Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Il Web è diventata una delle strade principali attraverso le quali persone, aziende e organizzazioni si scambiano informazioni. A causa della crescente mole di dati disponibili, trovare e riusare informazioni con il passare del tempo è diventato sempre più difficile. I motori di ricerca hanno tentato di occuparsi di questo problema indicizzando tutte le pagine presenti sul Web, ma devono comunque scontrarsi con alcuni problemi. Per prima cosa, risulta sempre più difficile tenere il passo con la crescita del Web. In secondo luogo, sono costretti ad ignorare la maggior parte delle informazioni disponibili, questo perchè devono occuparsi esclusivamente delle pagine statiche, non tenendo conto delle informazioni che sono "nascoste" e disponibili unicamente attraverso forms (le maschere), che permettono di accedere ai contenuti messi a disposizione da pagine dinamiche . Da non trascurare inoltre il fatto che la massima accuratezza della ricerca raggiungibile con questi sistemi, non può andare oltre la pagina Web.

Nell'affrontare il problema della ricerca su pagine dinamiche, alcuni servizi di integrazione dell'informazione forniti da portali commerciali (all'interno cioè di uno specifico dominio) hanno fornito prestazioni di livello superiore rispetto alla concorrenza, citiamo tra questi **jango.excite.com** e **cnet.com**. Questi servizi di integrazione offrono un accesso uniforme a collezioni eterogenee di pagine dinamiche messe a disposizione dai fornitori di contenuti, usando la tecnologia wrapper [8], con l'unico limite di agire all'interno di un dominio privato. Alcuni sforzi di ricerca in ambito universitario e non, hanno cercato di estrarre la

struttura del contenuto da documenti Web semi strutturati al fine di mettere a punto servizi di integrazione che possano uscire da un dominio privato. Gli strumenti ottenuti dall'implementazione pratica di queste ricerche, sezionano una pagina Web in oggetti di interesse, operazione che può venire effettuata dal programmatore oppure attraverso alcuni gradi di automazione. A titolo di esempio in [3, 6] sono presenti tool che scoprono i confini degli oggetti manualmente. In questo caso il programmatore per prima cosa esamina i documenti e trova i tags HTML che separano gli oggetti di interesse, quindi scrive un programma per separare l'una dall'altra, le regioni occupate dagli oggetti. Altri approcci [1, 2, 4, 5, 7, 9, 10] separano tali regioni attraverso alcuni gradi di automazione. Questi approcci si affidano principalmente all'uso di conoscenze sintattiche, come specifici tags HTML, per identificare i confini tra gli oggetti. Uno degli approcci a muoversi in questa direzione preso come riferimento è quello suggerito da Embley [5], il quale ha messo a punto un metodo di estrazione automatica degli oggetti basato su euristiche. Da notare che se da una parte la sua euristica gioca un ruolo critico nel raggiungimento di un livello accettabile di accuratezza, dall'altra lo sviluppo manuale di programmi in grado di separare gli oggetti sarebbe troppo costoso (si consideri uno sforzo nell'ordine di 2 settimane/uomo per ogni sito realizzato) [5]. Le comuni tecniche adottate per la costruzione dei programmi utilizzati da questi approcci, spesso richiedono progettisti che si dedichino alla comprensione dello specifico layout di presentazione o degli specifici contenuti delle pagine Web. Come risultato, molti dei servizi di integrazione dell'informazione non decollano perchè impiegano troppo tempo per incorporare nuovi fornitori di contenuti all'interno dell'esistente struttura di accesso all'integrazione.

L'estrazione di dati da pagine HTML al fine di renderli disponibili ad applicazioni software sta diventando molto importante anche nello sviluppo di alcuni servizi di e-commerce emergenti. A titolo di esempio possiamo citare quelli che vengono chiamati shopping agents, i quali sono moduli software che navigano in rete alla ricerca dei migliori prezzi

di articoli presenti su siti di e-commerce. Un altro importante esempio è rappresentato dal bisogno dei moderni portali di fornire un accesso integrato ad alcuni servizi forniti dal Web, attraverso un'interfaccia unica (es: home banking, trading on line, pagamento delle bollette, controlli della carta di credito etc.). Queste applicazioni hanno bisogno di manipolare dati presenti in pagine HTML, poichè l'obiettivo di queste sorgenti di dati sono le persone, e non le macchine, i dati codificati in HTML non possono essere direttamente utilizzati dalle applicazioni, essi devono essere estratti dalle pagine HTML e trasformati in un formato più strutturato (es: XML). Questo compito è affidato a programmmi che prendono il nome di wrappers.

La trasformazione di documenti in formato "human readable" HTML in un formato "machine readable" XML, permetterebbe l'implementazione di software in grado di effettuare autonomamente ricerche mirate ad un determinato contesto e di interpretare i dati individuati, con granularità ben maggiore di quella raggiungibile dagli attuali motori di ricerca, al fine di poter generare autonomamente risposte a queries complesse formulate dall'utente.

Risulta doveroso a questo punto sottolineare l'importanza del linguaggio XML che si sta affermando universalmente come standard per lo scambio di dati sul Web, offrendo un metodo facile per integrare applicazioni distribuite e fornendo a queste sofisticati servizi. Tra questi servizi, oltre a quelli appena citati, possiamo annoverare a titolo di esempio la maggior parte di quelli di E-commerce e di gestione della catena dei fornitori (supply-chain management). Questi nuovi servizi portano anche ad una rivoluzione per quanto riguarda la trasformazione dalle sorgenti di informazioni esistenti in altre accessibili attraverso programmi software e agenti.

Una domanda a questo punto sorge spontanea: "Da dove vengono questi dati in formato XML dal momento che la maggior parte delle informazioni tutt'ora disponibili sul Web è ancora espressa nel linguaggio human-oriented HTML e lo sarà nell'immediato futuro?". Un approccio

largamente adottato per risolvere il problema è stato quello di scrivere wrappers al fine di incapsulare l'accesso alle sorgenti e produrre dati più strutturati, come quelli espressi nel linguaggio XML. I wrappers sono gli strumenti chiave per effettuare la conversione di pagine HTML in documenti equivalenti in formato XML, semanticamente significativi e ben strutturati.

L'obiettivo di questa tesi è quello di effettuare un'analisi dei generatori di wrappers attualmente disponibili ed il loro relativo test, fornendo una panoramica di quelle che sono le principali soluzioni utilizzate da questi strumenti al fine di risolvere il problema dell'estrazione di dati presenti sul Web in formato HTML e della loro successiva traduzione in XML.

Degli strumenti presi in considerazione, alcuni richiedono un forte intervento del progettista nel processo di generazione dei wrappers, altri lavorano in modo completamente automatico o semiautomatico. La scrittura manuale di wrappers è un compito molto impegnativo ed i risultati così ottenuti sono abbastanza fragili, in quanto ogni piccolo cambiamento del sito può impedire al wrapper di funzionare correttamente, questo ha orientato le ricerche verso strumenti in grado di non richiedere, se non in quantità minime, input da parte di uno sviluppatore.

Prima di iniziare ad esaminare gli strumenti attualmente disponibili, è giusto far notare che l'approccio completamente automatico di estrazione di informazioni da pagine Web è solo una delle molte sfide da affrontare nella costruzione di un sistema di ricerca delle informazioni e aggregazione di servizi per il Web scalabile ed affidabile.

Questa tesi si inserisce all'interno di un progetto più ampio denominato **MOMIS** (**M**ediator Envir**O**nment for **M**ultiple **I**nformation **S**ources) [51, 52, 53, 54, 55], sviluppato con lo scopo di realizzare un processo semiautomatico di integrazione di sorgenti eterogenee e distribuite.
MOMIS adotta un'architettura a tre livelli con un *Mediatore* che ne occupa la parte centrale ed avente lo scopo di fornire una visione integrata degli schemi locali. Tale visione integrata degli schemi costitutivi le sin-

gole sorgenti permette di effettuare delle interrogazioni prescindendo dalla conoscenza della specifica sorgente, ma utilizzando le informazioni che derivano dall'unico schema, sintesi delle diverse eterogeneità. Il *Mediatore* rappresenta pertanto il cuore del sistema ed ha il compito di realizzare l'integrazione degli schemi e di provvedere alla gestione dei risultati delle operazioni di query.

Elementi indubbiamente innovativi di questo progetto sono rappresentati dall'impiego di un approccio *semantico* e dall'uso di logiche descrittive per la rappresentazione degli schemi delle sorgenti. Questi elementi introducono, infatti comportamenti intelligenti che permettono di sfruttare al meglio le conoscenze intensionali, semantiche ed estensionali sia inter-schema sia intra-schema, per generare una vista globale il più possibile espressiva e sulla quale potere effettuare delle interrogazioni significative.
Di fondamentale importanza è quindi l'impiego di , un ambiente software sviluppato presso l'Università di Modena e Reggio Emilia, in grado di realizzare la validazione di schemi ad oggetti e l'espansione semantica delle interrogazioni.

Se il mediatore rappresenta il cuore del processo di integrazione, il wrapper rappresenta il meccanismo attraverso il quale avviene l'interazione con la singola sorgente. Pertanto dovranno esistere diversi wrapper ognuno ottimizzato per presidiare una certa tipologia di sorgenti. I compiti del wrapper sono sostanzialmente due: in primo luogo deve fornire una rappresentazione, nella logica comune, ODL_{I^3}, dello schema della sorgente alla quale é connesso. Tale descrizione dovrà necessariamente essere il frutto di un compromesso, tra la salvaguardia della struttura della sorgente originaria e la rappresentazione di schemi attraverso il modello dati utilizzato. In secondo luogo deve permettere l'esecuzione delle cosiddette *query locali* e deve fornire i risultati ottenuti al mediatore.

Obiettivo della presente tesi è stata l'analisi di strumenti in grado di generare wrappers per sorgenti di dati di tipo HTML. Questi wrappers dovranno essere in grado di effettuare l'estrazione delle informazioni da

pagine HTML e di restituirle in formato XML. Tale linguaggio, prestandosi a modellare sia dati di tipo strutturato sia dati di tipo semistrutturato, permette la rappresentazione anche di realtà aventi complessità elevate. Le funzionalità necessarie per la successiva integrazione delle informazioni in formato XML in MOMIS sono già state realizzate attraverso la messa a punto di uno specifico wrapper [78], che consente la trasformazione dello schema rappresentativo della singola sorgente in quello corrispondente espresso nel linguaggio ODL_{I^3}. Per quanto riguarda l'esatta collocazione di questo studio all'interno del progetto MOMIS, è possibile posizionarlo sotto il livello wrapper.

La tesi è organizzata nel modo seguente:

Nel **Capitolo 1** viene dapprima introdotta l'architettura di riferimento per i sistemi di Integrazione di Informazioni, per poi illustrare le scelte implementative fatte in MOMIS. Viene anche presentato il componente usato al fine di introdurre, nel sistema, comportamenti intelligenti. Nella parte finale del capitolo vengono illustrate le fasi di interazione con il wrapper XML da parte del mediatore.

Nel **Capitolo 2** viene effettuata l'analisi di uno strumento semiautomatico per la generazione di wrappers in grado di estrarre il contenuto di pagine HTML e tradurlo in formato XML, il cui nome è XWRAP Elite, sviluppato al Georgia Institute of Tecnologies. Nella parte conclusiva, verranno illustrati i risultati di una serie di test effettuati utilizzando come sorgenti le pagine HTML di una serie di siti della realtà industriale modenese.

Nel **Capitolo 3** viene introdotto un sistema completamente automatico che risponde al nome di RoadRunner sviluppato dall'Università di Roma, la cui logica di funzionamento si basa sul confronto tra pagine per estrarre la struttura dei documenti in esame. Questo strumento fa uso di un agente di navigazione che all'interno di un sito classifica le pagine in

base alla loro struttura, creando cluster di pagine simili.

Il **Capitolo 4** illustra i principi di funzionamento del sistema Andes della IBM che fa affidamento su XPath ed XSLT per effettuare la conversione in XML. Questo strumento fa uso di un agente autonomo di navigazione, per certi aspetti simile a quello utilizzato da RoadRunner, per recuperare i documenti di interesse, al fine di poterli sottoporre alla successiva fase di traduzione in XML. Al contrario degli strumenti analizzati nei capitoli precedenti, il cui obiettivo era l'automatizzazione di tutte le fasi di generazione, in questo strumento è richiesto l'impiego di un progettista che dovrà fornire una serie di input. Queste informazioni aggiuntive saranno utilizzate per ottenere wrappers con un buon grado di robustezza ai cambiamenti strutturali delle sorgenti su cui è stato messo a punto. Contiene funzionalità per la sitesi di hyperlinks dinamici in hyperlink statici percorribili dal navigatore.

Nel **Capitolo 5** viene effettuato uno studio su un toolkit per la generazione di wrappers supervisionata da un operatore, il cui nome risponde a Lixto. Questo software verrà testato sulle pagine dei siti utilizzati per la sperimentazione di XWRAP Elite, al fine di poter confrontare i risultati ottenuti.

Nel **Capitolo 6** viene analizzato il sistema Lapis, nato con l'intento di fornire un valido ausilio all'estrazione di dati da documenti testuali e utilizzato in questo contesto per generare wrappers in grado di estrarre il contenuto di pagine HTML (che sotto certe ipotesi possono essere considerate alla stregua di documenti testuali).

Nel **Capitolo 7** verrà effettuata una comparazione delle principali caratteristiche degli strumenti in grado di estrarre dati da sorgenti HTML.

Sono inoltre presenti tre Appendici. In Appendice A viene riportato

un glossario dei termini usati in ambito I^3. In Appendice B un'analisi del sistema Omini, componente fondamentale del sistema XWRAP Elite analizzato nel capitolo 2. In Appendice C vengono presentati due progetti del Georgia Institute of Technologies, il primo prende il nome di XWRAP Original ed è il predecessore di XWRAP Elite, con il quale condivide parte della logica di funzionamento. Il secondo è OminiSearch, e illustra come XWRAP Elite verrà utilizzato in una struttura di estrazione più complessa.

