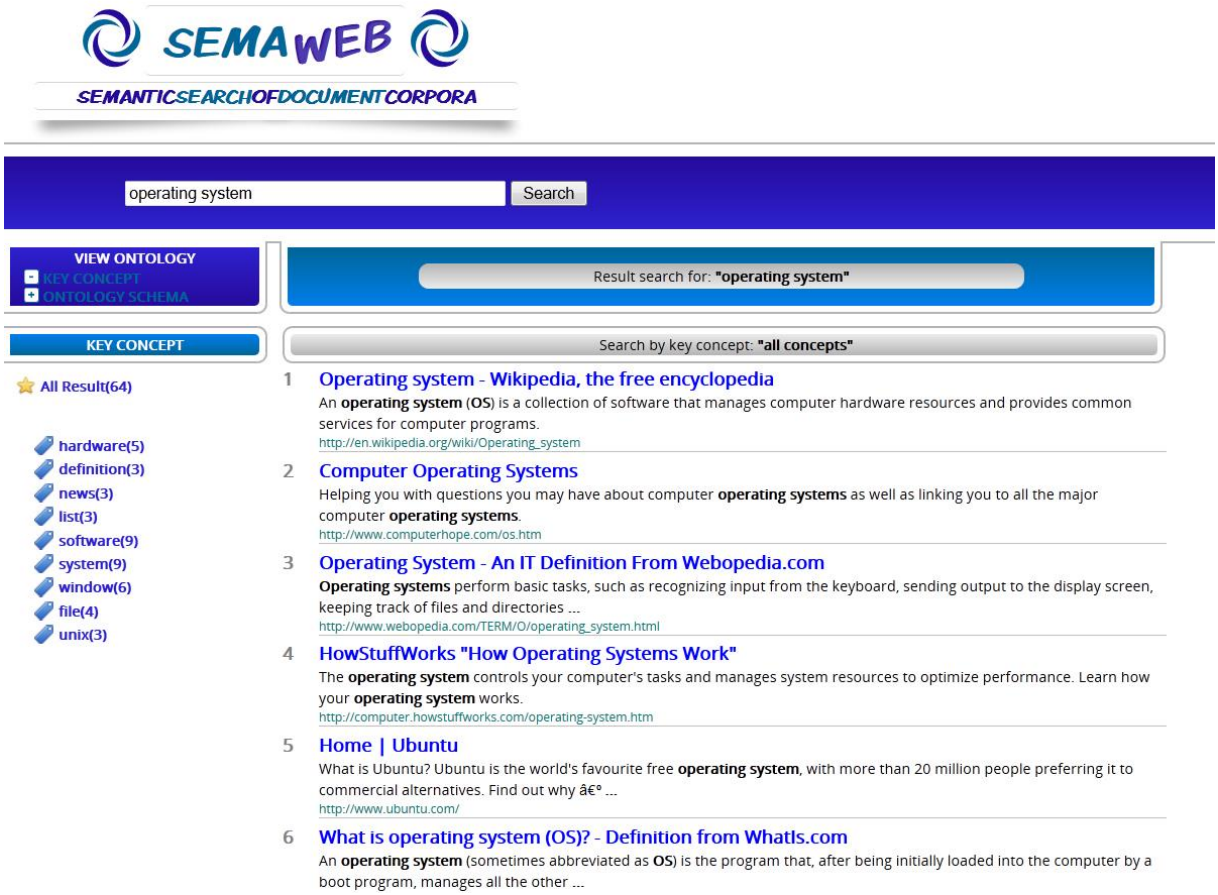

Figura 4.8 - Schermata dei risultati di una ricerca web

4.5.2 Estrazione concetti chiave

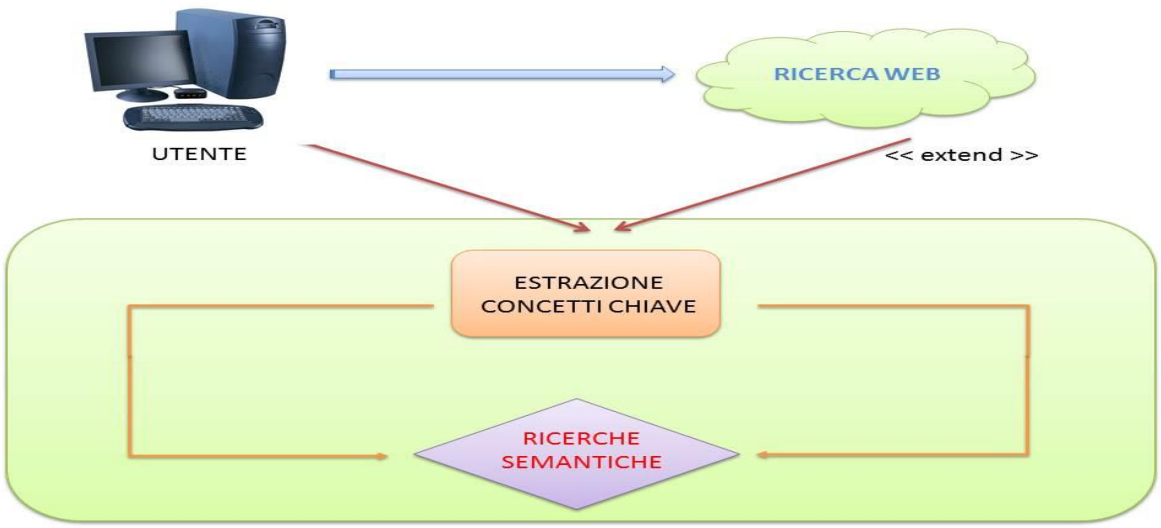

Figura 4.9 - Estrazione dei concetti chiave

Realizzata una ricerca web, il sistema effettua una elaborazione dei documenti, restituendo, grazie a ricerche semantiche, una lista di concetti chiave che suggeriranno il tipo di argomento trattato dai documenti inerenti a tale termine.

