TESI DI LAUREA

Cucina Semantica: Studio di un motore di ricerca semantico per ricette

Candidati:
Davide Iannello (matr. 653260)
Antonio Isolda (matr. 653270)

Relatore: Prof. Thimoty Barbieri

Politecnico di Milano, Polo Regionale di Como
Anno Accademico 2004-2005

Dipartimento di Elettronica e dell'informazione
email: nabucodonosor_82@yahoo.it
cantoreomero@yahoo.it
24 febbraio 2005

Al professor Barbieri,
che ci ha proposto l'argomento
e ci ha indicato la via

Il Web ha successo perché l'ipertesto è un mezzo talmente flessibile che il Web non limita il Sapere che cerca di rappresentare. Altrettanto deve valere per la rete di significati. In effetti, la rete di tutto quello che sappiamo e usiamo di giorno in giorno è assai complessa. Per rappresentarla ci serve la potenza di un linguaggio forte. [...]. Quando tale potenza verrà liberata, i computer della Rete Semantica acquisiranno prima la capacità di descrivere, poi di dedurre e infine di ragionare.

Tim Berners-Lee

Indice

Prefazione

Lo scopo fondamentale della nostra tesi è quello di sviluppare una web application che fornisca un primo prototipo di knowledge base. I documenti che vengono utilizzati ai fini del prototipo, e classificati usando il sistema delle ontologie, sono le ricette regionali della cucina italiana. Le ricette vengono suddivise per regione.
E' poi possibile fare una ricerca nella base di conoscenza usando un qualsiasi termine gastronomico, per categoria di cibo, per regione o per ingrediente.
Una volta fatta la ricerca e, dopo aver così ottenuto il nome della ricetta o delle ricette che ci interessano, il software deve essere in grado di connettersi al web e di trovare il testo completo della ricetta avvalendosi del motore di ricerca universale google.
Mediante l'ontologia sono possibili operazioni, prettamente semantiche: ad esempio dopo aver trovato una ricetta che contenga come ingrediente un limone possiamo cercare tutti le ricette che contengano al suo interno un ingrediente della categoria agrumi, a cui limone appartiene.

Possiamo quindi suddividere, fondamentalmente, il lavoro che intendiamo fare nelle seguenti fasi fondamentali:

- Fase di Ricerca dei documenti per l'ontologia: In questa fase si effettua una ricerca di tutti i documenti che vengono poi classificati e categorizzati con l'inserimento e la contestualizzazione all'interno di un'ontologia;
- Fase di Categorizzazione: La categorizzazione avviene utilizzando il sistema delle ontologie (OWL);
- Fase di Consultazione: La ricerca avviene sulle ontologie e mostra i risultati della ricerca in modo semantico e non in modo sintattico.

Quindi Partendo dalla nozione di Semantic Web, che abbiamo prima introdotto, si sono cercati di individuare gli strumenti che, affiancando Html, potenziano le funzionalità del Web: la ricerca ci ha portato ad individuare alcuni strumenti che stanno diventando di fatto degli standard.

La tesi è strutturata, quindi, nel seguente modo: il primo capitolo presenta quelle che sono le premesse al Semantic Web, sia dal punto di vista del web tradizionale sia da quello dell'intelligenza artificiale: in sostanza mostriamo il background che sta dietro al semplice, in apparenza, concetto di Web Semantic ed inoltre dell'ambito applicativo e scientifico nel quale è utilizzato; nel secondo capitolo descriveremo come la nostra tesi si inserisce nel contesto del background prima menzionato e le motivazioni per cui abbiamo approcciato il lavoro in un certo modo, mettendo anche in evidenza gli obiettivi che ci siamo stabiliti a priori da raggiungere con il nostro lavoro.

I successivi due capitoli analizzano e descrivono le specifiche del nostro lavoro e la progettazione tecnica del lavoro stesso e tutti i passi che abbiamo seguito per l'effettiva realizzazione del Semantic Web: la creazione delle ontologie, l'annotazione delle pagine, la raccolta delle informazioni e i modi di sfruttare le informazioni semantiche e mostriamo un'applicazione pratica, usando la sintassi di diversi linguaggi e i tools a nostra disposizione: sono analizzati i vari passi da compiere per la creazione dell'ontologia, le informazioni da annotare all'interno delle pagine e i tipi di risposte che si ottengono dall'interrogazione della base di conoscenza creata. Viene anche preso in esame un esempio già esistente di ontologia, nello specifico il prototipo conosciuto come Wine Agent.

Nel capitolo successivo mostriamo e descriviamo l'implementazione, in una sorta di manuale utente, mostrando come funziona il progetto implementato.

Nell'ultima sezione si mostrano i risultati ottenuti, mettendo in evidenza gli esiti raggiunti e mostrando le conclusioni a cui siamo arrivati ed i possibili sviluppi futuri. La bibliografia finale, infine, tenta di illustrare in modo esauriente tutti i documenti che si sono consultati durante la fase preliminare, di implementazione del progetto e di stesura di questa tesi, dando un occhio a tutti quei lavori che sono definiti come fondamentali in questo campo (materiale in lingua inglese) ma ponendo anche attenzione nella ricerca di materiale esistente in italiano, cercando di renderla il più completa possibile .

Capitolo 1

Introduzione

Una indagine autorevole di qualche anno fa stimava che il numero di utenti internet raddoppiasse ogni 18 mesi. Questo dato ci fa immediatamente rendere conto di come il web sia diventato uno strumento di comunicazione e di conoscenza sempre piú rilevante.

Nel navigare sul web si seguono dei link, che portano a quella che formalmente viene detta risorsa (resource), identificata univocamente da un URI.
Nel linguaggio corrente una risorsa viene anche detta documento, per mettere in evidenza il fatto che sia leggibile da un essere umano, o oggetto, per mettere in evidenza che è leggibile da una macchina. Qualunque sia il termine utilizzato, la risorsa non è una entitá a sé, ma è accompagnata da informazioni che la descrivono. Le informazioni sulla risorsa vengono generalmente dette Metadati.

Si puó quindi dire che i metadati sono informazioni, comprensibili dalla macchina, relative a una risorsa web o a qualche altra cosa. Il punto chiave è costituito appunto dal fatto che i metadati sono comprensibili dalla macchina (machine understandable).
Di conseguenza, i metadati costituiscono un tipo di informazione che puó essere utilizzata dai software agent, per fare un uso appropriato delle risorse, rendendo piú semplice e veloce il funzionamento del Web, aumentando la nostra fiducia in esso.

A titolo di esempio, quando si reperisce un documento (o un oggetto) sul web, utilizzando il protocollo HTTP, è possibile che il server invii alcune informazioni sulla risorsa, quali la sua data di aggiornamento, la data massima di validitá dell'informazione, il suo autore, etc.

Quindi il Web, come insieme di risorse e di informazioni sulle risorse (cioè metadati) è giá una realtá alla quale siamo abituati. Va tenuto presente che i metadati sono dati, e questo fatto ha alcune conseguenze. La prima è che i metadati possono essere memorizzati come dati, in una risorsa, che puó quindi contenere informazioni relative a se stessa o ad un'altra risorsa.

Attualmente esistono tre modi per acquisire i metadati:

1. i metadati sono contenuti nella risorsa medesima, come per esempio nella sezione HEAD di un documento HTML;

2. al momento del trasferimento della risorsa, le informazioni vengono trasferite dal server al client (GET) o dal client al server (PUT o POST);

3. i metadati vengono estratti da un'altra risorsa e quindi i metadati relativi ad un documento possono essere estratti dalla risorsa stessa, o da un'altra risorsa, o possono essere trasferiti con il documento.

La seconda conseguenza, del fatto che i metadati sono dati, è che i metadati possono essere descritti da altri metadati, e cosí via. I metadati consistono in asserzioni sui dati, le quali vengono quindi rappresentate sotto forma di un nome di asserzione e un insieme di parametri. L'assioma è che i metadati sono rappresentati da un insieme di asserzioni indipendenti.

Nel caso vi siano piú asserzioni relative alla stessa risorsa, l'asserzione risultante è quella ottenuta mediante l'AND logico delle due asserzioni. Le asserzioni relative alle risorse vengono spesso riferite come attributi della risorsa.

Eppure attualmente tutte le informazioni che sono sul Web sono *"machine readable"* ossia sono informazioni, NON elaborabili automaticamente, relative a una risorsa o oggetto web. Nel Semantic Web invece le informazioni devono essere *"machine understandable"* cioè informazioni elaborabili automaticamente, relative a una risorsa web o a qualche altra cosa. Nel Web abbiamo milioni di risorse e oggetti di tipo sconosciuto, tutte collegate tra di loro.

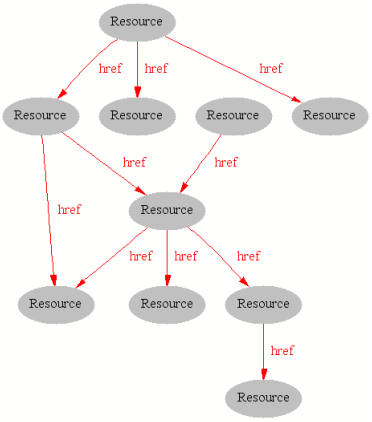

Figura 1.1: Il web era così ...

La figura sottostante ci fa vedere esattamente questo: mostra come nel web tutte le risorse siano collegate tra loro mediante link (rappresentati nella figura come frecce chiamate href). Ogni risorsa della rete è accessibile tramite questi link, passando attraverso oggetti intermedi presenti nel web; notando che partendo da un oggetto sorgente ci possono essere diversi percorsi alternativi per arrivare a un oggetto destinazione.
Invece, mediante il Semantic Web, noi creiamo una struttura basata su risorse e associazioni di tipo definito.

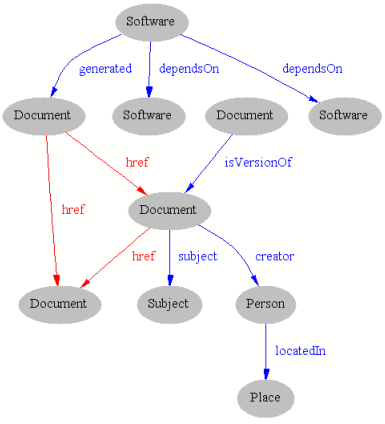

Figura 1.2: ... in futuro sarà così

Se osserviamo la figura possiamo chiarire meglio il concetto di Web Semantico: Il reperimento delle risorse può essere basato sulle loro proprietà, piuttosto che indicizzando tutte le parole presenti. In sostanza ogni risorsa rientra in una categoria semantica, la quale ha certe proprietá che possono definire e caratterizzare la risorsa stessa.

È utilizzando proprio queste proprietá delle risorse che noi possiamo creare una struttura dati, simile a quella presentata in figura 1.2, dove dall'oggetto sorgente passiamo all'oggetto destinazione, senza passare per tutti i campi intermedi (muovendosi tra decine di link creati dall'indicizzazione di ogni termine o parola) ma in modo agevole e veloce mediante pochi link diretti, che contraddistinguono una proprietá particolare di quel dato oggetto, che rientra in una data categoria semantica.

RDF è un linguaggio per descrivere il significato delle risorse. Il concetto di Semantic Web venne creato da Tim Berners-Lee.

Era il 1994, e si stava ancora costituendo il W3C quando Tim Berners Lee rifletteva già sul dare un senso compiuto a ogni informazione presente sul web. Nella storica conferenza internazionale del WWW al CERN di Ginevra, Tim Berners-Lee illustró la sua visione (molto condivisa) del futuro del web e del W3C, ed introdusse un nuovo concetto che a sua veduta sarebbe stato alle basi della diffusione di massa del web.

Immediatamente molte delle migliori menti del W3C compresero le potenzialitá di quel concetto e si misero all'opera indipendentemente da tutte le altre iniziative del W3C per poter concretizzare quella seconda geniale intuizione dell'ideatore del web. Questo lavoro si concretizzo nel 1997, quando in modo non del tutto ufficiale si costituí il piccolo gruppo di lavoro che inizió a lavorare sul metalinguaggio chiamato XML, che in seguito sarebbe stato alla base di tutti i futuri linguaggi del W3C. Chiaramente, il tutto rientrava perfettamente nella visione del semantic web e voleva creare l'infrastruttura progettuale (XML) che ne avrebbe in seguito permesso la creazione.

Nel 2001 Tim Berners Lee in un intervista a Scientific American[1] illustró chiaramente il futuro del web e conió il termine Semantic Web descrivendolo così: "Il Web Semantico è un estensione del web corrente, dove ogni informazioni è giá autodefinita: questo abilita la possibilitá che i computer e gli utenti lavorino in cooperazione".
Subito venne presentato ufficialmente il gruppo di lavoro[2] che si sarebbe occupato del Semantic Web e il W3C inizió ufficialmente a lavorare su questa futura visione del web.

[1] www.businessweek.com/bwdaily/dnflash/oct2004/nf20041022_6972_db083.htm
[2] www.w3.org/2001/sw/

1.1 Cos'è il Semantic Web ?

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation".
Tim Berners-Lee, abbiamo visto, che ha definito così il web semantico, ovvero "...è un estensione del web attuale in cui le informazioni sono strutturate con un senso compiuto, migliorando il lavoro tra le persone e i computer".

Nella pratica il web semantico è una sezione di lavoro del W3C[3] che si occupa di produrre documenti, soluzioni ed esperienze per consentire al prossimo web di acquisire un senso piú compiuto, contribuendo a fornire alle persone uno strumento piú completo, efficace, utile.

In realtà gli studi e le ricerche che il Consorzio Web sta portando avanti in questi anni potrà portare un serio e concreto beneficio a tutti i livelli informatici, non solo quelli basati su web. In sostanza è la realizzazione e utilizzo di procedure, modelli e linguaggi basati su XML come RDF, DAML+OIL, OWL e altri ancora.

Poiché in realtá il lavoro del W3C è piú ampio di come lo abbiamo prima descritto, il panorama possibile che l'utilizzo e l'applicazione concreta delle ontologie[4] e della semantica puó creare ed offrire è davvero ampio ed interessante.

Per usare un esempio concreto e alla portata di tutti vedremo come potrebbe evolversi il panorama dei motori di ricerca.

Se oggi pensiamo alle ricerche che quotidianamente facciamo e ai risultati che spesso sono ben lontani da quello che cerchiamo, riusciamo anche ad aver presente la difficoltá che troviamo nel reperire informazioni interessanti nel web.
Basta pensare ad una ricerca che contenga la parola "Milano" e i risultati possono essere molteplici:

- un portale su Milano;
- un sito che produce materiale Milanese;
- un sito che ha sede a Milano;
- un documento che parla di Milano;
- una persona che ha cognome "Milano" e una propria pagina personale.

Potremmo continuare e complicare maggiormente la cosa, ma giá questo ci fa capire che non sempre le informazioni che realmente cerchiamo vengano soddisfatte almeno al 60%, soprattutto quando la nostra ricerca contiene una parola molto comune.
Con il Semantic Web possiamo aggiungere alle nostre pagine un senso compiuto, un significato che va oltre le parole scritte, una "personalitá" che puó aiutare ogni motore di ricerca ad individuare ad esempio un "portale su Milano", scartando di fatto gli altri risultati che non soddisfano la nostra richiesta.

[3] www.w3c.org
[4] il termine ontologia, in filosofia, indica la teoria che studia quali tipi di cose esistono.

Capitolo 2

Le premesse del Semantic Web

La domanda è: se il Semantic Web è una estensione del Web tradizionale, quali sono i mezzi che dobbiamo usare per concretizzare questo sviluppo del Web ?

Dobbbiamo semplicemente fare ricorso alle nozioni e agli strumenti dell'intelligenza artificiale, questo è tutto ció che ci occorre.

2.1 I motori di Ricerca e il Web Semantic

Ogni giorno milioni di persone nel mondo accedono ai siti web per cercare ogni genere di informazioni. Per la presenza, nei server della rete delle reti (Internet), di miliardi di pagine web indicizzate attraverso la rete dei motori di ricerca, sta diventando complicato ottenere in breve tempo e senza ambiguità le informazioni cercate. Un italiano su due, ormai ha accesso a internet da casa. Qualcuno ha detto che su internet c'è tutto, basta saperlo cercare, ma spesso il problema è proprio questo. I siti Web sono decine, centinaia di milioni, ma non servirebbero a niente se alcuni di loro non funzionassero come giganteschi cataloghi, permettendo di trovare in poco tempo ciò che si cerca. Sono i motori di ricerca.

Questi particolari siti sono di due tipi: le directory e i motori di ricerca veri e propri. Le directory, come ad esempio Yahoo, raggruppano i siti per tipologie secondo una struttura "ad albero": l'utente può partire da una categoria molto generale (attualità, economia, sport, tempo livero, ecc.) e restringere via via la ricerca. I motori di ricerca veri e propri, invece, di cui google è l'esempio più noto, hanno una grafica molto semplice, con uno spazio in evidenza dove inserire la parola, o le parole attinenti al tema di interesse, le "parole chiave". In poche frazioni di secondo il motore fornisce un numero più o meno grande di link alle pagine Web che contengono quella parola. Gli archivi dei motori di ricerca di norma sono più ampi di quelli dell directory, e arrivano a dimensioni impressionanti: Google recentemente ha superato gli otto miliardi di pagine. Nonostante questo però, nessun motore potrà mai "schedare" l'intero Web.

Le Directories si basano su una logica completamente differente, in quanto dipendono unicamente dal lavoro di persone che valutano ed inseriscono i vari siti proposti in ogni categoria da ogni responsabile del sito. Ossia, in questo caso tutto parte dalla proposta del web master (o chi per lui) che chiede ad una Directory di valutare il proprio sito web e di inserirla in una categoria precedentemente indicata. Un sito Web viene inserito comunque in un Motore di Ricerca, mentre i responsabili delle Directories possono tranquillamente rifiutare l'inserimento di un sito. Esistono dei criteri in base ai quali un sito puo' essere rifiutato o meno.

Nel caso di DMOZ [1], per esempio, ogni categoria viene descritta in modo tale che si sappia quali devono essere le principali caratteristiche di ogni categoria.

[1] www.dmoz.org

Per quel che riguarda Yahoo![2] la situazione è diversa innanzitutto perchè non viene indicata la descrizione di ogni sezione ma si può vedere solo il cosiddetto "strillo", che evidenzia il percorso che porta alla sezione finale.

Attualmente esistono moltissimi Motori di Ricerca e alcune Directories importanti.

Quattro Motori controllano circa l'ottanta per cento del traffico online.

Si tratta dei seguenti Motori:

- Google (www.google.it);
- Fast (www.alltheweb.com);
- Altavista (www.altavista.it);
- Inktomi (www.inktomi.com).

Per quel che concerne le Directories la situazione ideale si raggiunge quando si è inseriti in: DMOZ e Yahoo!, poichè molti Motori si basano anche su queste Directory per "completare" il proprio database.

La tendenza per il futuro sta già tutta nell'analisi della situazione attuale, con quattro Motori di Ricerca attorno ai quali ruota l'ottanta per cento del traffico del Web. Nel tentativo di contrastare questa forza, i "competitors" minori si stanno accorpando sempre più spesso unendo i propri database.

Il settore dei motori di ricerca sta ora crescendo molto, tanto che diverse ricerche hanno indicato proprio uno di essi, e cioé Google, come sito Web più famoso al mondo. Nonostante questo, ormai sono rimasti solo tre concorrenti di spessore internazionale e con tecnologie proprio: Google, Yahoo! e Microsoft. A ulteriore dimostrazione di quanto siano importanti i search engine, Microsoft ha investito miliardi di dollari per realizzare un nuovo motore di ricerca per il suo portale Msn (Al momento MSN dichiara di aver indicizzato nel suo database 5 miliardi di pagine web, mentre Google ne ha al momento catalogato oltre 8 miliardi).

Altavista, il pioniere dei grandi motori di ricerca, e il norvegese Alltheweb, per anni tra le più innovative realtà del settore, sono stati comprati l'anno scorso da Yahoo!, che fornisce le sue tecnologie anche a Lycos ed Excite. Anche in Italia i portali più frequentati, come Virgilio, Libero, Tiscali, Supereva e Kataweb, offrono servizi di ricerca (e molti anche le directory), ma nessuno ha tecnologie proprie (tranne Virgilio per la parte di directory e Il Trovatore per i siti Web italiani).

Quello che sia il motore scelto quel che importa è che la ricerca sia precisa e veloce, per evitare di trovarsi migliaia di link, la maggior parte dei quali inutili. La scelta delle parole chiave è determinante. La prima regola è evitare di usare parole generiche: se per esempio siamo fan dei Police e cerchiamo il testo del loro brano "Every Breath You Take", mettendo i temini "Police" o "breath" su un motore di ricerca si ottengono milioni di link praticamente inutilizzabili.

Le cose da fare sono altre. Si può cercare tramite una directory, partendo dalla categoria "musica" o equivalenti, o dettagliare la ricerca inserendo accanto a "Police" una o più parole del titolo del brano. Un'altra strada è la funzione "trova la seguente frase", e inserire l'intero titolo del brano o una parte di una strofa del testo.

Altre funzioni molto diffuse in questi siti sono la possibilità di chiedere pagine solo in italiano (o in un 'altra lingua), o solo quelle aggiornate negli ultimi tre o sei mesi, o le "pagine simili", cioé siti non ancora visitati e correalati ai risultati precedenti.

[2] www.yahoo.it

In realtà però le possibilità sono innumerevoli: si possono cercare solo determinati tipi di file o richiedere la traduzione della pagina trovata, addirittura, se il link porta a una pagina non più esistente, cliccare "versione cache", che vi riproporrà quella pagina nello stato in cui era qundo è stata schedata dal motore.

O addirittura ci sono anche i "metamotori" (come Dogpile, Meta-Crawler o Mamma.com) che permettono di lanciare una ricerca contemporaneamente su diversi search engine; oppure può optare per un "motore specializzato".

La domanda quindi che possiamo farci ora è: qual'è il futuro di questo settore in crescita ?

Questo stralcio di un intervista[3] a Frank van Harmelen[4] ci fa intravedere l'avvenire e le prospettive dei motori di ricerca:

> "Facciamo un esempio: immaginate di stare cercando l'indirizzo di lavoro di Frank van Harmelen. Forse con Google non lo troverete, perchè sono stato troppo pigro per mettere quest'informazione sul mio sito Web. Ma se il computer sapesse che io lavoro per la Vrije Universiteit, e se potesse trovare l'indirizzo dell'università, allora potrebbe dedurre che questo è un indirizzo utile da fornire come risposta alla vostra domanda. Naturalmente, tutto ciò dipende dall'informazione di sfondo (la conoscenza) che abbiamo dato al computer. Dunque, in effetti, tutto dipenderà dalla qualità di quella che chiamiamo l'ontologia: una collezione di termini e di relazioni tra questi termini ...
>
> La sola cosa che si noterà navigando sul Web è che la qualità dei risultati che vengono restituiti dai motori di ricerca sarà molto migliore rispetto al passato. I motori di ricerca attuali funzionano molto bene per quanto riguarda il recall: trovano tutto quello che c'è da trovare. Ma non sono altrettanto soddisfacenti per quanto riguarda la precisione: oltre a trovare quello che stavate cercando, vi danno un sacco di cose in più che non volevate. Naturalmente sto un po' esagerando, ma il livello attuale di precisione potrebbe essere molto migliorato. E mi aspetterei anche un miglioramento del modo in cui l'informazione viene presentata all'utente. Per esempio, se io ora digito il mio nome Van Harmelen' in un motore di ricerca, ottengo due tipi di risultati: alcuni riguardano me e il mio lavoro scientifico, mentre altri riguardano il villaggio olandese Harmelen. Il problema è che il motore di ricerca non può distinguere tra i due, e semplicemente mette tutta l'informazione in un singolo grande elenco. Quando il Semantic Web sarà ultimato, il motore di ricerca sarà in grado di determinare che ci sono in realtà due tipi di risultati, e che questi dovrebbero essere mostrati separatamente; oppure, meglio ancora, dovrebbe chiedermi che cosa stavo cercando, se la persona Frank van Harmelen o il villaggio Harmelen."

> E poi aggiunge: "Ho sentito di recente alcune persone di Google e sono stato sorpreso nel vedere quanto fossero ... beh, direi garbatamente riluttanti. Conoscevano bene gli ultimi sviluppi, ma ci hanno detto che non stavano sfruttando attivamente la tecnologia. Ma nel contempo si può vedere che stanno sperimentando. Avete mai sentito parlare della Open Directory? un progetto dove migliaia di volontari classificano innumerevoli pagine Web secondo decine di migliaia di categorie. Ebbene, Google sta già collegando i suoi risultati di ricerca secondo la gerarchia tematica della Open Directory. Per molti risultati delle vostre ricerche troverete già un collegamento ipertestuale alla categoria nella quale quel risultato è classificato secondo nella Open Directory; questo ci dà la possibilità di andare a vedere le altre voci che appartengono a quella categoria.

[3] $http://www.cs.vu.nl/\sim frankh/$ (la traduzione dell'intervista è stata approntata da Margherita Benzi)

[4] AI Department, Division of Mathematics and Computer Science, Faculty of Sciences, Vrije Universiteit Amsterdam, de Boelelaan 1081a, 1081HV Amsterdam , The Nether-lands

Quindi, senza ammetterlo, stanno già usando la tecnologia del Semantic Web, ed è naturale che non possano permettersi di ignorarla: dopotutto la popolarità di un motore di ricerca è determinata soltanto dalla qualità dei risultati che esso produce."

2.2 L'architettura del Semantic Web

Il Semantic Web, nella visione di Berners-Lee, ha una architettura a livelli (si veda fig. 2.1).

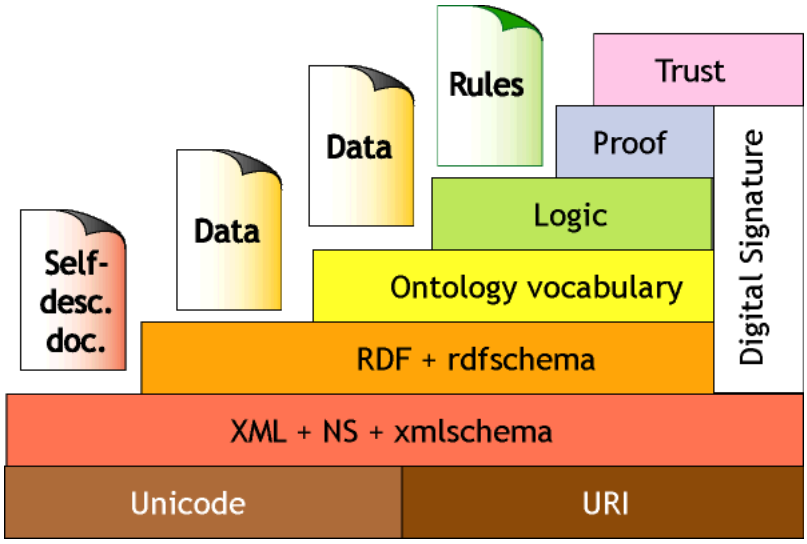

Figura 2.1: L'architettura del Semantic Web

Per chiarezza di terminologia, va ricordato che la filosofia di base del Web è quella di uno spazio informativo universale, navigabile, con un mapping da URI (Uniform Resource Identifier) alle risorse. Nel contesto del Semantic Web, il termine semantico assume la valenza di "elaborabile dalla macchina" e non intende fare riferimento alla semantica del linguaggio naturale e alle tecniche di intelligenza artificiale. Il Semantic Web è, come l'XML, un ambiente dichiarativo, in cui si specifica il significato dei dati, e non il modo in cui si intende utilizzarli. La semantica dei dati consiste nelle informazioni utili perchè la macchina possa utilizzarli nel modo corretto, eventualmente convertendoli.

Il Semantic Web potrà funzionare solo se le macchine potranno accedere ad un insieme strutturato di informazioni e a un insieme di regole di inferenza da utilizzare per il ragionamento automatico[5].

La sfida del semantic web, quindi, è fornire un linguaggio per esprimere dati e regole per ragionare sui dati, che consenta l'esportazione sul web delle regole da qualunque sistema di rappresentazione della conoscenza.

Esaminando più in dettaglio la figura 2.1, si può notare il ruolo di base giocato da XML (con Name Space e xmlschema), che consente di dare ai documenti una struttura arbitraria, mentre RDF può essere usato per esprimere il significato, asserendo che alcuni particolari elementi hanno delle proprietà (p.es. autore-di). Un terzo componente è l'Ontology Vocabulary (livello ontologico), inteso come il contenitore che definisce in modo formale le relazioni fra i termini. Una ontologia permette di descrivere le relazioni tra i tipi di elementi (per es. "questa è una proprietà transitiva"), senza però fornire informazioni su come utilizzare queste relazioni dal punto di vista computazionale.

Le ontologie possono svolgere un ruolo fondamentale nel migliorare il funzionamento del Web (ricerca di concetti, collegamento delle informazioni contenute in una pagina alle strutture di conoscenza associate, etc.). Il linguaggio definito dal W3C per definire ontologie strutturate, in architettura web, per consentire una migliore integrazione dei dati tra applicazioni in settori diversi è OWL (Ontology Web Language).