Capitolo 1

L'Integrazione delle Informazioni

La crescita delle sorgenti di dati sia all'interno dell'azienda che sulla rete, ha reso possibile l'accesso ad una quantità molto ampia di informazioni. La probabilità di reperire un dato di interesse è di conseguenza aumentata vertiginosamente, allo stesso tempo bisogna registrare una diminuzione altrettanto importante della probabilità di venirne in possesso nei tempi e soprattutto nei modi desiderati. Questo principalmente a causa della grande eterogeneità dei dati disponibili, sia per quanto riguarda la natura (testi, immagini, etc.), sia il modo in cui vengono descritti.

Gli standard esistenti (TCP/IP, ODBC, OLE, CORBA, SQL, etc.) risolvono parzialmente i problemi relativi alle diversità hardware e software, dei protocolli di rete e di comunicazione tra i moduli; rimangono però irrisolti quelli relativi alla modellazione delle informazioni. I modelli e gli schemi dei dati sono infatti differenti e questo crea una eterogeneità semantica (o logica) non risolvibile da questi standard.

Un'importante area, sia di ricerca che di applicazione, riguarda l'integrazione di DataBase eterogenei ed anche i *datawarehouse* (magazzino di dati). Questi lavori studiano la possibilità di materializzare presso l'utente finale delle viste, ovvero porzioni di sorgenti, replicando fisicamente i dati e affidandosi a complicati algoritmi di mantenimento ai fini

di garantirne la consistenza.

Con il termine Integrazione delle Informazioni (I^2) si indicano in letteratura quei sistemi che al contrario di quelli appena citati, basandosi sulle descrizioni dei dati, combinano tra loro informazioni provenienti da diverse sorgenti (o parti selezionate di esse) senza dover ricorrere perciò alla duplicazione fisica [61].

Nelle pagine seguenti verrà descritta la proposta dell'ARPA (Advanced Research Projects Agency)[56] per un'architettura che favorisca da una parte l'autonomia delle sorgenti, ma che dall'altra assicuri flessibilità e riusabilità. Si presenterà inoltre l'approccio seguito nel progetto MOMIS.

1.1 L'Integrazione Intelligente delle Informazioni

L'integrazione delle informazioni va dunque distinta da quella dei dati e dei DataBase; per ottenere risultati selezionati è richiesta *conoscenza* ed *intelligenza* volte alla scelta delle sorgenti e dei dati, nonchè alla loro fusione e alla conseguente sintesi. Quando l'Integrazione delle Informazioni fa uso di tecniche di Intelligenza Artificiale si parla allora di Integrazione Intelligente delle Informazioni (*Intelligent Integration of Information, I^3).* Al contrario di quanto accade con l'uso di tecniche tradizionali (che si limitano ad una semplice aggregazione), questa forma di integrazione si prefigge lo scopo di accrescere il valore delle informazioni gestite (ottenendone anche di nuove dai dati utilizzati).

1.1.1 Il Programma I^3

Il programma I^3 è un'ambiziosa ricerca finalizzata ad indicare un'architettura di riferimento che realizzi l'integrazione di sorgenti eterogenee in maniera automatica ed è sviluppato dall'ARPA, l'agenzia che fa capo al

Dipartimento di Difesa americano [56]. In quel contesto si é potuto osservare che l'integrazione aumenta il valore dell'informazione ma richiede una forte adattabilità realizzativa: si devono infatti riuscire a gestire i casi di aggiornamento e sostituzione delle sorgenti, dei loro ambienti e/o piattaforme, della loro ontologia e della semantica. Le tecniche sviluppate dall'*Intelligenza Artificiale*, potendo efficacemente dedurre informazioni utili dagli schemi delle sorgenti, diventano pertanto uno strumento prezioso per la costruzione automatica di soluzioni integrate flessibili e riusabili.

Progettare la costruzione di supersistemi ad hoc che interessino una grande quantità di sorgenti non correlate semanticamente, è faticoso ed il risultato è un sistema scarsamente manutenibile o adattabile, strettamente finalizzato alla risoluzione dei problemi per cui è stato implementato. Secondo il programma I^3, una soluzione a questi problemi può essere trovata mediante l'introduzione di architetture modulari sviluppabili secondo i principi proposti da uno standard. Tale standard deve porre le basi dei servizi che devono essere realizzati attraverso l'integrazione e deve cercare di abbassare i costi di sviluppo e di mantenimento. Pertanto costruire nuovi sistemi risulta realizzabile con minor difficoltà e minor tempo (e quindi costi) se si riesce a supportare lo sviluppo delle applicazioni riutilizzando la tecnologia già sviluppata. Per la riusabilità è fondamentale l'esistenza di interfacce ed architetture standard. Il paradigma suggerito per la suddivisione dei servizi e delle risorse nei diversi moduli, si articola su due dimensioni:

- *orizzontalmente* in tre livelli: livello utente, livello intermedio che fa uso di tecniche di IA, livello delle sorgenti di dati;
- *verticalmente*: diversi domini in cui raggruppare le sorgenti.

I domini dei vari livelli non sono strettamente connessi, ma si scambiano dati ed informazioni la cui combinazione avviene a livello dell'utilizzatore, riducendo la complessità totale del sistema e permettendo lo sviluppo di applicazioni con finalità diverse.

I^3 si concentra sul livello intermedio della partizione, quello che media tra gli utenti e le sorgenti. In questo livello sono presenti vari moduli, tra i quali è giusto evidenziare:

- **Facilitator** e **Mediator** (le differenze tra i due sono sottili ed ancora ambigue in letteratura): ricercano le fonti *interessanti* e combinano i dati da esse ricevuti;
- **Query Processor**: riformula le query aumentando le loro probabilità di successo;
- **Data Miner**: analizza i dati per estrarre informazioni intensionali implicite.

Nell'Appendice A è presente un glossario dei termini comunemente usato in ambito I^3 , allo scopo di spiegare quei termini che dovessero risultare ambigui o poco chiari, visto il campo recente ed in evoluzione in cui si muove il progetto.

1.2 Architettura di riferimento per sistemi

L'obiettivo del programma I^3 é quello di ridurre il tempo necessario alla realizzazione di un integratore di informazioni, fornendo una raccolta e una formalizzazione delle soluzioni prevalenti nel campo della ricerca. In particolare due sono le ipotesi che rappresentano la base del progetto I^3 . La prima é connessa con la difficoltà che si ha nel ricercare delle informazioni all'interno della molteplicità delle sorgenti di informazione che in questo momento é possibile individuare. Il secondo aspetto é legato al fatto che le fonti di informazione e i sistemi informativi, pur essendo spesso semanticamente correlati tra di loro, non lo sono in una forma semplice né preordinata. Di conseguenza il processo di integrazione delle informazioni puó risultare molto complesso.

Pertanto é necessario proporre una architettura di riferimento che rappresenti alcuni dei servizi che un integratore di informazioni deve contenere e le possibili interconnessioni fra di essi. Tale descrizione non vuole imporre né delle soluzioni implementative, né é da ritenersi esaustiva delle funzionalità che un sistema di integrazione deve includere.

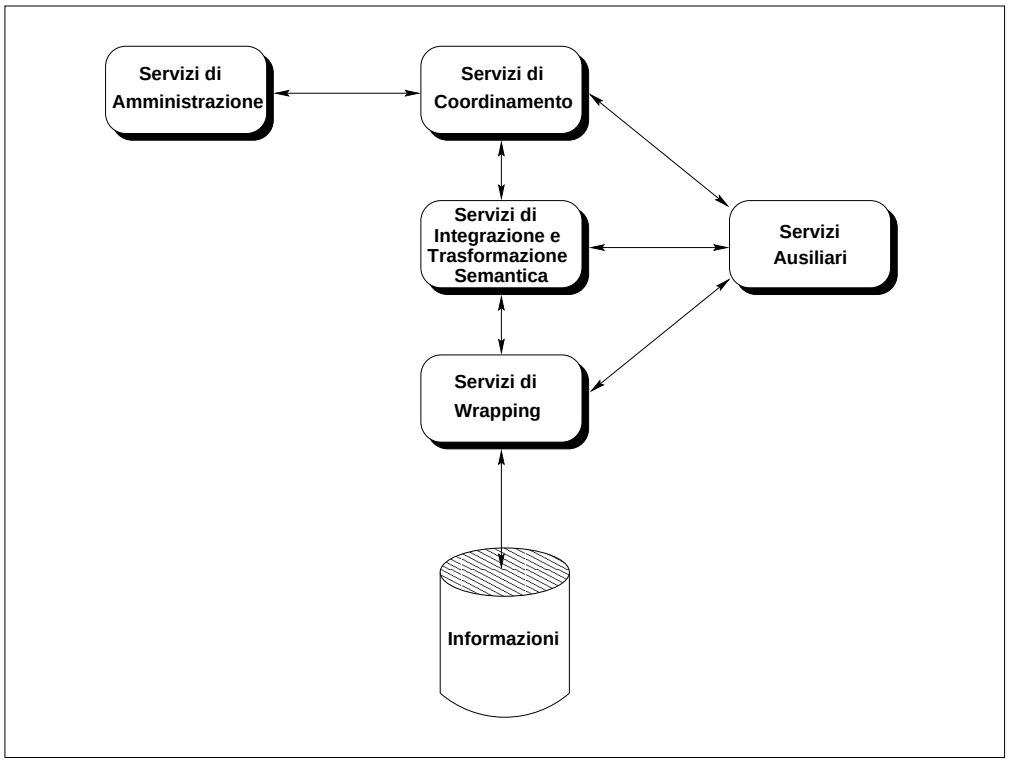

Figura 1.1: Diagramma dei servizi I^3

In Figura 1.1 viene riportata l'architettura di un sistema I^3 . L'architettura di riferimento dà grande rilevanza ai Servizi di Coordinamento. Questi servizi giocano infatti due ruoli: come prima cosa, possono localizzare altri servizi I^3 e fonti di informazioni che possono essere utilizzate per costruire il sistema stesso; secondariamente, sono responsabili di individuare ed invocare a run-time gli altri servizi necessari a dare risposta ad una specifica richiesta di dati.
Sono comunque in totale cinque le famiglie di servizi che sono state rap-

presentate in questa architettura. I servizi individuati, così come rappresentati nell'architettura proposta, possono essere soggetti a due differenti chiavi di lettura: la lettura seguendo l'asse orizzontale e quella seguendo l'asse verticale mettono in evidenza diversi aspetti e diversi compiti del sistema.
Se si percorre l'asse verticale, si può intuire come avviene lo scambio di informazioni nel sistema: in particolare, i servizi di *wrapping* provvedono ad estrarre le informazioni dalle singole sorgenti. Tali informazioni vengono poi impacchettate ed integrate dai Servizi di Integrazione e Trasformazione Semantica, per poi essere passate ai servizi di Coordinamento che ne avevano fatto richiesta. L'asse orizzontale mette invece in risalto il rapporto tra i servizi di Coordinamento e quelli di Amministrazione, ai quali spetta infatti il compito di mantenere informazioni sulle capacità delle varie sorgenti (che tipo di dati possono fornire ed in quale modo devono essere interrogate). Funzionalità di supporto, che verranno descritte successivamente, sono invece fornite dai Servizi Ausiliari, responsabili dei servizi di arricchimento semantico delle sorgenti.
Vengono ora analizzate in dettaglio le funzionalità specifiche di ogni servizio e le problematiche che devono essere affrontate.

1.2.1 Servizi di Coordinamento

I servizi di Coordinamento sono quei servizi di alto livello che permettono l'individuazione delle sorgenti di dati *interessanti*, ovvero che probabilmente possono dare risposta ad una determinata richiesta dell'utente. A seconda delle possibilità dell'integratore che si vuole realizzare, possono essere rappresentati da meccanismi che includono dalla selezione dinamica delle sorgenti (o brokering, per Integratori Intelligenti) fino al semplice *Matchmaking*, in cui il mappaggio tra informazioni integrate e locali è realizzato manualmente ed una volta per tutte. Vediamo alcuni esempi.

1. **Facilitation e Brokering Services**: l'utente manda una richiesta al

sistema e questo usa un deposito di metadati per ritrovare il modulo che può trattare la richiesta direttamente. I moduli interessati da questa richiesta possono essere o uno solo alla volta (nel qual caso si parla di Brokering) oppure più di uno (e in questo secondo caso si tratta di facilitatori e mediatori, attraverso i quali a partire da una richiesta ne viene generata più di una da inviare singolarmente a differenti moduli che gestiscono sorgenti distinte, e reintegrando poi le risposte in modo da presentarle all'utente come se fossero state ricavate da un'unica fonte). In questo ultimo caso, in cui una query può essere decomposta in un insieme di sottoquery, si farà uso di servizi di Query Decomposition e di tecniche di Inferenza (mutuate dall'Intelligenza Artificiale) per una determinazione dinamica delle sorgenti da interrogare, a seconda delle condizioni poste nell'interrogazione.
I vantaggi che questi servizi di Coordinamento portano stanno nel fatto che non è richiesta all'utente del sistema una conoscenza del contenuto delle diverse sorgenti, ma viene fornita un'unica rappresentazione delle sorgenti e in questo modo un sistema omogeneo che gestisce direttamente la sua richiesta. Pertanto l'utente non é obbligato a conoscere i domini con i quali i vari moduli I^3 si trovano ad interagire. Risulta quindi evidente la considerevole diminuzione di complessità di interazione col sistema che deriva da una architettura di questo tipo.

2. **Matchmaking**: il sistema viene configurato manualmente da un operatore in fase di inizializzazione. Successivamente a quella fase, tutte le richieste verranno trattate allo stesso modo. Sono definiti gli anelli di collegamento tra tutti i moduli del sistema, e nessuna ottimizzazione è fatta a tempo di esecuzione.