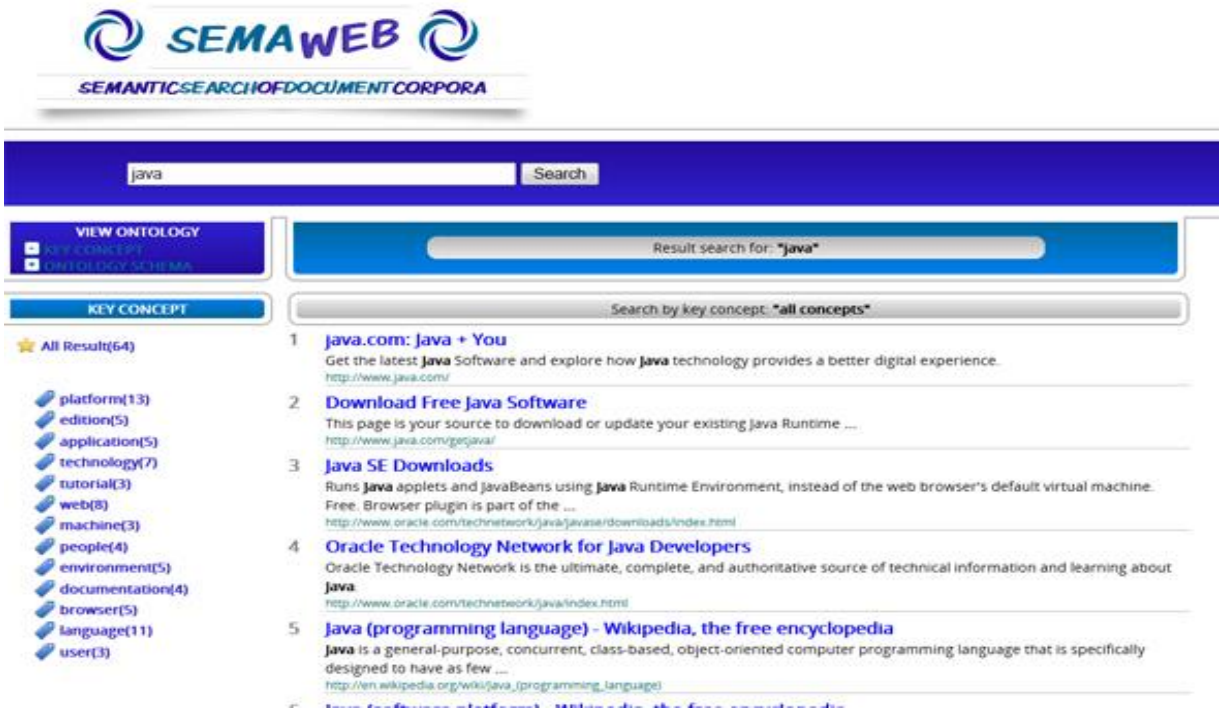

Figura 4.10 - Schermata di estrazione dei concetti chiave (box di sinistra)

4.5.3 Albero delle relazioni ontologiche

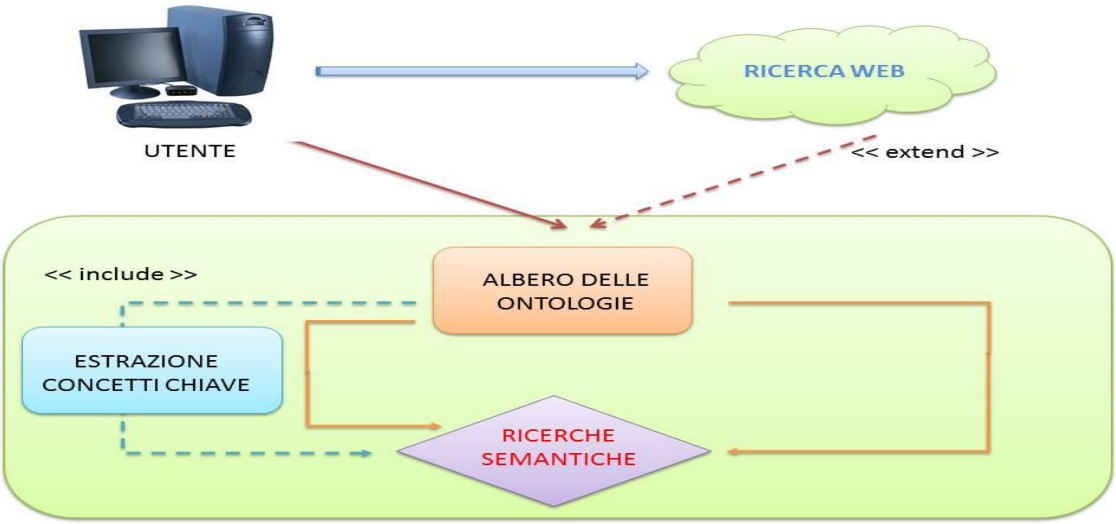

Figura 4.11 - Creazione albero ontologico (relazione tra concetti chiave)

Selezionando dal menù opzionale la visualizzazione dello schema ontologico, si accede ad una sezione dove è possibile visualizzare la gerarchia iperonima di ciascun concetto chiave ricavato dalla ricerca. Come si nota, sono presenti diversi tipi di visualizzazione, quella generale dove sono elencati tutti i concetti e i loro relativi termini iperonimi, quella relativa ad ogni singolo concetto dove sarà possibile visualizzare la gerarchia ontologica inerente ed infine la visualizzazione dell'albero ontologico.