[5]Berners-Lee T., Hendler J., Lassila O.: The Semantic Web, Scientific American, May 2001, *http* : *//www.scientificamerican.com*/2001/0501*issue*/0501*bernerslee.html*

2.3 Il Semantic Web ed i suoi linguaggi

Figura 2.2: Tim Berners-Lee

Oggi il Web è essenzialmente un insieme di standard come l'HTTP (HyperText Transfer Protocol) e l'HTML (HyperText Markup Language), usati per trasmettere e ricevere documenti ipertestuali e multimediali adatti all'utilizzo attraverso l'intervento di un essere umano. Secondo la visione di Tim Berners-Lee, l'inventore del Web, questo aspetto limita di molto le sue potenzialità, con il Web Semantico, invece, si favorirà lo sviluppo di agenti software altamente specializzati [6].
Gli agenti dovrebbero perciò riuscire ad interpretare tutti quei documenti che sono scritti o indicizzati attraverso formalismi di rappresentazione espressi in linguaggi specifici di rappresentazione della conoscenza, che permettono di definire una concettualizzazione ovvero un'ontologia.

Tradizionalmente questi linguaggi sono stati sviluppati nell'area dell'Intelligenza artificiale e focalizzano la loro base formale sui paradigmi del calcolo dei predicati del primo e secondo ordine, la logica descrittiva o anche i paradigmi object oriented. Sono le diverse proprietà espressive e computazionali che differenziano poi questi linguaggi.

Il linguaggio XML, la cui prima bozza è del 1996, è un linguaggio di Markup pensato per scambiare in modo semplice documenti via web; consente a chiunque di progettare il proprio formato dei documenti e di scrivere poi in quel formato. Le componenti di XML sono: il contenuto (XML); le specifiche che riguardano gli elementi ossia la struttura (DTD, Document Type Definition o XMLSchema) e le specifiche che riguardano la visualizzazione ossia lo stile (XSL eXtensible Style Language).

L'XML permette la creazione di "marcatori' (tag) all'interno delle pagine Web in grado di definire una struttura coerente e non ambigua e che rifletta lo specifico dominio semantico del documento.
Marcare in una pagina web le parole "Giacomo Puccini" con il tag <compositore> permetterebbe già di fare un grosso passo in avanti ai motori di ricerca, che riuscirebbero a fornire documenti pertinenti, escludendo quelli in cui non si parla del famoso musicista ma di uno sconosciuto omonimo.

[6] Berners-Lee et al. 2001

I marcatori vengono definiti da appositi Data Type Definitions (DTDs)o da più specifici XML-Schema.

In primo luogo, XML viene utilizzato spesso nella rappresentazione di documenti testuali, anche di carattere letterario o genericamente umanistico, diffondendosi quindi in un settore nel quale la rappresentazione digitale delle informazioni costituisce in larga misura un elemento di novità. In secondo luogo, la versatilità di XML, nel generare definizioni di diverse tipologie di documento, produce la convinzione diffusa che tale linguaggio possa costituire una sorta di strumento "neutro" con il quale rappresentare la conoscenza contenuta nelle diverse fonti informative, dagli ordini di spedizione commerciali ai manoscritti latini tardo medievali.

L'RDF e gli RDF-Schema vengono invece usati per esplicitare le relazioni tra documenti e tra metadati, permettendo di compiere asserzioni che contribuiscono ad incrementare delle vere e proprie basi di conoscenza. Queste basi conterranno anche le relative regole per dedurla ed inferirla.

Tra quelli piú rilevanti citiamo Ontolingua, Loom, OCML e Flogic. Poiché XML è diventato "de facto" lo standard per lo scambio di dati tra le applicazioni, sono stati sviluppati linguaggi di rappresentazione della conoscenza per il Web che rendono possibile la pubblicazione delle ontologie usando la sintassi di questo standard. In questo modo, per quanto è possibile, si evita l'arduo compito di definire dei parser appositi. I linguaggi basati sull'XML sono candidati ideali per il Semantic Web. Alcuni di questi sono il Simple HTML Ontology Extensions (SHOE), l'Ontology Exchange Language (OML) e il Resource Description Framework Schema (RDFS).

Essi ereditano le caratteristiche dell'XML e incorporano nuove caratteristiche che migliorano l'espressività del modello dati iniziale. Ulteriori proposte di linguaggi estendono RDF e RDFS come: Ontology Interchange Language (OIL) e il suo successore (DAML+OIL). Visto che l'XML impone la necessità di restrizioni strutturali (una base comune), nell'idea di fornire metodi esenti da errore per le espressioni semantiche, RDF fornisce l'infrastruttura per permettere la codifica, il riuso e lo scambio di metadati strutturati.

In questo modo, l'RDF é il modello piú promettente per associare informazioni al contenuto delle risorse Web. Ed é improbabile che un singolo linguaggio possa coprire tutte le necessità e i requisiti presentati nel Semantic Web. Un'analisi delle caratteristiche maggiormente auspicate, riguardanti l'espressività e la potenza dei meccanismi di ragionamento, ci fornirà il profilo per un efficiente linguaggio per il Semantic Web. Viene fatta una chiara distinzione tra i termini: rappresentazione della conoscenza e il ragionamento.

La formalizzazione della conoscenza é portata avanti, in molti casi, attraverso l'uso di concetti, relazioni n-arie, funzioni, procedure, istanze, assiomi, regole di produzione ed attraverso la semantica formale. Questi fattori determinano l'espressività del linguaggio.

La tabella 1 (figura 2.3) mostra un confronto tra questi elementi; il simbolo +' indica un aspetto supportato dal linguaggio, il simbolo -' indica invece un aspetto non supportato, +/-' indica un aspetto non direttamente supportato (ma che può essere modellato con le risorse fornite dal linguaggio) e NA' si riferisce ad aspetti non supportati di piccola rilevanza. Come é possibile notare i linguaggi basati sull'XML non forniscano di solito la possibilità di definire funzioni, procedure e assiomi, eccetto qualche piccola assiomatizzazione come le regole deduttive nel caso di SHOE. Essi perdono anche la semantica formale inerente al linguaggio stesso. Tutto ci ò rende difficile l'implementazione di meccanismi efficienti di ragionamento.

In questo senso Ontolingua é forse il piú espressivo di tutti i formalismi presentati, sebbene attualmente non esista nessun motore inferenziale che l'implementi.

Altri interessanti confronti possono essere affrontati in relazione ai meccanismi di ragionamento che i linguaggi permettono. La tabella 2 (figura 2.4) considera questi aspetti. OIL ha un automatico sistema di classificazione (desiderabile nel caso delle ontologie), Flogic implementa il trattamento di eccezioni ed entrambi questi linguaggi presentano tipici sistemi di inferenza. Confrontati ai linguaggi basati sull'XML, i linguaggi tradizionali supportano l'implementazione di procedure, il mantenimento di restrizioni, ed entrambi le valutazioni top-down e bottom-up.

	Ontolingua	OCML	LOOM	FLOGIC	XOL	SHOE	RDF(S)	OIL
Concepts	+	+	+	+	+	+	+	+
n-ary Relations	+	+	+	+/–	–	+	+	+
Functions	+	+	+	+/–	–	–	–	+
Procedures	+	+	+	–	–	–	–	–
Instances	+	+	+	+	+	+	+	NA
Axioms	+	+	+	+	–	–	–	NA
Rules of Production	+	+	+	–	–	–	–	NA
Formal Semantics	+	+	+	+	+	–	–	–

Figura 2.3: Confronto tra linguaggi relativamente agli elementi per la rappresentazione della conoscenza, presentato in [Corcho/Gmez 2000].

In tal modo esiste un importante compromesso tra completezza ed espressività dei meccanismi di inferenza usati e la loro efficienza. Tutto ciò rende molto interessante lo studio e lo sviluppo di tecniche per la valutazione distribuita di programmi logici in questo contesto e le tecniche che forniscono supporto all'elaborazione delle interrogazioni basate sulle ontologie nel semantic web.

	ONTOLINGUA	OCML	LOOM	FLOGIC	XOL	SHOE	RDF(S)	OIL
Inference Mechanisms								
Correct	–	+	+	+	–	–	–	+
Complete	–	–	–	+	–	–	–	+
Classification								
Automatic Classification.	–	–	+	–	–	–	–	+
Exceptions								
Use of Exceptions	–	–	–	+	–	–	–	–
In heritance								
Monotonic	+	+	+	+	NA	+	NA	+
Non-Monotonic	+/–	+/–	+	+	NA	–	NA	–
Simple Inheritance	+	+	+	+	NA	+	+	+
Multiple Inheritance	+	+	+	+	NA	+	+	+
Procedures								
Implementation of Procedures	+	+	+	–	–	–	–	–
Restrictions								
Examination of Restrictions	+	+	+	+	–	–	–	–
Evaluation Model								
Top-Down	–	+	+	+	–	NA	–	–
Bottom-Up	–	+	+	+	–	NA	–	–

Figura 2.4: Meccanismi di ragionamento nei linguaggi, presentati in Corcho/Gmez 2000

2.4 La rappresentazione della conoscenza

Alla base del programma di rappresentazione della conoscenza nell'Intelligenza Artificiale vi è l'ipotesi "riflessiva" di McCarthy (1968): l'intelligenza dipende essenzialmente dal fatto di essere in grado di rappresentare con il linguaggio i fatti che riguardano situazioni, obiettivi e possibili azioni; in altre parole, per essere intelligente un sistema deve poter rappresentare le proprie operazioni e le proprie strutture. A queste ipotesi va aggiunto un principio che Minsky (1985) ha cosí, paradossalmente, riassunto: il modo piú efficiente di risolvere un problema è quello di sapere giá come risolverlo; la conoscenza che si vuole fornire alla macchina è conoscenza relativa a come risolvere problemi: quanto più essa è specifica per la soluzione di una certa classe di problemi, tanto piú è utile.

Apparentemente la struttura tramite cui gli esseri umani reperiscono, elaborano e comunicano conoscenza è il linguaggio. Anche se ció fosse vero, il linguaggio non potrebbe essere utilizzato da una teoria computazionale della conoscenza per almeno tre ragioni:

- le frasi del linguaggio presentano delle ambiguitá (per esempio, "punto g" ha due significati), e non è ancora chiaro come la mente risolva tali ambiguitá;
- né la sintassi né la semantica di alcun linguaggio umano sono state interamente determinate;
- occorrono risorse computazionali infinite per riuscire a trattare la varietá di forme consentita dal linguaggio naturale.

È pertanto necessario definire un nuovo "linguaggio" che non presenti i tre ostacoli di cui sopra, ovvero postulare che le strutture di rappresentazione della conoscenza non coincidano necessariamente con quelle del linguaggio naturale. È ovviamente infinito il numero di possibili convenzioni per rappresentare la conoscenza che soddisfano il requisito di essere:

- non ambigue;
- specificate;
- finite.

Soltanto la plausibilitá psicologica e l'adeguatezza computazionale possono aiutare a selezionare quelle piú opportune, nonostante Hector Castaneda abbia sostenuto (1964) che nessuna logica della conoscenza puó essere psicologicamente plausibile. La conoscenza è, innanzitutto, sempre relativa a "qualcosa": la conoscenza di "qualcosa" è la capacitá di formare un modello mentale di quella "cosa".

Questa definizione non identifica peró in maniera univoca la strutture che devono essere impiegate dalla mente: per "modello mentale" di un oggetto di conoscenza si puó intendere, per esempio, tanto una struttura che esprima le proprietá di tale oggetto, quanto uno struttura che determini le azioni che si possono compiere a fronte di tale oggetto, o ancora una struttura che ne sintetizzi la relazione con il resto del mondo; o, infine, una struttura che dichiari tutto ció che è "rilevante" (ai fini computazionali) di quell'oggetto.

Esistono tuttavia dei punti fermi su come la mente rappresenti la conoscenza, in una forma o nell'altra le teorie computazionali assumono che esistano strutture dinamiche ed adattative che:

- hanno origine dalle esperienze passate;
- vengono in qualche modo generalizzate, per valere in un numero piú ampio di casi;
- determinano l'azione da compiere a fronte di una nuova situazione.

Finora i programmi che simulano comportamenti intelligenti sono stati dotati di strutture che possono essere ricondotte a due tipi fondamentali: le formule logiche (approccio logistico) e gli schemata (approccio associativo).

Forse il primo tentativo scientifico di definire una struttura simbolica fu quello degli "schemata". Il concetto venne introdotto da Otto Selz e da Henry Head negli anni Venti, per rendere conto dei fenomeni scoperti durante esperimenti sul rapporto fra percezione e azione.

Head postuló l'esistenza nella memoria di una rappresentazione flessibile delle esperienze passate, una rappresentazione del passato tale per cui la mente potesse facilmente ricondurre una nuova situazione a una situazione nota e pertanto inferire l'azione da compiere nella nuova situazione sulla base dell'azione compiuta nell'occasione precedente.

Selz concepí lo schemata come un elenco di concetti che descrivono la situazione e ipotizzó che il problema di determinare l'azione da compiere a fronte di una nuova situazione consistesse nel formare un nuovo schemata con i concetti relativi a tale situazione e nel completare i concetti mancanti basandosi sugli schemata giá noti relativi a situazioni passate.[7]

Con l'avvento del modello a schemata sorse l'esigenza di qualificare meglio il processo di "comprensione": se comprendere significa, in ultima analisi, reperire gli schemata piú idonei ed applicarli alla situazione in corso, allora l'intera intelligenza puó essere identificata con la comprensione. Anche lo schemata di Jean Piaget[8] era la rappresentazione interna di una generalizzazione di situazione che si presta ad inferire per analogia come comportarsi in tutte le situazioni simili, e che determina a sua volta come verrá ricordata una nuova situazione.

Claparede, prima di Piaget, aveva notato nel 1911 come il riconoscimento di una scena richieda una "ricostruzione" a partire dagli oggetti familiari. Piaget osservó (1968) che un bambino riusciva addirittura a ricordare una configurazione di bastoncini con maggiore accuratezza dopo sei mesi che non dopo una settimana, e lo attribuí al fatto che, nel ricostruire il ricordo, il bambino poteva servirsi di un maggior numero di schemata.

Neisser (1976) pose l'accento sul carattere "anticipatorio" dell'elaborazione a schemi, giá sottolineato da Selz: uno schemata utilizzato a fronte di una situazione che si sta formando consente di "prevedere" come la situazione si completerá. Nel confrontare lo schemata alla situazione corrente "ci si aspetta" che alcuni eventi previsti nello schemata si verifichino.

L'approccio logistico deriva da Gottlob Frege (1848-1925) il quale definisce una logica che, eccetto alcune modifiche di notazione, é la logica del prim'ordine usata in gran parte dei sistemi odierni di rappresentazione della conoscenza.

Il metodo logistico rappresenta la conoscenza sfruttando l'idea che le leggi del pensiero siano le leggi della logica. Nel far ció assume che il linguaggio della logica soddisfi i due requisiti di McCarthy: ossia consenta di esprimere tutto ció che sappiamo esprimere e consenta di eseguire calcoli su ció che viene cosí espresso.

Nell'approccio logistico pertanto rappresentare la conoscenza di un dominio significa tradurla in un insieme di regole e fatti.
Utilizzando fatti e regole è possibile rappresentare della conoscenza in maniera del tutto separata dal proprio contesto. Soltanto quando il sistema di produzione andrá ad esaminarle per risolvere uno specifico problema in quel dominio, le varie regole verranno messe in relazione fra di loro e con la situazione corrente. Il principio che giustifica questa operazione è stato definito così da McCarthy: "la conoscenza necessaria ai programmi intelligenti puó essere espressa in una forma dichiarativa indipendente dall'uso che se ne fará".

In definitiva tramite i predicati (riveduti sotto forma di fatti e di regole) è pertanto possibile costruire una astrazione dello stato dell'universo in funzione di due grandezze elementari: oggetti e relazioni fra oggetti. Lo stato in un certo istante è espresso da una formula che elenca tutte le relazioni fra gli oggetti che compongono l'universo; ovvero è rappresentato da una congiunzione di fatti: per esempio, "l'auto è ferma, e io sono a bordo, e il cielo è sereno, e una signora vende fiori sul marciapiede, e due ragazzi stanno parlando, e...". Due stati del mondo in tempi diversi sono rappresentati da due formule che differiscono nelle formule atomiche della propria congiunzione.

[7] Frederic Bartlett - 1932
[8] psicologo svizzero, Neuchtel 1896 - Ginevra 1980

Notiamo infine che esiste una chiara analogia fra i linguaggi di programmazione dei computer e i linguaggi della logica matematica: entrambi sono linguaggi artificiali inventati per poter rappresentare il mondo ed eseguire ragionamenti su tale rappresentazione. McCarthy, inventando il primo linguaggio di programmazione che consentisse di elaborare simboli (il LISP), trasformó l'analogia in un'equivalenza.

2.4.1 Le Reti Semantiche

La rappresentazione tramite reti semantiche (termine coniato da Margaret Masterman) venne introdotta da Ross Quillian nel 1965 per simulare computazionalmente il modello "associativo" in cui la memoria umana organizza le informazioni semantiche. Quillian era alla ricerca della soluzione a uno dei problemi piú vecchi e complessi dell'Intelligenza Artificiale: come venga ricordato il significato di una parola. La soluzione che propose è assai semplice: si ricorda il significato di una parola associando a quella parola tutte le altre parole che in qualche modo servono a definire il suo significato. Il significato non sarebbe pertanto qualcosa di assoluto, ma sempre qualcosa di relativo agli altri significati disponibili.

La scomposizione del significato delle parole in componenti minimi si allaccia al problema dell'ontologia di tali componenti, cioè se si tratti di espedienti descrittivi o di proprietà effettivamente presenti o in natura o nel sistema cognitivo dei parlanti. La psicologia cognitiva e sperimentale hanno un interesse particolare per il modo in cui i concetti e le relative proprietà vengono memorizzati. Nel 1968 lo psicologo Ross-Quillian formuló un'ipotesi di struttura di memoria, detta rete semantica, che doveva rendere conto sia della memorizzazione che delle operazioni di accesso. Una rete semantica è l'espressione di relazione tra concetti con le relative proprietà.

Così, ad esempio, concetti come animale, uccello, canarino si rappresentano come segue:

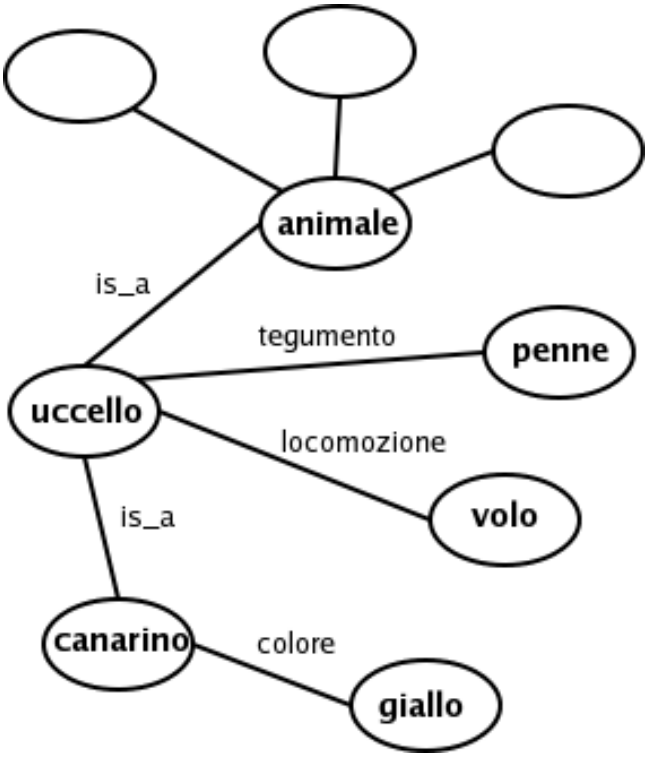

Figura 2.5: Esempio di rappresentazione di relazione in una rete semantica

che indica una relazione di iponimia di uccello rispetto a animale e di canarino rispetto a uccello.

La relazione is_a non indica, peró, solo l'iponimia, ma anche l'eredità da parte dell'iponimo delle proprietà associate all'iperonimo. Così, in virtù di questa linea di ereditarietà, canarino eredita da uccello le proprietà tegumento: penne e locomozione: volo. Questa distribuzione delle proprietà, sempre associata al concetto più alto, cioè più generale, nella gerarchia, costituirebbe una forma di economia della memoria, in quanto evita di memorizzare tutte le proprietà in associazione con tutti i concetti.

Inoltre spiegherebbe il fatto, comprovato sperimentalmente, che occorre più tempo per dichiarare vera l'associazione tra un iponimo e la proprietà di un iperonimo, che non una sua proprietà; così occorre più tempo a verificare (dire se è vera o falsa) la frase "un canarino vola" che la frase "un canarino è giallo". Questo perché il cammino, in termini di connessioni, è più lungo da canarino a volo che da canarino a giallo.

Si ricordi, in ogni caso, che i dati sperimentali in psicologia sono sempre dubitabili. Un'altra caratteristica delle reti semantiche è che non esiste un insieme chiuso di proprietà. A parte il legame is_a, che è caratteristico, ma non indispensabile, tutti gli altri sono puramente convenzionali e possono essere ampliati a piacere, pur di creare l'etichetta opportuna sul legame e di collegarlo con il concetto relativo.

Così a canarino posso aggiungere verso: canto, orgine: Canarie, dimora: gabbia, ecc. in modo da caratterizzare un canarino nella nostra esperienza urbana, piuttosto che il canarino che vive in libertà. Questo significa che le reti semantiche possono essere usate per rappresentare ogni sorta di proprietà di concetti, anche quelle che scaturiscono da esperienze individuali.

Le reti semantiche costituiscono un formalismo per rappresentare concetti e proprietà, ma non impongono alcuna restrizione su che cosa rappresentare. Non si puó, quindi, chiedere alle reti semantiche una risposta su quale sia il significato di un dato elemento lessicale, ma occorre definire tale significato con metodi autonomi, prima di tradurlo in termini di reti semantiche. Con ció la distinzione tra denotazione e connotazione[9] perde significato, in quanto tutto è rappresentato uniformemente. Un problema particolare è posto proprio dalla rigidezza dell'ereditarietà. Se, infatti, accanto a canarino avessimo pinguino questo erediterebbe la proprietà di volare, fatto che non è vero.

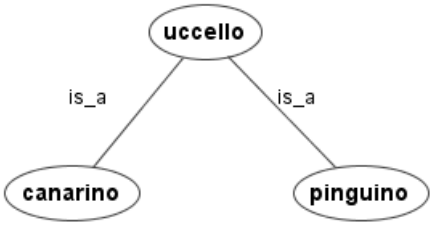

Figura 2.6: Esempio di ereditarietà

La soluzione, che consisterebbe nel degradare la proprietà locomozione: volo a tutti gli iponimi di uccello eccetto pinguino, non è praticabile perché annulla un'importante generalizzazione ("tutti gli uccelli volano"). D'altro canto, anche la frase "questo canarino non puó volare perché gli sono state tarpate le ali" è in contradizione con l'ereditarietà delle proprietà. In termini più astratti le reti semantiche, come pure le definizioni universali ("tutti gli uccelli volano", "tutti gli uccelli hanno penne"), non dispongono di meccanismi per il trattamento dell'eccezione (il pinguino) e la proprietà transitoria (ali tarpate).

[9] In semiologia il termine polisemia indica le varietà di significati di un termine, denotazione designa il significato primario oggettivo (ad esempio volpe = animale) mentre connotazione esprime il significato metaforico, esteso, indiretto, polisemico di una parola (ad esempio volpe = persona scaltra); un testo letterario ha un livello denotativo (significato letterale) e un livello connotativo (uno o più ulteriori livelli di lettura)

2.4.2 Dopo Quillian

Nel modello di memoria associativa di Quillian erano presenti l'idea di molte delle caratteristiche che diventeranno tipiche dei sistemi a rete semantica successivi. Ad esempio, gli archi che rappresentano relazioni di sottoclasse fra concetti diventeranno un costrutto fondamentale di pressochè tutte le reti semantiche. Essi sono gli antesignani di quelli che in seguito verranno chiamati di volta in volta archi isa (dove isa sta per "is a", vale a dire "è un"), archi ako ("a kind of", cioè : "un tipo di"), o archi superc, ossia archi di superconcetto.

Nel modello di Quillian è in un certo senso già implicito il principio in base a cui, mediante tali archi, si possa descrivere un concetto facendo riferimento a concetti di portata più generale. In vari lavori successivi Quillian ha ulteriormente perfezionato il modello che abbiamo presentato. Ad esempio, in Collins e Quillian (1970) viene presentato un tipo di rete che puó essere considerato un antesignano diretto di gran parte dei sistemi successivi. In quel lavoro emerge chiaramente il ruolo privilegiato degli archi di tipo superset. Essi strutturano la rete come una gerarchia di concetti.

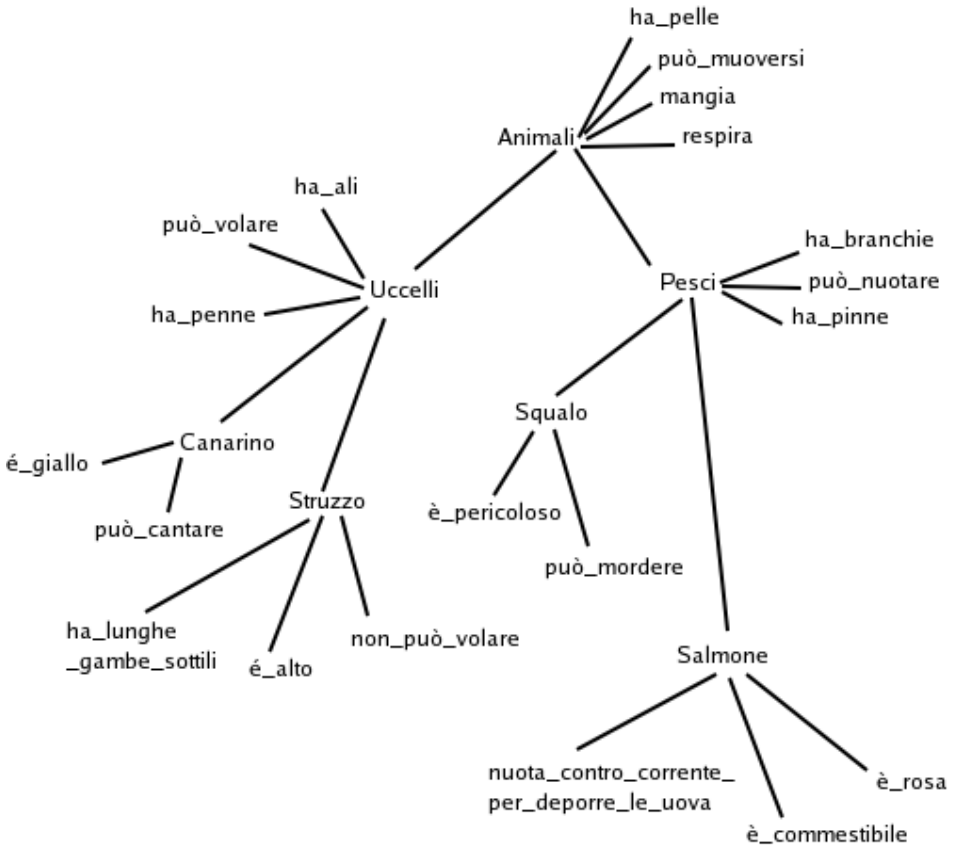

Figura 2.7: Esempio di gerarchia di concetti

Ad ogni nodo è associato un insieme di proprietà che caratterizzano il concetto corrispondente (si veda fig. 2.5 a pag. 24[10]). Ad esempio, Uccello è descritto come sottoclasse di Animale, e gli sono attribuite le proprietà ha_ali, può_volare e ha_penne. Le proprietà più generali sono introdotte ai livelli più alti della tassonomia, e diventa esplicito il meccanismo dell'ereditarietà, in base al quale, di norma, un concetto eredita le proprietà associate ai suoi superconcetti (ad esempio, Uccello, in quanto sottoconcetto di Animale, eredita ha_pelle, e così via).

[10] Collins e Quillian 1970

Viene inoltre introdotto esplicitamente il meccanismo per cui proprietà definite a livelli più specifici possono "cancellare" proprietà incompatibili che dovrebbero essere ereditate dai livelli superiori. Ad esempio, il concetto Uccello ha associata la proprietà volo, mentre per Struzzo, che è sottoclasse di Uccello, vale invece la proprietà non_può_volare.

Carbonell (1970) utilizza una rappresentazione a reti semantiche simile a quella di Quillian per la base di conoscenza di SCHOLAR, un programma per la didattica assistita dal calcolatore. Le reti utilizzate da Carbonell presentano alcune importanti innovazioni; ad esempio, compare la distinzione fra concept units e example units, cioè fra nodi che rappresentano concetti generali e nodi che rappresentano istanze specifiche, ossia singoli individui.

Un concetto può essere appreso in tre modi fondamentali:

- Induzione: generando una regola da un insieme di esempi pratici del concetto (Patrick Winston, 1970);
- Deduzione: costruendo la regola sulla base di una descrizione del concetto (Gerlad DeJong, 1981);
- Abduzione: trasferendo alla regola le proprietà di un concetto analogo (Patrick Winston, 1979).