1.2.2 Servizi di Amministrazione

Sono servizi usati dai Servizi di Coordinamento per localizzare le sorgenti *utili*, per determinare le loro capacità, e per creare ed interpretare TEMPLATE. I Template sono strutture dati che descrivono i servizi, le fonti ed i moduli da utilizzare per portare a termine un determinato task. Sono quindi utilizzati dai sistemi meno "intelligenti", e consentono all'operatore di predefinire le azioni da eseguire a seguito di una determinata richiesta, limitando al minimo le possibilità di decisione del sistema.
In alternativa all'uso dei Template, possono essere utilizzate le **Yellow Pages**: si tratta di servizi di directory che memorizzano le informazioni sul contenuto delle varie sorgenti e sul loro stato (attiva, inattiva, occupata). Consultando queste Yellow Pages, il mediatore è in grado di spedire alla giusta sorgente la richiesta di informazioni, ed eventualmente di rimpiazzare questa sorgente con una equivalente nel caso non fosse disponibile. Fanno parte di questa categoria di servizi il Browsing: si permette all'utente di "navigare" attraverso le descrizioni degli schemi delle sorgenti, recuperando informazioni su queste. Il servizio si basa sulla premessa che queste descrizioni siano fornite esplicitamente tramite un linguaggio dichiarativo leggibile e comprensibile all'utente. Altri moduli interessanti sono:

- **Resource Discovery:** sono in grado di riconoscere e ritrovare gli strumenti che gestiscono una determinata richiesta a run time. Questi servizi sono in grado di acquisire ed aggiornare dinamicamente informazioni sui tool (nomi e servizi forniti) e sul loro stato. Inoltre acquisiscono e mantengono le informazioni sui domini informativi.

- **Iterative Query Formulation:** sono di supporto all'utente in fase di formulazione di query particolarmente complesse sullo schema integrato; specialmente qualora la query formulata non abbia prodotto informazioni interessanti, suggeriscono quali condizioni rilasciare, come rendere più specifica o più generale la query.

- **Primitive di costruzione delle configurazioni:** servono a scegliere due cose: da una parte i servizi di tool e le sorgenti appropriate per svolgere un opportuno task, dall'altra il giusto modo di collegarli tra loro per creare una configurazione.

1.2.3 Servizi di Integrazione e Trasformazione Semantica

Questi servizi supportano le manipolazioni semantiche necessarie per l'integrazione e la trasformazione delle informazioni. Il tipico input per questi servizi é rappresentato da una o più sorgenti di dati, e l'output é la "vista" integrata o trasformata di queste informazioni. Tra questi servizi si distinguono quelli relativi alla trasformazione degli schemi (ovvero di tutto ciò che va sotto il nome di *metadati*) e quelli relativi alla trasformazione dei dati. Sono spesso indicati come servizi di Mediazione, essendo tipici dei moduli mediatori.

1. Servizi di **integrazione degli schemi**. Supportano la trasformazione e l'integrazione degli schemi e delle conoscenze derivanti da fonti di dati eterogenee. Ne fanno parte i servizi di trasformazione dei vocaboli e dell'ontologia, usati per arrivare alla definizione di un'ontologia unica che combini gli aspetti comuni alle singole ontologie usate nelle diverse fonti. Queste operazioni sono molto utili quando devono essere scambiate informazioni derivanti da ambienti differenti, dove molto probabilmente non si condivide un'unica ontologia. Fondamentale, per la fase di creazione dell'insieme dei vocaboli condivisi, è la fase di individuazione dei concetti presenti in diverse fonti, e la riconciliazione delle diversità presenti sia nelle strutture, sia nei significati dei dati.

2. Servizi di **integrazione delle informazioni**. Provvedono alla traduzione dei termini da un contesto all'altro, ovvero dall'ontologia di partenza a quella di destinazione. Possono inoltre occupar-

si di uniformare la "granularità" dei dati (come possono essere le discrepanze nelle unità di misura, o le discrepanze temporali).

3. Servizi di **supporto al processo di integrazione**. Sono utilizzati nel momento in cui una query è scomposta in molte subquery, da inviare a fonti differenti, e nel momento in cui i risultati provenienti dalle singole subquery devono essere integrati. Comprendono inoltre tecniche di *caching*, per supportare la materializzazione delle viste (problematica molto comune nei sistemi che vanno sotto il nome di datawarehouse).

1.2.4 Servizi di Wrapping

Sono utilizzati per fare sì che le fonti di informazioni aderiscano ad uno standard, che può essere interno o proveniente dal mondo esterno con cui il sistema vuole interfacciarsi. Si comportano come traduttori dai sistemi locali ai servizi di alto livello dell'integratore e viceversa quando si interroga la sorgente di dati. In particolare, sono due gli obiettivi che si prefiggono:

1. permettere ai servizi di coordinamento e di mediazione di manipolare in modo uniforme il numero maggiore di sorgenti locali, anche se queste non sono state esplicitamente pensate come facenti parte del sistema di integrazione;

2. essere il più riusabili possibile. Per fare ciò, dovrebbero fornire interfacce che seguano gli standard più diffusi (e tra questi, si potrebbe citare il linguaggio SQL come linguaggio di interrogazione di basi di dati, e CORBA come protocollo di scambio di oggetti). Questo permetterebbe alle sorgenti estratte da questi wrapper "universali" di essere accedute dal numero maggiore possibile di moduli mediatori.

In pratica, compito di un wrapper è modificare l'interfaccia, i dati ed il comportamento di una sorgente, per facilitarne la comunicazione con il mondo esterno. Il vero obiettivo è quindi standardizzare il processo di wrapping delle sorgenti, permettendo la creazione di una libreria di fonti accessibili; inoltre, il processo stesso di realizzazione di un wrapper dovrebbe essere standardizzato, in modo da poter essere riutilizzato da altre fonti.

1.2.5 Servizi Ausiliari

Aumentano le funzionalità degli altri servizi descritti precedentemente: sono prevalentemente utilizzati dai moduli che agiscono direttamente sulle informazioni. Vanno dai semplici servizi di monitoraggio del sistema (un utente vuole avere un segnale nel momento in cui avviene un determinato evento in un database, e conseguenti azioni devono essere attuate), ai servizi di propagazione degli aggiornamenti e di ottimizzazione.

1.3 Il mediatore

Secondo la definizione proposta da Wiederhold in [63] "un mediatore è un modulo software che sfrutta la conoscenza su un certo insieme di dati per creare informazioni per una applicazione di livello superiore.... Dovrebbe essere piccolo e semplice, così da poter essere amministrato da uno, o al più pochi, esperti".

Compiti di un mediatore sono allora:

- assicurare un servizio stabile, anche nel caso di cambiamento delle risorse;
- amministrare e risolvere le eterogeneità delle diverse fonti;
- integrare le informazioni ricavate da più risorse;

- presentare all'utente le informazioni attraverso un modello scelto dall'utente stesso.

Il progetto MOMIS, di cui questa tesi fa parte, ha tra i suoi obiettivi la realizzazione di un Mediatore. Durante la fase di progettazione e realizzazione del sistema, é stato necessario introdurre una ipotesi che restringesse il campo applicativo del sistema stesso (e di conseguenza che restringesse il campo dei problemi a cui dare risposta). Tale ipotesi consiste nel fatto che il mediatore, per il momento, si occupi esclusivamente di sorgenti di dati di testo strutturati e semistrutturati, quali possono essere basi di dati relazionali, ad oggetti, file di testo, pagine HTML ed XML. L'approccio architetturale scelto è quello *classico*, che si sviluppa principalmente su 3 livelli:

1. utente: attraverso un'interfaccia grafica l'utente pone delle query su uno schema globale e riceve un'unica risposta, come se stesse interrogando un'unica sorgente di informazioni;
2. mediatore: il mediatore gestisce l'interrogazione dell'utente, combinando, integrando ed eventualmente arricchendo i dati ricevuti dai wrapper, ma usando un modello (e quindi un linguaggio di interrogazione) comune a tutte le fonti;
3. wrapper: ogni wrapper gestisce una singola sorgente, ed ha una duplice funzione: da un lato converte le richieste del mediatore in una forma comprensibile dalla sorgente, dall'altro traduce informazioni estratte dalla sorgente nel modello usato dal mediatore.

Facendo riferimento ai servizi descritti nelle sezioni precedenti, l'architettura del mediatore che si è progettato è riportata in Figura 1.2. In particolare, in questa tesi, é stato effettuato uno studio volto all'individuazione di generatori di wrappers in grado di estrarre dati da sorgenti HTML e di convertirli in formato XML. Il passo successivo, ossia l'integrazione di dati in formato XML all'interno del global schema, viene

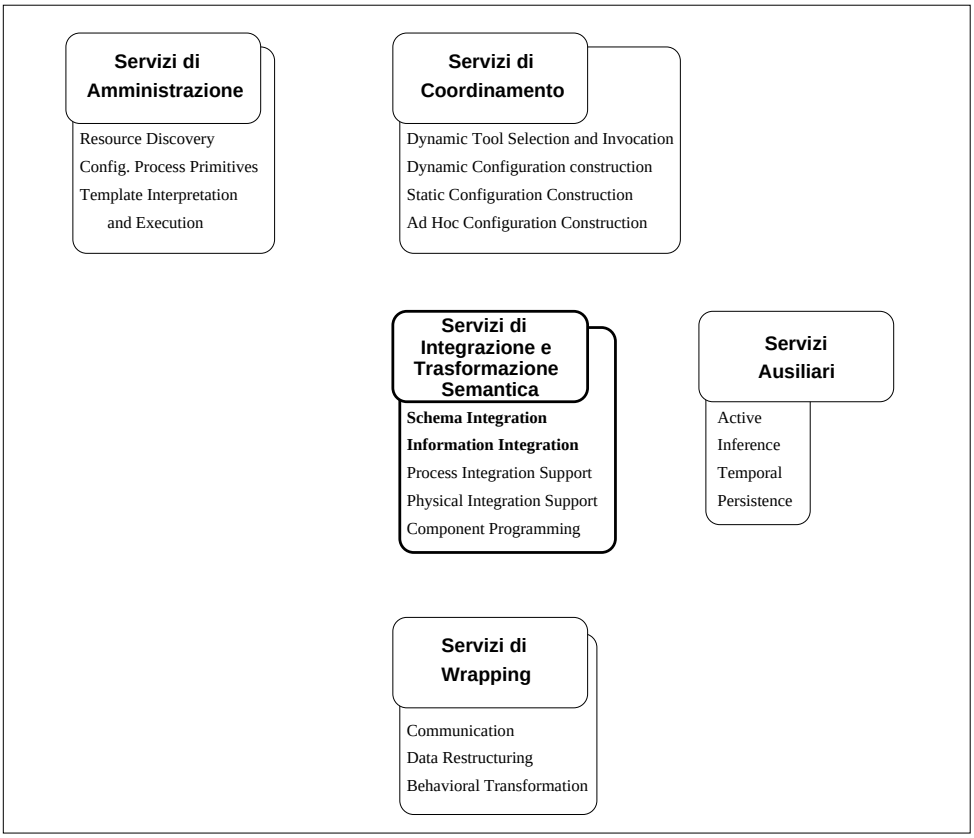

Figura 1.2: Servizi I^3 presenti nel mediatore

effettuata mediante l'impiego di un apposito wrapper, realizzato in una precedente tesi [78], che svolge la funzione di interfaccia tra il nucleo del mediatore, in grado di fornire i servizi di integrazione e trasformazione semantica e la singola sorgente XML. Tale wrapper restituisce la descrizione in ODL_{I^3} della sorgente XML alla quale è connesso e permette l'esecuzione a livello locale delle queries generate dal query manager. È possibile quindi affermare che i wrappers generati dagli strumenti oggetto di questo studio, si collocano un gradino sotto il livello wrapper, così come inteso nelle pagine precedenti.

L'impostazione architetturale presentata dimostra l'intenzione di progettare un mediatore che si distacchi dall'approccio *strutturale*, cioè sintattico, tuttora dominante tra i sistemi presenti sul mercato. L'approccio

strutturale, adottato da sistemi quali TSIMMIS [57, 58, 59, 60], è caratterizzato dal fatto di usare un self-describing model per rappresentare gli oggetti da integrare, limitando l'uso delle informazioni semantiche alle regole predefinite dall'operatore. In pratica, il sistema non conosce a priori la semantica di un oggetto che va a recuperare da una sorgente (e dunque di questa non possiede alcuno schema descrittivo) bensì è l'oggetto stesso che, attraverso delle etichette, si autodescrive, specificando tutte le volte, per ogni suo singolo campo, il significato che ad esso è associato. Questo approccio porta quindi ad un insieme di vantaggi, tra i quali possiamo identificare:

- la possibilità di integrare in modo completamente trasparente al mediatore, basi di dati fortemente eterogenee e magari mutevoli nel tempo: il mediatore non si basa infatti su una descrizione predefinita degli schemi delle sorgenti, bensì sulla descrizione che ogni singolo oggetto fa di sè. Oggetti simili provenienti dalla stessa sorgente possono quindi avere strutture differenti, cosa invece non ammessa in un ambiente tradizionale object-oriented;

- per trattare in modo omogeneo dati che descrivono lo stesso concetto, o che hanno concetti in comune, ci si basa sulla definizione manuale di rule, che permettono di identificare i termini (e dunque i concetti) che devono essere condivisi da più oggetti.

Altri progetti, e tra questi quello proposto, seguono invece un approccio definito *semantico*, che è caratterizzato dai seguenti punti:

- il mediatore deve conoscere, per ogni sorgente, lo schema concettuale (metadati);

- informazioni semantiche sono codificate in questi schemi;

- deve essere disponibile un modello comune per descrivere le informazioni da condividere (e dunque per descrivere anche i metadati);
- deve essere possibile una integrazione (parziale o totale) delle sorgenti di dati.

In questo modo, sfruttando opportunamente le informazioni semantiche che necessariamente ogni schema sottintende, il mediatore può individuare concetti comuni a più sorgenti e relazioni che li legano.