L'albero delle relazioni ontologiche è costruito attraverso una ricerca gerarchica dei concetti iperonimi e dei termini stessi. Tale schema visualizza le relazioni in comune appartenenti a ciascun concetto. Questo tipo di ricerca è in grado di specificare il ramo di appartenenza di ogni singolo termine e di visualizzare le relative categorie di interesse nelle quali essi sono contenuti.

Di seguito è possibile visualizzare tali risultati analizzando le relative immagini.

Figura 4.12 – Visualizzazione lista concetti iperonimi

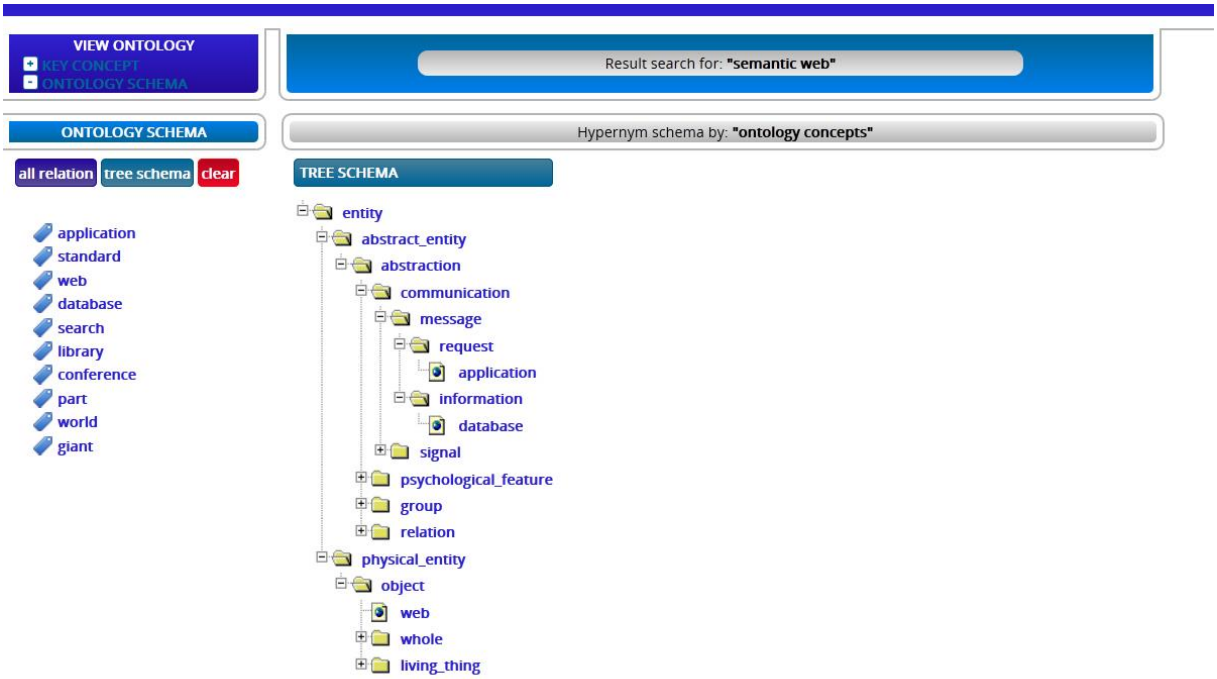

Figura 4.13 – Albero relazioni ontologiche

4.6 SemaWeb: componenti del sistema

Nel paragrafo precedente abbiamo analizzato le funzionalità dell'applicazione dal punto di vista dell'utente tralasciando i dettagli implementativi; in questo paragrafo tratteremo invece i componenti di cui è costituito il sistema; successivamente verrà mostrato come questi componenti interagiscono tra di loro per realizzare le varie funzionalità.

4.6.1 Classi

Di seguito verranno elencate le classi esterne utilizzate, necessarie per compiere determinate azioni:

- *Jwnl.jar*: libreria utilizzata come interfaccia di connessione tra l'applicazione ed il database di WordNet;
- *LibJson.jar*: libreria utilizzata per gestire i risultati in formato json;
- *Ejml.jar*: (Efficient Java Matrix Library) libreria di algebra lineare utlizzata per la gestione delle matrici all'interno dell'applicazione;
- *Qtag.jar*: libreria utilizzata per il POS(Part of Speech) tagging della parte testuale dei documenti ricavati;
- *Jenkov-PrizeTags*: libreria utilizzata per la gestione e la realizzazione dell'albero delle relazioni ontologiche all'interno dell'applicazione.

Le classi implementate invece sono le seguenti:

- *LibWordNet.jar*: classe che si occupa di effettuare la connessione al database WordNet utilizzando la libreria di riferimento "Jwnl" e di gestire tutte le relative funzionalità di ricerca e recupero ed elaborazione delle informazioni per i termini;
- *LibVSM.jar*: classe che gestisce l'elaborazione del Vector Space Model ed in particolare, le operazioni da effettuare attraverso il metodo di SVD (Singular Value Decomposition) della matrice inversa dei termini;

- *LibStopWord.jar*: classe che gestisce la ricerca delle StopWord dato un relativo termine;
- *BuildResult.jar*: classe che estende "httpservlet", rappresenta la servlet dell'applicazione e gestisce le principali azioni svolte dal programma.