Un altro dei primi sistemi a rete semantica è legato proprio alle ricerche di Winston (1970, 1975) sull'apprendimento automatico e fa riferimento al primo dei tre modi appena menzionati per l'apprendimento. Lo scopo di Winston era sviluppare programmi in grado di apprendere in base ad esempi semplici generalizzazioni in un mondo di blocchi. Il problema di apprendere a riconoscere strutture, in particolare archi, in un ambiente formato da blocchi fu introdotto come caso di studio da Winston (1975) ed è diventato un classico nella spiegazione dei meccanismi coinvolti nell'apprendimento. Come strumento di rappresentazione della conoscenza, veniva utilizzato un sistema di rappresentazione a rete che presenta caratteristiche comuni a molti sistemi successivi.

Nelle reti di Winston, alcuni nodi rappresentano oggetti, o classi di oggetti, mentre altri nodi rappresentano relazioni (ad esempio il nodo left of). Di conseguenza, anche gli archi hanno funzioni diverse: alcuni rappresentano relazioni fra oggetti (ad esempio, l'arco one part is), altri servono semplicemente a collegare nodi che rappresentano oggetti o classi a nodi che rappresentano relazioni. Non sempre la notazione di Winston è del tutto coerente, e puó accadere che la stessa relazione compaia talvolta rappresentata come arco, talvolta come nodo. Inoltre, la notazione non distingue tra relazioni di portata logica generale, quale quella di superconcetto, e relazioni specifiche dipendenti dal dominio, rendendo in questo modo impossibile, ad esempio, un trattamento soddisfacente del meccanismo di ereditarietà.

In questo, anche il lavoro di Winston è sintomatico dei limiti dei primi sistemi a rete semantica, limiti che verranno in seguito messi in luce da W. Woods. Un ulteriore linea di sviluppo nella storia delle reti semantiche prende l'avvio dal lavoro di Filmore (1968) sulle strutture a casi in linguistica. In questo tipo di sistemi le reti semantiche vengono usate principalmente per rappresentare il significato di verbi e di enunciati linguistici.

2.4.3 Le prime applicazioni del Semantic Web: i LOM

L'esigenza di rendere riutilizzabile, accessibile ed interoperabile la conoscenza codificata nei diversi sistemi di e-learning, ha portato a riconsiderare le modalità con le quali i contenuti per l'apprendimento vengono creati su sistemi software proprietari, inutilizzabili cioè in altri contesti. In un futuro prossimo lo studente o il docente potrebbero ad esempio muoversi liberamente e selezionare risorse educative all'interno di più sistemi differenti.

In una prospettiva di questo tipo, lo scambio, l'integrazione e la costruzione collaborativa di conoscenza si potranno attuare creando degli ampi "repositori di conoscenze" a cui si potrà accedere in modo rapido ed intuitivo attraverso interfacce e software adeguati. Esistono già dei software come Edutella [11], in grado di gestire servizi peer-to-peer per lo scambio di risorse educative on-line.

Per raggiungere questi obbiettivi è stato necessario ripensare ai modelli utilizzati per la rappresentazione della conoscenza e di descrizione dei contenuti, il che ha portato alla definizione di oggetto di apprendimento o "learning object" (LO).
Essi sembrano appunto offrire una soluzione a questi problemi sia dal punto di vista degli utenti che degli sviluppatori: per gli utenti in quanto possono offrire una modalità adattiva (adaptive) per la creazione di courseware "su misura" in base ai bisogni e agli stili di apprendimento propri di ciascuno; per gli autori in quanto soddisfano le esigenze di condivisione e riutilizzo delle risorse, facilità di aggiornamento, risparmio di tempo e di costi [12]. Un learning Object può essere definito come un modulo composto da una piccola unità di apprendimento su di un argomento specifico completa di materiale didattico e di una parte dedicata alla valutazione dell'apprendimento, completa di esercizi e di soluzioni. Il modulo così creato può poi essere usato in molti courseware diversi. In questa prospettiva i LO sono intesi come elementi modulari da utilizzare liberamente prelevandoli da appositi "Learning Object Repositories".

Esistono in già molti "repositori" su web dove è possibile recuperare learning objects, come ad es. il sito MERLOT (Multimedia Educational Resource for Learning and Online Teaching) [13]. Attraverso un'interfaccia di ricerca il sistema propone i LO che soddisfano gli specifici requisiti inseriti dagli utenti.

Questo del recupero efficiente degli oggetti di apprendimento è un problema molto sentito perchè vicino al problema della ricerca di documenti attraverso i motori e gli indici di Rete. Cercare informazioni in Internet comporta un altissimo tasso di "rumore", cioè di documenti non pertinenti.

Ed è per questo che fin da subito, si è stabilito i LO dovessero essere dotati di un sistema di classificazione il più possibile completo ed efficiente: i metadati, o meglio i LOM (Learning Objects Metadata) appunto. Un aspetto interessante è che i LOM sono scritti in XML: il nuovo linguaggio a marcatura che dovrebbe aggiungere della semantica alle pagine Web e che costituirà l'ossatura del cosiddetto Semantic Web.

A differenza di una scheda bibliografica, i LOM devono riuscire a comunicare oltre che i riferimenti essenziali, anche tutta una serie di informazioni utili per le potenziali applicazioni educative dell'oggetto di apprendimento in questione. Un aspetto interessante è che i LOM sono scritti in XML, e a loro volta, anche i Learning Object stessi possono essere "scritti" in questo linguaggio con notevoli vantaggi in termini di flessibilità nella gestione del contenuto.

Esempi di learning objects sono contenuti multimediali, contenuti didattici, software didattico e strumenti software ai quali ci si riferisce durante l'apprendimento basato sul computer. In senso più ampio, i learning objects possono includere obiettivi d'apprendimento, persone, organizzazioni, o eventi. Lo standard LOM si focalizza sul minimo insieme di propriet à necessario per la gestione, l'allocazione e la valutazione dei learning object. Lo standard permette la possibilità di estendere localmente il set minimo di proprietà per applicazioni particolari.

[11] http://edutella.jxta.org
[12] Petrucco, 2002
[13] http://www.merlot.org

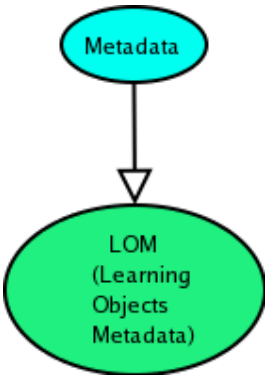

Figura 2.8: I LOM come metadati

Proprietà rilevanti dei learning objects includono il tipo di oggetto, l'autore, il proprietario legale, i termini di distribuzione e il formato. Dove applicabile, i LOM possono includere proprietà pedagogiche, come lo stile di insegnamento o di interazione, il livello di apprendimento, il livello e i prerequisiti. Ogni learning object può avere più di una descrizione.

Lo standard può includere attributi circa la sicurezza, la privacy, il commercio e la valutazione, ma solo nella misura in cui i campi dei metadata siano forniti per specificare caratteristiche descrittive legate a queste aree. Allo standard non competono i metodi di realizzazione di queste caratteristiche. Lo standard LOM fa riferimento agli esistenti standard aperti e ai lavori in corso nelle relative aree.

Questo standard si occupa solo della conformità di una istanza di Learning Object Metadata, non della conformità di applicazioni che processano tale istanze, come generatori (creano istanze di metadata), ritrasmettitori (accettano e ritrasmettono istanze di metadata) o lettori (accettano istanze di metadata, ma non necessariamente le processano). Un'istanza Learning Object Metadata può essere passata fra queste applicazioni.

2.4.4 Un ulteriore passo verso il Web Semantico: il ruolo del linguaggio RDF

Ma la descrizione a marcatori dell'XML non è sufficiente a convogliare una semantica che esprima relazioni fra concetti. Con esso è possibile formalizzare la conoscenza attraverso espressioni del tipo "soggetto-predicato-oggetto" i cui termini e restrizioni operative sono definite in apposite strutture chiamate RDF-Schema. Potremmo ad esempio asserire che "Giacomo Puccini è il compositore della Tosca" e nello schema RDF relativo vi saranno asserzioni che specificano cosa significa "essere un compositore" e che ne limitano l'applicazione ad un campo semantico ristretto che è quello delle "opere teatrali". In sostanza mentre l'XML aiuta a dichiarare una struttura di conoscenza, l'RDF aggiunge ad essa l'aspetto relazionale. Con RDF, ogni oggetto descritto è una risorsa identificata in modo univoco su Web, attraverso la sua URL o URI cioè degli indirizzi in grado di localizzare esattamente e senza ambiguità una ed una sola risorsa. Ad esempio una asserzione del tipo: "William Gibson è l'autore di Neuromante" può anche essere rappresentata in RDF sotto forma di grafo orientato, un sistema di rappresentazione molto vicino a quello delle reti semantiche e delle mappe concettuali (vedi figura).
Sembra evidente quindi, come il Semantic Web si proponga quale formalismo universale e standardizzato di rappresentazione della conoscenza attraverso la Rete.

```
<?xml version="1.0"?>
<biblioteca>
    <libri>
        <libro collocazione="1">
            <titolo>Neuromante</titolo>
            <autori>1</autori>
        </libro>
        <libro collocazione="2">
            <titolo>Il talismano</titolo>
            <autori>2</autori>
            <autori>3</autori>
        </libro>
    </libri>
    <autori>
        <autore codAutore="1">
            <nome>William</nome>
            <cognome>Gibson</cognome>
        </autore>
        <autore codAutore="2">
            <nome>Stephen</nome>
            <cognome>King</cognome>
        </autore>
        <autore codAutore="3">
            <nome>Peter</nome>
            <cognome>Straub</cognome>
        </autore>
    </autori>
</biblioteca>
```

Sopra abbiamo riportato un breve frammento di XML, il linguaggio che sta alla base di tutti gli altri linguaggi per la definizione dei metadati.

2.4.5 Le ontologie

L'RDF, pur essendo un linguaggio potente per la rappresentazione della conoscenza, non ha di per sè, alcun modo di operare inferenze o deduzioni. Bisogna rifarsi ad un ulteriore livello che riesca ad associare i concetti a regole logiche d'uso attraverso software specifici: questo rappresentato dalle ontologie. Esse sono considerate l'ultimo scalino verso il web semantico. Un'ontologia [Guarino, 1995] è una esplicitazione formale di un determinato campo del sapere umano e nella comunità dei ricercatori che si occupano del Semantic Web è intesa come base di conoscenza progettata con l'obiettivo di essere condivisa e riutilizzabile concretamente nel mondo reale. Ed è prevedibile che in una prima fase, le ontologie divengano sempre più importanti nei sistemi di gestione aziendali, per poi essere utilizzate in ambiti generici per esempio come le sofisticate interfacce dei motori di ricerca o nei sistemi di e-learning.

Anche i contesti scolastici più tradizionali saranno coinvolti poiché l'utilizzo delle ontologie nei software didattici permetterà di condividere e analizzare facilmente le strutture della conoscenza di un determinato dominio disciplinare del sapere, formalizzando appositi vocabolari condivisi. Un'ontologia può essere infatti definita anche come un vocabolario di termini con precise definizioni e un insieme di assiomi formali relativi alla loro interpretazione e al loro uso.

Per dare un'intuizione di che cosa un'ontologia aggiunga a un linguaggio di annotazione, si può pensare alla differenza esistente tra un semplice elenco di parole italiane e un dizionario concettuale. Il primo ci dice solo quali parole è lecito usare, il secondo fornisce anche un insieme di relazioni logico-grammaticali con altri termini (per esempio, che ci dica che Carlo è un professore, che professore è un ruolo accademico e sociale, che ogni professore afferisce a un Dipartimento accademico, che i Dipartimenti sono strutture organizzative delle Università, che l'Università è un tipo di istituzione, ecc.). Le ontologie pensate per il Web hanno spesso la forma di tassonomie corredate di un insieme di regole di inferenza, e forniscono definizioni formali delle relazioni tra termini.
Le ontologie rappresentano la base concettuale sulla quale dei software possono operare delle inferenze: tornando all'esempio di Giacomo Puccini, un agente intelligente che cerchi sul Web informazioni sulle sue opere, potrebbe scoprire che un documento che parla della famosa "aria del Nessun Dorma" si riferisce proprio ad un suo melodramma e non all'aria che respiriamo. Questo grazie ad un'estensione del concetto di "opera" presente nell'ontologia di dominio che sta consultando. Wordnet [14] è forse la più nota ontologia esistente: sviluppata dal laboratorio di Scienze Cognitive della Princeton University, tenta di descrivere i concetti (sinonimi, contrari e relazioni tra concetti) contenuti nella lingua inglese. WordNet si configura come un sistema incrociato di riferimenti lessicali basato sulle più recenti teorie psicolinguistiche della memoria lessicale umana. Partito con la lingua inglese, Wordnet conta ormai la sua estensione a molte altre lingue, compreso l'Italiano. Nomi, verbi, aggettivi sono organizzati in insiemi di sinonimi, ognuno dei quali rappresenta un concetto lessicale sottostante.

Se ad esempio consultiamo Wordnet sui significati del termine "opera" otteniamo due definizioni, in realtà non molto diverse da quelle che potremmo trovare all'interno di un dizionario qualunque. Ma ció che rende unico il sistema, è la possibilità di effettuare ricerche di "dipendenza concettuale".
Come ad esempio le relazioni di tipo meronimico (parte-tutto), ricchissime di implicazioni semantiche relative alla rete di associazioni concettuali che il concetto di opera, esplica, venendo inteso sia nel senso di "opera teatrale" che nel senso di luogo fisico dove si tengono gli spettacoli.

[14] http://www.cogsci.princeton.edu/ wn/

2.5 Agenti software

Gli agenti software sono programmi che dovrebbero sfruttare la conoscenza contenuta nei metadati per fornire servizi complessi agli utenti del Semantic Web.

Per far ciò, gli agenti devono essere dotati di un buon livello di autonomia (non possono rivolgersi continuamente all'utente per decidere cosa fare) e di capacità di ragionare in modo complesso sull'informazione a cui hanno accesso.

Questo significa al minimo essere in grado di rappresentare gli obiettivi di un certo utente, di mettere in atto una sequenza di azioni che possa soddisfare tali obiettivi, ed eventualmente di cooperare con altri agenti per ottenere tale risultato.

Naturalmente, ognuno dei punti elencati comporta l'emergere di molti problemi, alcuni dei quali di difficile soluzione. Il più generale di tutti ha a che fare con un concetto solo apparentemente tecnico, vale a dire in concetto di interoperabilità. Interoperabiltà significa non solo l'abilità di programmi progettati indipendentemente e implementati su piattaforme diverse di scambiarsi dati (cosa che già oggi in parte avviene), ma soprattutto di cooperare tra di loro sulla base di una "comprensione" del significato dei dati oggetto di scambio (in questo senso si parla di interoperabilit 'a semantica).

In altre parole, ciò richiede che i programmi per il Semantic Web (e in particolare gli agenti software) debbano "capirsi" senza presupporre un accordo a priori su cosa significhi un certo dato (detto altrimenti, due agenti diversi potrebbero non condividere la rappresentazione del mondo in base a cui agiscono), e su quale uso verrà fatto di un certo dato.

Inoltre, agenti molto evoluti dovrebbero anche essere capaci di dedurre, a partire dall'analisi di un documento e grazie a processi inferenziali, nuovi metadati non presenti in esso, di acquisire nuove "conoscenze e capacità" quando vengono in contatto con nuove ontologie e di ricavare informazioni importanti anche da servizi che rispondano solo parzialmente alle richieste dell'agente stesso.

In aggiunta, ci si aspetta che la cooperazione tra questi agenti avvenga anche in base a una capacità di ragionare sulle intenzioni degli utenti umani e degli altri agenti software, e nel rispetto di norme di tipo etico e legale.

Capitolo 3

Motivazioni & obiettivi

3.1 Motivazioni

Come a molte persone con cui abbiamo provato a discutere di questo argomento, quasi tutti, inizialmente, guardandoci dubbiosi ci chiedevano: "Web Semantic, e che cos'è ? di cosa si tratta ?" e noi stessi eravamo titubanti inizialmente a valutare tra le varie proposte una tesi che avesse come oggetto una "tecnologia" relativamente nuova (anche se molte delle sue idee base affondano le radici nel filone dell'intelligenza artificiale e del web) come il Web Semantic.

Il nostro relatore ci presenta l'idea e i concetti base e ci parla di un esempio, ormai divenuto l'esempio standard per tutti coloro che iniziano a studiare questa materia: un prototipo piuttosto completo conosciuto comunemente su Web come Wine (vedi paragrafo 3.2).

Guardiamo l'esempio e cerchiamo di capire come funziona e tutte le implicazioni a cui esso porta. Inizialmente pensiamo: "sembra un comunissimo motore di ricerca su web, per parola chiave, come goggle, solo che è limitato alla categorizzazione e indicizzazione dei vini ..."; ma poi continuando ad analizzare e dopo aver esaminato con attenzione questo esempio, uno dei pochi completi disponibili su internet, scorgiamo con sempre maggior interesse, le caratteristiche che ci hanno fatto interessare a questo progetto.

Questa tesi non ha destato in noi la nostra attenzione solo perchè rientra nel filone informatico, inerente alle tecnologie ipermediali legate al web, ma proprio in virtù del fattore di novità che il progetto stesso rappresenta e che in un prossimo futuro i motori di ricerca su web, assorbiranno in maniera più o meno diretta. Quelle stesse caratteristiche che quando vennero illustrate da Tim Berners-Lee al convegno della W3C suscitò e creò un ondata d'entusiamo in relazione al nuovo progetto che si potrebbe definire incredibile. Quante volte quando noi effettuiamo una ricerca su Internet ci capita di trovarci di fronte una serie smisurata di indirizzi di siti che sono poco pertinenti o che assolutamente non c'entrano nulla con l'argomento che cercavamo o che solo in senso lato ci interessano, perchè trattano l'argomento che ci interessa solo in modo marginale? Uno dei problemi pi comuni che i navigatori del Web si trovano a dover affrontare quasi quotidianamente è la ricerca di informazioni in maniera accurata. Proprio il fatto che il Semantic Web con le sue innovazioni permette di scremare in qualche modo tutti i risultati, facendo si di ottenere di fatto, ricerche più attinenti con quel che cercavamo la rende oltremodo interessante per tutti coloro che usufruiscono quotidianamente e non dei servizi che la rete offre. Con il crescere del numero di pagine pubblicate sul Web, i classici motori di ricerca cominciano a diventare sempre meno idonei nella ricerca di informazioni. Sarà sicuramente capitato più volte di cercare informazioni su un determinato argomento, ad esempio sulla coltivazione dei "girasoli", e di ottenere come risultato una serie di link a pagine che hanno a che fare con la vendita diretta di semi di girasole, a pagine pubblicitarie su una determinata marca di olio di girasole, alla home page del signor "Girasoli" o a quella dell'hotel "Soli e Girasoli" e cosí via.

Il fatto è che i motori di ricerca classificano le informazioni contenute in una pagina Web basandosi sulle parole in essa contenute. Quindi la presenza del termine "girasoli" all'interno di una pagina web fa s che questa sia candidata ad essere proposta a chi effettui una ricerca

in base a questo termine. Per consentire una ricerca più accurata occorrerebbe classificare le pagine Web ricorrendo a descrizioni del contenuto delle pagine stesse, tramite quelli che tecnicamente vengono chiamati metadati.

Nel Semantic Web, invece, si vuole ottenere maggiore flessibilità, a scapito della completezza delle risposte che si possono ricavare, esattamente come nel Web tradizionale la mancanza di struttura e di controllo centralizzato il prezzo da pagare per avere ampia disponibilit‘a delle informazioni. Aggiungere la semantica ai contenuti del Web richiede la creazione di un linguaggio che permetta di esprimere dati e regole per i ragionamenti: tale linguaggio deve permettere che le regole proprie di un sistema di rappresentazione della conoscenza possano essere utilizzate sul Web.

Naturalmente, per la natura stessa del Web, è possibile che dati concettualmente diversi, siano rappresentati con lo stesso nome: un programma dovrebbe essere in grado di distinguere la situazione e quindi elaborare le informazioni in maniera appropriata. Perché questo possa avvenire, si devono creare delle “ontologie”.

Il termine ontologia, preso in prestito dal linguaggio filosofico, indica un documento condiviso, che contiene la descrizione formale dei concetti di un dato dominio; identifica le classi più importanti, le organizza in una gerarchia, specifica le loro proprietà (che caratterizzano anche gli oggetti appartenenti alla classe) e descrive anche le relazioni più significative, che legano queste classi. Le convenzioni usate per presentare queste descrizioni, vanno dal linguaggio naturale a formalismi logici, ma è chiaro che la regolarità e una specifica formale facilitano la comprensione via software.

In definitiva, la struttura dei dati e la semantica introdotta dalle ontologie migliorano le potenzialità del Web: i programmi di ricerca, basandosi su un preciso concetto, cercano e trovano le pagine che effettivamente si riferiscono a quel concetto, anzichè quelle che contengono parola-chiave ambigue o generiche; in questo modo la ricerca è più accurata: i programmi di ricerca, che nel Web tradizionale riportano una serie di pagine tra le quali l’utente deve ulteriormente cercare quelle d’interesse, con queste nuove funzionalità danno le pagine che si riferiscono a un preciso concetto; inoltre le interrogazioni possono riguardare informazioni che non risiedono sulla stessa paginaWeb: il programma di ricerca, inferendo sulle regole specificate, può individuare il dato richiesto e rispondere all’interrogazione.

Le ontologie consentono anche di migliorare le prestazioni di altre applicazioni basate sul Semantic Web: ad esempio, visitando i siti di commercio elettronico, il confronto di informazioni si ottiene solo visitando negozi diversi, analizzando quelle informazioni che compaiono nel layout della pagina e trascurando quelle di pi‘u difficile reperimento; l’introduzione di una o più ontologie, per uniformare le informazioni disponibili, permette un confronto più immediato dei cataloghi e anche un’analisi via software dei dati. Ed è chiaro che HTML, il linguaggio standard per la creazione di pagine Web, non ha le caratteristiche per strutturare i dati in base alla loro semantica, poiché specifica come devono apparire le informazioni, e pertanto deve essere affiancato da un altro strumento per la strutturazione semantica: tutto questo comporta la separazione tra il contenuto e il layout delle pagine e permette l’indipendenza tra i due.

Le potenzialità del Semantic Web sarebbero comunque inutili se non ci fossero degli agenti software in grado di raccogliere le informazioni dalle diverse sorgenti, elaborarle e scambiare i risultati con altri programmi. La potenza di questi agenti sarà tanto più grande quanto più i contenuti del Web e i servizi che li usano sono resi disponibili.

3.2 Un esempio di Ontologia: Wine

Abbiamo già accenato in precedenza all'esempio del Wine e ora ne vediamo quali sono le caratteristiche principali.

Il Wine è nato come progetto interno all'università di Stanford, ed in particolare come lavoro dell'assistente ricercatore, del Knowledge Systems Lab, Eric I. Hsu. Inizialmente questo progetto era inquadrato in un contesto più ampio ed era stato ideato, progettato, elaborato e incastonato nel settore della rappresentazione della conoscenza e del ragionamento, se non fosse che poi da questo esperimento iniziale sono partiti, in seno all'università di Stanford, una miriade di progetti analoghi che si differenziano soltanto leggermente negli scopi e negli obbiettivi a cui questi progetti sono preposti: come ad esempio la creazione di tools, per il governo Usa, che siano in grado di tirare fuori delle conclusioni a partire da serie di dati rilevati da sensori, in diversi contesti.

Ora se noi andiamo alla pagina iniziale del Wine Agent[1], abbiamo la seguente schermata iniziale.

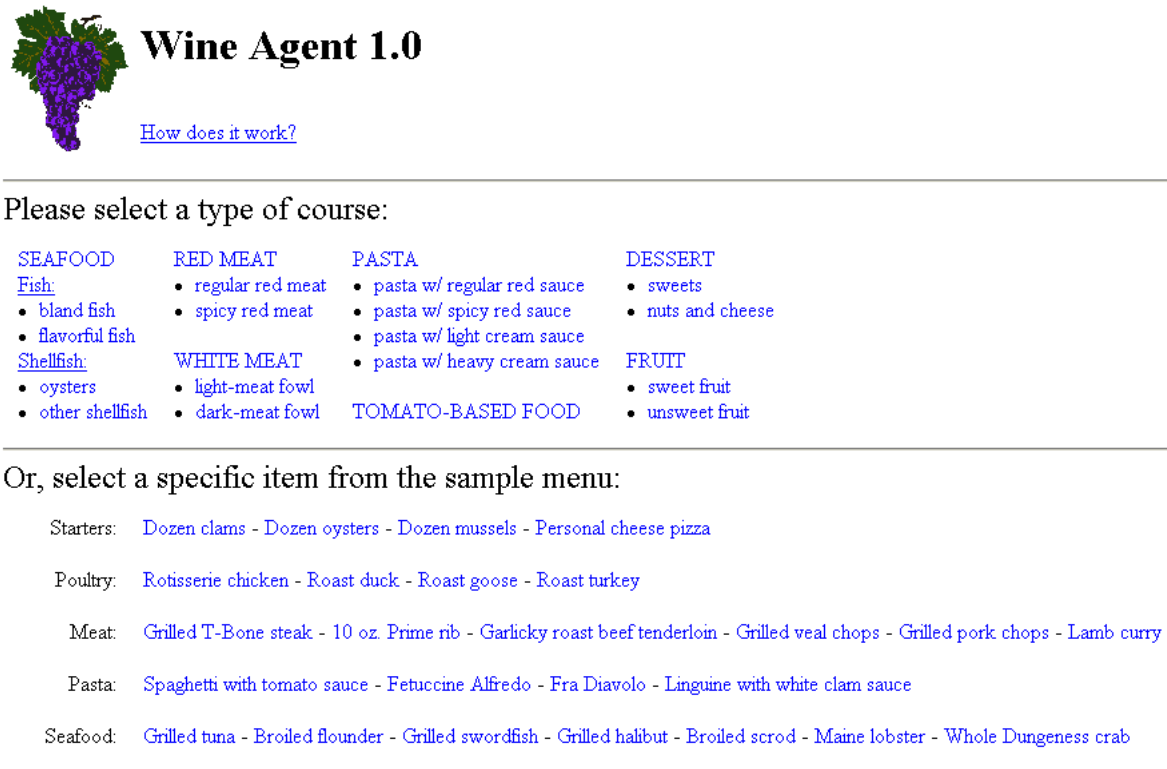

Figura 3.1: Index del Wine Agent

Ora, come vediamo dalla figura, osserviamo che a una intestazione iniziale (con il nome del progetto, il logo e un link che spiega in modo piuttosto stringato i principi alla base del lavoro) seguono due sezioni:

- Nella prima sezione osserviamo che sono presenti una serie di categorie di cibi (tipi di portata), nel quale vediamo che alle sette tipologie di portate principali sono associate un certo numero di sottocategorie di piatti (principalmente basati sul cibo della categoria

[1]Si accede all'index del Wine Agent al seguente indirizzo: $http://onto.stanford.edu:8080/wino/index.jsp$

principale cui il piatto appartiene), al quale vanno aggiunte le categorie intermedie di Cibi di Mare (ossia Pesci e Molluschi).

Ad esempio la sottocategoria Dessert è stata suddivisa nelle sottocategorie:

- Dolci (sweets)
- Noci e formaggio (nuts and cheese)

Oppure la sezione Carni Bianche:

- carne bianca di cacciagione (light-meat fowl)
- carne scura di cacciagione (dark-meat fowl)

O la sezione carne rossa:

- Carne rossa normale (regular red meat)
- Carne rossa piccante (spicy red meat)

Basta clikkare su una categoria o sottocategoria, ad esempio Fish (pesce) e andiamo ad una schermata come questa.