1.3.1 L'approccio Semantico all'Integrazione

Il cuore dell'architettura I^3 è rappresentato dai Servizi di Trasformazione ed Integrazione Semantica. Essi si occupano dell'integrazione vera e propria delle informazioni cercando di risolvere le differenze tra gli schemi, i modelli ed i tipi di dati, mentre le differenze dalla prospettiva delle piattaforme (Hw, DBMS, API) sono risolte dai protocolli di rete e dagli standard: SQL, OLE (Object Linking Embedded), per legare oggetti in applicazioni contenitori, ODBC (Object Database Connectivity): per la connessione fra basi di dati eterogenee, ODMG, che fornisce un modello dei dati ed i linguaggi di descrizione, manipolazione ed interrogazione per basi di dati orientate agli oggetti, CORBA, per lo scambio degli oggetti fra sorgenti eterogenee. Un secondo problema cui si dedica la Semantica riguarda il potere espressivo delle interrogazioni: molti sistemi, specialmenti quelli accessibili via Web, supportano solo un ristretto numero di query predefinite, aggiungendo conoscenza semantica è possibile ricondurre le query più complesse in quelle predefinite o in un loro sovrainsieme.

L'integrazione degli schemi porta alla definizione di viste omogenee: queste rappresentano un superschema, virtuale nella maggior parte dei casi, degli schemi delle sorgenti integrate. Le viste possono essere *convenzionali* se rappresentano in maniera omogenea gli schemi delle sorgenti o parti di esse, altrimenti sono dette *integrate*: in questo caso gli

schemi sono fusi e combinati e non è più facilmente riconoscibile il contributo portato dalle diverse sorgenti. La fusione degli schemi può essere completata dalla materializzazione dei dati, come ad esempio si verifica nei data warehouse: in cui i dati sono duplicati e fusi per essere conservati presso l'integratore. Questa scelta presenta vantaggi in termini di velocità di risposta, specialmente quando l'esecuzione dei dati richiede l'esecuzione di join su molti attributi, però diventa pesante il mantenimento della consistenza tra le sorgenti e la vista. In ragione di queste considerazioni si sta diffondendo sempre più l'approccio virtuale: esso consiste nel non replicare i dati, ma nell'eseguire a run-time la decomposizione della query globale, l'interrogazione delle sorgenti e la fusione delle informazioni.

1.3.2 Trattamento dei Metadati e delle Ontologie

Il concetto su cui si basa l'integrazione semantica di sorgenti, autonome nel fine ed eterogenee nella struttura, riguarda la sovrapposizione dei rispettivi domini, che si traduce in corrispondenze fra gli schemi delle sorgenti. Per introdurre i problemi legati al processo di individuazione delle similitudini fra questi schemi introduciamo i concetti di *Metadato* ed *Ontologia.*

Metadati: rappresentano informazioni relative agli schemi: significato delle classi (o relazioni) e delle loro proprietà (attributi), relazioni sintattiche e/o semantiche esistenti tra i modelli nelle sorgenti, caratteristiche delle istanze nelle sorgenti: tipi degli attributi, valori null. Utilizzare i metadati può essere di supporto nell'affrontare problematiche relative ad aspetti semantici. Vediamo di approfondire l'argomento. Pur ipotizzando che anche sorgenti diverse condividano una visione similare del problema da modellare, e quindi un insieme di concetti comuni, nulla assicura che i diversi sistemi usino esattamente gli stessi vocaboli per rappresentare questi concetti, ne tantomeno le stesse strutture dati. Al contrario, dal momento che le diverse sorgenti sono state progettate

e modellate da persone differenti, è molto improbabile che queste persone condividano la stessa concettualizzazione del mondo esterno: in altre parole non esiste nella realtà una semantica univoca a cui chiunque possa riferirsi.

Come riportato in [62] la causa principale delle differenze semantiche (anche se non l'unica) è identificabile nelle diverse concettualizzazioni del mondo esterno che persone distinte possono avere. Una medesima realtà può essere rappresentata su DBMS differenti che utilizzano modelli diversi: partendo così dalla stessa concettualizzazione, determinate relazioni tra concetti avranno strutture diverse a seconda che siano realizzate, ad esempio, attraverso un modello relazionale o ad oggetti.

L'obiettivo dell'integratore di fornire un accesso integrato ad un insieme di sorgenti, si traduce quindi nel difficile compito di identificare i concetti comuni all'interno delle sorgenti e risolvere le eventuali differenze semantiche.

Possiamo classificare queste incoerenze semantiche in tre gruppi principali:

1. *eterogeneità tra le classi di oggetti:* benchè due classi in due differenti sorgenti rappresentino lo stesso concetto nello stesso contesto, possono usare nomi diversi per gli stessi attributi o per i metodi, oppure avere gli stessi attributi con domini di valori differenti o ancora avere regole diverse sui valori;

2. *eterogeneità tra le strutture delle classi:* comprendono le differenze nei criteri di specializzazione, nelle strutture per realizzare un'aggregazione, ed anche le discrepanze schematiche, quando cioè valori di attributi sono invece parte dei metadati in un altro schema (ad esempio, l'attributo SESSO presente in uno schema può essere assente in un altro schema che rappresenta le persone attraverso le classi MASCHI e FEMMINE);

3. *eterogeneità nelle istanze delle classi:* ad esempio, uso di diverse unità

di misura per i domini di un attributo, o la presenza/assenza di valori nulli.

Ontologie: come riportato in Appendice A, per ontologia si intende, in questo ambito, "l'insieme dei termini e delle relazioni esistenti in un dominio per denotare concetti ed oggetti". Un'ontologia relativa ad un insieme di sorgenti è costituita da tutti quei termini che identificano in maniera non ambigua lo stesso concetto o la stessa porzione di conoscenza. Un'ontologia può essere più o meno generale a seconda del livello cui ci si riferisce: se ci si riferisce ad una precisa applicazione l'ontologia sarà limitata dipendendo dal dominio e dall'obiettivo della stessa. Le ontologie di livello superiore sono più vaste riferendosi solo al dominio ma essendo indipendenti dall'obiettivo, quelle di livello ancora superiore sono indipendenti anche dal dominio e definiscono concetti molto generali.

Più precisamente, mutuando la classificazione fatta da Guarino [64, 65] è possibile individuare i seguenti livelli di ontologia:

1. *top-level ontology*: descrive concetti molto generali come spazio, tempo, evento, azione..., che sono quindi indipendenti da un particolare problema o dominio: si considera ragionevole, almeno in teoria, che anche comunità separate di utenti condividano la stessa top-level ontology;

2. *domain e task ontology*: descrive, rispettivamente, il vocabolario relativo a un generico dominio (come può essere un dominio medico, o automobilistico) o a un generico obiettivo (come la diagnostica, o le vendite), dando una specializzazione dei termini introdotti nelle top-level ontology;

3. *application ontology*: descrive concetti che dipendono sia da un particolare dominio sia da un particolare obiettivo.

È lecito supporre che, integrando sorgenti seppur autonome, il livello dell'ontologia sia vincolato all'interno di un certo dominio; questa re-

strizione è congruente con l'idea di integrare i domini comuni di sorgenti che descrivono campi almeno parzialmente sovrapposti.

Non disponendo delle informazioni sui Metadati e sulle Ontologie risulta impossibile realizzare una buona integrazione. Questa affermazione risulta facilmente comprensibile osservando che, già trattando una sola sorgente, la rappresentazione del dominio per la costruzione ad esempio di un DB sarà fatta in una infinità di modi diversi a seconda del progettista incaricato, delle informazioni disponibili e della specifica applicazione richiesta; questo fatto risulta ancora più evidente dovendo integrare domini che solo parzialmente sono sovrapposti, che sono stati realizzati da persone diverse, in momenti diversi e con fini del tutto autonomi.

Se dunque si desidera integrare è indispensabile poter disporre di strumenti in grado di dedurre, direttamente dagli schemi, informazioni sui metadati e sulle ontologie. Le ontologie di supporto a questo processo sono basate su tecniche di manipolazione della conoscenza, studiate nell'area dell'intelligenza artificiale (AI): queste tecniche sono in grado di simulare *intelligenza* trasformando problemi di apprendimento in un equivalente problema di ricerca. Una volta ricavata questa conoscenza si possono stabilire dei mapping intersorgente e costruire lo schema integrato.

1.4 Il sistema MOMIS

Considerando le problematiche descritte nei paragrafi precedenti, nonché alcuni sistemi preesistenti [59, 66, 67, 68, 69], si è giunti alla progettazione di un sistema intelligente di integrazione di informazioni da sorgenti di dati strutturati e semistrutturati denominato **MOMIS** (**M**ediator Envir**O**nment for **M**ultiple **I**nformation **S**ources). Il contributo innovativo di questo progetto, rispetto ad altri similari, risiede nella fase di analisi ed integrazione degli schemi sorgenti, realizzata in modo semiautomatico [70, 51, 71]. Un lavoro approfondito è stato svolto anche ri-

guardo alla fase di *query processing* ([72, 52, 73]), cioè per il processo che dalla query posta sullo schema unificato provvede a generare automaticamente le sottoquery da inviare alle sorgenti ed ad integrare i risultati. MOMIS nasce all'interno del progetto MURST 40% INTERDATA dalla collaborazione tra i gruppi operativi dell'Università di Modena e Reggio Emilia e di quella di Milano.

1.4.1 L'approccio adottato

MOMIS adotta un approccio di integrazione delle sorgenti *semantico* e *virtuale* [72]. Il concetto di semantico è stato illustrato nella sezione 1.3. Con virtuale si intende invece che la vista integrata delle sorgenti, rappresentata dallo schema globale, non viene *materializzata*, ma il sistema si basa sulla decomposizione delle query e sull'individuazione delle sorgenti da interrogare per generare delle subquery eseguibili localmente; lo schema globale dovrà inoltre disporre di tutte le informazioni atte alla fusione dei risultati ottenuti localmente per poter ottenere una risposta significativa.

Le motivazioni che hanno portato all'adozione di un approccio come quello descritto sono varie:

- la presenza di uno schema globale permette all'utente di formulare qualsiasi interrogazione che sia con esso consistente;
- le informazioni semantiche che comprende possono contribuire ad una eventuale ottimizzazione delle interrogazioni;
- l'adozione di una semantica *type as a set* per gli schemi permette di controllarne la consistenza facendo riferimento alle loro descrizioni;
- la vista virtuale rende il sistema estremamente flessibile, in grado cioè di sopportare frequenti cambiamenti sia nel numero che nel tipo delle sorgenti, ed anche nei loro contenuti (non occorre prevedere onerose politiche di allineamento);

Inoltre si è deciso di adottare un unico modello dei dati basato sul paradigma ad oggetti, sia per la rappresentazione degli schemi sia per la formulazione delle interrogazioni. Il modello comune dei dati utilizzato nel sistema (ODM_{I^3}) è di alto livello e facilita la comunicazione tra il mediatore ed i wrapper. Per definire questo modello si è cercato di seguire le raccomandazioni relative alla proposta di standardizzazione per i linguaggi di mediazione, nata in ambito I^3: un mediatore deve poter essere in grado di gestire sorgenti dotate di formalismi complessi (ad es. quello ad oggetti) ed altre decisamente più semplici (come i file di strutture), è quindi preferibile l'adozione di un formalismo il più completo possibile.

Per la descrizione degli schemi si è arrivati a definire il linguaggio ODL_{I^3}che si presenta come estensione del linguaggio standard ODL proposto dal gruppo di standardizzazione ODMG-93. Questo permette di cogliere le indicazioni emerse in ambito I^3 ed al contempo di discostarsi il meno possibile dalle proposte del gruppo appena citato. Una trattazione esauriente su ODL_{I^3} può comunque essere trovata in [52, 70, 51, 71].

Per quanto riguarda il linguaggio di interrogazione si è adottato OQL_{I^3} che adotta la sintassi OQL senza discostarsi dallo standard. Questo linguaggio richiede un maggiore sforzo dal punto di vista dello sviluppo di moduli per l'interpretazione e la gestione delle interrogazioni (implementando le funzionalità tipiche di un ODBMS), ma risulta altresì estremamente versatile ed espressivo fornendo la possibiltà di sfruttare le informazioni rappresentate nello schema globale. Le maggiori difficoltà implementative sono quindi ampiamente giustificate da una maggiore versatilità e da una migliore facilità d'uso per l'utente finale.

Riassumendo: in MOMIS i wrapper si incaricano di tradurre in ODL_{I^3} le descrizioni delle sorgenti, i moduli del mediatore di costruire lo schema globale, mentre tutte le interrogazioni sono poste dall'utente su questo schema utilizzando il linguaggio OQL, o meglio OQL_{I^3}, disinteressandosi dell'effettiva natura ed organizzazione delle sorgenti; sarà il sistema stesso che si incaricherà di tradurre le interrogazioni in un lin-

guaggio comprensibile dalle singole sorgenti e di riorganizzare le risposte ottenute in una unica corretta e completa da fornire all'utilizzatore.

1.4.2 L'architettura generale di MOMIS

In Figura 1.3 è illustrata dettagliatamente l'architettura generale di MOMIS. Lo schema evidenzia l'organizzazione a tre livelli utilizzata.

Livello Mediatore. Il nucleo centrale del sistema è costituito dal **Mediatore** (o *Mediator*) che presiede all'esecuzione di diverse operazioni. Per meglio comprendere i suoi compiti, è opportuno a questo punto illustrare le due fasi ben distinte in cui si articola la sua attività.

La prima funzionalià del Mediatore è quella di generazione dello Schema Globale. In questa fase il modulo del Mediatore denominato **Global Schema Builder** riceve in input le descrizioni degli schemi locali delle sorgenti espressi in ODL_{I^3}e forniti ognuno dal relativo wrapper. A questo punto (utilizzando strumenti di ausilio quali ODB-Tools Engine, WordNet, ARTEMIS) il Global Schema Builder è in grado di costruire la vista virtuale integrata (**Global Schema**) utilizzando tecniche di clustering e di Intelligenza Artificiale. In questa fase è prevista anche l'interazione con il progettista il quale, oltre ad inserire le regole di mapping, interviene nei processi che non possono essere svolti automaticamente dal sistema (come ad es. l'assegnamento dei nomi alle classi globali, la modifica di relazioni lessicali, ...). Oltre allo Schema Globale, altri "prodotti" di questa fase sono le **Mapping Table**, tabelle che descrivono il modo in cui gli attributi globali presenti nello schema generato hanno corrispondenza (*mappano*) nei vari attributi locali presenti negli schemi delle sorgenti.