4.6.2 Pagine JSP

Le pagine jsp sono servlet scritte ad un livello più alto di astrazione ed è giusto quindi inserire la loro descrizione in questo contesto.

L'interazione con l'utente è realizzata attraverso due pagine jsp:

- *Index.jsp*: costituisce la pagina di presentazione del progetto, essa la si può considerare statica in quanto non vengono gestiti nessun tipo di dati;
- *Resultsearch.jsp*: pagina che rappresenta il nucleo dell'applicazione, in essa è possibile visualizzare ed interagire con i risultati di ricerca;
- *Onto_tree.jsp*: pagina integrata all'interno della "resultsearch.jsp", gestita attraverso chiamate asincrone(ajax), permette di visualizzare e compiere azioni sull'albero delle relazioni ontologiche;

4.7 Analisi delle funzionalità

In questo paragrafo viene descritta l'implementazione delle funzionalità del sistema attraverso l'utilizzo di diagrammi UML e mostrando le parti del codice più rilevanti.

4.7.1 Ricerca web

La ricerca web viene realizzata tramite l'interazione tra la pagina *search.jsp* oppure *resultsearch.jsp* e la richiesta di recupero dati alle API di Google, in particolare vengono utilizzati i seguenti meteodi:

- *buildResultSearch*:

```
private JSONArray buildResultSearch(HttpURLConnection connection)
        throws IOException, JSONException
{
    String line;
    StringBuilder builder = new StringBuilder();
    String response;
    InputStream inputStream;
    try
    {
        inputStream = connection.getInputStream();
    }
    catch(IOException e)
    {
        inputStream = connection.getErrorStream();
        e.printStackTrace();
    }

    BufferedReader buffreader = new BufferedReader(new InputStreamReader(inputStream));
    while((line = buffreader.readLine())!= null)
    {
        builder.append(line);
    }
    response = builder.toString();
    JSONObject json = new JSONObject(response);
    JSONArray ja = json.getJSONObject("responseData").getJSONArray("results");

    return ja;
}
```

Come spiegato in precedenza, i dati della ricerca web sono restituiti in formato JSON; questa funzione serve per la lettura del flusso di dati provenienti dal servizio di Google (la cui connessione è contenuta nell'oggetto di tipo `URLConnection`) e per l'inserimento di questi ultimi in un oggetto di tipo JSONArray in maniera da facilitare la gestione e l'estrazione di dati dal formato JSON.

- *resultSearch*:

```
public void resultSearch(String query)
{
    int i=0;
    int j;
    HttpURLConnection connection;
    JSONArray ja;
    try{
        query = URLEncoder.encode(query, "UTF-8");
        while(i<this.resultSize)
        {
            connection = getConnection(query,i);
            ja = buildResultSearch(connection);

            for(j=0;j<ja.length();j++)
            {
                JSONObject jo = ja.getJSONObject(j);
                resultTitleFormatted.add(jo.getString("title"));
                resultContent.add(jo.getString("content"));
                resultContentFormatted.add(jo.getString("content"));
                resultTitle.add(jo.getString("titleNoFormatting"));
                resultURL.add(jo.getString("url"));
            }
            i = i+8;
        }
    }
    catch(Exception e)
    {
        this.resultTitle = null;
        this.resultURL = null;
        e.printStackTrace();
    }
}
```

Metodo che svolge diversi compiti; innanzitutto codifica la query di ricerca in formato UTF-8 per poter essere trasmessa tramite una richiesta http; lo step successivo si ha con la chiamata al metodo `buildResultSearch()` che preleva i risultati dall'oggetto `JSONArray` e li inserisce all'interno di oggetti parametrizzati di tipo `ArrayList<String>`; questi saranno i dati che verranno visualizzati all'utente come risultato di ricerca, quindi vengono estratti il titolo, l'anteprima ed il link di riferimento al documento.

Come detto in precedenza, la API di Google consentono di estrarre solamente 8 risultati per volta, per questo motivo l'interrogazione e l'estrazione vengono eseguite tramite l'utilizzo di un ciclo `while`.

4.7.2 Estrazioni dei concetti chiave

Questa operazione è una delle più complesse e costose da punto di vista progettuale e computazionale.