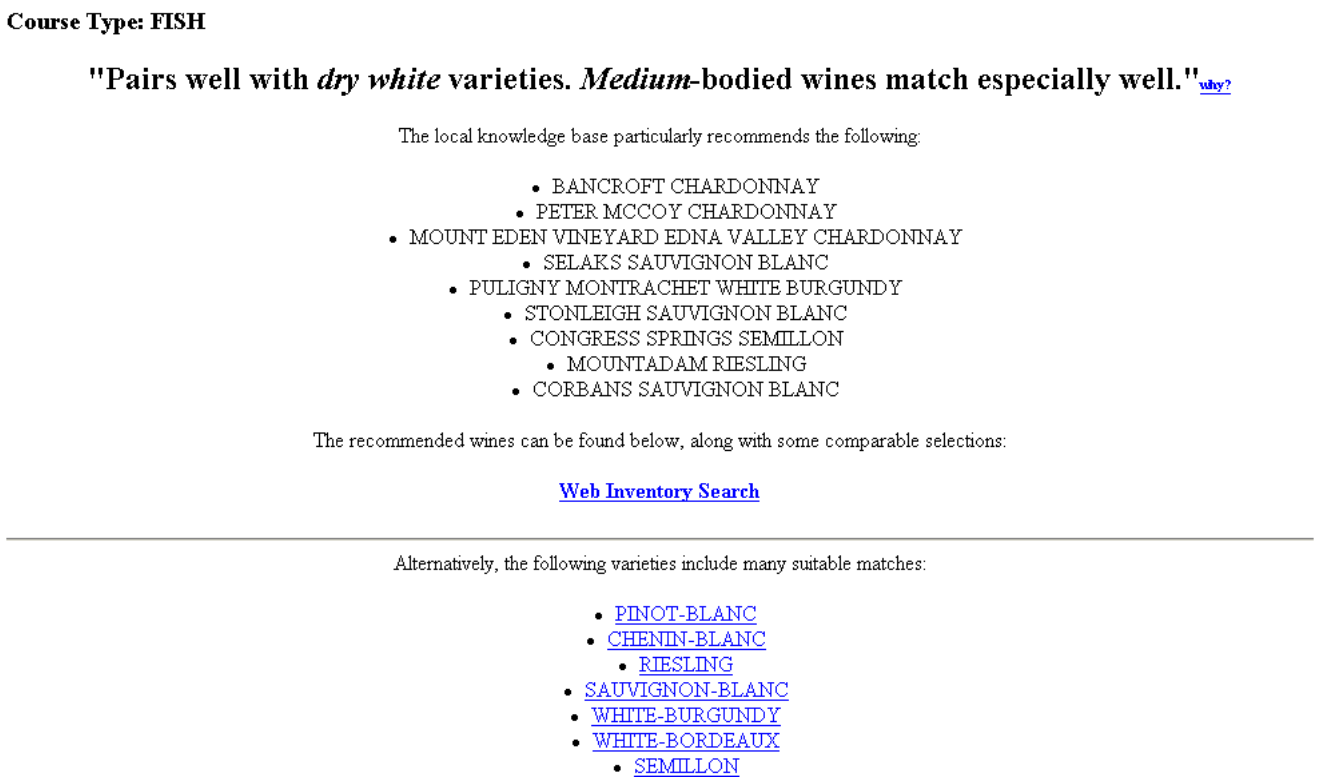

Figura 3.2: Vini associati al tipo di portata pesci

Vediamo che quindi vengono, prima di tutto, indicate le tipologie di vino che meglio si abbinano alla categoria di cibo selezionato. Poi vengono indicati i vini che vengono particolarmente raccomandati, sulla base dell'ontologia (base di conoscenza) ed infine sotto vengono anche indicati una serie di vini che si possono scegliere in alternativa ai vini raccomandati.

- Nella seconda sezione, dell'index del Wine Agent, invece abbiamo un menu, molto più semplice e diretto, che indica direttamente solamente le categorie principali più importanti, e alcuni piatti associati a ogni categoria.
  - Starters: Dozen clams - Dozen oysters - Dozen mussels - Personal cheese pizza
  - Poultry: Rotisserie chicken - Roast duck - Roast goose - Roast turkey
  - Meat: Grilled T-Bone steak - 10 oz. Prime rib - Garlicky roast beef tenderloin - Grilled veal chops - Grilled pork chops - Lamb curry
  - Pasta: Spaghetti with tomato sauce - Fetuccine Alfredo - Fra Diavolo - Linguine with white clam sauce
  - Seafood: Grilled tuna - Broiled flounder - Grilled swordfish - Grilled halibut - Broiled scrod - Maine lobster - Whole Dungeness crab
  - Dessert: Double chocolate cake - Apple pie - Fruit plate - Baked apples - Bananas Foster - Peach cobbler - Assorted nuts - Assorted cheeses

Se ora noi ad esempio selezionamo dalla categoria Poultry il piatto Rotisserie chicken, arriviamo alla seguente schemata.

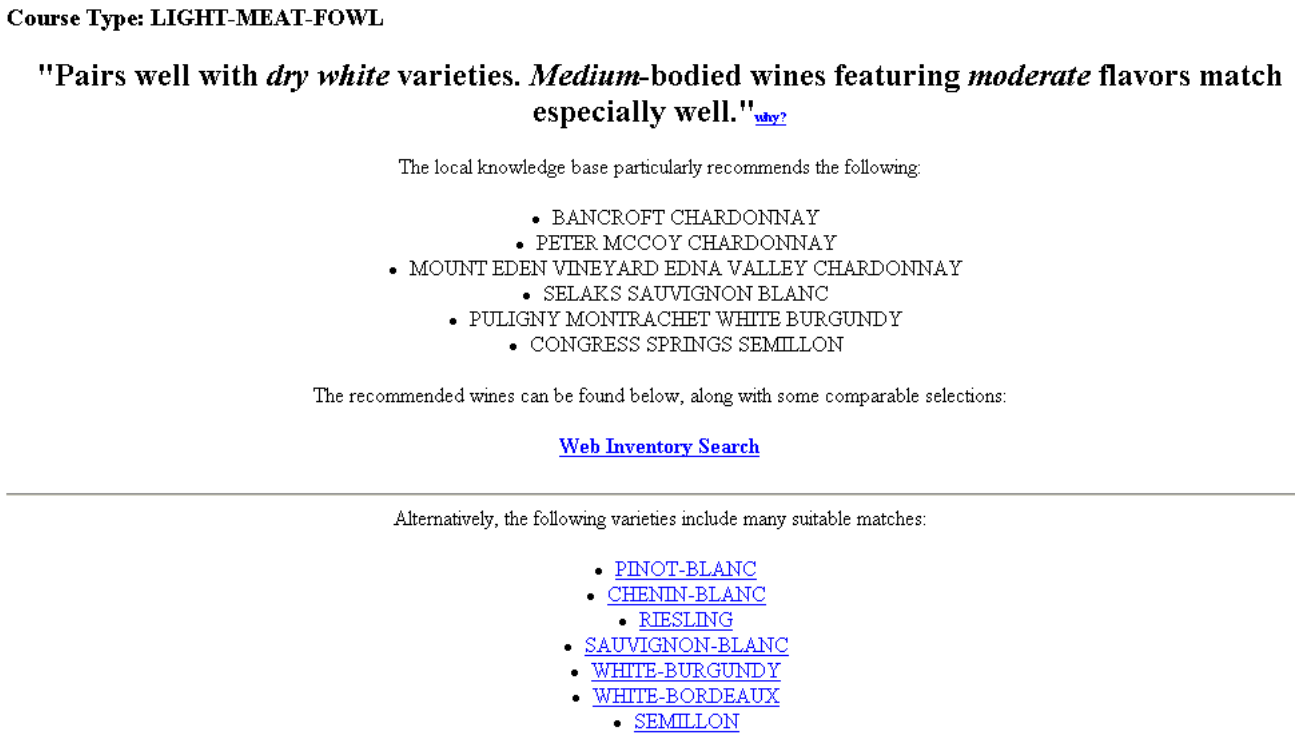

Figura 3.3: Vini associato all'item Rotisserie chicken, Pollo Arrosto

Come prima vediamo che quindi vengono, prima di tutto, indicate le tipologie di vino che meglio si abbinano alla categoria di cibo selezionato. Poi ancora vengono indicati i vini che vengono particolarmente raccomandati ed infine sotto vengono anche indicati una serie di vini che si possono scegliere in alternativa ai vini raccomandati.

Una sezione a parte spiega, in modo forse eccessivamente riduttivo e limitato, come funziona. In sostanza il Wine Agent è basato su un ontological mark-up language e ogni operazione di esso possiamo vedere che è basato su un processo che possiamo suddividere in tre fasi:

- Consultare l'ontologia;
- Eseguire le queries;
- Output dei risultati.

Per quanto riguarda l'ontologia[2] essa contiene le gerarchie e le descrizioni delle categorie di vini e cibi, insieme con le restrizioni su come particolari istanze devono essere messe insieme. Vediamo nella seguente figura il grafo che rappresenta l'ontologia del Wine Agent.

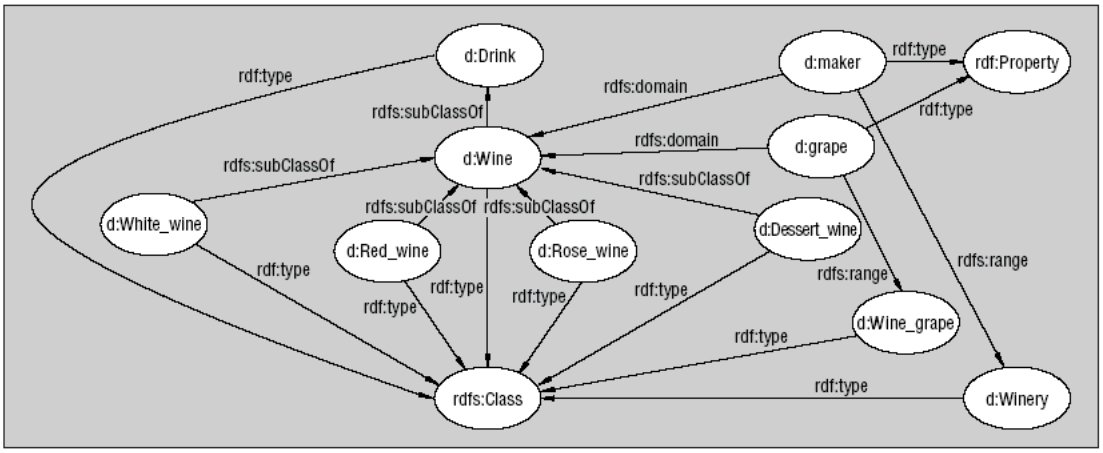

Figura 3.4: An RDF Schema graph representing the Wine ontology.

Come vediamo dalla figura tutti gli oggetti sono sostanzialmente di due tipi:

- Classi (rdf:Class, in basso)
- Proprietà (rdf:Property, in alto a destra)

Quindi la figura mostra per tutti gli oggetti è definita una relazione che definisce il tipo di oggetto stesso, in particolare vediamo che quasi tutti gli item dell'ontologia hanno definito un rdf:type di tipo rdfs:Class mentre vi sono solo due oggetti che hanno rdf:type di tipo rdf:property (dmaker e dgrape).

Osserviamo in particolare nel macroblocco di classi a sinistra che vi è una classe principale Drink che ha come sottoclasse la classe Vini, dal quale a sua volta discendono altre quattro sottoclassi (tutte le relazioni di discendenza sono specificate da una relazione del tipo rdfs:subClassOf):

- d:White_wine
- d:Red_wine
- d:Rose_wine
- d:Dessert_wine

[2]che è liberamente accessibile all'indirizzo web: *http* : *//www.daml.org/ontologies/*76

La classe Wine ha due proprietà, ossia un attributo, chiamati d:maker (di tipo Winery) e d:grape (di tipo Wine_grape).

Questo lo si capisce in quanto ad esempio l'elemento d:maker è stato definito come rdf:Property, con dominio (rdfs:domain) la classe d:Wine e con codominio (rdfs:range), d:Winery mentre d:grape è stato definito come rdf:Property, con dominio (rdfs:domain) la classe d:Wine e con codominio (rdfs:range), però, Wine_grape.

L'ontologia, inoltre, in accordo con le specifiche evidenzia che il vino un liquido potabile prodotto da almeno un produttore di tipo Winery, ed è fatto almeno di un tipo di uva. In aggiunta vengono anche aggiunte informazioni, come la regione nel quale il vino viene prodotto, e ,cosa pi importante, tutte le informazioni relative al vino come: colore, zucchero, corpo e fragranza.

Il concetto di pietanza in un pasto è alla base dell'accoppiamento di un alimento con un vino. Ogni portata contiene almeno un alimento ed almeno una bevanda, di solito un vino. Quando l'utente seleziona un tipo di cibo naturalmente, o un alimento specifico, il Wine agent consulterà le definizioni associate per conseguire le informazioni sul miglior vino che si può assocciare a una data vivanda, per formare una portata.

Ad esempio nel caso della pasta con salsa rossa piccante, abbiamo come restrizioni sul vino da associare al piatto:

- La bevanda deve appartenere alla categoria vini rossi;
- La bevanda deve avere corpo pieno;
- La bevanda deve avere un forte aroma;
- La bevanda deve avere gusto secco, asciutto.

Uno dei vini che corrisponde a queste caratteristiche è lo Chateau Lafite Rothschild Pauillac. Nell'ontologia sono, inoltre, a disposizione delle informazioni aggiuntive sul questo vino come ad esempio che è un vino di Medoc proveniente da Bordeaux, in Francia ed è di colore rosso.

Per esempio, il concetto di Pauillac specifica che questi vini sono caratterizzati da sapori forti ed e sono fatti interamente dall'uva del Cabernet-Sauvignon. Inoltre, Pauillacs è un sottoinsieme particolare di Medocs, distinguendo l'origine di questo vino dagli altri prodotti nella regione di Pauillac.

Tramite il processo sopra descritto, l'agente quindi identifica le proprietà e gli attributi di un vino adatto e sulla base di queste viene selezionata la bevanda adatta per accompagnare ogni vivanda. Adesso il tutto si riduce ad un problema di trattamento testi per generare la pagina portale, che mostra a video i risultati.

La funzionalità fornita dal Wine Agent non è dissimile da quello che potrebbe essere fornita da una semplice look-up table. Effettivamente, le coppie cibo/vino sono pubblicati, abitualmente, in forma tabulare in cui i contrassegni compaiono alle intersezioni delle colonne e delle file che rappresentano le varietà compatibili di alimento e di vino.

Il Wine Agent dimostra che, almeno, questa semplice operazione può essere compiuta con l'ausilio di tecnologia semantica, ma la difficoltà rimane dimostrare come questo può essere migliorato in avvenire.

Supponiamo che se non l'intero web, allora almeno un certo numero di parti in cooperazione usino contrasségni semantici, ai fini di questo esempio. Ora con questo approccio rispetto al metodo tradizionale, di provare a costruire un database enorme degli alimenti e dei vini, le definizioni sarebbero distribuite tra queste parti cooperanti partecipanti.

Un ristorante o un rivenditore che offrono un menu in linea hanno marcato ogni item alimento con delle definizioni standard, machine-readable. Tali marcature comporterebbero benefici dai vantaggi ben noti delle ontologie.

Per esempio, con subclassing, aggiungere un nuovo vino di Pauillac all'inventario non richiederebbe wine.com per contrassegnare tutte le proprietà del vino; sarebbe un altro vino proveniente da Pauillac come specificato nell'esempio visto sopra, più tutte le caratteristiche differenzianti. Ma più importante, in termini del Software Agent, è che tutte le marcature sarebbero rilevabili meccanicamente e possono essere maneggiate dai sistemi di ogni organizzazione.

3.3 Obiettivi

Ora, come abbiamo visto nel paragrafo introduttivo di questa sezione, non ci sono di fatto molte applicazioni effettive di Web Semantic o dell'uso di ontologie sul Web o comunque disponibili per l'uso presso il grande pubblico.

Addirittura l'intervista al professor Frank van Harmelen ha fatto intravvedere come perfino un grande motore di ricerca, come google, stia valutando solo ora le potenzialità del Web Semantic e non ne abbia ancora implementato la tecnologia, utilizzando ancora il metodo di indicizzazione di tutte le pagine sul web.

Per questo assume una certa rilevanza il nostro tentativo (se pur nel nostro piccolo e senza i mezzi che sono a disposizione delle medie e grandi aziende nazionali o internazionali) di costruire un piccolo prototipo che implementi e sfrutti le tecnologie che il Web Semantic rende disponibili, cercando di mostrare tutte le potenzialità che essa comporta.

L'esempio esistente, del Wine Agent, che siamo andati ad analizzare, è un esempio che se pur mostra il funzionamento e i principi sul quale è basato il Semantic Web (illustrando anche il concetto di ontologia), non mostra appieno come in futuro potrebbe evolversi e svilupparsi questa diramazione del Web, proposta con tanto entusiamo da Berneers-Lee.

Questo è un punto essenziale del nostro lavoro: infatti il nostro progetto non si limita a costruire un ontologia (un pò come nel Wine Agent), ma la s'impiega mostrandone poi di fatto gli sviluppi e i progressi che in futuro un motore di ricerca avrà, riuscendo a trovare molti più risultati mirati e di "qualità" migliore, utilizzando e adottando proprio i mezzi semantici. In sostanza il nostro obiettivo primario è stata la creazione di un prototipo di applicazione ipermediale su un web service di ricerca.

Questa è una tecnologia relativamente nuova che è ancora poco utilizzata e approfondita. Essa può essere impiegata in moltissimi contesti, dimostrando la sua efficacia in diversi ambiti. Ad esempio può essere utilizzata non solo nei motori di ricerca, ma anche nei siti commerciali: il potenziale di questa tecnologia è enorme.

Senza contare poi i problemi di portabilità delle applicazioni Web, dovuti di solito al fatto che quasi tutte le nuove tecnologie non sono standard bensì soluzioni proprietarie più o meno diffuse che necessitano per essere utilizzate di software particolari o di particolari versioni di esso. La portabilità è un obiettivo tecnicamente difficile da raggiungere.

La portabilità deve essere garantita:

- A livello software: trasporto il file sorgente da una macchina all'altra (in quanto file di testo) e lo compilo col compilatore di quella macchina.
- A livello di codice binario: possibilit di eseguire un programma (compilato) su qualsiasi tipo di computer.

Mentre la portabilità software è garantita da quasi tutti i linguaggi (C, C++, Pascal, Fortran, Cobol...), la portabilità a livello di codice binario è tipica di Java.

Le tecnologie con il quale è stato implemenentato il nostro progetto, sono basate, infatti, sui linguaggi di programmazione Java e XML (RDF è basato su XML, come abbiamo visto, permettendo di definire la semantica dei tag XML): garantendo così la massima portabilità possibile del software.

Capitolo 4

Specifiche e requisiti di sistema

Nei capitoli precedenti ci siamo occupati di descrivere la semantica nel contesto informatico e i relativi sviluppi, adesso analizziamo un esempio di web semantic, ovvero di applicazione web che sfrutta le ontologie per effettuare ricerche di tipo semantico.

Prima di procedere con la specifica software dell'applicazione da noi realizzata, è bene spiegare brevemente il linguaggio Rdf/Owl e Rdql.

4.1 Rdf/Owl

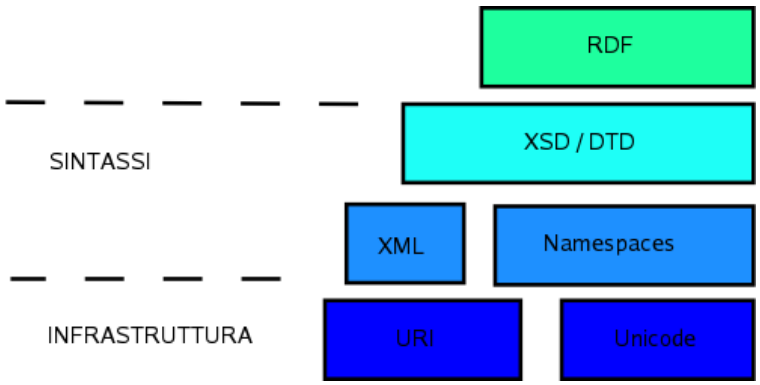

Figura 4.1: Supporti base per il semantic web

I metadati, cioè dati sui dati, descrivono quegli aspetti delle risorse che ci interessa rendere processabili in maniera automatica. Rappresentano conoscenza in quanto informazione utilizzata dal sistema durante l'elaborazione.

Ci sono due tipi di metadati, sostanzialmente:

Metadati interni al documento

- Es.: i tag che inserisco in un documento di testo per descrivere particolari proprietà dello stesso;
- Uso procedurale: "questo testo è in grassetto";
- Uso informativo: "estrai i titoli di tutti i capitoli";

Metadati esterni

- Costruiscono un descrittore esterno;
- Distinguono il contenuto dalla descrizione del contenuto;
- Tipicamente si guadagna in potenza espressiva;
- Si può standardizzare il linguaggio rispetto al formato del contenuto.

Prima di porsi il problema di come rappresentare la conoscenza occorre risolvere due problemi:

- Come riferirsi agli oggetti del nostro modello formale ? (Anche se i metadati sono interni, la descrizione può richiedere di far riferimento ad oggetti esterni);
- Come garantire l'interoperabilità per lo meno sintattica ?

Le risposte a queste due richieste sono:

- URI (Uniform Resource Identifiers) ed URIref;
- XML (eXtensibleMarkup Language) ed Unicode risolve il problema di distingure dati da meta-dati;

RDF è un modello di descrizione astratto e non pone vincoli sulla sintassi e sul significato delle descrizioni di una risorsa. Questo vuol dire che ognuno potrebbe proporre un qualsiasi meccanismo per descrivere risorse utilizzando le astrazioni previste da RDF. Ad esempio, una descrizione RDF potrebbe essere espressa tramite una rappresentazione grafica delle risorse, delle proprietà e delle asserzioni.

Tuttavia, per ragioni pratiche la definizione formulata del W3C propone XML come metalinguaggio per la rappresentazione del modello RDF, cioè propone RDF come linguaggio basato su XML. In pratica, una descrizione RDF è un documento XML contenente alcuni *tag predefiniti* (ad esempio <rdf:RDF>) e una lista di *tag liberi* per la descrizione vera e propria.

RDF descrive non solo ciò che è presente nel Web, come le pagine o parti di esse, ma anche tutto quello a cui può essere associato un URI: ad esempio, se si può associare un URI ad una persona, tutte le informazioni ad essa relative potranno essere rappresentate mediante RDF. Un URI reference è un URI seguito, opzionalmente, da un identificatore di elemento (fragment identifier), ad esempio: http://www.server.com/index.jsp#title1.

In RDF spesso si usano le URI per idenficare il documento RDF di specifica, mentre i fragment identifier per riferirsi alla specifica di una particolare risorsa. Nella descrizione RDF si fa uso della tecnica dei namespace di XML. Un namespace è un contesto che definisce un insieme di tag da utilizzare in un documento XML.

Nel precedente esempio, il tag <rdf:RDF> indica che tale tag è definito nel namespace identificato dal prefisso rdf, la cui dichiarazione si trova nella prima riga del documento XML. In tale riga è presente la direttiva di elaborazione che indica anche l'indirizzo presso cui è pubblicata la definizione del namespace.

RDF ha un sistema a classi, paragonabile a quelli della programmazione ad oggetti: una collezione di classi è detta schema e le classi formano una gerarchia, che, con il meccanismo delle sottoclassi, consente la specializzazione dei concetti e la condivisione delle definizioni dei metadati. Inoltre, è permessa l'ereditarietà multipla, che consente la vista multipla dei dati e lo sfruttamento di definizioni create da altri.

Il modello dei dati è rappresentato con un grafo, che schematizza la tripla *risorsa-proprietà-valore*: i nodi del grafo sono le risorse descritte da RDF (pagine Web, insieme di pagine, oggetti non direttamente accessibili via Web), gli archi sono etichettati con il nome della proprietà relativa alla risorsa e sono direzionati dal nodo che rappresenta la risorsa verso quello che indica il valore (che può essere di tipo atomico oppure un'altra risorsa, ottenendo in questo caso dele relazioni fra le risorse). La risorsa (identificata univocamente da un URI) insieme con una proprietà ad essa associata e al corrispondente valore forma uno *statement*, in cui i tre elementi prendono il nome di soggetto, predicato e oggetto.

Per esprimere una affermazione dobbiamo identificare:

- L'**oggetto** che vogliamo descrivere;
- La specifica **proprietà** dell'oggetto (o tra oggetti) su cui vogliamo predicare;
- Il **valore** assunto dalle proprietà o l'**oggetto** con cui viene messa in relazione l'entità su cui stiamo predicando.

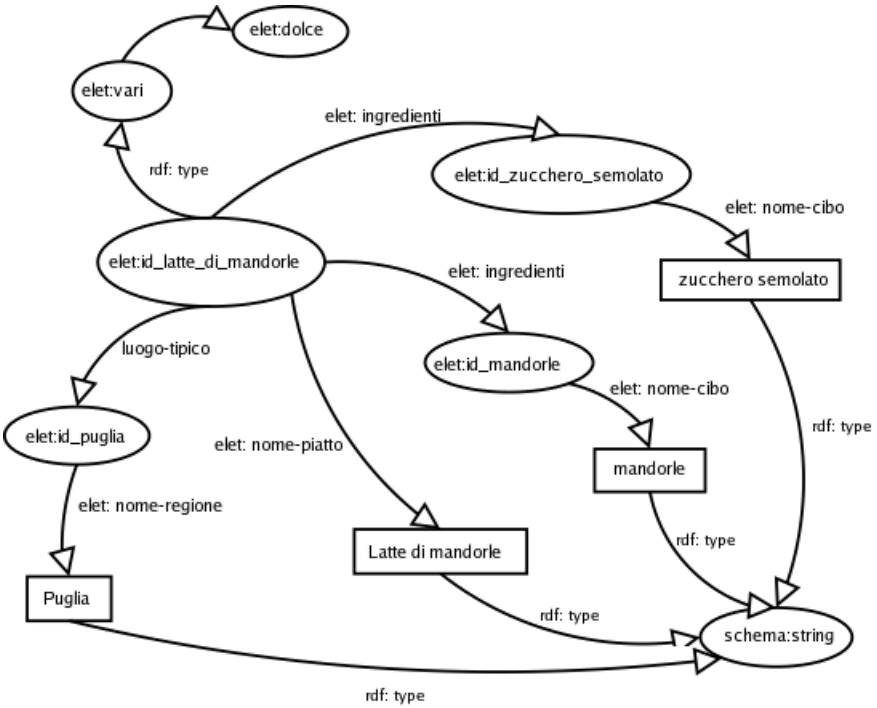

Figura 4.2: Esempio di statement

Nell'esempio (figura 4.2 a pag. 46) vengono evidenziate le relazioni tra le risorse (rappresentate da ovali), valori atomici (rappresentati da rettangoli) e le proprietà (sono gli archi).