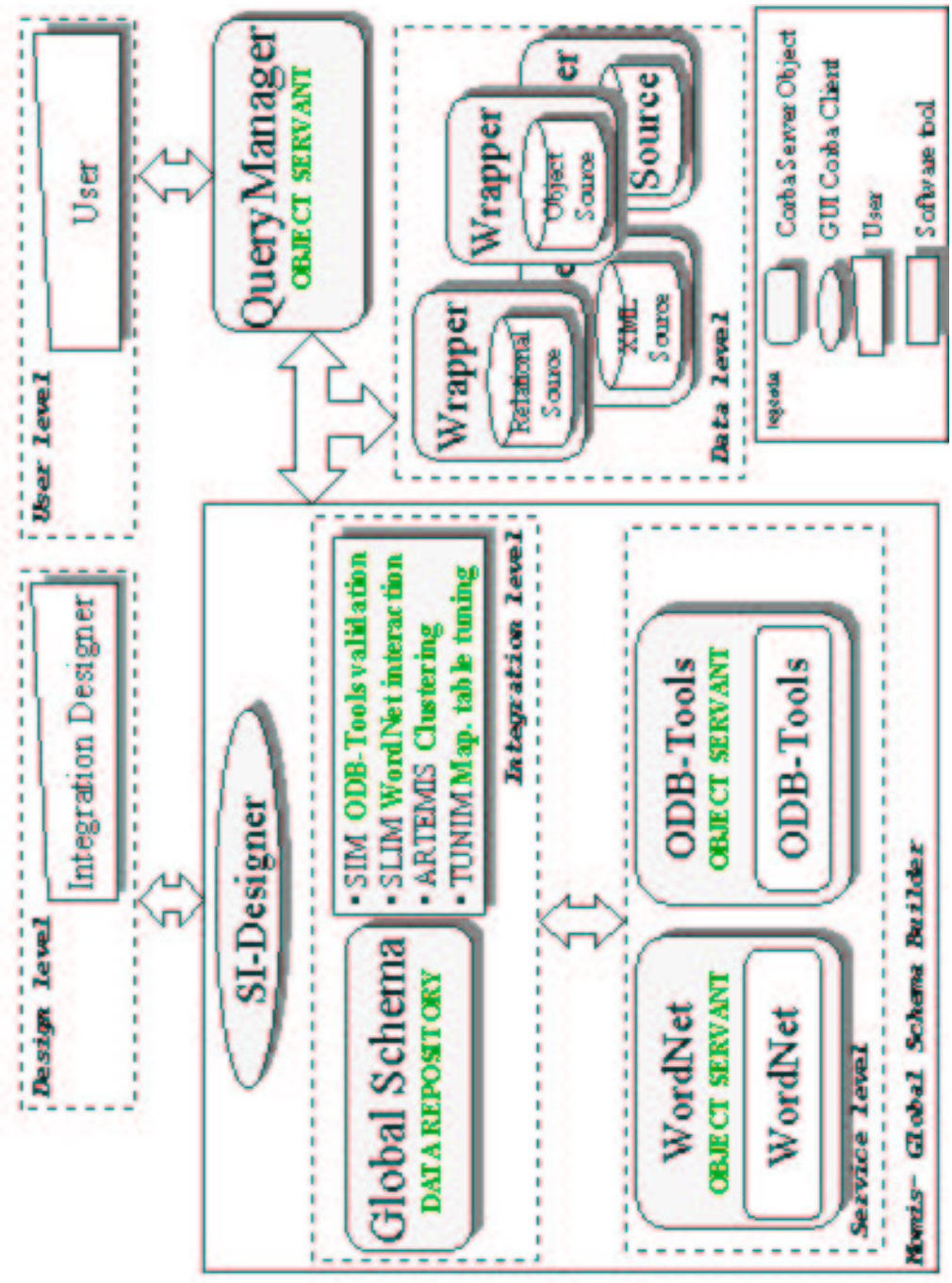

Figura 1.3: Architettura generale di MOMIS

Un secondo importante modulo che compone la struttura del mediatore è il **Query Manager** che presiede alla fase di query processing. In questa fase la singola query posta in OQL_{I^3}dall'utente sullo Schema Globale (che chiameremo *Global Query*) sarà rielaborata in più *Local Query* (anche'esse espresse in OQL_{I^3}) da inviare alle varie sorgenti, o meglio ai wrapper predisposti alla loro traduzione. Questa traduzione avviene in maniera automatica da parte del Query Manager utilizzando tecniche di logica descrittiva.

L'ultimo modulo del Mediatore è rappresentato dall'**Extensional Hierarchy Builder** il quale si occupa della generazione della Conoscenza Estensionale (Gerarchie Estensionali e Base Extension) necessaria per ottimizzare le interrogazioni.

Livello Wrapper. I **Wrapper** costituiscono l'interfaccia tra il mediatore e le sorgenti; ad ogni sorgente corrisponde un determinato wrapper ed ogni wrapper deve essere disegnato esclusivamente per la sorgente (o la tipologia di sorgenti) che sovrintenderà. Ogni wrapper ha due compiti ben precisi:

- in fase di integrazione deve fornire al Global Schema Builder la descrizione della sorgente da integrare in formato ODL$_{I^3}$;
- in fase di query processing deve tradurre la local query (rivolta alla "sua" sorgente) che gli è stata indirizzata dal Query Manager (e che è espressa in OQL$_{I^3}$) nel linguaggio di interrogazione specifico della sorgente per la quale é stato progettato.

Collegate ai wrapper sono quindi le **Sorgenti**, per questo a volte si parla anche di quattro livelli. Esse sono le fonti da integrare, possono essere DataBase (ad oggetti o relazionali) o parti di essi, file system ed anche sorgenti semistrutturate.
Il lavoro svolto nella seguente tesi ha avuto come obiettivo l'analisi di una serie di generatori di wrappers attualmente in circolazione, i

wrapper generati dovevano essere in grado di estrarre i dati contenuti in sorgenti di tipo HTML e di convertirli in formato XML. Una volta in questo formato è stato possibile sfruttare le funzionalità di un wrapper precedentemente messo a punto [78] al fine di portare a termine l'integrazione. Seguendo questa nuova prospettiva i livelli si possono considerare 4 (o 5 nel caso si voglia considerare anche quello delle sorgenti), in quanto sotto il livello wrapper non si ha più una sorgente ma si ha un ulteriore livello wrapper che serve a rendere disponibili i dati della sorgente in esame in un formato che permetta l'integrazione, in questo caso XML.

Livello Utente L'utilizzatore del sistema dovrà potere interrogare lo schema globale. L'accesso ai dati si propone del tutto simile a come avvengono le usuali interrogazioni di un DataBase tradizionale: le sorgenti ed il modo in cui i dati vengono recuperati risultano all'utente del tutto trasparenti, in quanto é il sistema ad occuparsi di tutte le operazioni necessarie per reperire le informazioni e combinare le risposte in un'unica risposta corretta, completa e non ridondante.

In precedenza si sono citati alcuni tool di ausilio per il mediatore, vediamo una loro rapida descrizione:

- *ODB-Tools* è uno strumento software sviluppato presso il dipartimento di Ingegneria dell'Università di Modena e Reggio Emilia [74, 75]. Esso si occupa della validazione di schemi e dell'ottimizzazione semantica di interrogazioni rivolte a Basi di Dati orientate agli Oggetti (OODB). Facciamo riferimento alla figura 1.4 per una breve spiegazione. ODB-Designer si occupa della validazione di schemi: si può inserire la descrizione di uno schema di DataBase (in ODL) ed il sistema realizzerà automaticamente la sua validazione e la sua riclassificazione; vale a dire che si occuperà di verificare che non esistano classi incoerenti (cioè che non possano essere popolate

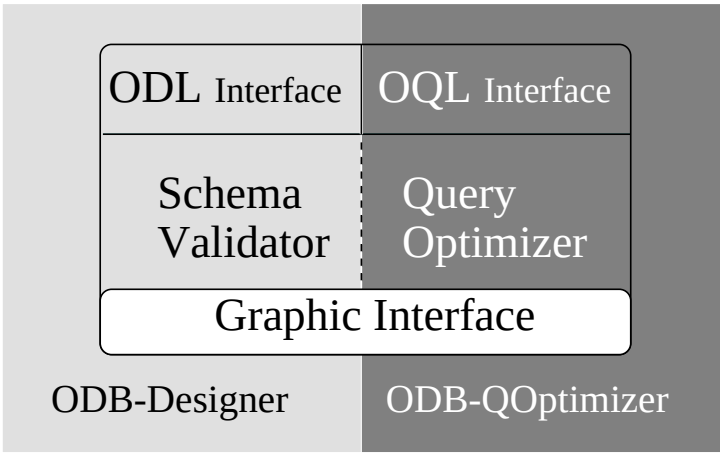

Figura 1.4: Architettura ODB-Tools

da nessun oggetto) e di determinare eventuali relazioni di specializzazione non esplicitate dallo schema stesso. ODB-Qoptimizer si occupa invece dell'ottimizzazione semantica delle interrogazioni: se si inserisce una query (in OQL) posta su di un determinato schema, questa viene automaticamente riformulata in una equivalente, ma più efficiente, sfruttando l'espansione semantica ed i vincoli di integrità.

- *WordNet* [76] è un DataBase lessicale on-line in lingua inglese. Esso è capace di individuare relazioni semantiche fra termini; cioè, dato un insieme di termini, WordNet è in grado di identificare l'insieme di relazioni lessicali che li legano.

- *ARTEMIS* [77] riceve in ingresso il *thesaurus*, cioè l'insieme delle relazioni terminologiche (lessicali e strutturali) precedentemente generate, e sulla base di queste assegna ad ogni classe coinvolta nelle relazioni un coefficiente numerico indicante il suo grado di affinità. Questi coefficienti verranno utilizzati per raggruppare le classi locali in modo tale che ogni gruppo (*cluster*) comprenda solo classi con coefficienti di affinità simili.

Capitolo 2

XWRAP Elite

Nella parte introduttiva è stato visto come la grande maggioranza delle informazioni disponibili in rete attualmente e con tutta probabilità anche nel prossimo futuro, sia costituita da documenti in formato "human readable" HTML. Al fine di poter utilizzare queste informazioni per qualcosa che vada oltre la navigazione e la ricerca effettuata "manualmente" dall'utente di internet, esse devono essere convertite in un formato "machine readable". Il compito di effettuare questa conversione viene attribuito ai wrappers.

La costruzione manuale di wrappers richiede una grande mole di lavoro e per sua natura non consente di avere un controllo diretto sugli errori commessi, tutto questo associato alla velocità di evoluzione dei contenuti sul Web non consentirebbe ad un qualunque approccio basato su questi strumenti software di fornire prestazioni accettabili.

Al fine di evitare, quando possibile, i problemi legati all'implementazione manuale di wrappers, sono stati sviluppati sistemi di generazione di wrappers semi-automatici che migliorano il processo di sviluppo di questi software anche se, a causa del costo dei miglioramenti forniti dal programmatore (successivamente verrà spiegato in dettaglio in cosa consistono), sono ancora ben lontani dall'essere sufficientemente scalabili per tenere il passo della crescita di pagine, siti e applicazioni Web. Altro problema con cui ci si deve confrontare nell'utilizzo dei wrappers

è la loro intrinseca fragilità al variare del contenuto delle pagine Web per cui sono stati creati.

In questo capitolo, sarà analizzato un aproccio per la costruzione di un sistema automatico per la generazione di wrappers per sorgenti informative Web: XWRAP Elite. Nel capitolo 7 sarà possibile trovare l'analisi che ha portato alla scelta di questo strumento come di quelli che saranno esaminati e testati nei capitoli successivi.

Questo strumento è una evoluzione di XWRAP Original, col quale condivide una parte della logica di funzionamento per poi distaccarsi completamente in quello che è il target dell'applicazione, ossia i tipi di pagine su cui l'estrazione è efficace. Se da una parte infatti XWRAP Original (che sarà per complettezza descritto in Appendice C), aveva nelle pagine di contenuto generico, come ad esempio le home pages dei siti personali, la sua killer application, dall'altra XWRAP Elite trova nelle pagine ricche di data-object come quelle create dinamicamente per rispondere a ricerche con parole chiave, il suo ambiente di funzionamento ideale.

L'obiettivo di XWRAP Elite è quello di fornire una metodologia per la trasformazione dello "human oriented" HTML nel "machine readable" e semanticamente significativo XML, questa trasformazione permetterà l'utilizzo sul documento sorgente di applicazioni basate su XML quali sofisticati sistemi di query, sistemi informativi basati sugli agenti e sistemi informativi basati su mediatori. Tra questi ultimi possiamo annoverare il sistema **MOMIS** sviluppato dall'Università di Modena.

La sfida principale è quella di generare la trasformazione automaticamente, richiedendo la minor quantità possibile di miglioramenti da parte del programmatore. Il principale contributo fornito è costituito da un insieme di algoritmi ed euristiche che bilanciano la richiesta di input semantici con l'automatizzazione di processi di estrazione dell'informazione e di generazione di wrappers.

Una delle caratteristiche dei wrappers generati automaticamente da XWRAP Elite è quella di fornire un buon grado di robustezza ai cam-

biamenti nei siti sui quali sono stati generati, questo soprattutto grazie alle modalità implicite nel loro processo di creazione. Questi software sezionano il processo di conversione in tre fasi:

- scoprire dove sono posizionati i dati all'interno di una pagina HTML e una volta trovati separarli in data-object di interesse
- decomporre gli oggetti individuati al passo precedente nei loro elementi costituenti
- effettuare la marcatura di oggetti ed elementi trovati in un formato di output.

XWRAP Elite automatizza le prime due procedure e richiede l'intervento dell'utente per migliorare le operazioni di marcatura dei dati di output al fine di poter individuare la maggior quantità di contenuti semantici possibile. In aggiunta a tutto ciò è stato inserito nel tool un componente in grado di generare il codice necessario per incapsulare tutte le parti create in un wrapper stand-alone.