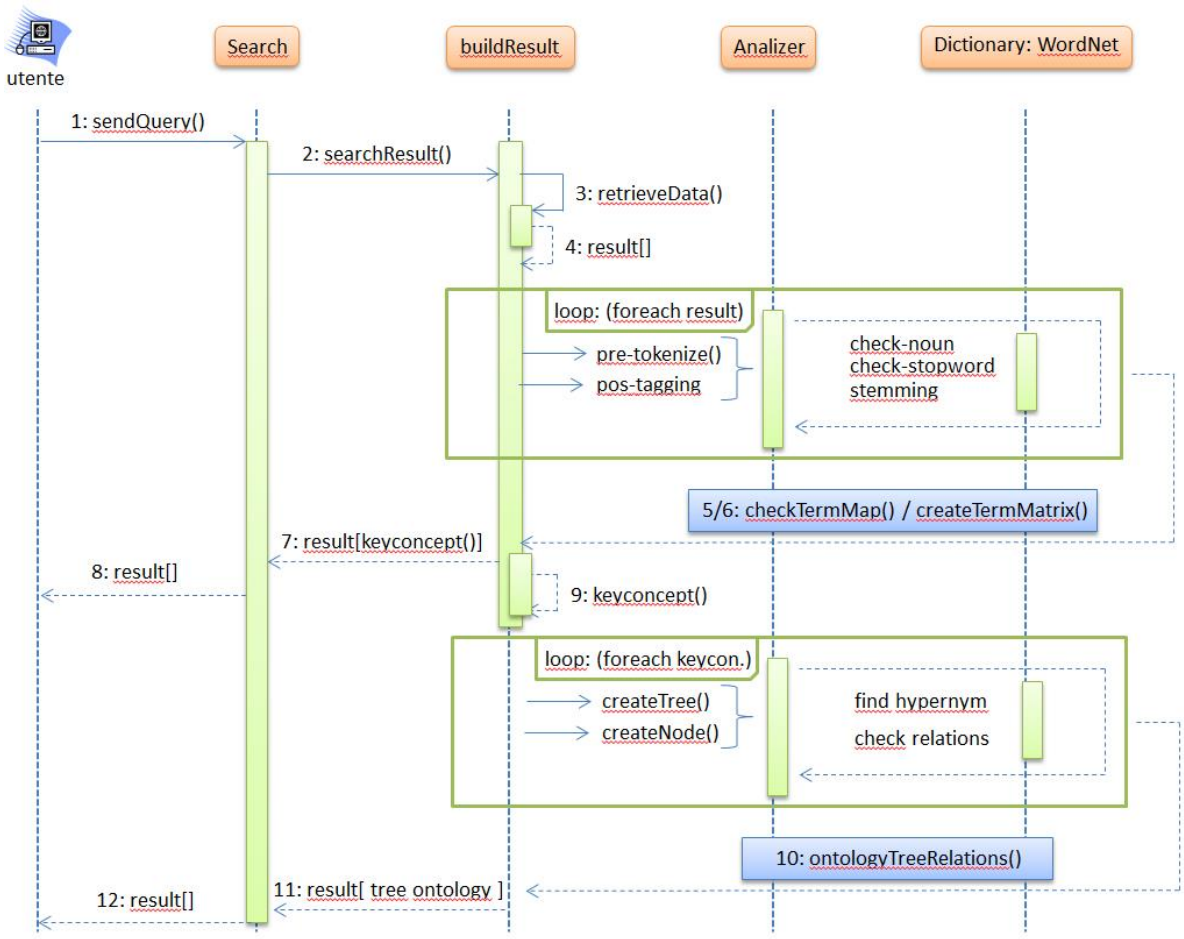

Figura 4.14 – Diagramma dell'estrazione dei concetti chiave e generazione albero ontologico

Come si può notare analizzando il diagramma, una volta effettuata una ricerca, vengono elaborati i dati di riscontro ed effettuati diversi controlli sui singoli termini di ciascun documento. Ricavata la lista dei concetti chiave e realizzato l'oggetto contenitore inerente, tali

risultati vengono poi restituiti alla pagina "*resultSearch.jsp*" che visualizzerà e permetterà di gestire tutte le funzionalità messe a disposizione.

Per raggiungere il traguardo, uno degli step più significativi è rappresentato dalla funzione *createTermMatrix()*:

```
public void createTermMatrix()
        throws NullPointerException, JWNLException
{
    int totDoc = contentTerm.size();
    Set set_tm = termMap.keySet();
    Iterator iter_tm = set_tm.iterator();
    while(iter_tm.hasNext())
    {
        Object key = iter_tm.next();
        termFrequent.add(key.toString());
    }
    int totTerm = termFrequent.size(), totTermDoc, frequency = 0;
    double matrixTerm[][] = new double[totTerm][totDoc];
    for(int k=0; k<totTerm; k++)
    {
        String term = termFrequent.get(k);
        totTermDoc = calcDF(term);
        for(int i=0; i<totDoc; i++)
        {
            int numTermDoc = contentTerm.get(i).size();
            for(int j=0; j<numTermDoc; j++)
            {
                String checkTerm = contentTerm.get(i).get(j);
                if(checkTerm.equals(term))
                    frequency++;
            }
            double tf = 0;
            if(numTermDoc != 0)
                tf = (double)frequency/(double)numTermDoc;
            double idf = Math.log((double)totDoc/(double)totTermDoc), w = tf*idf;
            matrixTerm[k][i] = w;
            frequency = 0;
        }
    }
    keyConceptInduction(matrixTerm);
}
```

Come si può notare analizzando il codice, il risultato finale è una matrice dei termini contenente il peso dei singoli concetti ricavati. Per costruire tale matrice, vengono eseguiti i seguenti controlli:

- inizialmente si tiene traccia del termine da analizzare;
- calcolo del numero totale di documenti recuperati;
- calcolo del numero totale dei termini presenti in un singolo documento;
- creazione della matrice contenente i pesi dei singoli termini;
- per ogni termine viene calcolato il suo document-frequency (DF);
- calcolato il DF, si scorre l'array che contiene i documenti e si tiene traccia del totale dei termini di ogni singolo documento;
- si calcola il term-frequency (TF), ovvero la frequenza del termine all'interno del singolo documento;
- calcolato il TF, si procede con il calcolo del term-frequency/inverted-document-frequency (TFIDF);
- calcolato il TFIDF, si ricava il peso del termine in esame moltiplicando TF*IDF;
- ottenuto il relativo peso, lo si inserisce nella matrice dei termini, la quale sarà poi sucessivamente elaborata dalla funzione *keyConceptInduction(matrixTerm)*.

La matrice viene poi passata come parametro ad una successiva funzione che effettua l'induzione dei concetti chiave analizzando i relativi pesi tramite la tecnica di LSA, ed analizzando l' SVD (Singular Value Decomposition) e recuperando i relativi termini sinonimi ed iperonimi.