Per comodità vengono utilizzate delle entità per non dover scrivere tutte le volte il namespace che risulta essere lungo e difficilmente ricordabile:

elet = http://www.elet.polimi.it/ontology.owl#

schema = http://www.w3.org/2001/XMLSchema#

rdf = http://www.w3.org/1999/02/22-rdf-syntax-ns#

Possiamo vedere il relativo frammento di codice Rdf/Owl che descrive la risorsa "latte di mandorle":

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.elet.polimi.it/ontology.owl#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xml:base="http://www.elet.polimi.it/ontology.owl">

    <owl:Class rdf:ID="vari">
       <rdfs:subClassOf rdf:resource="#dolce"/>
    </owl:Class>

    <owl:ObjectProperty rdf:ID="luogo-tipico">
        <rdfs:range rdf:resource="#Italia"/>
        <rdfs:domain rdf:resource="#ricette"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:ID="ingredienti">
      <rdfs:range>
        <owl:Class>
           <owl:unionOf rdf:parseType="Collection">
             <owl:Class rdf:about="#cibi"/>
             <owl:Class rdf:about="#ricette"/>
           </owl:unionOf>
        </owl:Class>
      </rdfs:range>
      <rdfs:domain rdf:resource="#ricette"/>
    </owl:ObjectProperty>

    <owl:DatatypeProperty rdf:ID="nome-piatto">
       <rdfs:domain rdf:resource="#ricette"/>
       <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:DatatypeProperty>

    <vari rdf:ID="id_latte_di_mandorle">
       <luogo-tipico rdf:resource="#id_puglia"/>
       <ingredienti rdf:resource="#id_zucchero_semolato"/>
       <ingredienti rdf:resource="#id_mandorle"/>
       <nome-piatto rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                    Latte di mandorle
       </nome-piatto>
    </vari>
</rdf:RDF>
```

Il codice sopra riportato è solo una piccolissima parte del file owl che è stato utilizzato per l'implementazione del nostro esempio di applicazione semantica. Come primo passo è utile definire una serie di namespaces per la costruzione dell'ontologia, infatti in questo modo è possibile riferirsi a delle risorse dove sono contenute le definizioni dei costrutti per la creazione dell'ontologia (ad esempio nel namespace owl viene contenuta la definizione di owl:Class).

Nelle righe successive andiamo a definire la classe "vari" come sotto-classe di "dolce"[1]. A questo punto iniziamo a definire delle proprietà, che come abbiamo visto in precedenza, permettono un collegamento tra le risorse o tra una risorsa e un valore atomico, ad esempio la prima proprietà *luogo-tipico* è un collegamento tra due risorse, ovvero tra ricette e Italia e il verso viene specificato dai costrutti *rdfs:range* e *rdfs:domain*, in particolare il verso è dalle ricette all'Italia[2]. Successivamente definiamo la proprietà *"ingredienti"*, questa collega un cibo o una ricetta con un'altra ricetta, dato che una ricetta potrebbe avere come ingrediente una serie di ingredienti di tipo cibo (ad esempio, cipolla, alloro, basilico, ...) oppure perché accanto ai cibi potrebbe avere come ingrediente un'altra ricetta (ad esempio per fare la pasta tipica di Genova può essere utile avere come ingrediente il "pesto alla genovese", quest'utimo definito come ricetta.). La successiva proprietà *nome-piatto* permette il collegamento tra una risorsa (ricetta) e un valore atomico (il nome della ricetta che è una stringa di caratteri). Infine viene definita l'istanza *"id_latte_di_mandorle"* della classe vari e viene poi definito il *luogo-tipico*, gli *ingredienti* e il *nome-piatto*.

Nella nostra ontologia sono state dichiarate 742 istanze così ripartite:

Italia

- Nord 7;
- Centro 5;
- Sud 6.

cibi

- carne
  - animali-da-cortile 1;
    - pollame 13;
  - da-macello
    - bovini 1;
      - bue 2;
      - manzo 4;
      - vitello-da-latte 3;
      - vitellone 7;
    - ovini;
      - agnello 2;
    - suini 16;
      - suino-castrato 2;
    - selvaggina;
      - da-pelo 2;
      - da-penna 11;

[1] Nel frammento di codice a pag.47 non è riportata la parte di codice che descrive il fatto che la classe dolce è a sua volta sotto-classe della classe ricette

[2] Nel frammento di codice a pag.47, non è riportata la parte di codice che descrive il fatto che l'Italia è a sua volta composta da tre sotto-classi: *Nord*, *Centro* e *Sud*. Queste contengono le istanze delle regioni.

- cereali-e-derivati
  - cereali 9;
    - risi 1;
      - fino 2;
      - semifino 2;
      - superfino 2;
  - derivati 1;
    - pane 6;
    - pasta-alimentare
      - corta 7;
      - lunga 11;
      - piccola 4;
      - ripiena 4;
    - sfarinati 8;
    - sostitutivi-del-pane 9;
- cefalopodi 5;
- condimenti-ausiliari 5;
- crostacei 8;
- erbe-aromatiche 29;
- frutta
  - a-guscio 8;
  - a-polpa 15;
  - agrumi 6;
  - di-bosco 5;
  - esotica 9;
- funghi 2
  - parassiti 3;
  - saprofiti 3;
  - simbionti 4;
- grassi
  - animali 6;
  - vegetali 7;
- latticini 4
  - formaggio
    - formaggi-a-lunga-maturazione 3;
    - formaggi-a-media-maturazione 6;
    - formaggi-a-pasta-erborinata 1;
    - formaggi-a-pasta-filata 4;
    - formaggi-freschi 4;
- liquidi 6
  - liquori 9;
  - vini 3;
- molluschi 10;
- pesce
  - acqua-dolce 13;
  - acqua-mare 25;
- prodotti-dolciari 19;
- rane-lumache 2;

- spezie 13;
- uova 5;
- verdura
  - a-foglie-verdi 12;
  - legumi-a-fiori-ed-a-frutta 12;
  - leguminose 9;
  - tuberi-radici-bulbi 7;

ricette

- antipasti 26;
- dolce
  - cucchiao 7;
  - pasticceria 9;
  - torte 10;
  - vari 14;
- primi
  - minestre 22;
  - pasta-ripiena 6;
  - pastasciutta 26;
  - risotti 12;
- salse 11;
- secondi 52.

Come abbiamo visto nell'esempio precedente, i tag propriamente descrittivi di una descrizione RDF sono liberi, nel senso che ciascuno è libero di inventarsi i tag che preferisce. Tuttavia questo comporterebbe la possibilità di ritrovarsi tag diversi con lo stesso significato.

Ad esempio, il nostro tag <ingredienti> potrebbe essere definito come <ingredients> da un nostro collega anglosassone. Per evitare il sovrapporsi di tag diversi con lo stesso significato, RDF prevede un meccanismo per creare vocabolari o schemi, cioè insiemi di proprietà ed elementi che definiscono un contesto per la descrizione di determinate categorie di risorse.

Un noto esempio di schema RDF è il Dublin Core Metadata: esso definisce i tag da utilizzare per la descrizione di documenti elettronici (libri, articoli, rapporti, ecc.).
Il seguente è un esempio di descrizione del sito Html.it in RDF utilizzando lo schema Dublin Core Metadata:

```
<?xml:namespace ns="http://www.w3.org/1999/02/22-rdf-syntax-ns#" prefix="rdf" ?>
<?xml:namespace ns="http://purl.org/metadata/dublin_core#" prefix="dc" ?>
<rdf:RDF>
  <rdf:Description about="http://www.html.it">
    <dc:Title>HTML.IT</dc:Title>
    <dc:Description>
         Il portale italiano su HTML e dintorni
    </dc:Description>
    <dc:Publisher>HTML.IT srl</dc:Publisher>
    <dc:Format>text/html</dc:Format>
    <dc:Language>it</dc:Language>
  </rdf:Description>
</rdf:RDF>
```

Come è evidente dall'esempio, anche per far riferimento ad uno specifico schema RDF si fa uso dei namespace. In questo esempio, oltre al namespace individuato dal prefisso rdf, viene utilizzato il namespace individuato dal prefisso dc e definito dallo schema RDF del Dublin Core Metadata.

Gli schemi RDF possono essere riutilizzati ed estesi per la descrizione di altre categorie di risorse. Possono includerer vincoli sui possibili valori di una proprietà ed ereditare caratteristiche da altri schemi.

4.1.1 HTML e RDF

Ma come possiamo associare una descrizione RDF ad una pagina HTML?

Abbiamo essenzialmente due modi:

Creare una descrizione esterna e mettere all'interno della pagina un riferimento alla descrizione tramite il tag <LINK>, come nel seguente esempio:

```
<link rel="meta" href="descrizione.rdf">
```

Oppure inserire la descrizione all'interno della pagina HTML, come in quest'altro esempio:

```
<?xml:namespace ns="http://www.w3.org/1999/02/22-rdf-syntax-ns#" prefix="rdf" ?>
<?xml:namespace ns="http://purl.org/metadata/dublin_core#" prefix="dc" ?>
<rdf:RDF>
  <rdf:Description about="http://www.html.it">
    <dc:Title>HTML.IT</dc:Title>
    <dc:Description>
        Il portale italiano su HTML e dintorni
    </dc:Description>
    <dc:Publisher>HTML.IT srl</dc:Publisher>
    <dc:Format>text/html</dc:Format>
    <dc:Language>it</dc:Language>
  </rdf:Description>
</rdf:RDF>
```

E' opportuno inserire la descrizione RDF nella sezione <HEAD> della pagina HTML per evitare che qualche browser visualizzi il suo contenuto all'utente.

4.2 Rdql

Rdql è un linguaggio di query per documenti Rdf/Owl. La sintassi è SQL-like, per spiegare questi concetti faremo alcuni esempi.

```
SELECT ?ricetta
WHERE (?codRicetta, elet:nome-piatto, ?ricetta)
      (?codRicetta, elet:ingredienti, ?codIngredienti)
      (?codIngredienti, tipo:type, elet:verdura)
      (?codRicetta, tipo:type, elet:primi)
      (?codRicetta, elet:luogo-tipico, ?codRegione)
      (?codRegione, tipo:type, elet:Nord)
USING elet FOR <http://www.elet.polimi.it/ontology.owl#>
      tipo FOR <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

Le variabili sono precedute dal simbolo *?*. Il frammento di codice sopra riportato significa che vogliamo estrarre il nome delle ricette che hanno almeno un ingrediente di tipo "verdura", inoltre vogliamo che la ricetta appartenga ai "primi" e che la regione di appartenenza della ricetta appartenga al "Nord" (Italia). Con l'istruzione **SELECT** indichiamo quale variabile vogliamo ottenere in output, nella clausola **WHERE** indichiamo tutte le condizioni che devono essere soddisfatte nella query (un pò come avviene in SQL). Siccome quando ci riferiamo ad una risorsa occore specificare anche l'URI, spesso questa risulta essere molto lunga, quindi conviene definire dei prefissi che ci consentano di riferirsi a questa in modo molto più comodo, questo è quanto viene fatto nell'istruzione **USING**.

Tutte le condizioni nella clausola where sono delle triplette:

1. il primo parametro indica la risorsa di partenza;
2. il secondo parametro indica la proprietà che ci interessa;
3. il terzo parametro indica la risorsa o il dato atomico di arrivo;

Il corrispondente *join* effettuato in SQL, in RDQL viene realizzato andando a scrivere in più triplette lo stesso identificativo, ad esempio:

```
(?codRicetta, elet:nome-piatto, ?ricetta)
(?codRicetta, elet:ingredienti, ?codIngredienti)
(?codIngredienti, tipo:type, elet:verdura)
```

Mettendo in due righe "?codRicetta" e "?codIngredienti" effettuiamo il join, inoltre nella prima riga viene detto che la variabile "?ricetta" conterrà il nome del piatto, nella seconda riga si specifica che "?codIngrediente" conterrà l'URI e l'identificativo generico dell'ingrediente e con la terza riga andiamo ad imporre che il tipo dell'ingrediente sia "verdura".

Vediamo adesso un altro esempio:

```
SELECT  ?ingredienti
WHERE (?codRicetta, elet:nome-piatto, "Torta di nocciole e mandorle")
      (?codRicetta, elet:ingredienti, ?codIngredienti)
      (?codIngredienti, elet:nome-cibo, ?ingredienti)
USING elet FOR  <http://www.elet.polimi.it/ontology.owl#>
```

Con il codice sopra riportato si cerca prima l'identificativo della risorsa contenuta nel namespace elet avente come valore atomico della proprietà *nome-piatto* il valore "Torta di nocciole e mandorle", a questo punto si cercano gli identificativi degli ingredienti e in seguito si cercano i corrispettivi nomi.

Vediamo un ultimo esempio:

```
SELECT ?ricette
WHERE (?codRicetta, elet:nome-piatto, ?ricette)
AND ?ricette =~ /torta/i
USING elet FOR  <http://www.elet.polimi.it/ontology.owl#>
```

In questo esempio cerchiamo tutte le ricette che contegano nel nome della ricetta la parola "torta", la **i** finale indica il fatto che la ricerca è case insensitive. L'operatore $=\sim$ può essere sostituito con $! \sim$, in questo caso cercherà l'esatto opposto, ovvero tutte le ricette che non contengano nel nome la parola torta.

4.3 Architettura

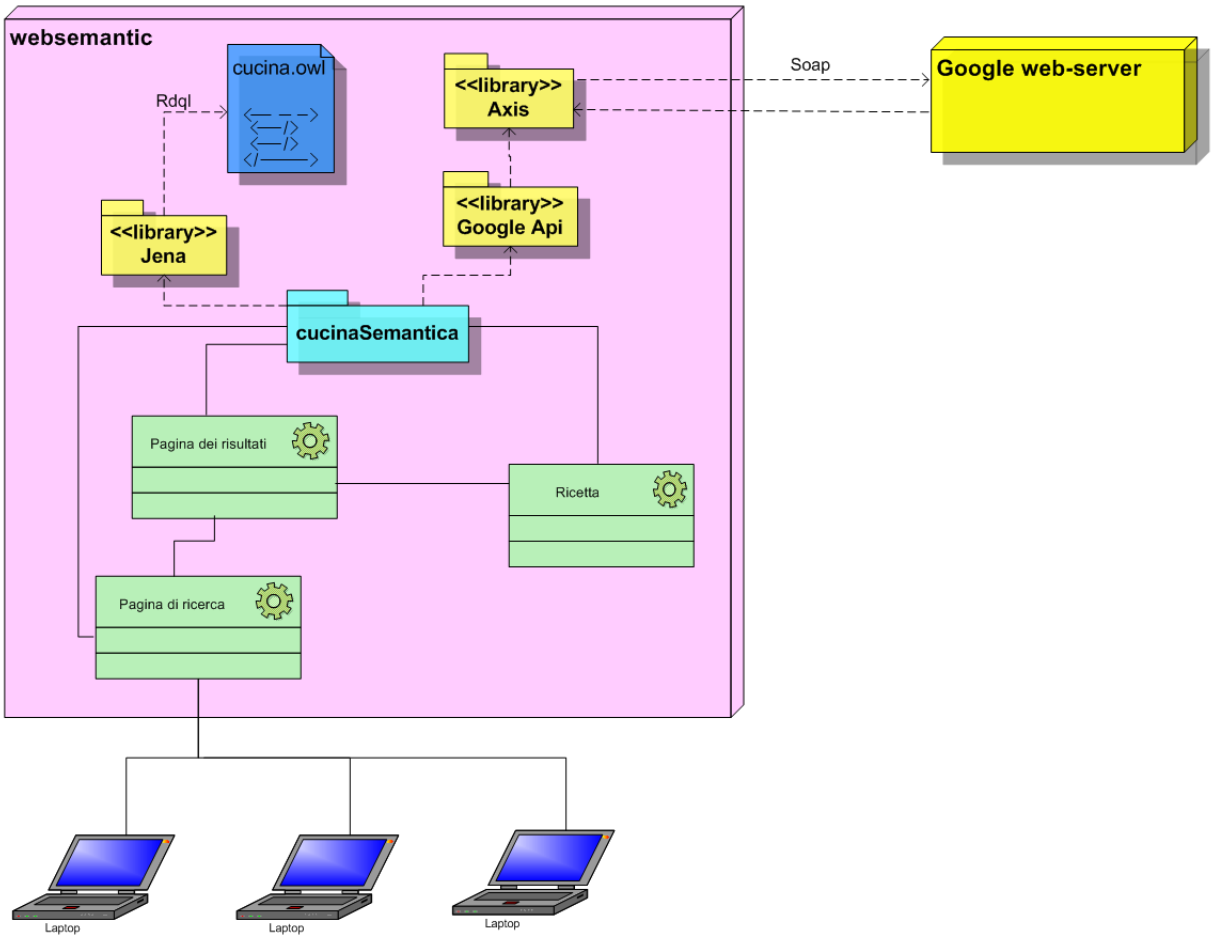

Figura 4.3: Architettura

In figura 4.3 viene mostrata l'architettura software della nostra applicazione. Con la nostra applicazione è possibile effettuare una serie di ricerche semantiche sulla base di poche informazioni, infatti tipicamente l'utente non conosce a priori il nome delle ricette che gli interessano, ma sà che vuole delle ricette sulla base di certi criteri.

Ad esempio l'utente potrebbe essere allergico (oppure non gradire) le spezie ed inoltre, può darsi, che voglia fare una ricetta tipica del Nord, poiché invita a cena degli amici provenienti dall'Italia settentrionale. Con informazioni così vaghe è difficile ottenere una lista di ricette da un normale motore di ricerca (*Google*, *Yahoo*, ...), questo perché se ad esempio proviamo ad inserire come frase di ricerca le seguenti parole:

ricette + primi + Nord -spezie

Otteniamo una serie di risultati, mostrati in figura 4.4.

Come abbiamo visto una semplice ricerca non ci ha portato ai risultati che volevamo. Con la nostra applicazione che sfrutta i concetti delle ontologie possiamo ottenere dei risultati migliori e più coerenti: anzichè effettuare una ricerca su un classico motore di ricerca, cerchiamo tutte le ricette sull'ontologia e poi visioniamo i documenti che contengono la descrizione della ricetta con l'ausilio di un qualsiasi motore di ricerca (nella nostra applicazione Google).

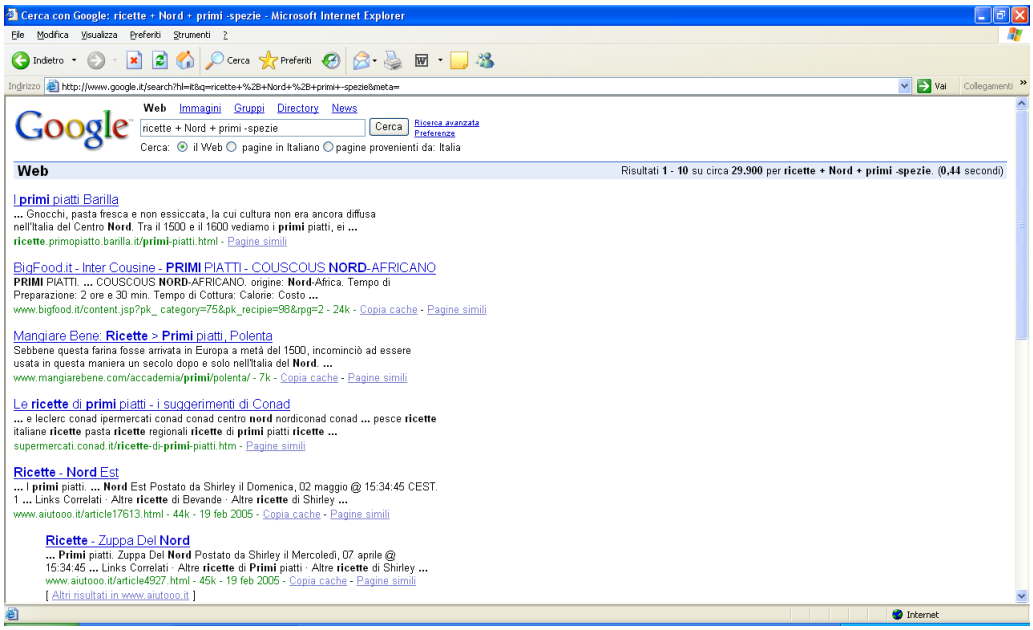

Figura 4.4: Output di google per la ricerca di ricette di primi del Settentrione, che non contengano spezie

In pratica il nostro obiettivo è quello di fare una serie di ricerche semantiche avendo pochi dati e, successivamente, ricerchiamo poi i documenti effettivi attraverso l'uso del pattern-matching. Quello che cerchimo di fare è di pre-filtrare le informazioni vaghe dell'utente e successivamente dettagliarle per ottenere i risultati a cui mirava l'utente.

Uno dei vantaggi dell'utilizzo del nostro software, è il fatto che i *metadati* sono contenuti nell'ontologia, mentre i documenti che descrivono l'implementazione delle ricette sono su web: così facendo si ha disposizione una miriade di pagine web in costante aggiornamento e in continuo aumento.

Tornando all'esempio di prima, con le stesse informazioni, se proviamo stavolta ad effettuare la ricerca con il nostro motore di ricerca, troveremo i seguenti risultati:

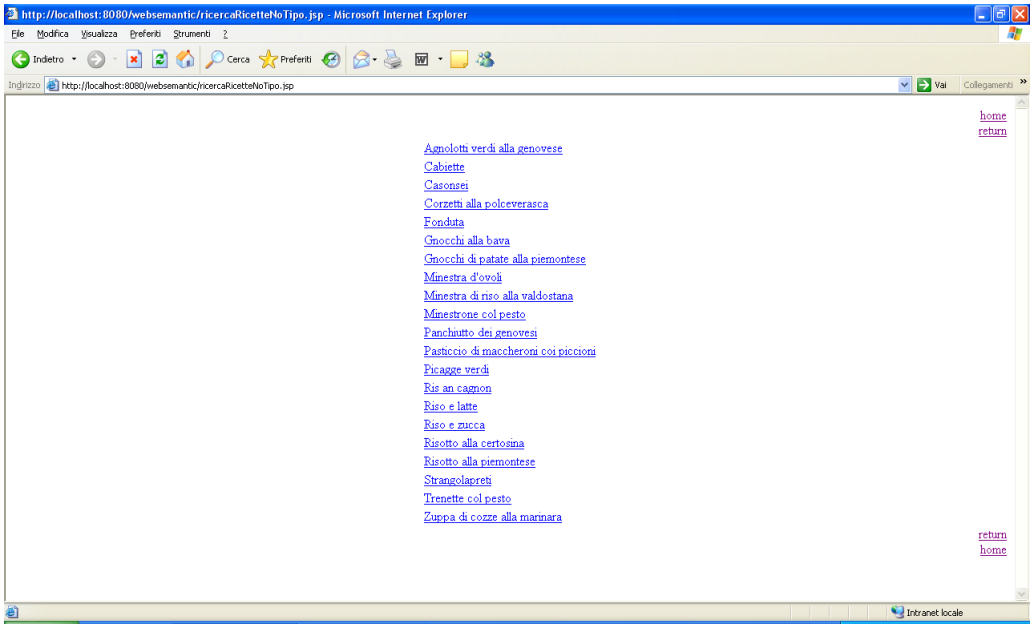

Figura 4.5: Risultati ottenuti con il nostro motore semantico

Come si vede in figura 4.5, il risultato è nettamente inerente con quello che cercavamo. Questo perché il nostro software è come se si frapponesse tra l'utente e il web, consentendo all'utente di effettuare prima una ricerca all'interno dell'ontologia e in una fase successiva sfruttare un classico motore di ricerca per ottenere la descrizione della ricetta che ha questo punto risulta essere nota. Con una struttura di questo tipo un qualunque motore di ricerca porta ai risultati sperati in breve tempo.

4.4 Use Case

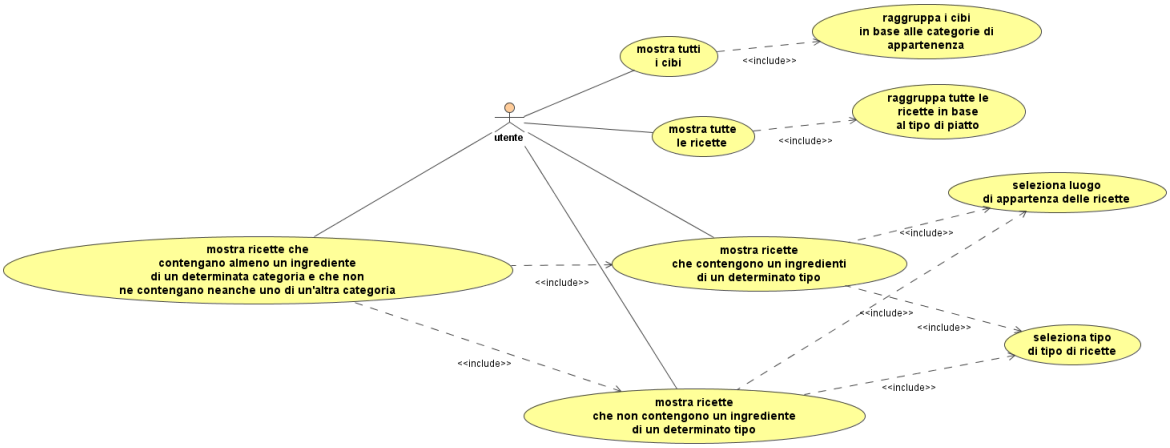

Figura 4.6: Use case

L'utente generico che si connette al sito ha la possibilità di effettuare una serie di operazioni di ricerca, in particolare deve essere possibile:

- mostrare tutti i cibi: per una migliore visualizzazione dei risultati è meglio raggruppare i cibi in base alle categorie di appartenenza;
- mostrare tutte le ricette: per una migliore visualizzazione dei risultati è meglio raggruppare le ricette in base al tipo di piatto;
- mostrare le ricette che contengono almeno un ingrediente di un determinato tipo: deve essere possibile la selezione del luogo di appartenenza delle ricette e il tipo di ricette;
- mostrare le ricette che non contengono un ingrediente di un determinato tipo: deve essere possibile la selezione del luogo di appartenenza delle ricette e il tipo di ricette;
- mostrare le ricette che contengano almeno un ingrediente di una determinata categoria e che non ne contengano neanche uno di un'altra: questo caso è visto come una combinazione dei due punti precedenti.

4.5 Mappa del sito

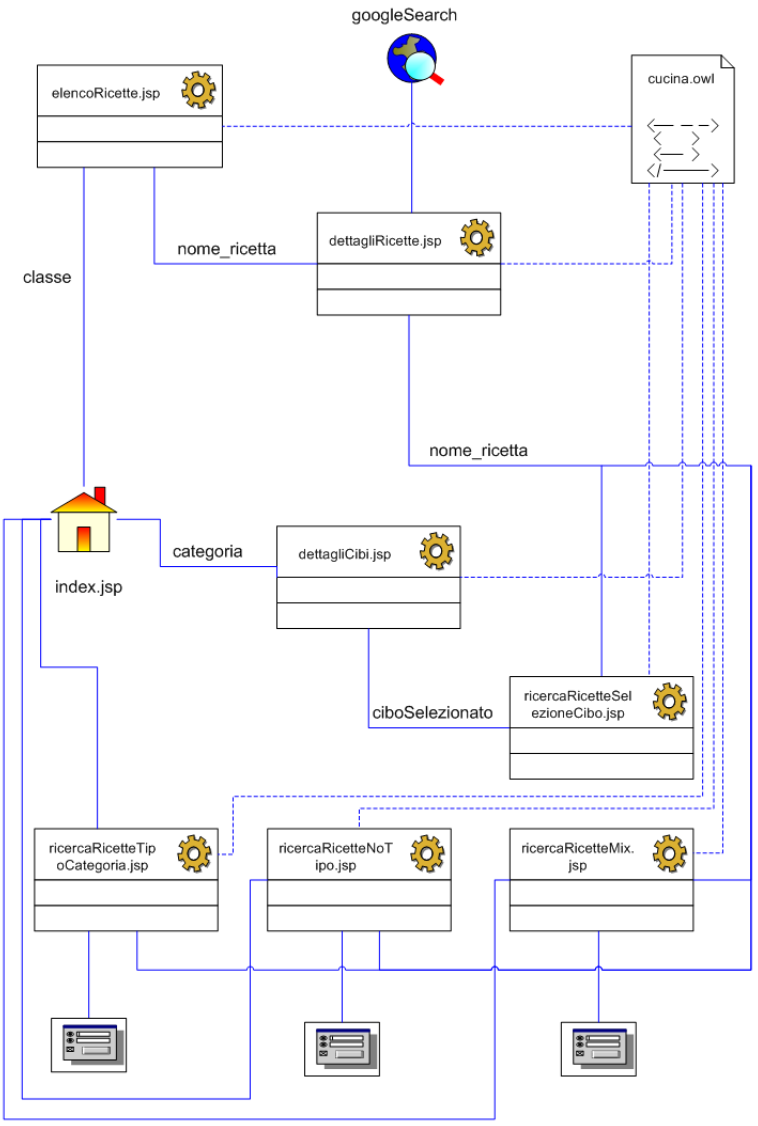

Figura 4.7: Mappa del sito

In figura 4.7 a pag. 58 è mostrata la mappa del sito, la tecnologia utilizzata è jsp[3]. L'**home page** è raggiungibile in qualunque punto del sito, non sono quindi riportati i link direzionali dalle altra pagine alla home. L'home page è suddivisa in tre sezioni:

categoria cibi : in questa parte della pagine vengono mostrati una serie di link che riportano tutte macro-categorie di cibi presenti nell'ontologia. Cliccando su uno di questi link si naviga verso la pagina **dettagliCibi.jsp**, inoltre si passa alla pagina di arrivo il nome della *categoria* selezionata, verranno quindi visti tutti i cibi appartenenti a quella determinata categoria di cibo;

tipo piatti : in questa parte della pagina vengono mostrati una serie di link che corrispondono alle macro-classi di apparteneza delle ricette (ad esempio, primi, secondi, . . .).

[3]http://java.sun.com

Cliccando su un link si raggiunge la pagina **elencoRicette.jsp**, inoltre viene inviato alla pagina di destinazione, nella query string, il parametro *classe* che rappresenta la classe di cui si vuole l'elenco delle ricette;

query : in questa sezione è possibile raggiungere tre pagine in cui è possibile effettuare ricerche il cui potere espressivo è superiore ad un normale motore di ricerca.

La pagina **dettagliCibi.jsp** mostra tutti i cibi presenti in una determinata categoria, da questa pagina è possibile spostarsi in una successiva pagina, **ricercaRicetteSelezioneCibo.jsp**. Questa pagina conterrà tutte le ricette che abbiamo almeno un ingrediente appartenente alla categoria di cibi presenti nella pagina *dettagliCibi.jsp*. Le ricette compaiono sotto forma di link (viene mostrato il nome della ricetta) che portano ad una successiva pagina, ovvero *dettagliRicetta.jsp*, questa mostra il luogo di appartenenza, gli ingredienti e i link ai documenti che ne descrivono la ricetta (preparazione, tempi di cottura, ...), la ricerca di questi ultimi link avviene effettuando una ricerca su *Google*[4].

La pagina **elencoRicette.jsp** contiene una lista di ricette tutte dello stesso tipo (ad esempio tutti i primi), le ricette appaiano sotto forma di link (viene mostrato il nome della ricetta). Cliccando su una ricetta vengono mostrati i dettagli nella pagina *dettagliRicetta.jsp*.

Analizziamo adesso i tre link che portano, dall'home page, ad altrettante pagine jsp che consentono di effettuare alcune query.

Il primo link porta alla pagina **ricercaRicetteTipoCategoria.jsp**, in questa pagina viene mostrata una form in cui è possibile selezionare il tipo di ingrediente che si vuole nella ricetta (ad esempio, carne o verdura o ...), il tipo di ricetta (ad esempio, primi o secondi o ...), e il luogo tipico di appartenenza della ricetta (ad esempio Nord). La stessa pagina jsp mostrerà i risultati sotto forma di link (ancora una volta vengono mostrati i nomi delle ricette che soddisfano le condizioni). Cliccando su un link viene mostrata la pagina *dettagliRicetta.jsp*, che conterrà la descrizione della ricetta. Dai risultati della ricerca, è anche possibile ritornare direttamente alla form di immissione dati senza dover passare dalla home page.

Il secondo link porta alla pagina **ricercaRicetteNoTipo.jsp**, anche in questa pagina, come in quella precedentemente descritta, compare una form molto simile a quella vista nel precedentemente. Tuttavia vi è una sostanziale differenza nella prima selezione, infatti questa volta si potrà selezionare il tipo di ingrediente che non si vuole assolutamente nella ricetta, mentre prima ci potevano essere da un minimo di un ingrediente ad un massimo di n (dove n è rappresenta il numero degli ingredienti di una ricetta) ingredienti di un determinto tipo. Anche qui i risultati vengono mostrati nella medesima pagina jsp, e i risultati compaiono nella stessa forma vista precedentemente. E' possibile anche in questo caso passare dalla pagina dei risultati alla pagina di visualizzazione della form senza passare dall'home page.