2.1 Architettura del sistema

La Figura 2.1 mostra l'intera architettura del sistema XWRAP Elite. Il tool prende in input un semplice documento e genera un wrapper in grado di convertire l'HTML nel formato di dati XML. Questo processo si articola in quattro passi:

Primo passo

- viene eseguito il parsing del documento ed estratto l'HTML tree (cui ci si riferirà anche come tag tree)
- viene individuato il sottoalbero che contiene i data-object di interesse

- vengono scoperte regole che consentono di dividere il sottoalbero trovato al punto precedente in una serie di oggetti individuali (extraction rules)

Durante il primo passo viene generato un componente, chiamato *Object Extraction,* basato sull'individuazione del sottoalbero minimo e sull'insieme di extraction rules. Gli oggetti individuati vengono estratti e passati allo step successivo.

Secondo passo, XWRAP Elite studia gli oggetti estratti per ottenere un gruppo di elementi separatori e quindi decompone gli oggetti in elementi. Gli elementi separatori includono sia tags HTML che stringhe puramente testuali. In questa fase si determina un componente chiamato *Element Extraction.*

Terzo passo

- vengono analizzati gli elementi degli oggetti rimasti al fine di individuare modelli di elementi basati su espressioni regolari e sull'ordine degli elementi, e generare regole di allineamento per raggruppare elementi simili appartenenti ad oggetti diversi nella stessa locazione.
- Un programmatore di wrappers inserirà regole di marcatura assegnando il nome di un elemento ad ogni gruppo.

In questo passo viene generato un componente chiamato *Element Tagging* che soddisfi le regole di allineamento e le regole di tagging.

Quarto passo, viene realizzato un unico pacchetto che contiene il wrapper, formato dai seguenti componenti:

- Object Extraction
- Element Extraction

- Output Tagging

Il wrapper ottenuto può convertire una pagina HTML simile a quella fornita come campione all'inizio del processo di generazione, in un equivalente documento XML.

La procedura di estrazione automatizzata degli oggetti verrà discussa in dettaglio in Appendice B, nella sezione 2.3 verrà analizzata la fase di estrazione degli elementi dai singoli oggetti, nella 2.4 quella di arricchimento semantico tramite la fase di tagging.

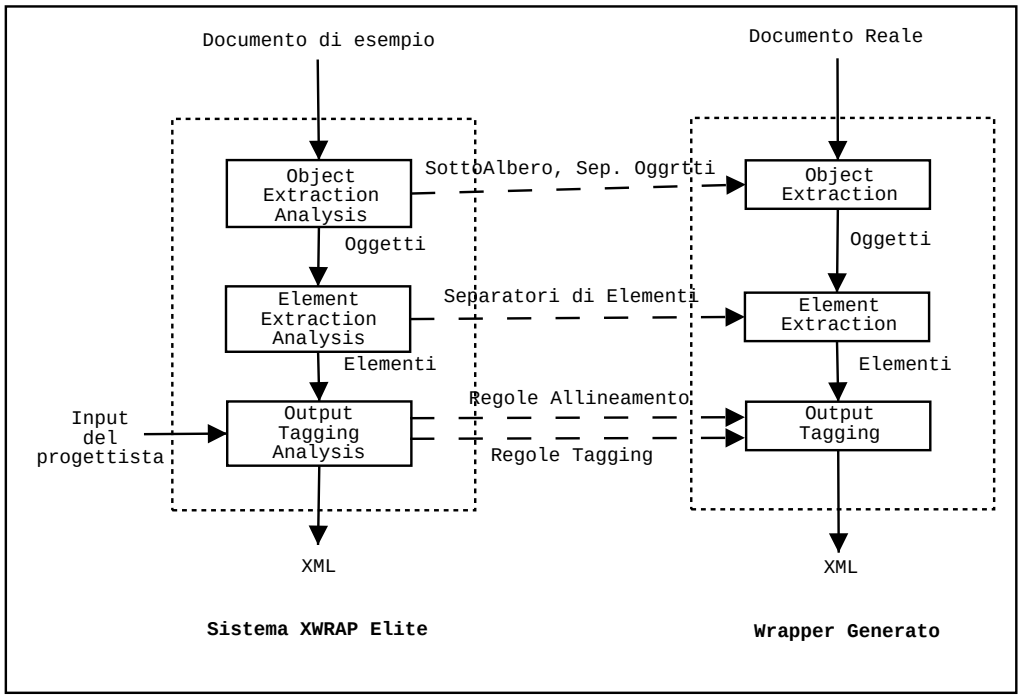

Figura 2.1: Architettura del Sistema XWRAP Elite

2.2 Estrazione degli oggetti

In questa fase XWRAP Elite utilizza le funzionalità offerte dal sistema Omini, anch'esso sviluppato dall'equipe di Georgia Tech. Essendo il

ruolo svolto da questo componente di fondamentale importanza all'interno dello strumento in esame ne verrà effettuato uno studio esaustivo in Appendice B.

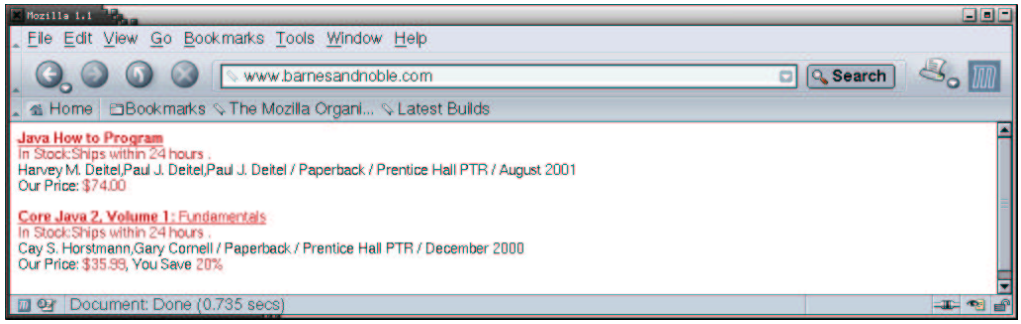

Figura 2.2: Due Data Objects Provenienti dal sito barnesandnoble.com

2.3 Element Extraction

Dopo che gli oggetti individuati sono stati estratti da una pagina, il passo successivo è quello di identificare gli elementi all'interno degli oggetti. Questo processo prende il nome di *Element Extraction.* Un data-object tipicamente consiste in un gruppo di elementi che sono separati da uno o più separatori. Un elemento separatore può essere sia un tag HTML, come <td> in una tabella, oppure un delimitatore puramente testuale, come il simbolo della barra in una stringa come questa: "Cay S. Horstmann, Gary Cornell / Paperback / Prentice All PTR / December 2000".

Dalle osservazioni effettuate su pagine create dinamicamente si è giunti alla conclusione che gli oggetti appartenenti alla stessa regione di contenuto in una pagina Web sono spesso omogenei, ciò significa che condividono la stessa struttura e lo stesso gruppo di elementi separarori. Questo permette di decomporre tutti i data objects in elementi, una volta che è stato correttamente individuato un gruppo comune di tags separatori. Nell'approccio XWRAP Elite, saranno quindi cercati tutti i separatori di

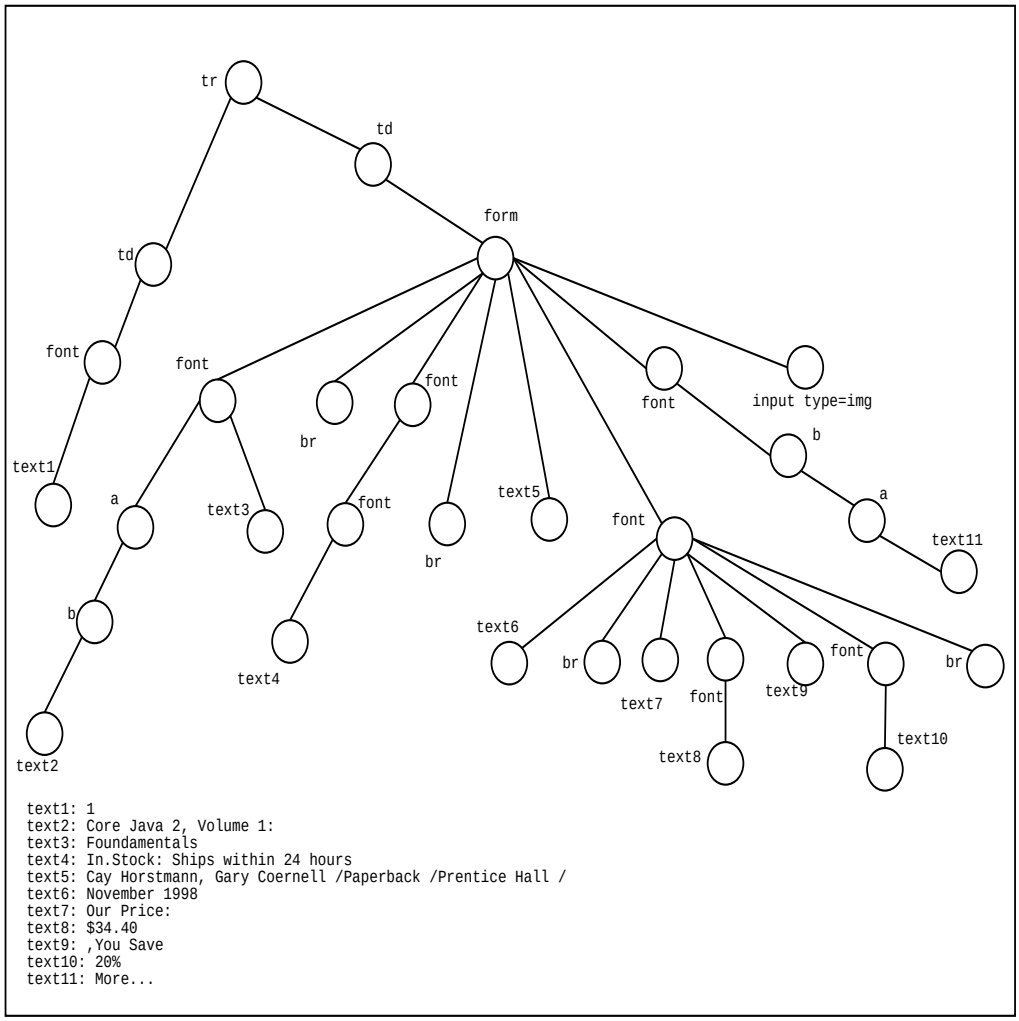

Figura 2.3: La struttura ad albero per il secondo oggetto in Figura 2.2

tag e i separatori di testo per ogni data-object e verrà estratto un gruppo di separatori adatti a tutti gli oggetti.

2.3.1 Separazione degli Elementi

Il set di euristiche utilizzate dal sistema Omini per scoprire un separatore di oggetti, non può essere applicato alla ricerca del gruppo di elementi separatori, per i seguenti motivi:

- gli oggetti sono simili tra loro, mentre il formato degli elementi in un oggetto è soggetto a forti variazioni.
- l'assunzione che sia presente sempre e solo un tag HTML che individua il confine tra oggetti, non è applicabile agli elementi.
- come è stato visto precedentemente, un separatore di elementi può anche essere un delimitatore testuale.

Sono stati scelti due approcci differenti, uno per scoprire separatori di tags e l'altro per quelli di parti testuali.

Tag Separators: sono stati costruiti tag trees per gli oggetti ed applicata un'euristica iniziale sugli alberi, al fine di ottenere un gruppo di separatori di tag. Quindi, viene applicato un set di altre tre euristiche complementari per raffinare il procedimento.

Text Separators: vengono estratte le stringhe di testo dagli oggetti e quindi analizzate per mezzo di un'euristica basata sulle stringhe, per ottenere i separatori nei contenuti testuali.

2.3.1.1 Separatori di tag

Si consideri un data-object e la corrispondente rappresentazione HTML tag-tree, i separatori di tag scompongono l'albero in sottoalberi che contengono elementi di interesse in forma completa. Questo significa che non separano nessun elemento in parti. Per portare un esempio, in Figura 2.3 si vede che <font> è un separatore di tag mentre <b> non lo è in quanto <b> contiene solo una parte del titolo del libro, "core Java, Volume 1: Fundamentals". L'appropriata combinazione è ottenuta incrementalmente:

- vengono individuati i separatori di tipo tags comunemente usati che sono sempre separatori all'interno dell'oggetto, come
 e <a>.

- sono applicate le euristiche complementari che verranno elencate di seguito, con l'intento di scoprire tag-separators supplementari

Highest Count Tag Heuristic

Il tag avente il maggior numero di occorrenze è quello con probabilità più alta di essere un separatore. Questa euristica tratta quindi il tag con numero di occorrenze maggiori come un tag-separator.

Repeating Pattern Heuristic

Alcuni tags spesso sono presenti più volte all'interno dello stesso oggetto nella forma di coppie aventi relazioni di adiacenza o padre-figlio. Se un tag appare di frequente in un pattern ripetitivo e molto meno di frequente appare isolatamente, è molto probabile che sia un tag-separator. La frequenza con cui un tag appare all'interno di un pattern ripetitivo è inversamente proporzionale alla differenza tra le occorrenze del tag e le occorrenze del pattern ripetitivo. Più piccola è tale differenza e più alta è la frequenza con cui il tag appare, di conseguenza più alta è la probabilità che sia un tag separatore.

Standard Deviation Heuristic

La deviazione standard dell'intervallo tra due occorrenze dello stesso tag è indice della regolarità del tag. Più piccola è la deviazione standard dell'intervallo di un tag e più regolari sono le presenze del tag, e quindi più è probabile che sia un tag-separator.

2.3.1.2 Separatori di testo

Gli elementi all'interno di stringhe puramente testuali sono solitamente separati da alcuni simboli, come "/" o " ". E' stata costruita una lista di separatori di tipo testuale comunemente usati, basata sui documenti Web che sono stati analizzati. La lista include i seguenti simboli: "\", ":", " ", ";".

Non sono stati introdotti separatori di linea, coè "**\n**", all'interno della lista perchè l'HTML tratta i separatori di linea come semplici spazi, così il separatore di linea è spesso usato anche all'interno di un elemento.