Di seguito saranno elencati i principali passaggi effettuati per ottenere la lista dei termini elaborata dalla funzionalità descritta sopra. Tali metodi sono contenuti all'interno della funzione *processContentSearch()*:

- *pre-tokenizzazione della stringa*: effettuo un primo controllo sulla stringa da elaborare, eliminando la punteggiatura e i caratteri speciali;
- *divisione della stringa in token*: suddivido ogni stringa nei relativi token;
- *pos tagging*: effettuo una prima analisi sui termini e ne deduco il POS (part of speech), ovvero se il termine corrisponde ad un nome, un aggettivo o un avverbio. Successivamente attraverso il supporto del database lessicale WordNet per avere un doppio riscontro sulla deduzione fatta in precedeza, scarto tutti i vocaboli che non rappresentano nomi;

- *controllo delle stopword*: vengono eliminati tutti quei termini che non hanno un valore descrittivo come congiunzioni, articoli, numeri in forma letterale, ecc..;
- *controllo delle unusual word*: vengono eliminati quei termini inusuali, il cui significato non è fondamentale per la comprensione del testo, o quei termini che possono creare ridondanze durante la fase di estrazione dei concetti chiave;
- *lemmatizzazione*: viene portato il termine alla sua forma canonica;

Come detto in precedenza, ogni vocabolo nel linguaggio naturale può assumere diversi significati; questi significati nel database WordNet sono rappresentati dai *Synset*.

Un passaggio fondamentale durante l'elaborazione dei concetti chiave, è dato dai seguenti passaggi:

- per ogni vocabolo vengono cercati i synset in cui esso è contenuto attraverso il metodo getSenses();
- per ogni synset ottenuto, vengono cercati i synset che hanno una relazione semantica di tipo *PointerType*, nel nostro caso vengono ricercate le relazioni di tipo SIMILAR_TO e HYPERNIM;
- per ogni nuovo synset, viene cercato al suo interno se ci sono termini presenti nel dizionario; nel caso cui venga riscontrato esito positivo, significa che due termini hanno una relazione di sinonimia, quindi il termine di partenza, o quello dei due con peso meno rilevante viene eliminato dal dizionario (evitiamo inutili ridondanze come accennato sopra);

4.7.3 Albero delle relazioni ontologiche

L'analisi per la costruzione dell'albero delle dipendenze ontologiche viene eseguita partendo dall'array contenente i concetti chiave definitivi.

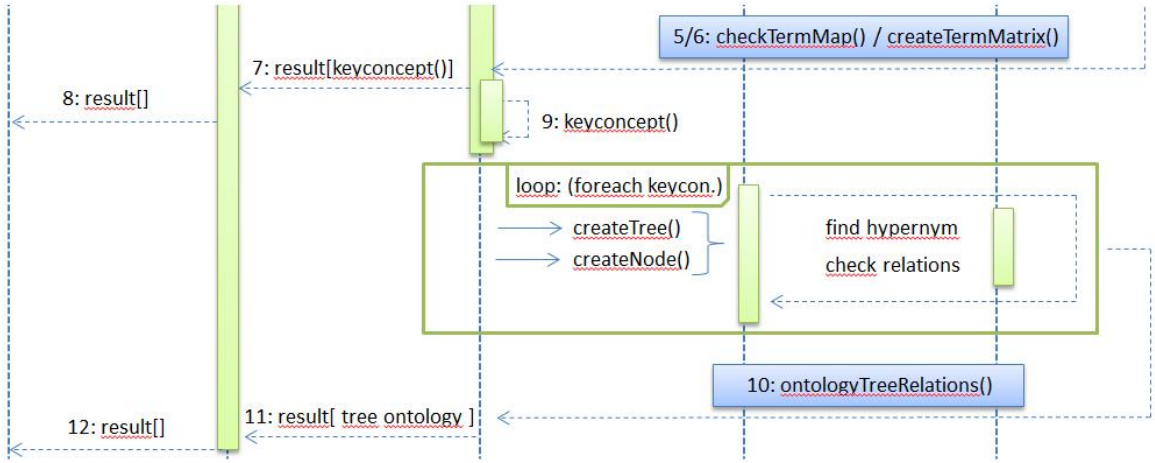

Figura 4.15 – Diagramma albero delle relazioni ontologiche

Questa funzionalità, consente di costruire l'albero ontologico delle dipendenze tramite i concetti chiave e l'interfacciamento di alcune funzionalità offerte da WordNet, quali la ricerca dei termini iperonimi.