Il terzo link porta alla pagina **ricercaRicetteMix.jsp**, in questa pagina viene mostrata una form in cui si chiede:

- un tipo di ingrediente che si vuole nella ricetta;
- un tipo di ingrediente che **non** si vuole nella ricetta;
- il tipo di ricetta;
- il luogo tipico di appartenenza della ricetta.

I risultati appaiono nella stessa forma vista precendentemente (ricercaRicetteTipoCategoria.jsp e ricercaRicetteNoTipo.jsp) ed anche la navigazione è la stessa.

[4] http://www.google.com

Un aspetto molto particolare da notare è il fatto che tutte le pagine jsp fanno riferimento all'ontologia *cucina.owl*, questo perché tutti i dati vengono estratti dall'ontologia.

4.6 Class Diagram

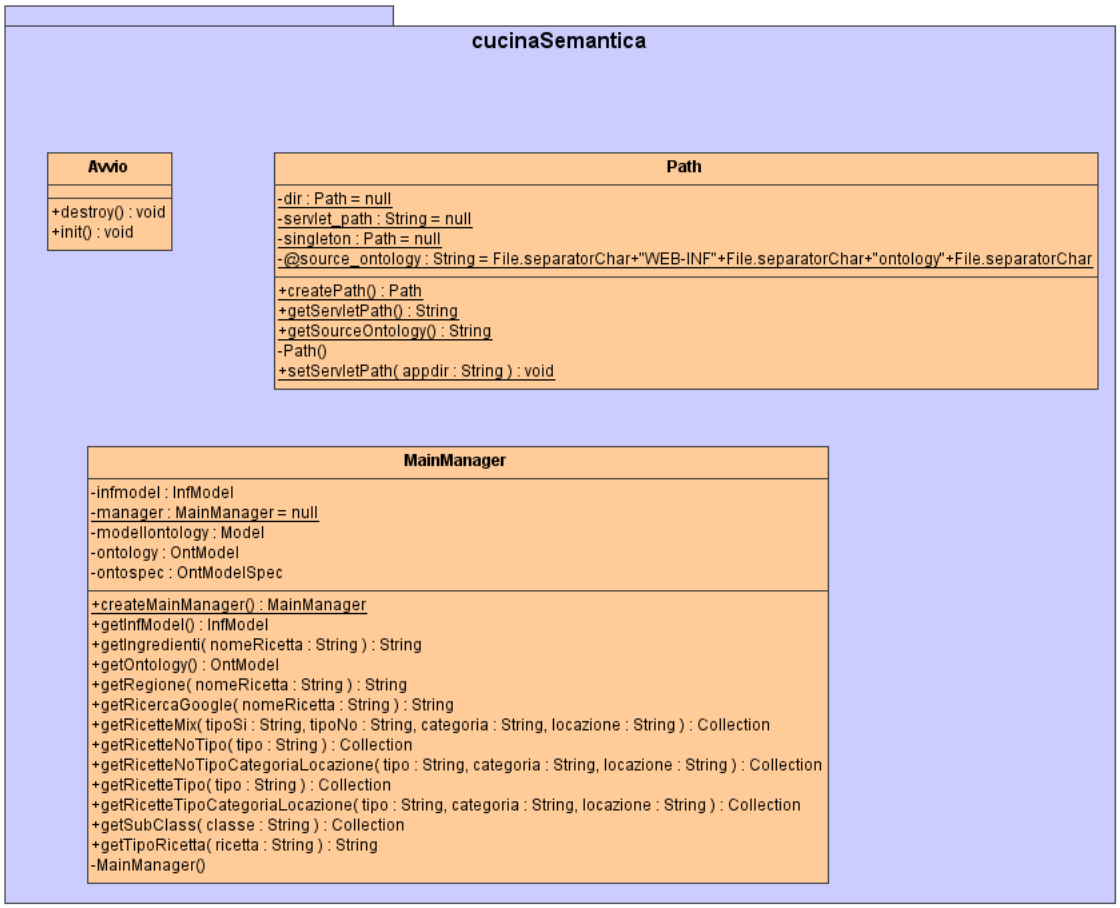

Figura 4.8: Class Diagram

In figura 4.8 viene riportato il package *cucinaSemantica* contenente tre classi, che mettono a disposizione dell'applicazione web una serie di metodi.

Le tre classi sono:

- Avvio;
- MainManager;
- Path.

Che nel seguito verranno descritte nel dettaglio di tutti i metodi e le proprietà di ogni singola classe.

4.6.1 Avvio

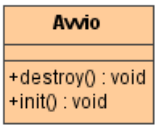

Figura 4.9: Avvio

Questa classe in realtà è una *servlet* che viene invocata automaticamente all'avvio del servlet-container. Lo scopo di questa servlet è quello di impostare alcuni oggetti della classe Path (vedi 4.6.3 a pag. 64). I suoi metodi sono soltanto *init()* e *destroy()*.

Il primo metodo avrà il compito di inizializzare gli oggetti della classe Path, il secondo metodo, invece, viene invocato dal servlet-container in fase di chiusura.

Non vi sono i classici metodi caratteristici di una servlet (*doGet()*, *doPost()*, *service(. . . , . . .)*, . . .), poichè il suo scopo non è quello di interagire con un client, ma è quello di settare una serie di oggetti utili all'applicazione.

4.6.2 MainManager

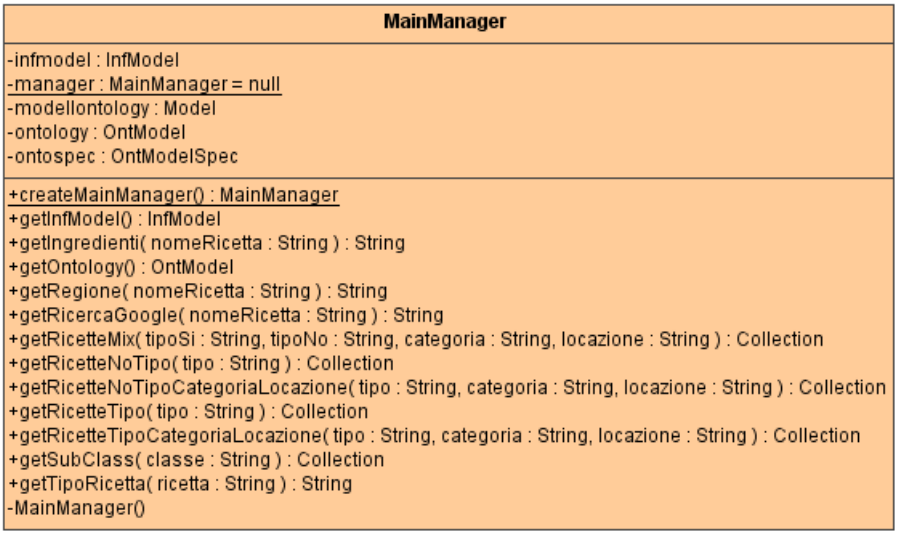

Figura 4.10: MainManager

Questa classe mette a disposizione una serie di metodi utili per tutte le paginejsp, infatti ha il compito di agire da interfaccia tra il mondo, le pagine web che mostrano i risultati delle interrogazioni, e l'ontologia che contiene i dati.

Analizziamo i metodi principali:

createMainManager : questo metodo statico implementa il concetto di *Singleton-Patterns*, ovvero si vuole un'unica istanza di questa classe. Questo per fare si che ci sia un unico oggetto in memoria ad avere accesso all'ontologia e quindi far si che tutte le pagine jsp accedano, tramite questa classe, ai dati

contenuti nell'ontologia.

All'interno di questo metodo viene fatto un controllo sull'oggetto privato *manager*:

```
se manager==null allora
   manager = new MainManager()
return manager
```

Il significato di questo pseudo-codice è il seguente: l'oggetto manager di default vale null, se vale ancora null significa che la classe MainManager non è mai stata richiamata da nessun altra classe o jsp prima di quel momento, quindi si procede alla sua istanziazione e, a questo punto, si restituisce l'istanza.

getIngredienti : questo metodo restituisce una lista di ingredienti di una data ricetta (nomeRicetta), l'implementazione di questo metodo è ottenuto con un'interrogazione rdql;

getRegione : questo metodo restituisce la regione d'appartenenza di una ricetta di cui si conosce il nome (nomeRicetta);

getRicercaGoogle : questo metodo permette di ricercare in rete attraverso il motore di ricerca Google, i documenti che descrivono le ricette. Le "parole" di ricerca sono del tipo: "nome ricetta" + categoria, dove la categoria indica il tipo di ricetta (ad esempio primi o secondi o ...). Nella query di ricerca viene messo anche il tipo perchè abbiamo visto empiricamente che mettere solo il nome della ricetta spesso porta a risultati che non riguardano le ricette e la cucina, questo è proprio dovuto al fatto che il motore di ricerca Google si limita ad effettuare ricerche di tipo pattern-matching e non di tipo semantico, come avviene nella nostra applicazione;

getRicetteMix : questo metodo consente di ricercare nell'ontologia tutte quelle ricette che hanno almeno un ingrediente di tipo *tipoSi* e che non abbiano neanche un ingrediente di tipo *tipoNo*, dove per tipo s'intende ad esempio carne, verdura, ... Inoltre tra i parametri è possibile specificare la categoria a cui deve appartenere la ricetta e la locazione di appartenenza (ad esempio vogliamo che le ricette siano del Nord);

getRicetteNoTipo : questo metodo permette la ricerca di tutte le ricette che non contengono nessun ingrediente di un certo tipo;

getRicetteNoTipoCategoriaLocazione : questo metodo permette di ottenere tutte le ricette che non abbiano neanche un ingrediente di un certo tipo e che la ricetta appartenga ad una data categoria e ad un dato luogo;

getRicetteTipo : questo metodo permette di ottenere tutte le ricette che abbiano almeno un ingrediente di un tipo specificato;

getRicetteTipoCategoriaLocazione : permette di ottenere tutte le ricette che abbiano almeno un ingrediente di un dato tipo e che le ricette appartengano ad una data categoria, inoltre le ricette devono appartenere ad un dato luogo;

getSubClass : questo metodo permette di ottenere la lista delle sotto-classi di una data classe. I cibi e le ricette sono strutturati in più classi, ogni classe può avere più sotto-classi (la relazione è classe *padre* - classe *figlia*). Questo metodo consente di ottenere tutte le sotto-classi (dirette) di una data classe; se ad esempio abbiamo una classe A ha come sotto-classe la classe B e quest'ultima ha come sotto-classe la classe C, se invochiamo il metodo getSubClass(A) otterremo come risultato la classe B ma non la classe C;

getTipoRicetta : questo metodo permette di conoscere il tipo di una data ricetta (dove per tipo s'intende primi, secondi, ...);

MainManager : questo metodo è il costruttore, è ed un metodo privato, poiché viene richiamato solo dal metodo *createMainManager* nel momento in cui viene instanziato in memoria l'oggetto manager;

4.6.3 Path

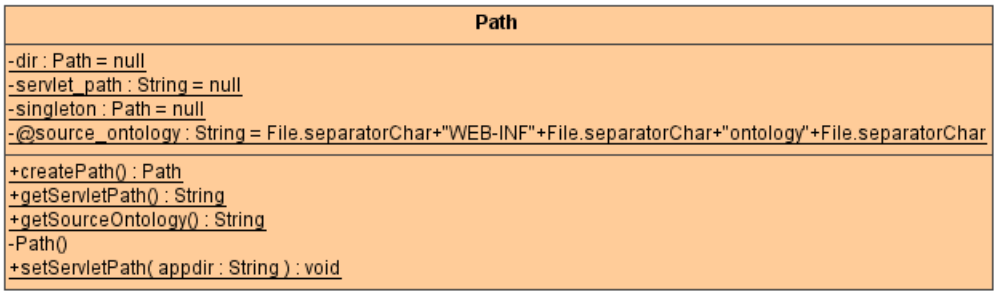

Figura 4.11: Path

Questa classe mette a disposizione una serie di metodi utili per l'individuazione di alcuni percorsi utili per l'individuazione di file (ad esempio l'ontologia) all'interno della web-application. Gli oggetti privati vengono "settati" all'avvio della servlet *avvio* (vedi 4.6.1 a pag. 62).

Questa classe non deve avere dei path dipendenti dalla piattaforma (sistema operativo) su cui viene installata l'applicazione. Infatti una delle punti di forza della nostra applicazione, è il fatto che essa è multipiattaforma (basta che ci sia una *java virtual machine* compatibile con il sistema operativo).

4.7 Deployment Diagram

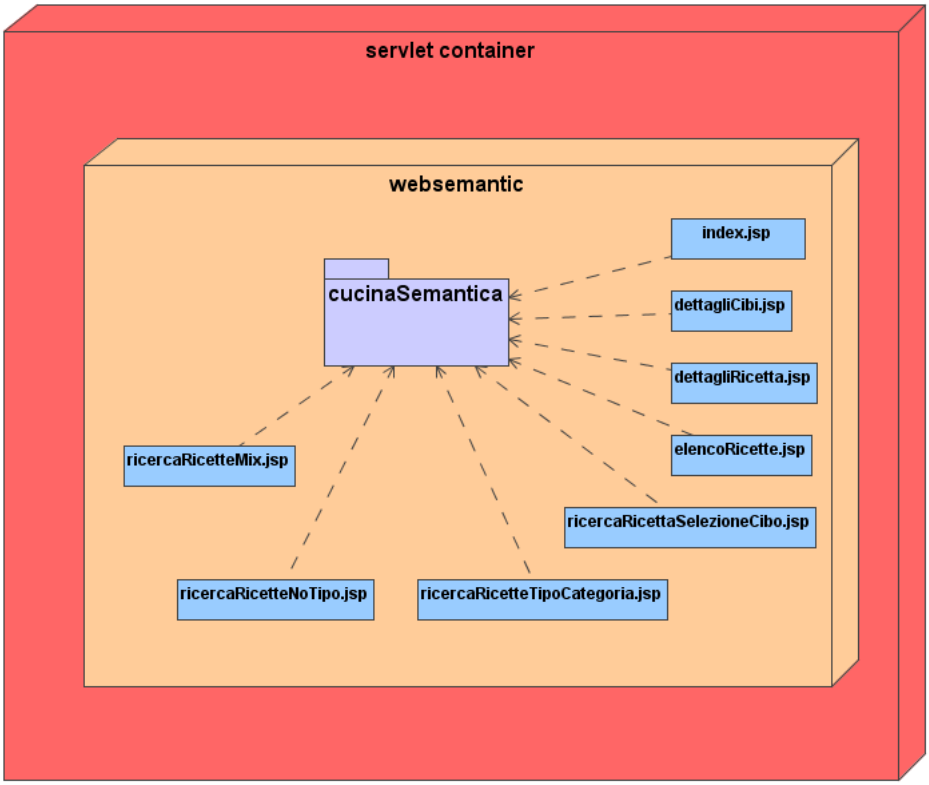

Figura 4.12: Deployment Diagram

Il servlet-container può essere uno qualunque (tomcat, resin, ...), al suo interno, nella cartella web-apps, verrà installata l'applicazione, *websemantic*. Questa contiene una serie di pagine jsp le quali includono il package *cucinaSemantica*, che contine le classi analizzate nei paragrafi precedenti.

Capitolo 5

Implementazione

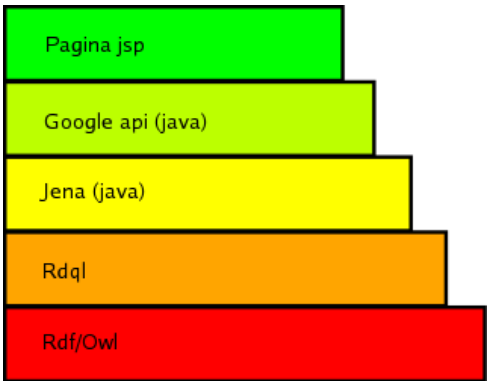

Figura 5.1: Tecnologie utilizzate

5.1 Creazione dell'ontologia

5.1.1 Definizione classi

Per la scrittura dell'ontologia è stato utilizzato il sofware Protegé realizzato nell'università di Standford.
Per la definizione delle classi si sfrutta il tab *OWLClasses*, in questa sezione è possibile definire la gerarchia delle classi e le loro proprietà.

Nella nostra ontologia vengono definite tre macro-classi:

Italia

cibi

ricette

Queste tre macro-classi a loro volta si suddividono ulteriormente: ad esempio, la classe *Italia* è a sua volta suddivisa in *Nord*, *Centro* e *Sud*. La proprietà comune a tutte le regioni è il *nome-regione* che indica appunto il nome di ciascuna regione, ovviamente sarà di tipo *String*. Infatti quando andremo ad istanziare una regione in una delle tre classi, sarà necessario definirne il nome. La classe *cibi* è suddivisa a sua volta in più classi, ciascuna delle quali conterrà cibi di stessa natura, ad esempio nella classe *spezie* verranno inserite le istanze di tutte le spezie. La proprietà comune a tutti i cibi è il *nome-cibo* (anche questo di tipo *String*), infatti, anche qui, nel momento in cui andiamo a definire un nuovo cibo, e identifichiamo la classe di appartenenza, dobbiamo anche specificare il relativo nome. La classe *ricette* è suddivisa al suo interno in

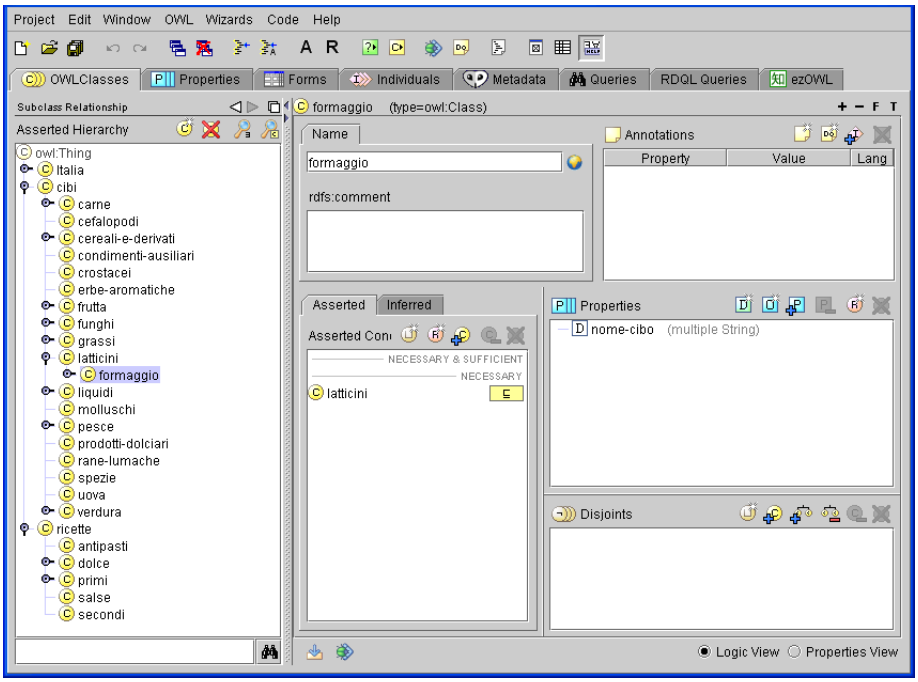

Figura 5.2: Diagramma classi

una serie di classi che descrivono il tipo di ricetta, ad esempio i *primi*, ... Le ricette hanno come caratteristica il *nome-ricetta*, la lista degli *ingredienti* e il *luogo-tipico*. In particolare gli ingredienti possono essere *cibi* o *ricette*, questo perchè tutti i cibi hanno degli ingredienti, ma possono anche avere come ingrediente altre ricette. Ad esempio se definiamo la ricetta della pasta frolla, potrebbe essere utile averla disponibile come ingrediente per una torta. Il luogo-tipico si riferisce alla regione di appartenenza definita in Italia.

Con il software Protegé è possibile in qualunque momento definire una nuova classe, abbiamo quindi deciso di chiamare Italia la classe che contiene le istanze di tutte le regioni italiane. Ciò non toglie il fatto che in un prossimo futuro si possa definire una classe Nazioni che rappresenti tutte le nazioni del Mondo inclusa l'Italia.

In figura 5.2 é mostrata la definizione della classe *formaggi*, il corrispettivo codice owl sarà:

```
<owl:Class rdf:ID="formaggio">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="latticini"/>
    </rdfs:subClassOf>
</owl:Class>
```

Il seguente listato di codice descrive le proprietà:

```
<owl:ObjectProperty rdf:ID="luogo-tipico">
    <rdfs:range rdf:resource="#Italia"/>
    <rdfs:domain rdf:resource="#ricette"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ingredienti">
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#cibi"/>
```

```
            <owl:Class rdf:about="#ricette"/>
          </owl:unionOf>
        </owl:Class>
      </rdfs:range>
      <rdfs:domain rdf:resource="#ricette"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="nome-piatto">
      <rdfs:domain rdf:resource="#ricette"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="nome-cibo">
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <rdfs:domain rdf:resource="#cibi"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="nome-regione">
      <rdfs:domain rdf:resource="#Italia"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

5.1.2 Definizione istanze

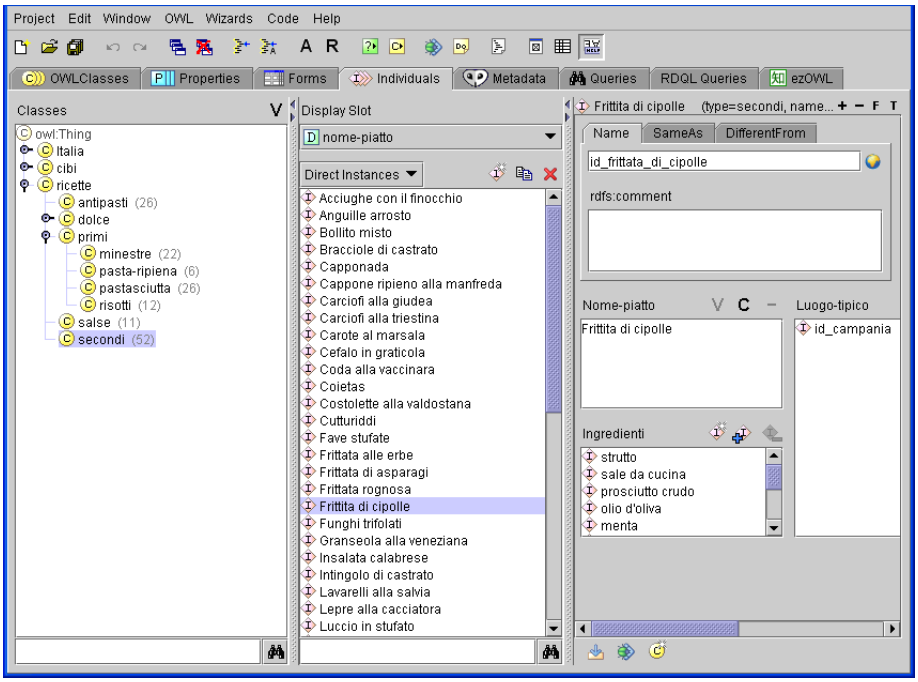

Figura 5.3: Creazione istanze

Adesso analizziamo come poter aggiungere un'istanza in una data classe, per far ciò utilizziamo la sezione *Individuals*, sempre all'interno del Protegé. Quando istanziamo un oggetto in una classe, ci viene chiesto di inserire le relative proprietà. In base alla classe di appartenenza dell'istanza avremo differenti proprietà. Se ad esempio, aggiungiamo la *Frittata di cipolle* alla classe *secondi*, quest'ultima é sotto-classe di *ricette*, dovremo specificare il *nome-piatto*, il *luogo-tipico* e gli *ingredienti*, tutto avviene tramite un'interfaccia grafica.

Il software Protegé provvederà lui a scrivere nell'ontologia il seguente listato di codice:

```
<secondi rdf:ID="id_frittata_di_cipolle">
    <nome-piatto rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
          Frittita di cipolle
    </nome-piatto>
    <luogo-tipico rdf:resource="#id_campania"/>
    <ingredienti rdf:resource="#id_strutto"/>
    <ingredienti rdf:resource="#id_sale"/>
    <ingredienti rdf:resource="#id_prosciutto_crudo"/>
    <ingredienti rdf:resource="#id_olio-oliva"/>
    <ingredienti rdf:resource="#id_menta"/>
    <ingredienti rdf:resource="#id_parmigiano_reggiano"/>
    <ingredienti rdf:resource="#id_cipolla"/>
    <ingredienti rdf:resource="#id_pepe"/>
    <ingredienti rdf:resource="#id_pomodori"/>
    <ingredienti rdf:resource="#id_uova_gallina"/>
    <ingredienti rdf:resource="#id_basilico"/>
</secondi>
```

5.2 Jena

Il package Jena mette a disposizione dell'utente (programmatore) una serie di metodi java per la manipolazione dei dati contenuti in un'ontologia.

Come prima cosa definiamo le librerie che ci serviranno per accedere ai metodi definiti nel package Jena.

```
import com.hp.hpl.jena.ontology.*;
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.rdql.*;
```

Nel frammento di codice seguente viene definito tutto quello che serve per potere accedere ad un'ontologia.