A causa del fatto che a volte questi separatori di testo vengono utilizzati per rappresentare simboli matematici (es: $15/5 = 3$) o per la marcatura di valori temporali (es: $"13 : 20p.m."$), le euristiche sono state implementate in modo da evitare questo problema, operando nel modo seguente:

- viene eseguito il parsing di un oggetto
- se ogni simbolo nella lista appare nell'oggetto senza che sia posizionato tra due cifre, il simbolo viene inserito in un gruppo di text-separators candidati che diventerà la lista valida per quell'oggetto.

Gruppo di Separatori Rappresentativo

Dopo che sono stati ottenuti i tag-separators ed i text-separators per ogni oggetto, viene costruito un gruppo di separatori rappresentativo il quale contiene solo separatori che compaiono nella maggior parte dei gruppi. Il criterio di selezione è dato dalla scelta dei separatori che compaiono in almeno due gruppi e in più del 5% dei guppi totali. Questa soluzione ha lo scopo di evitare che un simbolo venga riconosciuto come separatore di testo quando invece è parte integrante di un elemento di testo, come ad esempio la virgola in Figura 2.2.

2.4 Output Tagging

Dalla fase di estrazione sono stati ottenuti gli elementi costituenti ciascun data object. Al fine di poter fornire gli oggetti e gli elementi corrispondenti come output in formato XML, tali oggetti ed elementi devono essere marcati indicandone il loro significato semantico. Questo processo è chiamato Output Tagging. Verrà assunto che i data-object siano omogenei, in modo da poterli marcare tutti con lo stesso nome. Più

difficoltosa risulta la marcatura degli elementi in quanto un oggetto ha elementi multipli e alcuni elementi possono mancare in un oggetto, cosicchè si rende necessario identificare gli elementi prima di assegnare loro un nome.

Esistono due metodi per identificare gli elementi:

1. Eseguendo il matching con espressioni regolari.

   Espressioni regolari accurate sono molto difficili da ottenere automaticamente, in particolare quando gli elementi sono simili (come può capitare nei casi in esame), tutto ciò, unito al fatto che espressioni regolari non corrette possono portare alla perdita dei dati contenuti negli elementi, renderebbe inappropriato questo approccio per il problema in questione.

2. In base all'ordine di apparizione nell'oggetto.

   Questo è un buon metodo per specifici domini applicativi come per i logs risultanti da esperimenti di laboratorio, ma trova scarsa applicazione in altri campi.

La strada percorsa è quella di utilizzare un approccio ibrido, che minimizzi i problemi creati dall'utilizzo di ciascuno dei due approcci citati presi singolarmente. L'approccio di XWRAP Elite sfrutta quindi sia espressioni regolari che l'ordine di apparizione degli elementi negli oggetti. Si assegna un numero ad ogni elemento in base all'ordine in cui appare nell'oggetto. Alcuni modelli di espressioni regolari sono generati automaticamente, con lo scopo di fornire un ausilio alle operazioni di allineamento degli indici che devono essere effettuate quando in un oggetto manca un elemento rispetto all'oggetto di riferimento. Elementi appartenenti ad oggetti diversi ma aventi lo stesso ordine, verrebbero infatti marcati alla stessa maniera, come nell'esempio di tabella 2.1.

2.4.1 Individuazione di Modelli di Espressioni Regolari

I modelli di espressioni regolari possono essere di due tipi:

Strong-matching: normalmente implica una alta possibilità che due elementi possano essere allineati quando soddisfano tale modello. Di questo tipo sono elementi contenenti valori costanti quali "Our Price" nel nostro esempio ("constant-value elements"). Alcuni elementi contengono stringhe costanti con alcuni numeri variabili, per esempio "Ships with 24 hours". Se scartiamo le variabili (il valore "24" in questo caso), questi elementi possono essere considerati appartenenti ad un modello simile al precedente, ci riferiremo ad elementi di questo tipo come a "constant-value-like elements". Gli elementi constant-value e constant-value-like appaiono in una grande percentuale di oggetti, quindi possiamo facilmente individuarli dalle statistiche effettuate contando il numero delle loro occorrenze.

Weak-matching: indica che due elementi potrebbero non essere allineati se solo uno dei due soddisfa il modello. E' possibile riconoscere quattro tipi di modelli weak-matching, i quali corrispondono a quattro differenti classi di elementi:

- un link ipertestuale
- un'immagine
- un valore dollaro (ogni elemento inizia con "$" ed è seguito da numeri, es: "$34.00")
- stringhe comuni (gli elementi restanti)

Questi patterns ci aiutano ad evitare errori nell'individuazione degli elementi simili a quelli che si sarebbero compiuti marcando gli oggetti secondo la divisione in elementi restituita in Tabella 2.1.

2.5 Allineamento degli elementi

Questa fase ha l'obiettivo di stabilire una corrispondenza tra gli elementi di tutti gli oggetti appartenenti ad una pagina e quelli dell'oggetto aven-

te il maggior numero di elementi (quest'ultimo verrà anche chiamato oggetto di riferimento). L'insieme di elementi di quest'ultimo conterrà infatti l'insieme di elementi di qualunque altro oggetto.

L'idea di base è quella di assegnare ad ogni elemento un indice, il quale è l'ordine dell'elemento corrispondente all'interno dell'oggetto avente il maggior numero di elementi. Ad elementi aventi lo stesso indice viene assegnato lo stesso nome.

Si procede ora a spiegare la tecnica con la quale dato un elemento Elem, viene individuato il suo corrispondente all'interno dell'oggetto di riferimento.

- Se Elem è di tipo constant-value-like e nell'oggetto di riferimento si trova un elemento strong-matching, si assegna ad Elem l'indice dell'elemento incontrato nell'oggetto di riferimento. Se c'è più di un elemento strong-matching, si assegna ad Elem l'indice dell'elemento più simile al suo (la cui differenza è minore). Se non c'è un elemento corrispondente oppure Elem non è di tipo constant-value-like, si passa allo step successivo.

- Se un elemento nell'oggetto di riferimento possiede un indice maggiore dell'indice dell'elemento immediatamente precedente Elem, e costituisce un weak-matching pattern con Elem, allore si assume che sia l'elemento corrispondente. Se esiste più di un elemento corrispondente, scegliamo quello con l'indice più piccolo. Se non ci sono elementi corrispondenti si passa allo step successivo

- Se un elemento nell'oggetto di riferimento ha un indice uguale o minore dell'indice dell'elemento che è immediatamente precedente Elem e stabilisce un weak-matching pattern con Elem, assumiamo che sia l'elemento corrispondente. Se ci sono più elementi corrispondenti, scegliamo quello con l'indice maggiore. Se non ci sono elementi corrispondenti l'elemento Elem viene marcato come "unknown".

In Tabella 2.1 sono mostrati gli elementi dei due oggetti mostrati in Figura 2.2. Oggetto1 ha 14 elementi, mentre Oggetto2 solamente 12, a causa della mancanza dei 2 elementi relativi allo sconto. Il 10^o elemento in Oggetto2 (<a href=”morelink”>) non incontra il 10^o di Oggetto1 (You Save), ma è dello stesso tipo (ipertesto) del 12^o elemento di Oggetto1, analogamente l'11^o elemento di Oggetto2 (more) ha lo stesso valore del 13^o in Oggetto1. La mancanza in Oggetto2 dei due elementi relativi allo sconto, presenti invece in Oggetto1, introduce uno sfasamento nelle corrispondenze tra i due oggetti che deve essere eliminando ricorrendo ad un allineamento.

Ordine	Oggetto1	Oggetto2
1	<a href=” ... “>	<a href=” ... “>
2	Core Java2, Volume 1: Foundamentals	Java How To Program
3	In-Stock: Ships with 2 hours	In-Stock: Ships with 24 hours
4	Cay Horstman, Gary Cornell	Harvey M. Deitel, Paul J, Deytel
5	Paperback	Paperback
6	Prentice Hall	Prentice Hall
7	November 1998	August 1999
8	Our Price:	Our Price:
9	$34.40	$64.75
10	You Save	<a href=”more link”>
11	20%	more
12	<a href=”more link”>	<img src=” .. “>
13	more	
14	<img src=” ... “>	

Tabella 2.1: Elementi dai due oggetti di Figura 2.2 prima dell'allineamento

È possibile, a questo scopo, allineare gli elementi 10^o, 11^o, 12^o in Oggetto2 con 12^o, 13^o, 14^o di Oggetto1, come mostrato in Tabella 2.2.

Ordine	Oggetto1	Oggetto2
1	<a href=" ... ">	<a href=" ...">
2	Core Java2, Volume1: Foundamentals	Java How to Program
3	In-Stock: Ships with 24 hours	In-Stock: Ships with 24 hours
4	Cay Horstmann, Gary Cornell	Harvey J. Deytel, Paul J. Deytel
5	Paperback	Paperback
6	Prentice Hall	Prentice Hall
7	November 1998	August 1999
8	Our Price:	Our Price:
9	$34.40	$64.75
10	You Save	
11	20%	
12	<a href=" more link">	<a href=" more link ">
13	more	more
14	<img src=" ... ">	<img src=" ... ">

Tabella 2.2: Elementi dai due oggetti di Figura 2.2 dopo l'allineamento

2.6 Risultati della Sperimentazione Diretta

In un primo momento per valutare l'efficacia dell'approccio adottato dal sistema Omini ed indirettamente di XWRAP Elite, sono stati presi come riferimento i risultati delle sperimentazioni effettuate dall'equipe di Georgia Tech che verranno illustrati in Appendice B. Per rendere esaustivo lo studio proposto in questa tesi, non poteva mancare una sperimentazione diretta sul tool, in grado di farci "toccare con mano" il modo di operare degli algoritmi e delle euristiche che sono stati descritti a livello teorico e dalla cui comprensione non si poteva prescindere, nell'intenzione di valutare con cognizione di causa le prestazioni effettive di questo strumento.

Il materiale su cui effettuare le prove era costituito da un paniere di siti Web riguardanti la realtà industriale modenese, diversi dei quali era-

no dotati di un motore di ricerca interno in grado di funzionare mediante l'input di parole chiave, altri fornivano direttamente un elenco di queste ultime tra cui effettuare la scelta, in entrambi i casi le pagine venivano costruite in maniera dinamica. I siti erano strutturalmente tutti differenti, per fornire una rappresentazione più fedele possibile della realtà che si può trovare in rete. In aggiunta a questi, sono stati considerati una serie di siti indicati dagli sviluppatori di XWRAP Elite.

Il toolkit viene messo a disposizione solo mediante collegamento remoto, molto meglio sarebbe stato avere la possibilità di scaricarlo direttamente su una macchina locale, in modo da avere il controllo sui collegamenti verso altri server. Uno dei problemi con cui ci si è dovuti scontrare, è stato infatti rappresentato dalle prestazioni del proxy collegato al server su cui il Toolkit risiede. Le pagine su cui si vogliono effettuare le analisi vengono scaricate sul server in modo che XWRAP Elite vi possa accedere direttamente, questa operazione è filtrata da un proxy che dopo innumerevoli tentativi, ha evidenziato una lunga serie di comportamenti che sembrano riconducibili a problemi di configurazione. Si è cercato di dare un senso logico al comportamento di questo componente, ma alla fine l'impressione è che nell'operare incontri problemi che non è in grado di risolvere da solo.

I problemi creati dal proxy, non sono comunque stati gli unici che si sono verificati, in seguito verrà fornita una breve descrizione dei limiti incontrati nell'uso di questo strumento, alcuni riconducibili alla sua logica di funzionamento, altri a metodologie atipiche di costruzione dei siti, di cui comuque non si può fare a meno di tenere in considerazione vista l'eterogeneità delle sorgenti disponibili sul Web.

Uno dei fattori di destabilizzazione del toolkit è costituito dalla presenza di immagini nelle pagine restituite dai motori di ricerca, caso tra l'altro piuttosto comune anche nei siti di riferimento messi a disposizione dagli stessi sviluppatori per testare lo strumento (un esempio può essere trovato in figura 2.4). Questi siti, che per la maggior parte riguardavano il commercio elettronico di libri, creavano infatti problemi nel

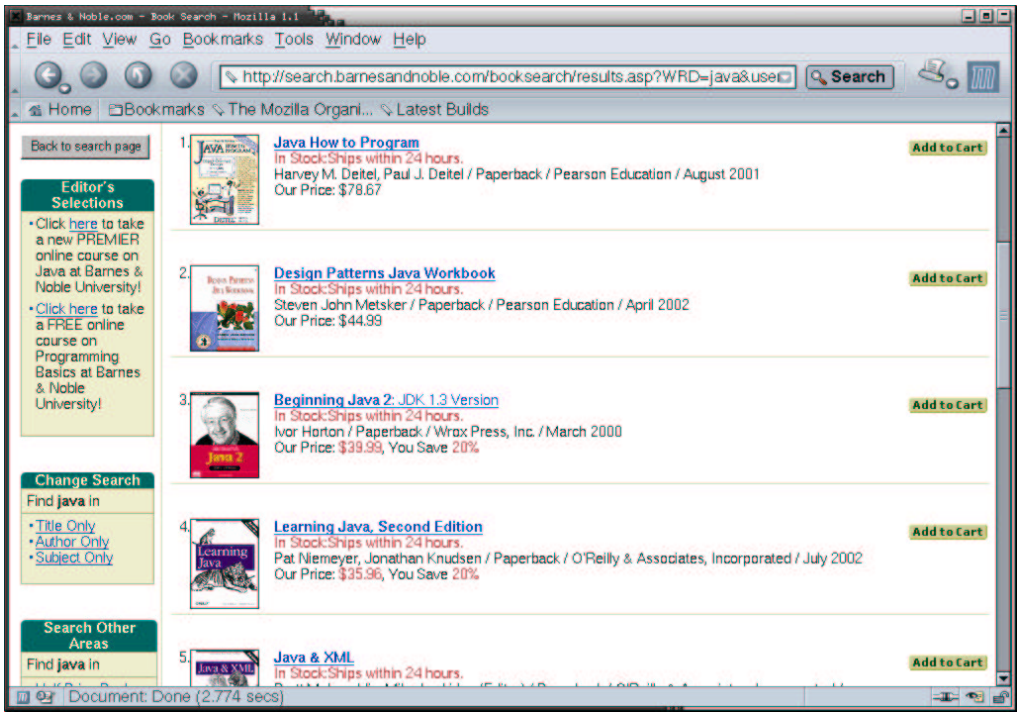

Figura 2.4: Risultati di una ricerca sul sito www.bn.com

momento stesso in cui ciò che veniva restituito dalla creazione dinamica della pagina (dopo l'immissione di una chiave) comprendeva l'immagine della copertina del libro e a volte un bottone per l'aggiunta dell'articolo al carrello, quest'ultimo, essendo gestito per lo più con linguaggi di scripting quali javascript, vbscript o simili, ha spesso disorientato il tool, che evidentemente non contiene strumenti per la gestione di questo tipo di oggetti.