Per motivi di complessità e di relativa lunghezza del codice inerente alla costruzione dell'albero delle dipendenze, riportiamo solo una piccola frazione del complesso, quella che consente di creare l'albero con i relativi nodi. In seguito verranno descritti brevemente i passaggi salienti delle parti non visualizzate:

```
public void createOntology()
    throws JWNLException, NullPointerException
{
    onto_tree = new Tree();
    ITreeNode root = null;
    boolean init = false;
    boolean first_node = true;
    for(String key:docAllocation.keySet())
    {
        if(!init)
        {
            ArrayList<String> tree_hypernym;
            root = new TreeNode("ID_root","entity","root");
            onto_tree.setRoot(root);
            ITreeNode node_child = new TreeNode("ID_"+key,key,"concept");
            ITreeNode node_prec = null;
            tree_hypernym = wordNet.getListHypernym(key);
            onto_map.put(key, tree_hypernym);
            int tot = tree_hypernym.size();
            tot = tot-2;
            for(int x=tot;x>=0;x--)
            {
                String hypernym = tree_hypernym.get(x);
                ITreeNode node_hypernym = new TreeNode("ID_"+hypernym,hypernym,"hypernym")
                if(first_node)
                {
                    root.addChild(node_hypernym);
                    first_node = false;
                }
                else
                {
                    node_prec.addChild(node_hypernym);
                }
                node_prec = onto_tree.findNode("ID_"+hypernym);
            }
            node_prec.addChild(node_child);
            init = true;
            first_node = true;
        }
        else
        {
            ArrayList<String> tree_hypernym;
            ITreeNode node_child = new TreeNode("ID_"+key,key,"concept");
            ITreeNode node_prec = null;
            tree_hypernym = wordNet.getListHypernym(key);
            onto_map.put(key, tree_hypernym);
            int tot = tree_hypernym.size();
            tot = tot-2;
            for(int i=tot;i>=0;i--)
            {
                String hypernym = tree_hypernym.get(i);
                ITreeNode node_hypernym = new TreeNode("ID_"+hypernym,hypernym,"hypernym")
                if(first_node)
                {
                    if(onto_tree.findNode("ID_"+hypernym) == null)
                    {
                        root.addChild(node_hypernym);
                    }
                    node_prec = onto_tree.findNode("ID_"+hypernym);
                    first_node = false;
                }
                else
                {
                    if(onto_tree.findNode("ID_"+hypernym) == null)
                    {
                        node_prec.addChild(node_hypernym);
                    }
                    node_prec = onto_tree.findNode("ID_"+hypernym);
                }
            }
            node_prec.addChild(node_child);
            first_node = true;
        }
    createDocOntology();
}
```

Questo metodo sostanzialmente istanzia un nuovo oggetto Tree(), ovvero il contenitore dell'albero. Una volta creato il nuovo albero, la radice e i nodi attraverso un ciclo, analizziamo la lista dei concetti chiave ed effettuiamo i seguenti controlli e passaggi:

- controlliamo se è stata creata la radice, altrimenti procediamo con la creazione della stessa;
- creiamo il nodo che contiene il concetto chiave;
- cerchiamo e ricavo tramite WordNet, l'albero che forma la scala degli iperonimi di tale termine;
- tengo traccia del numero totale degli elementi ricavati dalla ricerca sottraendo la radice degli iperonimi che non serve in quanto corrisponde a "root";
- scorro la lista ricavata in senso contrario e costruisco le dipendenze all'interno dell'albero settando le relative radici dal padre in questione;
- per ogni concetto ripeto lo stesso procedimento, inserendo ad ogni padre il relativo figlio comune se esiste;
- una volta terminata la verifica su tutti i concetti chiave, abbiamo a diposizione l'albero contenente le dipendenze ontologiche dei termini, e possiamo così vedere quali elementi accomunano i termini, se sono state riscontrate durante la ricerca degli elementi in comune;

Infine, dopo aver creato il nostro albero, creiamo anche una documentazione ontologica contenente per ogni singolo concetto, l'albero delle relative dipendenze iperonime, così da poterle confrontare una ad una.

Conclusioni

L'aumento esponenziale delle dimensioni di Internet, lo sviluppo continuo di nuove tecnologia e la facilità di accesso alla rete ha portato ad un enorme aumento della quantità di informazioni disponibili.

Con lo sviluppo dei motori di ricerca si è, in parte, data la possibilità agli utenti di trovare le informazioni desiderate; nonostante ciò, l'orientamento prettamente euristico dei motori di ricerca assieme l'elevato grado di ambiguità del linguaggio naturale, non è più sufficiente per ottenere dei risultati di ricerca di buona qualità.

La tendenza attuale, infatti è la produzione di sistemi che integrano al loro interno degli agenti semantici che riescono ad aggiungere delle capacità di ragionamento fondamentali per eseguire delle computazioni avanzate; nello stesso si cerca di far "partecipare" attivamente l'utente alla ricerca dando la possibilità di esprimere preferenze e giudizi.

SemaWeb ha cercato di fondere insieme queste caratteristiche in maniera di essere in grado di offrire l'immediatezza di un classico motore di ricerca con l'aggiunta di funzionalità personalizzabili, quali una ricerca più mirata verso una ristretta sezione derivante da quella eseguita o ad una visualizzazione di dipendenza tra i relativi documenti ricercati.

Occorre evidenziare che questo progetto può costituire una piccola base per lo sviluppo futuro di un'applicazione completa, in grado di potersi adattare alla continua evoluzione del Web e di tutto quello che lo costituisce.

Di seguito elencheremo dei limiti e delle possibili idee che potrebbero essere un punto di partenza per un ulteriore ampliamento del software:

- possibilità di scegliere una determinata categoria di ricerca: attualmente è possibile scegliere solo la ricerca web, ma questo, non è costituito unicamente da documenti, ma sono rappresentati svariati tipi di risorse, quali blogs, forum, ebook, video, immagini ed altro ancora…;

- ampliamento della ricerca da diversi motori: attualmente i risultati provengono unicamente dalla ricerca tramite API di Google;
- miglioramento della procedura di estrazione dei concetti chiave: attualmente la ricerca dei termini viene effettuata scansionando i titoli e le anteprime dei documenti. Utilizzando funzionalità che java dispone, è possibile poter leggere l'intero documento cui si riferisce l'anteprima ricevuta. In questo caso, si dovrebbero poi utilizzare tecniche per la sommarizzazione dei dati ancora più complesse e precise, come ad esempio un metodo astrattivo oppure di interpretazione delle sottostringhe vicine ad una determinata frase presa in esame, in quanto gli algoritmi che si basano su una supposizione statistica non dà la certezza di poter interpretare in modo completamente corretto il contenuto dell'informazione stesso;
- possibilità di visualizzare una lista o un insieme di documenti che descrivano la natura dei concetti chiave ricavati;

In conclusione SemaWeb potrebbe essere inserito all'interno degli studi e soluzioni dei limiti dei motori di ricerca di cui si è parlato inizialmente; è buona cosa precisare che l'obiettivo della realizzazione di questo tipo di varianti ai motori di ricerca, non ha lo scopo di doverli sostituire, ma di completarli; ognuno di essi, dai motori di ricerca sociali, a quelli collaborativi, hanno delle potenzialità e funzionalità differenti che si rivolgono a tipi di utenze ben specifiche.