```
Model modellontology = ModelLoader.loadModel("file.owl");
InfModel infmodel =
ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM_RULE_INF, modellontology);
OntModelSpec ontospec = new OntModelSpec(OntModelSpec.OWL_MEM_RULE_INF);
OntModel ontology = ModelFactory.createOntologyModel(ontospec, modellontology);
```

La classe **Model** mette a disposizione una serie di metodi per la creazione di risorse, proprietà e asserzioni, per l'aggiunta e la rimozione di asserzioni dal modello, per l'effettuazione di interrogazioni (query) sul modello. La classe **InfModel** consente un'estensione della classe Model, in particolare si specifica che il modello OWL è memorizzato in memoria e usa le regole di inferenza dell'OWL (*OWL_MEM_RULE_INF*). La classe **OntModel** è un'estensione della classe InfModel e mette a disposizione una serie di metodi per l'accesso ai dati contenuti nell'ontologia.

Adesso vediamo come si possono effettuare query sull'ontologia. Il listato di codice seguente contiene tutti gli oggetti necessari per una interrogazione:

```
String queryString = "...";
Query que=new Query(queryString);
que.setSource(infmodel);
QueryExecution qe = new QueryEngine(que);
QueryResults results = qe.exec();
```

```
for(Iterator iter=results; iter.hasNext();){
    ResultBinding res=(ResultBinding)iter.next();
    Sistem.out.println(res.get("...").toString());
}
qe.close();
```

Come primo punto occorre istanziare la classe **Query** passandogli la queryString, in questo modo sulla queryString verrà effettuato un controllo sintattico dal parser, inoltre con il metodo *setSource* viene specificato il modello contenente i dati. La classe **QueryExecution** permette l'istanziazione dell'*engine* che eseguirà la query.

Successivamente si ottengono i risultati, in particolare si ottiene un'oggetto di tipo **QueryResults**, che è molto simile all'oggetto *ResultSet*[1]. L'estrazione dei dati viene effettuato attraverso l'oggetto **ResultBinding**: quest'ultimo ci permetterà il mapping tra la variabile definita nella queryString e il valore contenuto in essa, in questo modo sarà possibile l'estrazione dei dati.

5.3 Google api

Una volta effettuata la ricerca sull'ontologia, ed avere quindi trovato le ricette che ci interessano, si desidera vedere la descrizione della ricetta a partire dal suo nome. Per far ciò ci appogiamo su uno dei motori di ricerca più importanti al mondo, ovvero Google[2].

Per potere interagire con il web-server occorre un'apposita registrazione, dopodichè si otterrà una password che permette a ciascun utente registrato di poter effettuare fino ad un massimo di 1000 query al giorno, bisogna inoltre scaricare le librerie che permettono l'interazione con il web-server.

Adesso analizziamo il codice che ci permette di inviare al web-server richieste di esecuzione di query, senza dover lavorare direttamente sui pacchetti SOAP[3]:

```
import com.google.soap.search.*;
```

Come prima cosa abbiamo incluso il package contenente gli oggetti che ci permettono la ricerca.

Il seguente listato di codice permette l'invio di una richiesta e la relativa ricezione:

```
GoogleSearch ricercaGoogle = new GoogleSearch();
ricercaGoogle.setKey(password);
String tipoRicetta = "...";
if(tipoRicetta!=null)
    ricercaGoogle.setQueryString("\""+nomeRicetta+"\" "+tipoRicetta);
else
    ricercaGoogle.setQueryString("\""+nomeRicetta+"\" ");
GoogleSearchResult risultati = ricercaGoogle.doSearch();
GoogleSearchResultElement[] elementi=risultati.getResultElements();
for(int i=0; i<elementi.length; i++){
    System.out.println(elementi[i].getURL());
}
```

La classe **GoogleSearch** permette l'interazione con il web-server, infatti attraverso essa è possibile specificare la password (ottenuta dalla registrazione) e la query. Nella nostra applicazione la query è del tipo:

"Agnolotti al tartufo"+primi

[1] Tipico oggetto ottenuto nell'esecuzione di query su database mediante jdbc/odbc

[2] http://www.google.it

[3] Le interazione con il web-server avvengono tramite pacchetti SOAP, sono semplicemente dei DTD che contengono la richiesta, a seguito di questa si ottiene a sua volta una risposta sempre di tipo DTD

ovvero specifichiamo il nome della ricetta tra apici e aggiungiamo la categoria di appartenenza della ricetta, questo perché abbiamo sperimentato empiricamente che mettere solo la ricetta senza apici porta a risultati che spesso non sono inerenti alla ricetta che cerchiamo, ad esempio (riferendoci alla query sopra) troviamo pagine web che descrivono i tartufi ma non la ricettta che cerchiamo; viceversa aggiungendo gli apici otteniamo risultati che si avvicinano molto alle ricette che vogliamo avere ma spesso, anche in questo caso, otteniamo risultati che non sono propriamente corretti, questo è dovuto al fatto che alcune ricette hanno il nome con duplice significato e quindi spesso i risultati ottenuti sono al di fuori del ambito della cucina.

A questo punto effettuiamo la ricerca invocando il metodo *doSearch()*. La classe **GoogleSearchResultElement** rappresenta un singolo risultato della ricerca, è possibile dalla classe GoogleSearchResult ottenere i risultati sotto forma di array di tipo GoogleSearchResultElement.

5.3.1 SOAP (Simple Object Access Protocol)

SOAP è un protocollo per lo scambio di messaggi XML tra le applicazioni. Inizialmente proposto da Microsoft e da IBM, oggi è invece un progetto "open source" in fase di standardizzazione presso il WorldWideWebConsortium(www.w3c.org).

SOAP definisce solo la struttura del messaggio ed alcune regole per elaborare quest'ultimo, rimanendo quindi ad un alto livello e completamente indipendente dal protocollo di trasporto sottostante. Per il trasporto dei messaggi di richiesta e risposta utilizza il protocollo HTTP sul quale i parametri e i comandi sono codificati con il linguaggio XML.

SOAP è composto da tre parti:

- SOAP *envelope construct*: definisce la struttura del messaggio;
- SOAP *encoding*: definisce il meccanismo usato per lo scampio dei dati;
- SOAP *RPC*: consente di invocare un metodo su un oggetto remoto (Remote Procedure Call).

Il principale obbiettivo di SOAP è garantire semplicità ed estensibilità, sfruttando le caratteristiche dei sistemi di rete senza tener conto del protocollo.

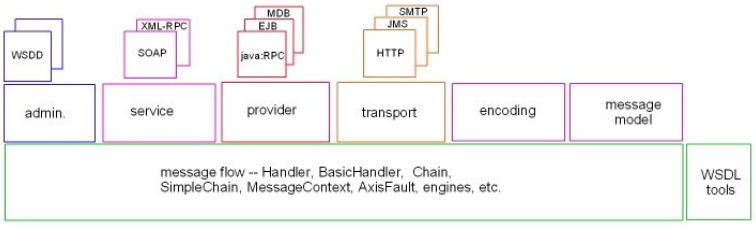

Figura 5.4: Tecnologie associate ad Axis

Un messaggio SOAP, che è un documento XML, deve contenere le seguenti parti:

Header (opzionale): consente l'aggiunta di caratteristiche al messaggio SOAP, quali l'*action*, ...;

Envelope (obbligatorio): costituisce l'elemento principale e rappresenta il messaggio all'interno del quale si trova il *Body* (obbligatorio). L'Envelope contiene quindi le informazioni necessarie al destinatario del messaggio e le informazioni sulla struttura e i contenuti dell'oggetto remoto richiesto;

WSDL (Web Services Description Language).

Il WSDL consente ai service provider di descrivere il formato delle richieste da inoltrare ai web services. Rappresenta il contratto tra cliente e fornitore di servizi. Il file WSDL associato ad un servizio web descrive cosa esso può fare (*metodi* e *parametri da invocare*, *valori di ritorno*,

...), dove esso risiede e come invocarlo. Gli elementi che compongono una definizione WSDL sono:

- i tipi (*types*): contiene le definizioni dei tipi di dati non predefiniti;
- i messaggi (*messages*): contiene i parametri di richiesta e di risposta del servizio;
- le operazioni (*portType*): descrive l'insieme delle operazioni supportate, ognuna delle quali costituita dai messaggi di richiesta e di risposta;
- i collegamenti (*binding*): descrivono il protocollo supportato, le operazioni consentite ed i relativi input ed output;
- la definizione del servizio (*services*): rappresenta la locazione del binding ad esso associato.

L'UDDI (Universal Description Discovery and Integration) permette di localizzare e conoscere i servizi offerti dalle aziende iscritte, come se fossere delle "*pagine gialle*" residenti sul web.

UDDI si basa sul protocollo SOAP in quanto sia le richieste che le risposte sono oggetti UDDI, incapsulati all'interno di messaggi SOAP. Più precisamente si tratta di una specifica per registry di informazioni sui servizi web (quale ad es. i dettagli per collegarsi ad esso): i fornitori di servizi e i servizi stessi sono descritti in XML. Lo scenario nel quale trova pieno utilizzo l'UDDI può essere ricondotto a tre fasi fondamentali:

- pubblicazione (*publising*): il fornitore del servizio per renderlo pubblico contatta il service broker, che provvede ad inserirlo nel registry tramite UDDI;
- ricerca (*finding*): alla richiesta di un servizio, il service broker provvede a cercare quelli che meglio rispondono alle esigenze del richiendente;
- collegamento (*binding*): nel momento in cui il service broker fornisce la risposta può stabilirsi il colleggamento tra il richiedente del servizio web e il fornitore.

Il corretto funzionamento di un qualsiasi web service richiede la presenza di specifiche librerie per la gestione dell'XML, SOAP e WSDL, quali **xerces.jar** e **axis.jar** e **wsdl4j.jar**.

Dopo aver importato le librerie necessarie e avere operato i giusti settaggi per una corretta esecuzione dele applicazioni, si deve implementare una classe java che rappresenta il web service. Detta classe, costituita dal file java e non dal consueto file bytecode, verrà posta all'interno della directory del web server adibita alla gestione delle richieste soap (axis) e successivamente rinominata con l'estensione jws (JavaWebServices). Con tale estensione si otterrà da parete di Axis il riconoscimento del web service, permettendone così la generazione del corrispondente codice xml relativo al descrittore WSDL, nonché la sua gestione come servizio erogato.

la richiesta dell'interfaccia wsdl ad un web service si effettua apponendo il suffisso "?WSDL" all'URL relativo al servizio in oggetto (es. http://host:port/axis/MyService.jws?WSDL). In tal modo, nel framework axis, oltre a ricavare le informazioni relative al suo descrittore WSDL, avviene anche la compilazione della classe e la generazione del file contenente il bytecode.

Ricavate le specifiche tecniche del servizio web, si può procedere alla realizzazione di un client fruitore.

Di seguito si riporta un'estratto di codice java relativo all'invocazione di un webService:

1. String endpoint = "http://host:port/MyService.jws";
2. Service service = new Service();
3. Call call = (Call) service.createCall();
4. call.setTargetEndpointAddress(new java.net.URL(endpoint));
5. call.setOperationName("*metodoDaInvocare*");
6. String ret = (String) call.invoke(new Object[]*parametri*);

Alla riga 1 viene inizializzata la variabile *endpoint* con l'URL (Uniform Resource Locator) relativo al web service da invocare. Nelle tre righe successive viene effettuata la chiamata all'indirizzo sopra riportato.
Alla riga 5, con il metodo *call.setOperationName* viene specificato come parametro d'ingresso il metodo da chiamare attraverso il web service.
La riga 6 rappresenta l'invocazione del web service tramite la quale vengono passati eventuali parametri richiesti dal metodo esposto nell'interfaccia wsdl. L'eventuale risposta del servizio web verrà castata esplicitamente al tipo relativo definito dal descrittore wsdl secondo quelli codificati dalle specifiche SOAP.
A questo punto il client ha ottenuto una rappresentazione nel linguaggio specifico invocazione (java) del dato ritornato. Questa è la grande rivoluzione SOAP: la serializzazione di oggetti appartenenti a qualsiasi linguaggio in una sintassi universale.

5.4 Pagine jsp

In questa sezione analizzeremo le pagine jsp che compongono la nostra applicazione web, che sono le pagine che vengono consultate effettivamente dagli utenti.

5.4.1 Home page

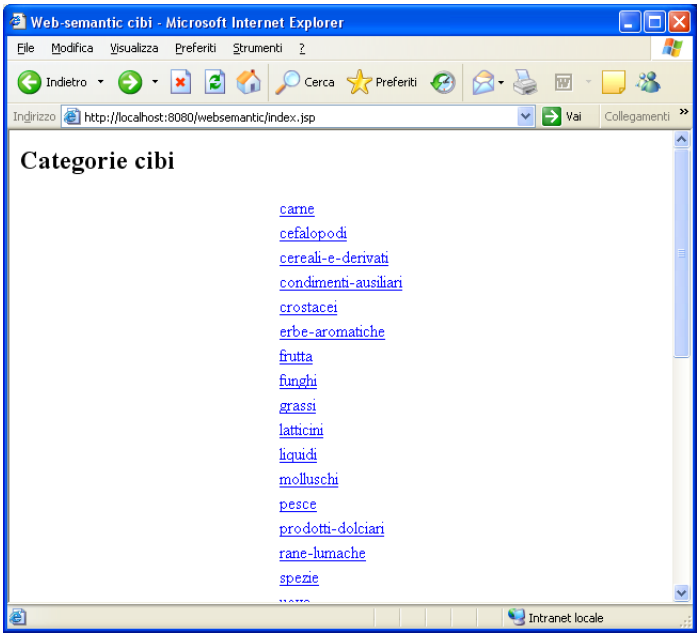

Figura 5.5: Home page

Nella figura 5.5 è mostrato la schermata iniziale della nostra applicazione funzionante, la Home Page.

La Home Page è stata suddivisa in tre sezioni:

- Categorie cibi;
- Tipologie di piatti;
- Query

Come chiaramente si vede dalla figura 5.5, la schermata di partenza mostra nella prima sezione l'elenco completo di tutte le categorie di cibi ed in particolare pone in evidenza, di fatto, tutti gli alimenti contenuti e rappresentati nella nostra ontologia.
Vi è un'intestazione a cui segue una lista di **categorie di cibi**. Ognuna di queste categorie di cibi compare sottoforma di link, che portano ad una successiva pagina jsp contenente tutta la lista dei cibi appartenente a quella data categoria (vedi 5.4.2 a pag. 77).

In figura 5.6 viene mostrata la parte "bassa" dell'Home page, che comprende la altre due sezioni:

Tipologie piatti : ovvero si possono visionare tutte le ricette appartenenti ad una data tipologia di piatti, ad esempio è possibile vedere tutte le ricette che sono primi piatti;

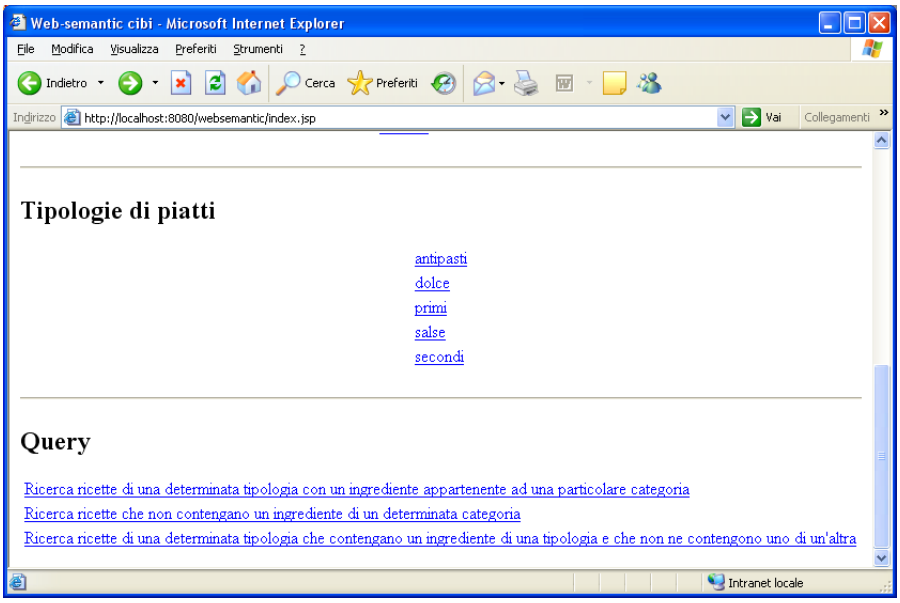

Figura 5.6: Home page, seconda schermata

Query : in questa sezione ci sono a disposizione tre link che portano a tre pagine di ricerca. Queste consentono di effettuare ricerche "semantiche": ad esempio è possibile vedere tutte i primi piatti che sono dell'Italia settentrionale e che contengono almeno un ingrediente di tipo verdura.

5.4.2 Dettagli cibi

Ora se noi andiamo a cliccare su uno degli elementi della lista, come ad esempio carne, appare a video una schermata come quella in figura 5.7

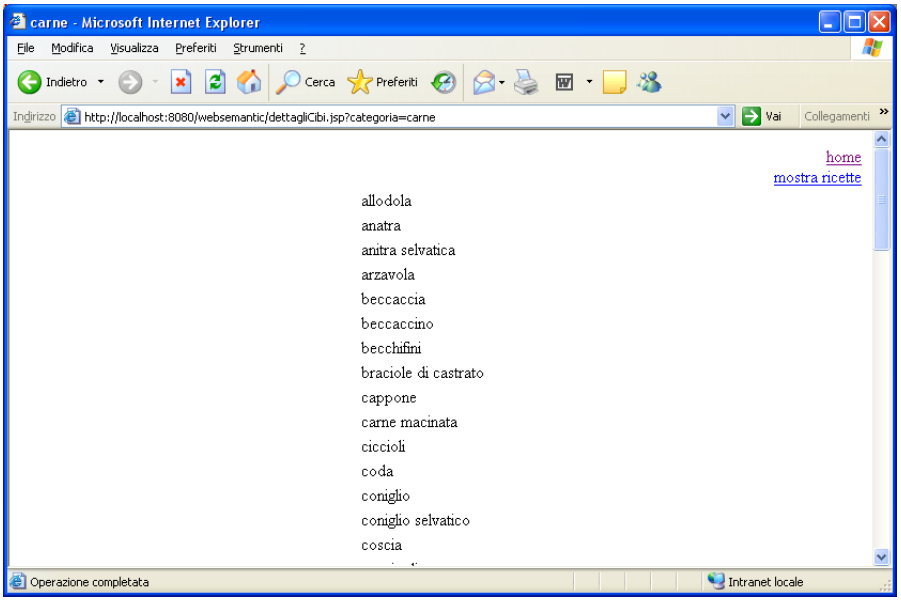

Figura 5.7: Lista dei cibi della categoria carne

Appare subito in evidenza come la lista che ci viene mostrato degli elementi di ogni categoria di cibi mostra tutti gli elementi di tale tipologia di alimento, indipendentemente dall'appartenenza ad una sottoclasse della classe principale carne.

Ad esempio, se vediamo il seguente stralcio della categoria carne:

- allodola
- anatra
- anitra selvatica
- arzavola
- beccaccia
- beccaccino
- becchifini
- braciole di castrato
- cappone
- carne macinata
- ciccioli
- coda
- coniglio
- coniglio selvatico

Vediamo come un elemento della sottoclasse selvaggina da penna, come l'allodola, è messo insieme ad esempio con elementi come il cappone, che fa parte della sottoclasse pollame, o anche le braciole di castrato, che fanno invece parte della sottosottoclasse suini della sottoclasse carni da macello.

All'utente interessa solo vedere la lista di tutti gli alimenti che appartengono alla classe carne, ma non gli interessa andare nello specifico tipo di carne.

Una coppia di link posti nella parte destra della pagina (sia in alto che in basso) ci consentono di raggiungere rapidamente l'Home page, oppure cliccando sull'altro link (*mostra ricette*) è possibile raggiungere una successiva pagina che mostra tutte le ricette aventi per ingredienti uno o più degli ingredienti appartenti alla stessa categoria di cibo (vedi il paragrafo 5.4.5 a pag. 81).

5.4.3 Elenco ricette

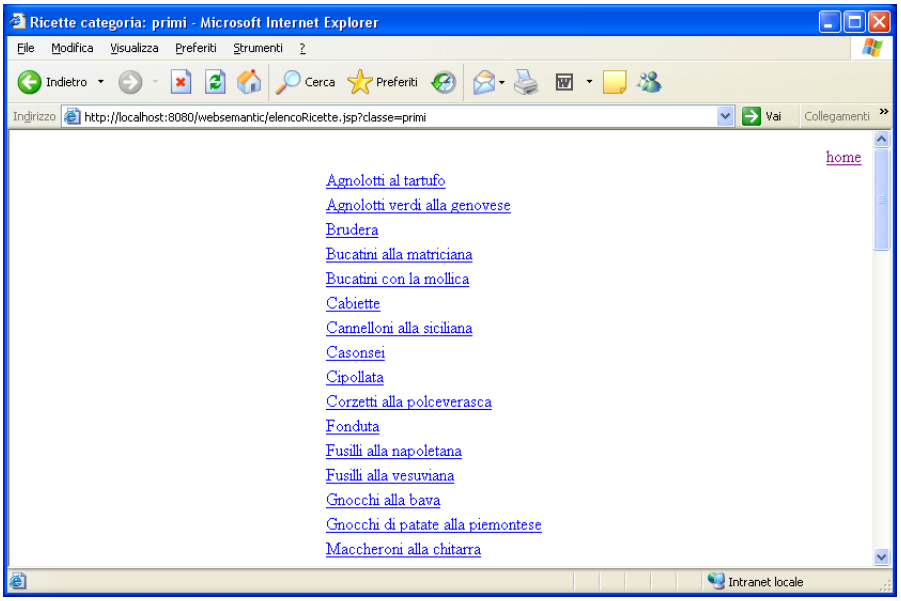

Figura 5.8: Lista della categoria primi

In questa pagina jsp è possibile vedere tutti le ricette appartenenti ad una determinato tipo di piatto. Ad esempio in figura 5.8 vengono mostrate tutte le ricette che appartengono alla categoria primi.

Vengono mostrati tutti i nomi delle ricette sotto forma di link, cliccando sulla ricetta è possibile vedere nel dettaglio la ricetta (regione di provenienza, ingredienti, ...). Nel paragrafo 5.4.4, a pag. 80, viene mostrato in dettaglio le caratteristiche di una ricetta.

5.4.4 Dettagli Ricetta

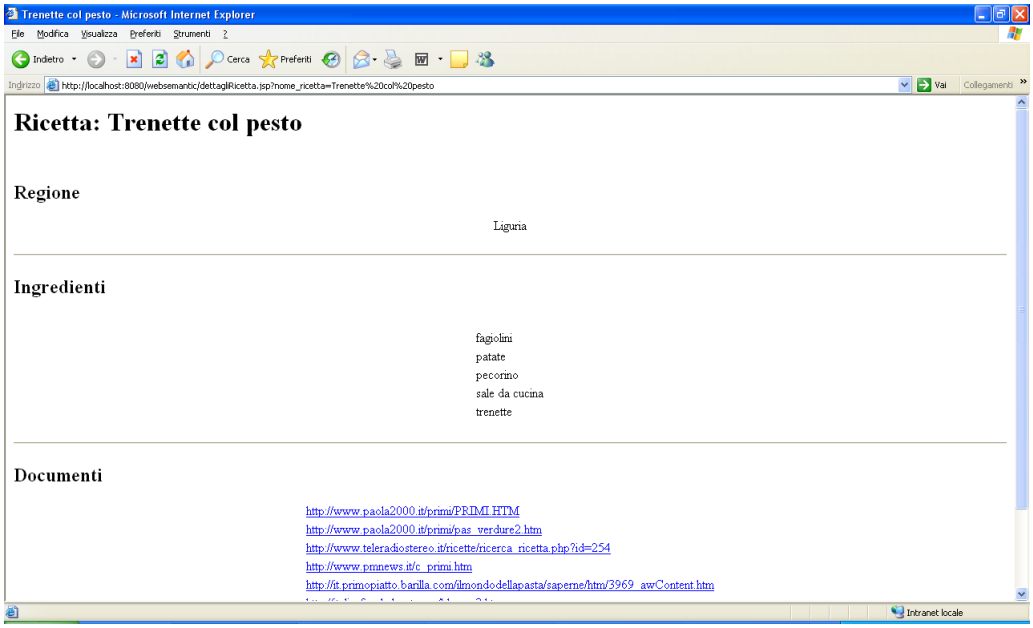

Figura 5.9: Dettagli ricetta

In questa pagina è possibile vedere in dettaglio le caratteristiche di una ricetta.

Nell'esempio in figura 5.9 viene mostrata in dettaglio le caratteristiche dell **Trenette col pesto**, in particolare vengono mostrate le seguenti informazioni:

Regione : ovvero la regione tipica di provenienza della ricetta;

Ingredienti : sono gli ingredienti della ricetta;

Documenti : sono link che portano a pagine web in cui viene descritta la ricetta. Questi link sono il risultato di una query effettuata con Google, quindi il numero di questi link può variare nel tempo, per il fatto che vengono determinati al momento della richiesta della descrizione della ricetta.

5.4.5 Ricerca ricette selezione cibo

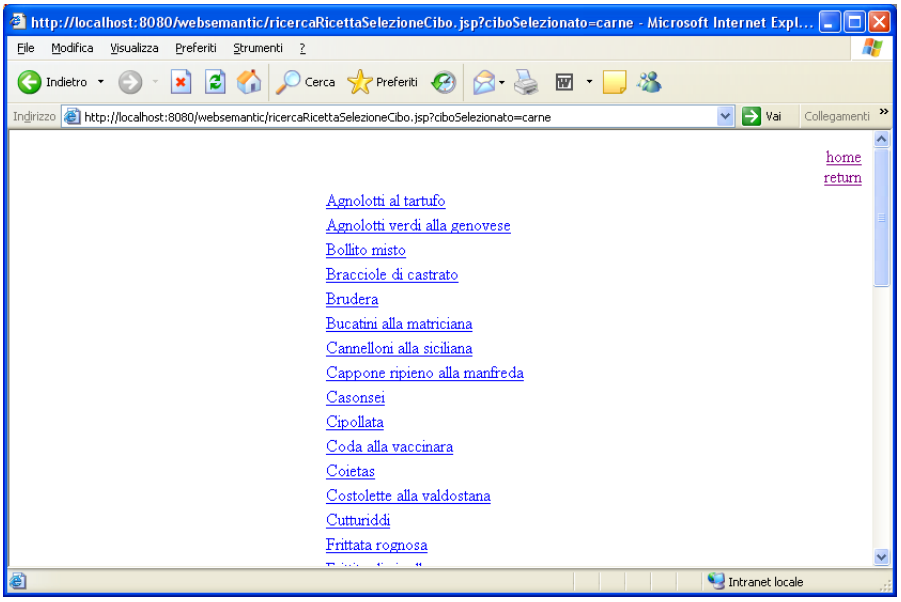

Figura 5.10: Lista delle ricette che contengono la carne

Questa pagina jsp consente di vedere tutte le ricette in cui almeno un ingrediente appartiene ad una data categoria di cibo. Nell'esempio, in figura 5.10, sono mostrate tutte le ricette che hanno almeno un ingrediente di tipo carne. Le ricette sono di ogni tipo (primi, secondi, ...), l'importante è che abbiano almeno un ingredienti appartenente ad una determinata categoria di cibo.

Vengono mostrati i nomi delle ricette sotto forma di link, cliccandoci sopra è possibile vedere nel dettaglio la ricetta (vedi 5.4.4 a pag. 80). Nella parte destra della pagina (sia in alto che in basso) compaiono una coppia di link che ci consentono di ritornare immediatamente all'home page (vedi 5.4.1 a pag. 75), oppure di ritornare alla pagina precedente, ovvero la pagina che mostrava tutti i cibi appartenenti ad una data categoria di cibo (vedi 5.4.2 a pag. 77).

5.4.6 Query

Ricerca ricette tipo categoria

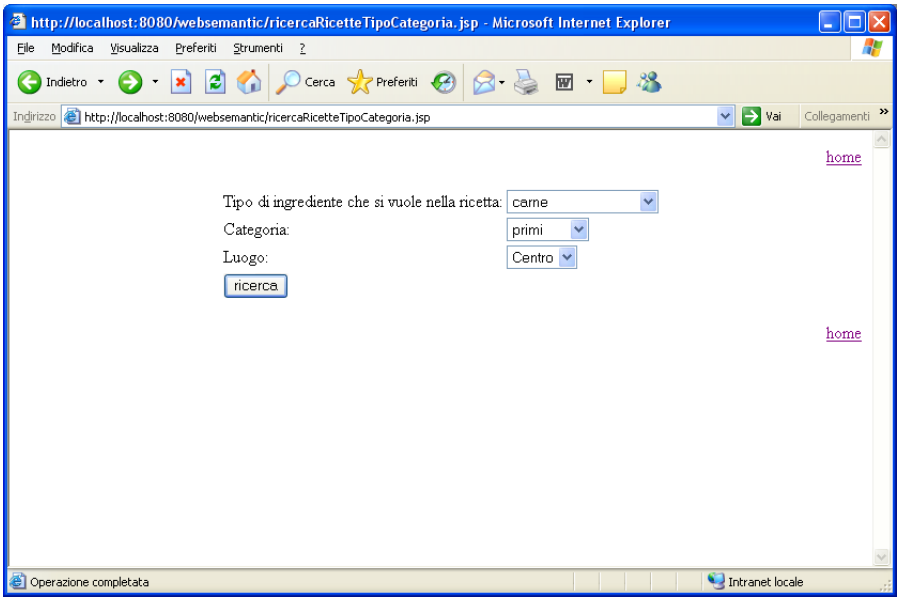

Figura 5.11: Esempio di funzionamento della prima query

In questa pagina viene proposta la possibilità di cercare tutte le ricette che soddisfino certi criteri, in particolare si vogliono delle ricette che abbiano **almeno** un ingrediente appartenente ad una determinata categoria, inoltre si vuole che le ricette appartengano ad una certa tipologia di piatti (ad esempio *primi*) ed infine le ricette devono essere tipiche di un'area geografica (ad esempio *Nord*).

Nell'esempio in figura 5.11 si cercano tutti i primi piatti che contengano almeno un ingrediente di tipo carne e che come regione d'appartenenza sia situata al Centro-Italia.

Se clicchiamo sul bottone *ricerca* otterremo il seguente risultato: Ovvero una serie di nomi

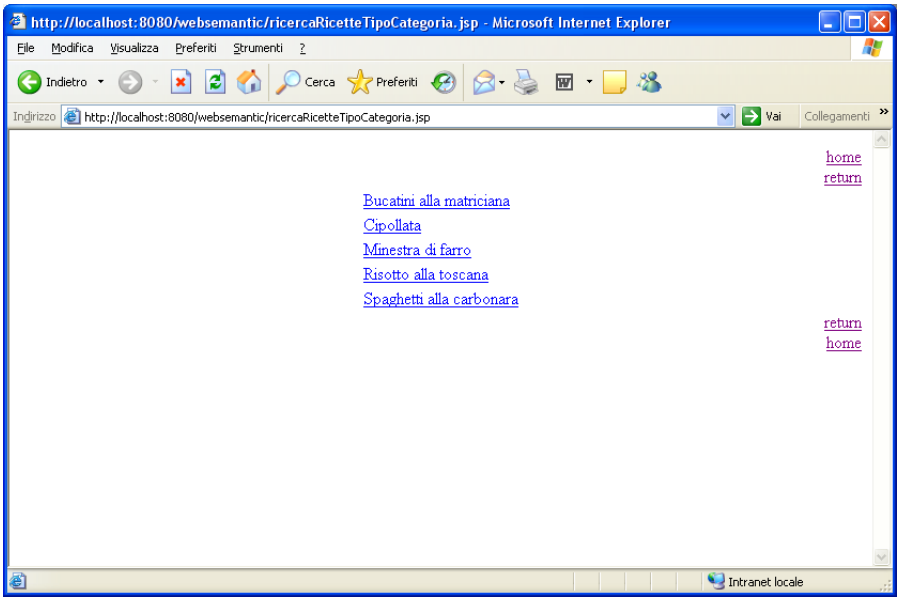

Figura 5.12: Risultati della prima query

corrispondenti alle ricette, ove cliccandoci sopra è possibile vedere nel dettaglio le ricette.

Nella parte destra della pagina vengono mostrate un paio di link:

home : consente di ritornare direttamente all'Home page (vedi 5.4.1 a pag. 75);

return : consente di ritornare alla pagina di ricerca.

Ricerca ricette no tipo

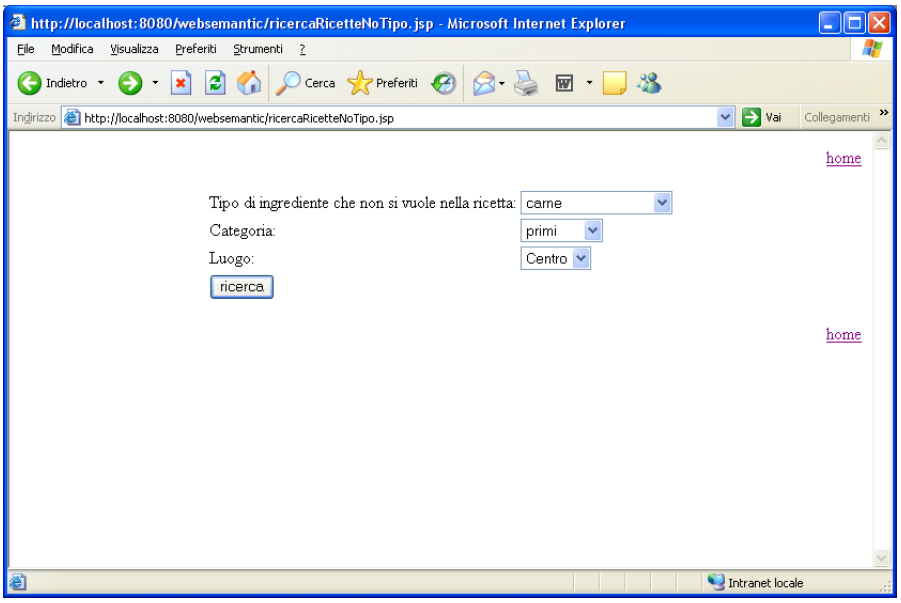

Figura 5.13: Esempio di funzionamento della seconda query

In questa pagina viene proposta la possibilità di cercare tutte le ricette che **non** contengono neanche un ingrediente di un tipo specificato, inoltre le ricette devono appartenere ad un determinata tipologia di piatto ed infine devono essere tipiche di una determinata area geografica.

Nell'esempio in figura 5.13, cerchiamo tutti primi piatti che non contegano neanche un ingrediente di tipo carne e che come regione tipica sia nel Centro-Italia.

Se clicchiamo sul bottone ricerca otterremo il seguente risultato:

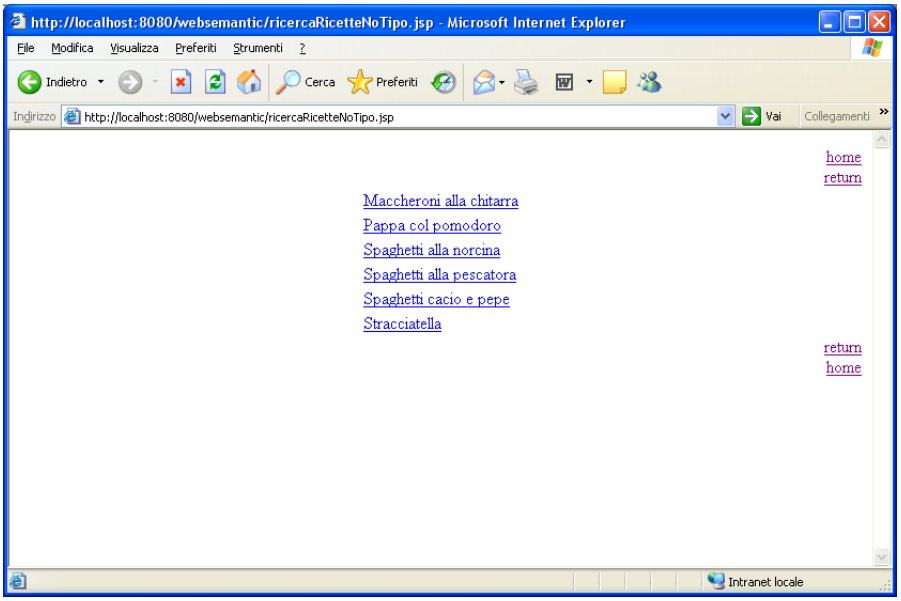

Figura 5.14: Risultati della seconda query

Notiamo subito che tutte le ricette trovate sono differenti rispetto alla precedente query (vedi 5.12 a pag. 85). Questo perché la query è l'esatto opposto della precedenti, infatti, in entrambe le query cerchiamo dei primi piatti del Centro-Italia, ma mentre prima dovevano contenere almeno un ingrediente di tipo carne, adesso vogliamo che neanche un ingrediente sia di tale tipo.

Ricerca ricette mix

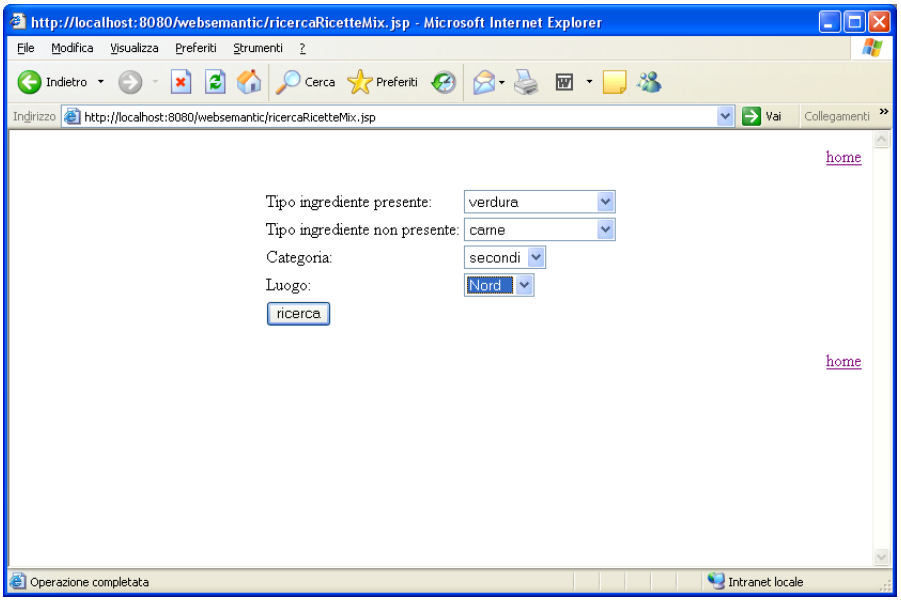

Figura 5.15: Esempio di funzionamento della terza query

In questa pagina viene proposta la possibilità di ricercare delle ricette appartenenti ad una determinata categoria e luogo (come nelle precedenti query) e che contengano **almeno** un ingrediente di un certo tipo e che **non** ne contengano neanche uno di un altro tipo.

Nell'esempio in figura 5.15 vengono cercati tutti i secondi piatti la cui regione d'appartenenza è situata al Nord e che abbiano almeno un ingrediente di tipo *verdura*, ma che non contengano neanche un ingrediente di tipo *carne*. Una ricerca di questo tipo è pensabile ad esempio per una persona vegetariana, la quale non gradisce mangiare piatti a base di carne. Da notare che se proviamo a cercare tutte le ricette che non contengono la carne (ad esempio usando la query vista in 5.13 a pag. 84) i risultati che otteniamo sono differenti da quelli che otteniamo con questa query. Questo perché adesso oltre a richiedere la non presenza di ingredienti di tipo carne, si vuole la presenza di almeno un ingrediente di tipo verdura, cosa che nella query vista in 5.13 non era possibile fare.

Se clicchiamo sul bottone ricerca otterremo i seguenti risultati:

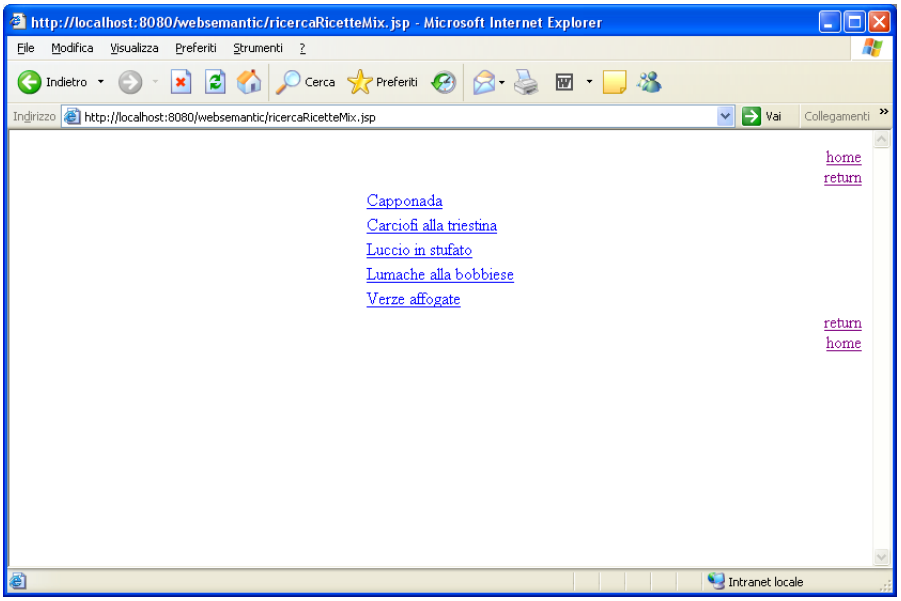

Figura 5.16: Risultati della terza query

In figura 5.16 vengono mostrate le ricette che soddisfano i vincoli imposti precedentemente (vedi 5.15).

Capitolo 6

Risultati e Sviluppi futuri

6.1 I Risultati

I contenuti di un applicazione ipermediale, del tipo che abbiamo progettato e realizzato, devono soddisfare le aspettative dell'utente. Una persona accede ad un sito per trovare delle informazioni, delle risorse, per ottenere un servizio; se non trova quello che cerca, quello che si aspetta, si rivolgerà altrove.

E' nostra opinione che il giudizio degli esperti non sia sufficiente a garantire ad un progetto la piena efficacia; una valutazione seria e completa deve prevedere la partecipazione di utenti tipici, in veste di "collaudatori".

Infatti, è molto importante ottenere un feedback relativo alle loro esperienze ed impressioni, per conoscere le difficoltà incontrate da persone che, essendo rimaste estranee alla fase di progettazione ed essendo in possesso di competenze inferiori a quelle di un esperto, sono in grado di rilevare eventuali mancanze sfuggite all'occhio di chi, a contatto quotidiano con i siti del Web, ne ha assorbito il modello concettuale sottostante.

L'utente tipico, quindi, è indispensabile:

- per mettere al riparo il valutatore esperto dal pericolo di lasciarsi sfuggire particolari cui abbia fatto l'abitudine con l'uso;
- per ottenere dati oggettivi e sottoponibili ad analisi quantitative;
- per garantire il rispetto del terzo punto della definizione ISO di usabilità, riguardante il parametro della piacevolezza e soddisfazione d'uso.

Abbiamo fatto provare il nostro software su un campione di 12 persone e abbiamo riscontrato che più dell'80% hanno ottenuto risultati soddisfacenti, ovvero sono riusciti ad ottenere una serie di ricette che li soddisfasse, cosa che con un normale motore di ricerca non sono riusciti.

Inizialmente gli utenti non capivano i vantaggi portati dal nostro motore di ricerca semantico, sostenendo che con il "loro google" potevano ottenere tutte le ricette che volevano, allora abbiamo provato a fargli la seguente domanda:

"Siete in grado di trovare una serie di ricette che non contengano carne, ma che contengano almeno un ingrediente di verdura?"

Subito gli utenti ci hanno risposto che era una domanda facilissima da risolvere, ma ancora non sapevano cosa li aspettava .... Dopo breve si sono accorti che una domanda, all'apparenza così semplice ed innocua, portava dietro sè una multitudine di problematiche.

Infatti nessun utente è riuscito a risolvere la nostra semplice domanda con un normale motore di ricerca, perché nel momento in cui cercavano le ricette a base di verdura ma che non contenessero alcun tipo di carne, ad esempio, spesso i risultati ottenuti erano incoerenti.

Questo perché il motore di ricerca cerca tutte le pagine che contengono la parola verdura ma non la parola carne, così ad esempio molti utenti hanno trovato la ricetta delle "Costolette di maiale" (piatto tipicamente a base di carne) come risultato della query!!!!

A seguito di ciò gli utenti hanno rivalutato il nostro software e hanno iniziato a farci domande su come fossimo riusciti a implementare un motore di ricerca così "intelligente".

Gli utenti ci hanno fatto notare che con un normale motore di ricerca è abbastanza difficile ottenere una lista di ricette conforme alle loro richieste, in particolare ci hanno fatto notare che per lo più ottenevano pagine web che spaziano dalle descrizioni di alcuni cibi, ai forum dove per caso viene menzionata la parola "ricette" o a siti di medicina dove si consiglia di mangiare certi cibi piuttosto che altri.

Di particolare interesse è il fatto che ci hanno fatto notare che le ricette più difficili da trovare con un normale motore di ricerca, sono quelle che non contengono neanche un ingrediente di un certo tipo. Ad esempio alcuni utenti hanno provato a cercare dei secondi piatti che non contenessero delle spezie e spesso ottenevano per risultato delle ricette che comunque contenevano degli ingredienti di tipo spezia. Questo perché quando effettuavano la ricerca mettevano nella query di ricerca **-spezie**, così facendo però ottenevano per risultato pagine web che non contengono la parola spezie, ma non che non contengono le spezie. D'altra parte è impensabile che un utente si metta a scrivere: **-pepe -cannella -...**, ovvero si metta ad elencare tutte le spezie.
Un utente è rimasto molto sorpreso quando a seguito di una ricerca ha ottenuto una serie di ricette e nel momento in cui visionava la ricette ha visto che la descrizione della ricetta era data da una serie di link ottenuti tramite Google. Infatti ci hanno domandato perché noi riusciamo ad ottenere dei risultati soddisfacenti con il nostro software che si appoggia ad un normale motore di ricerca, mentre quando loro cercavano di fare la stessa cosa direttamente con Google non ci riuscivano.

La nostra risposta è stata la seguente:

"Il software che abbiamo creato permette di fare ricerche mirate, ovvero consente di "inquadrare" le ricette e in un passo successivo, attraverso un normale pattern-matching, ottenere la descrizione della ricetta. In sostanza usiamo un normale motore di ricerca soltanto per effettuare ricerche di pura corrispondeza parola con parola."

6.2 Conclusioni

L'uso di RDF è agli inizi e non sono ancora apprezzabili gli effetti che questa tecnologia porterà sul Web. Non sono ancora molti i motori di ricerca che utilizzano RDF per classificare le pagine Web, ma sembra che l'interesse nei confronti di questa tecnologia sia molto alto. Quello che possiamo fare al momento è cercare di immaginare quello che potrà essere il Web con un uso accurato di RDF.

Un primo risultato sarà senz'altro la maggior precisione nella ricerca di informazioni tramite i motori di ricerca. Si apre anche la strada alla realizzazione di agenti software in grado di navigare su Internet alla ricerca di informazioni specifiche, come ad esempio il confronto di prezzi di determinati articoli.

Le ontologie, definendo rapporti di equivalenza tra oggetti, rappresentano la soluzione che elimina possibili ambiguità, garantendo l'univocità delle relazioni. Possono migliorare la rete, aumentare la precisione dei motori di ricerca, ricercando solo quelle pagine che si riferiscono ad un concetto preciso, o affrontare interrogazioni complicate le cui risposte non risiedono in una singola pagina. Le applicazioni più avanzate utilizzeranno le ontologie per collegare le informazioni di una pagina alle regole di inferenza.

Questo stralcio di un intervista[1] a Frank van Harmelen[2] ci fa intravedere l'avvenire e le prospettive del Web Semantic:

> "Predire quando sarà la grande svolta del Semantic Web è davvero una domanda difficile. Predire il futuro è sempre difficile, e lo è in modo particolare per il settore della IT [tecnologia dell'informazione].
>
> Ed in particolare predire gli eventi del World Wide Web è totalmente impossibile; Tim Berners-Lee, il padre del World Wide Web, usa la metafora della slitta da bob: all'inizio dovete spingere parecchio per farlo partire, ma una volta che è partito dovete saltarci dentro in fretta prima che parta senza di voi.
>
> Beh, in questo momento il Semantic Web si trova nella fase della spinta: dobbiamo parlare alle industrie per illustrare in modo convincente i suoi benefici; ma io non dubito che il Semantic Web ci sarà. Anche Tim Berners-Lee vede il Semantic Web come l'unico sbocco possibile per il World Wide Web. Ma lasciatemi specificare meglio: sarei molto deluso se nei prossimi due o tre anni non vedessimo nessuna applicazione per i comuni navigatori, in particolare per quanto riguarda l'area del commercio elettronico questo settore ha molto da guadagnare dalla personalizzazione. La conversione di tutto il Web in un Semantic Web richiederà molto più tempo, ma io sono convinto che avverrà".

In un futuro a noi prossimo, il Semantic Web estenderà le sue funzioni al mondo fisico, costituito da qualsiasi apparecchio elettronico, dal player MP3 alla TV, dal PC al cellulare, individuato dal proprio URI. La condivisione di una semantica comune e di schemi più versatili del PnP, grazie alle relazioni di equivalenza delle ontologie, permetterà alla nuova rete di acquisire più informazioni dai media oggi non accessibili dalla rete, aprendo un mondo di potenzialità che ha ancora un aspetto quasi fantascientifico. L'automazione della casa e il controllo remoto dei componenti saranno soluzioni tangibili: la gestione ottimale del riscaldamento o del condizionatore, ordini di acquisti via web di alimenti da parte del nostro stesso frigorifero e l'acquisto da parte del nostro PC del migliore pacchetto viaggi, secondo i nostri gusti ed impegni in agenda, sono solo alcuni esempi banali di ciò che il Semantic Web potrà offrirci con un intervento umano minimo. I primi passi concreti sono stati già fatti in questo campo, con lo sviluppo di uno standard per la descrizione delle capacità funzionali dei dispositivi e delle preferenze dell'utilizzatore, definendo in RDF il Composite Capability/Preference Profile (CC/PP).

[1] $http://www.cs.vu.nl/\sim frankh/$ (la traduzione dell'intervista è stata approntata da Margherita Benzi)

[2] AI Department, Division of Mathematics and Computer Science, Faculty of Sciences, Vrije Universiteit Amsterdam, de Boelelaan 1081a, 1081HV Amsterdam , The Nether-lands

Si tratta della creazione di profili descrittivi, messi a disposizione sulla rete, per ciascun apparecchio, in modo che da una parte il Semantic Web possa utilizzare tali risorse, e, dall'altra, che queste ultime possano condividere le risorse di Internet.

La potenza reale del Semantic Web sarà osservabile una volta che saranno creati software automatici cooperanti capaci di leggere i contenuti Web. Anche i servizi che non sono stati progettati appositamente per interagire fra loro potranno trasferire i dati che dispongono di semantica.

Le applicazioni potranno scambiarsi le informazioni e i programmi saranno in grado di raccogliere ed elaborare i contenuti del Web. Il Semantic Web, facendo riferimento ad ogni entità con un URI, permetterà a ciascuno di descrivere nuovi concetti, eventualmente basandosi su altri già esistenti e l'utilizzo di linguaggi standard, consentirà alle informazioni individuali di essere collegate al resto del Web. La struttura che si verrà a formare, imperniata sulla conoscenza e sullo studio degli agenti software, fornirà una serie di strumenti nuovi, in grado di comunicare, lavorare e "imparare" insieme.

Bibliografia

[0] Questo link propone l'articolo: Creating Semantic Web Contents with Protege-2000 per la Stanford University
http://www-smi.stanford.edu/pubs/SMI_Reports/SMI-2001-0872.pdf

[1] Questo articolo rimanda a un draft sull'evoluzione del web e sul web semantic
http://www.topquadrant.com/conferences/apr21_WCST/materials/Deborah_McGinness.pdf

[2] Questo link rimanda alla voce enciclopedica del WS riportata dalla Wikipedia
http://en.wikipedia.org/wiki/Semantic_Web

[3] Questo link rimanda alla principale applicazione attualmente disponibile su web basato su un'ontologia
http://wordnet.princeton.edu/wn2.0.shtml

[4] Questo link mostra l'ontologia del Wine Agent della Stanford University
http://www.daml.org/ontologies/76

[5] Questo link indirizza alla home page di Protege coordinato dalla Stanford University
http://protege.stanford.edu/

[6] Questa è la pagina ufficiale di Tim Berbers-Lee, il creatore del Html e del WS
http://www.w3.org/People/Berners-Lee/

[7] Questa è la pagina principale dedicata dal w3c al linguaggio RDF
http://www.w3.org/RDF/

[8] Questo link proprone un OWL Web Ontology Language Overview
http://www.w3.org/TR/owl-features/

[9] Questo link porta al WebServices - Axis
http://ws.apache.org/axis/java/architecture-guide.html

[10] Un'altro link raccomandato dal w3c sull'RDF
http://www.w3.org/TR/1999/REC-rdf-suntax-19990222

[11] Questo link è la home page dell'università degli studi di Trento, include documenti interessanti sul WS
http://dit.unitn.it/welcome

[12] La pagina principale dedicata al WS sul web
http://www.semanticweb.org/

[13] La pagina dedicata dal w3c al linguaggio XML
http://www.w3.org/XML/

[14] Un'altra pagina dedicata dal w3c al linguaggio XML
http://www.w3.org/TR/2000/REC-xml-20001006

[15] L'articolo HP Labs SemanticWeb Research
http://www.hpl.hp.com/semweb/

[16] Tutorial on OWL, an introduction
http://www.cs.man.ac.uk/~horrocks/ISWC2003/Tutorial/

[17] L'articolo di Oreste Signore: Strutturare la conoscenza: XML, RDF, Semantic Web
http://www.weblab.isti.cnr.it/people/oreste/

[18] Sito sulle Web Content Accessibility Guidelines, basato sulle idee di Lisa Seeman
http://ubaccess.com/lisa/rdf-tech-src-july.html

[19] Questo link porta alla pagina principale del W3C sul Web Semantic
http://www.w3.org/2001/sw/

Bibliografia

[1] Bertolotti Paola. Semantic web: analisi e utilizzo di strumenti per la sua realizzazione. Master's thesis, Università degli Studi di Torino, Scienze Matematiche Fisiche e Naturali, 2002-2003.

[2] Dr John Davies, Professor Dieter Fensel, and Professor Frank van Harmelen. *Towards the semantic web: Ontology-driven Knowledge Management.* John Wiley and Sons, ltd, 2003.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. *The Semantic Web.* Scientific American, May 2001.

[4] R. J. Brachman and J. G. Schmolze. *An Overview of the KLONE Knowledge Representation System.* Cognitive Science, 9, 1985.

[5] M. P. Evettand, W. A. Andersen, and J. A. Hendler. *Providing Computationally Effective Knowledge Representation via Massive Parallelism.* Elsevier Science Publishers,, 1994.

[6] M. P. Evett, J. A. Hendler, and L. Spector. *Parallel Knowledge Representation on the Connection Machine.* Journal of Parallel and Distributed Computing, 22(2):168184, 1994.

[7] D. Fensel, O. Lassila, F. van Harmelen, I. Horrocks, J. Hendler, and D. L. McGuinness. *The semantic Web and its languages.* IEEE Intelligent Systems, Nov/Dec 2000.

[8] C. F. Goldbard and P. Prescod. *XML.* IEEE Intelligent Systems, 1999.

[9] J. Heflin and J. Hendler. *Semantic Interoperability on the Web.* In Proc. of Extreme Markup Languages 2000, pages 111120, Alexandria, VA, 2000.

[10] B. Kettler, W. Andersen, J. Hendler, and S. Luke. *Using the Parka Parallel Knowledge Representation System (version 3.2).* Technical Report CS-TR 3485 (UMIACS TR-95-68), Department of Computer Science, University of Maryland, College Park, Maryland, 1995.

[11] E. Rich and K. Knight. *Artificial Intelligence.* McGraw-Hill, 1991.

[12] Frank Manola and Eric Miller. Resource description framework primer. http://www.w3.org/TR/rdf-primer/. W3C Recommendation.

[13] Smith Welty and Deborah McGuinness. Owl web ontology language guide. http://www.w3.org/TR/owl-guide/, 10 Feb 2004. W3C Recommendation.

[14] Dean Schreiber. Owl web ontology language reference. http://www.w3.org/TR/owl-ref/, 10 Feb 2004. W3C Recommendation.

[15] Stefano Emilio Campanini. Introduzione a owl. http://stefano.campanini.info/, 18 maggio 2004.

[16] Gruber T. R. *Toward Principles for the Design of Ontologies Used for Knowledge Sharing.* International Journal of Human and Computer Studies 43(5/6), pp. 907-928, 1995.

[17] Guarino N. *Formal Ontology in Information Systems. Formal Ontology in Informa-tion Systems.* Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998, Amsterdam, IOS Press, pp. 3-15., 1998.

[18] C. Masolo and A. Oltramari. *La prospettiva dell'ontologia applicata.* Rivista di Estetica(22), pp. 170-183, 2003.

[19] W. V. O. Quine. *Ontological Relativity and Other Essays,.* New-York, London, Columbia University Press., 1969.

[20] Van Benthem J. *The Logic of Time,.* Dordrecht Kluwer, 1983.

[21] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle. *The RDF Suite: Managing Voluminous RDF Description Bases.* Institute of Computer Science, FORTH, Heraklion, Greece, 2000. http://www.ics.forth.gr/proj/isst/rdf/rssdb/rdfsuite.pdf.

[22] A. Ankolenkar, M. Burstein, J.R. Hobbs, O. Lassila, D.L. Martin, D. McDermott, S.A. McIlraith, S. Narayanan, M. Paolucci, T.R. Payne, and K. Sycara. *DAML-S: web service description for the semantic web.* In The First International Semantic Web Conference (ISWC), June 2002.

[23] J.C. Arprez, O. Corcho, Fernandez-Lopez, M., and A. Gomez-Perez. *WebODE: a scalable workbench for ontological engineering.* In Proceedings of the First International Conference on Knowledge Capture (K-CAP) October 2123, Victoria, BC, Canada, June 2001.

[24] J. Banerjee, W. Kim, H.-J. Kim, and H.F. Korth. *Semantics and implementation of schema evolution in object-oriented databases.* SIGMOD Record. In Proceedings of the Conference on Management of Data, Vol. 16(3), pp. 311322, 1987.

[25] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. *OilEd: a reasonable ontology editor for the semantic web.* In Proceedings of the Joint German/Austrian Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence, no. 2174, pp. 396408. Berlin: Springer-Verlag, 2001.

[26] R. Benjamins, D. Fensel, S. Decker, and A. Gomez Perez. *(KA)2: building ontologies for the internet: a mid-term report.* International Journal of Human-Computer Studies, 51(3): 687712, 1999.

[27] T. Berners-Lee, J. Hendler, and O. Lassila. *The semantic web.* Scientific American, May, 2001.

[28] O. Corcho and A Gomez-Perez. *A roadmap to ontology specification languages.* In Dien, R. and Corby, O (eds.), Knowledge Engineering and Knowledge Management; Methods, Models and Tools, Proceedings of the 12th International Conference EKAW, Juan-les-Pins, France, October 26 LNCS 1937, pp. 8096., 2000.

[29] T. Berners-Lee, J. Hendler, and O. Lassila. *The semantic web.* Scientific American, May, 2001.

[30] M. Dalal. *Semantics of anytime family of reasoners.* In Wahlster, W. (ed.), Proceedings of ECAI96, pp. 360364., 1996.

[31] A. Das, McGuinness Wu, W., D.L., and A. Cheyer. *Industrial strength ontology management for e-business applications.* In the Proceedings of International Semantic Web Working Symposium (SWWS), July 30August 1, Stanford University, CA., 2001.

[32] S. Decker, M. Erdmann, Fensel, D., and R. Studer. *Ontobroker: ontology based access to distributed and semi-structured information.* In Meersman, R., Tari, Z. and Stevens, S. (eds.), Database Semantics: Semantic Issues in Multimedia Systems. Kluwer Academic, 1999.

[33] M Dimitrov. *XML standards for ontology exchange.* exchange. In Proceedings of OntoLex 2000: Ontologies and Lexical Knowledge Bases, September 810, Sozopo, 2000.

[34] J. Domingue. *Tadzebao and webonto: discussing, browsing, and editing ontologies on the web.* In Proceedings of the 11th Knowledge Acquisition for Knowledge-Based Systems Workshop, April 1823, Banff, Canada., 1998.

[35] J. Pan and I. Horrocks. *Metamodeling architecture of web ontology languages.* In the Proceedings of the International Semantic Web Working Symposium (SWWS), July 30August 1, Stanford University, CA., 2001.

[36] B. Shneiderman. *The eyes have it: a task by data type taxonomy of information visualizations.* In the Proceedings of the IEEE Symposium on Visual Languages 96, pp. 336343, September, Los Alamitos, CA, IEEE, 1996.

[37] S. Staab, H.-P. Schnurr, R. Studer, and Y. Sure. *Knowledge processes and ontologies.* IEEE Intelligent Systems, 16(1): 2635, 2001.

[38] M. Uschold and M. King. *Towards a methodology for building ontologies.* In Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95, Montreal, Canada, 1995.

[39] D. Mladenic. *Text-learning and related intelligent agents: a survey.* IEEE Intelligent Systems 14(4): 4454., 1999.

[40] Wiederhold G. Mitra, P. and M. Kersten. *In: Zaniolo, C., Lockemann, P., Scholl, M. and Grust, T. (eds.), Advances in Database Technology - EDBT 2000, 7th International Conference on Extending Database Technology, LNCS 1777, pp. 86100, Konstanz, Germany, March 2731, Springer-Verlag.* 2000.

Elenco delle figure

Indice analitico

Powered by LaTeX