Bibliografia

Capitolo 1

[WIK-WWW] Wikipedia, “World Wide Web”, http://it.wikipedia.org/wiki/Web

[WIK-MDR] Wikipedia, “Motore di ricerca”, http://it.wikipedia.org/wiki/Motori_di_ricerca

[CR-OS] C.Carpineto, G.Romani, “Verso i motori di ricerca di prossima generazione”, Mondo digitale, 2005, 2, pp.19-31

[WIK-PR] Wikipedia, “Page Rank”, http://it.wikipedia.org/wiki/PageRank

[WIK-MRS] Wikipedia, “Social Search”, http://en.wikipedia.org/wiki/Social_search

[WIK-MRC] Wikipedia, “Collaborative Search Engine”, http://en.wikipedia.org/wiki/Collaborative_search_engine

[WIK-WS] Wikipedia, “Web Semantico”, http://it.wikipedia.org/wiki/Web_semantico

[WIK-ON] Wikipedia, “Ontologia Informatica”, http://it.wikipedia.org/wiki/Ontologia_(Informatica)

[WIK-IR] Wikipedia, “Information Retrieval”, http://it.wikipedia.org/wiki/Information_Retrieval

Capitolo 2

[LUHN-58] H.P. Luhn 'The automatic creation of literature abstracts', IBM Journal of Research and Development, 1958, 2, pp. 159-165

[CHS-04] P. Cimano, S. Handschuh, and S. Staab, .Towards the self-annotating web,. in *WWW '04: Proceedings of the 13th international conference on World Wide Web*, 2004, pp. 462.471

[MIKA-05] P. Mika, .Ontologies are us: A uni_ed model of social networks and semantics,. in *The Semantic Web ISWC 2005*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005, pp. 522.536

[ACCS-09] Advanced Computer & Communication Systems, http://accsnet.com/hosts/, 2009

[RG-65] H. Rubenstein and J. Goodenough, “*Contextual correlates of synonymy*,” Commun. ACM, vol. 8, no. 10, pp. 627.633, 1965

[LN-98] D. Lewis, “Naive bayes at forty: The independence assumption in information retrieval,” in *Proceedings of the 10th European Conference on Machine Learning ECML-98*, 1998, pp. 4.15

[ITPFP-06] E. Iosif, A. Tegos, A. Pangos, E. Fosler-Lussier, and A. Potamianos, “Combining statistical similarity measures for automatic induction of semantic classes,” in *Spoken Language*

Technology, 2006. SLT '06. IEEE/ACL Workshop on, 2006, pp. 86.89

[CH-90] K. Church and P. Hanks, "Word association norms, mutual information, and lexicography," *Comput. Linguist.*, vol. 16, no. 1, pp. 22.29, 1990

[CV-07] R. Cilibrasi and P. Vitanyi, "The google similarity distance," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 3, pp. 370.383, 2007

[WIK-CA] Wikipedia, "Teoria della complessità algoritmica (Kolmogorov)", htpp://it.wikipedia.org/wiki/Teoria_della_complessità_alogitimica

Capitolo 3

[SB-07] http://www.sensebot.net

[CAR-02] http://project.carrot2.org

Capitolo 4

[WIK-JA] Wikipedia, "Java", http://it.wikipedia.org/wiki/Java_(linguaggio)

[HS-07] Herbert Schildt, "La creazione di Java", In "Java - La guida completa", McGraw-Hill, 2007, pp.7-12

[WIK-JS] Wikipedia, "JavaScript", http://it.wikipedia.org/wiki/JavaScript

[GASAPI-09] Google, "Google Ajax Search API - Developer's Guide" http://code.google.com/intl/it-IT/apis/ajaxsearch/documentation/, 2009

[WIK-JSON] Wikipedia, "Json", http://it.wikipedia.org/wiki/JSON

[JSON-99] http://www.json.org

[JLIB-06] http://jsonlib.sourceforge.net

[WIK-JQ] Wikipedia, "JQuery", http://it.wikipedia.org/wiki/JQuery

[EJML-11] http://code.google.com/p/efficient-java-matrix-library

[JEN-07] http://jenkov.com/prizetags

[WIK-WN] Wikipedia, "WordNet", http://it.wikipedia.org/wiki/Wordnet

[WNPU-09] Miller George A., "WordNet - About Us." http://wordnet.princeton.edu, Princeton University, 2009

[JWN-09] SourceForge, "JWordNet", http://sourceforge.net/apps/mediawiki/jwordnet/index.php

[WIK-LSA] Wikipedia, "Latent Semantic Analysis", http://en.wikipedia.org/wiki/Latent_semantic_analysis

[WIK-SVD] Wikipedia, "Singular Value Decomposition", http://en.wikipedia.org/wiki/Singular_value_decomposition

[SWY-75] G.Salton, A.Wong, CS.Yang, "*A vector space model for automatic indexing communication*", ACM, 18(11):613-620, 1975