Fonte di guai è stata anche la strutturazione delle pagine di alcuni siti, questa era dotata di struttura semplice solo ad uno sguardo superficiale della presentazione, andando a studiare il codice HTML si è scoperto che le informazioni erano organizzate sfruttando tabelle annidate che andavano anche oltre le due dimensioni (righe per colonne), creando strutture che XWRAP Elite non è in grado di decifrare correttamente. Dalle prove effettuate si nota infatti che solo le tabelle in due dimen-

Figura 2.5: Voce "Immobili/Attività" del sito www.tessilmoda.com

sioni sono supportate con risultati accettabili dalle euristiche di Omini oltre, la presenza di errori nella separazione degli oggetti diventa una certezza. Questo inconveniente si è verificato in maniera lampante su uno dei nostri siti test: tessilmoda.com (vedi figura 2.5). Selezionando a titolo di esempio la voce "Bacheche" viene restituita una pagina dove i diversi data object di interesse sono separati in tabelle. Questa situazione sembrerebbe l'ideale per poter ottenere buoni risultati dal separatore di oggetti. Dopo aver effettuato le prove con le euristiche a disposizione si scopre però, che nessuna è in grado di individuare il tag separatore tra le singole tabelle. Andando a vedere cosa c'è dietro la visualizzazione della pagina si scopre che quelle che sembrano semplici tabelle in realtà sono strutture composte da diverse tabelle HTML, annidate l'una dentro l'altra oppure in sequenza, al fine di restituire come risultato di visualizzazione un'unica tabella. Facendo riferimento al caso appena discusso,

possiamo notare che ad ogni annuncio è associato un data object, esaminando il codice HTML corrispondente emerge che ogni data object è strutturato come una tabella che ne contiene altre due al proprio interno, innestate allo stesso livello: una prima tabella è riservata all'intestazione orizzontale del data object, mentre una seconda tabella viene riservata al contenuto vero e proprio. Applicando XWRAP Elite, viene individuata correttamente la zona della pagina in cui sono contenute le informazioni di interesse, ossia la tabella che contiene tutti i data object, quando si passa alla fase di individuazione del corretto separatore di oggetti il tool marca come tale quello che separa le tabelle di livello più interno, identificando come singoli oggetti le intestazioni che però sono solo una parte del data object.

La realizzazione di un wrapper è stata possibile per la sezione del sito *"Settori di Attività"* relativamente alla voce *"Trasporto"*, bisogna però precisare che tale ricerca ha restituito un solo data object ed il componente di XWRAP Elite che sembra essere fonte di maggiori guai è proprio l'object extractor; una volta messo questo componente nelle condizioni di non poter sbagliare (fornendogli come nel caso in esame un documento con un unico oggetto) il resto del tool fornisce i risultati attesi. Un caso analogo lo incontriamo nella pagina *"NewEntry"* selezionando "Linea Erre" (vedi figura 2.7). Questi ultimi due esempi ci hanno permesso di riscontrare l'efficacia del componente element extractor del sistema, il quale ha fornito una separazione più che accettabile dell'oggetto in esame. In figura 2.6 viene riportato il contenuto del file XML ottenuto dalla voce trasporto.

Altro problema con cui ci si è dovuti confrontare in non poche occasioni è stato quello della determinazione del sottoalbero minimo nel quale sono presenti gli oggetti di interesse, se da una parte è vero infatti che un sottoalbero spesso viene individuato, è altrettanto vero che non sempre questo contiene tutti i dati di cui si voleva ottenere la conversione. Per come è stato definito il sottoalbero minimo, la situazione non

```
<ResultSet><sourceUrl>http://www.tessilmoda.com/
cnaptServ/ricerca_aziende2.jsp?
categoria_ricerca_1=%2B22</sourceUrl>
<httpMethod>get</httpMethod><content>
<record name="Trasporto">
<URL><link src="http://www.tessilmoda.com/
cnaptServ/mostra_azienda.jsp?cl_cliente=627"/>
</URL>
<Nome_Azienda>(1) SANA ALLESTIMENTI S.R.L.
</Nome_Azienda>
<Attività>Produzione Conto Proprio</Attività>
<Mercati>Mercati: Estero; Italia</Mercati>
<Notizie>Notizie Utili</Notizie>
<Laboratori>Ampiezza Laboratorio: Oltre 300mq
</Laboratori>
<Addetti>Numero Addetti: Oltre 10 Addetti</Addetti>
<Tipo_Trasporto>Trasporto Proprio: Si
</Tipo_Trasporto>
<Attività>Trasporto</Attività>
<Attività>Fornitori di Attrezzature</Attività>
<Attività>Servizi per la Moda: Studio tendenze moda
</Attività>
<null><img src="http://www.tessilmoda.com/
                cnapt/images/hr.gif"/>
</null>
</record>
</content></ResultSet>
```

Figura 2.6: Contenuto del file XML ottenuto dalla voce trasporto

sembra avere soluzione, bisogna quindi accontentarsi di disporre della parte dei dati più significativa. Per portare un esempio pratico, nel sito www.codysbooks.com dopo aver effettuato una ricerca attraverso una parola chiave, la prima parte del risultato è costituita dai 10 libri più venduti e questa parte non appartenendo al sottoalbero minimo non viene presa in considerazione da XWRAP Elite. La situazione peggiora quando la pagina alloca le informazioni di interesse su più di un frame, come accadrebbe nell'esempio precedente se per l'elenco dei 10 libri più venduti venisse creato un altro frame. La scelta è effettuata automaticamente dal tool senza possibilità di intervento da parte del programmatore di

Figura 2.7: Voce "New Entry/Linea Erre" del sito www.tessilmoda.com

wrapper. In questo caso il fatto che il tool sia completamente automatico rappresenta un'arma a doppio taglio.

Una modifica che mi sento di suggerire per quanto riguarda la ricerca del sottoalbero minimo contenente i data-object di interesse è quella di dare maggiore importanza alla divisione in frame della pagina Web. Quando questa divisione è presente, i risultati della ricerca sono spesso posizionati in basso a destra, ed è quindi in questo frame che bisogna andare ad approfondire la ricerca dei data-object di interesse.

Per quanto riguarda le prestazioni, nel testare XWRAP Elite viene subito alla luce che il principale collo di bottiglia nel processo di generazione dei wrapper, è il numero di iterazioni necessarie per raggiungere una significativa "copertura"del sito Web.

Un altro aspetto che suscita qualche perplessità nella logica di funzionamento è legato al fatto che la pagina di esempio a partire dalla quale è

stato generato il wrapper potrebbe essere troppo semplice per convertire pagine più complesse, anche se appartenenti alla stessa ricerca. Pensiamo ad esempio ad un wrapper dotato di più di una parola chiave, se è stato costruito prendendo come esempio le pagine della prima keyword non è detto che non incontri difficoltà nella traduzione di pagine restituite da una ricerca che utilizza una delle altre. Volendo portare un esempio pratico di questa situazione si pensi ad una ricerca su un sito di libri e articoli di cronaca, in cui la ricerca per autore di libri restituisce oggetti che contengono solo il nome dell'autore (come capita nella maggior parte dei casi per i romanzi), mentre in quella di articoli viene riportato sia il nome dell'autore che quello del fotografo, se il wrapper viene costruito su una pagina nella quale sono stati restituiti solo libri aventi un unico autore, nell'utilizzo del wrapper per ricerche di articoli è molto probabile che si verifichino problemi.

Il tool per quanto è stato possibile verificare, non fornisce nessun ausilio alla sicurezza. Nelle situazioni in cui è stata richiesta una connessione attraverso il protocollo HTTPS, in grado di supportare lo scambio di dati mediante SSL (Secure Sockets Layer) e che utilizza la porta 110 invece della 80, non viene fornito alcun tipo di risultato. Problema non trascurabile quest'ultimo dal momento che molti dei siti di commercio elettronico si stanno indirizzando verso questo tipo di connessione. In futuro quindi, se il tool non verrà modificato, potrà essere utilizzato solo su quei siti che gestiscono il catalogo attraverso connessioni al server di tipo HTTP e solo nella seconda fase, quella di acquisto, viene richiesta una connessione HTTPS. Quest'ultimo è uno dei problemi maggiori riscontrati tentando di testare lo strumento sul sito www.interpreta.it.

Utilizzando il sopracitato sito, sembrano creare qualche problema di troppo anche i frame. Nella voce *"avvenimenti"* infatti (vedi figura 2.8), il tool riconosce correttamente i due data-object presenti, ma non riesce ad uscire dalla fase di rifinitura dell'estrazione di oggetti ed elementi, per arrivare a quella successiva di tagging. Questo problema sembra essere riconducibile al fatto che rimanga visibile, anche se seminascosto in

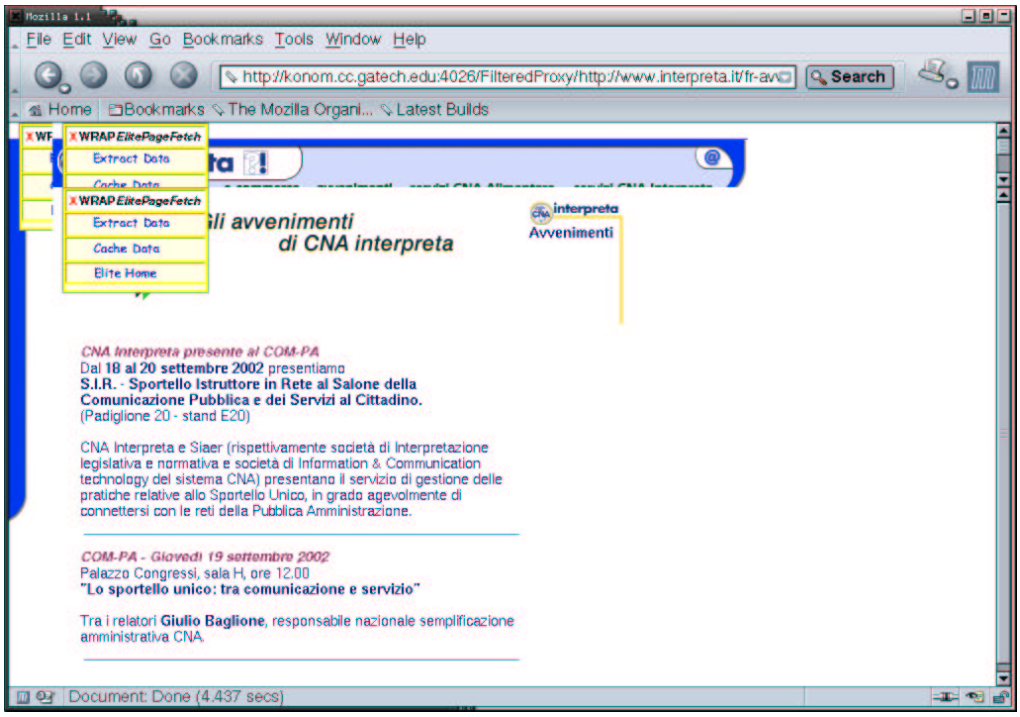

Figura 2.8: Voce "Avvenimenti" del sito www.interpreta.it

secondo piano, uno dei frame della pagina precedente, che ha portato al raggiungimento di quella in esame. Quest'ultimo problema, come quello relativo al mancato supporto del protocollo HTTPS, è stato risolto al fine di portare avanti la fase di testing, ed estrarre dati anche da pagine che facevano uso di frames non perfettamente riconosciuti e di connessioni sicure, caricando localmente le pagine sul server sparc20.ing.unimo.it e utilizzando come input di XWRAP Elite gli URL che partivano da questo indirizzo invece di quelli relativi agli indirizzi originali.

Un ultimo test è stato eseguito per valutare i risultati dell'applicazione di XWRAP Elite ad un tipo di pagina non data intensive come ad esempio la home page di tessilmoda.com (vedi figura 2.9). Per riuscire nell'intento di tradurre questa pagina bisognerebbe far riconoscere ad XWRAP ogni area in cui è suddiviso il sito come un oggetto, fatto questo bisognerebbe, per ogni oggetto individuato, identificare e dare signi-

Figura 2.9: Home page del sito www.tessilmoda.com

ficato semantico agli elementi che lo compongono. Il problema è che XWRAP Elite, una volta individuati gli oggetti, o quantomeno i possibili oggetti all'interno della pagina, attribuisce o chiede al programmatore di attribuire, i nomi e i significati agli elementi di uno di questi oggetti che verrà utilizzato come riferimento per assegnare tali valori anche agli altri oggetti della pagina che però in questo caso non sono oggetti simili. Questo problema è legato al fatto che XWRAP Elite è stato sviluppato per poter estrarre automaticamente il codice XML da pagine che contengono oggetti tutti dello stesso tipo, ben differenti quindi da quella in esame.

Ai risultati tutt'altro che incoraggianti ottenuti sui siti di riferimento, fanno da contraltare quelli ottenuti sui siti consigliati dagli sviluppatori stessi che con ogni probabilità, possiamo immaginare essere quelli su cui lo strumento ha fornito le prestazioni migliori. In effetti bisogna ammettere che in alcuni di questi casi il lavoro dell'operatore è stato veramente

minimo ed i risultati eccellenti. Per altri però non si può dire altrettanto, evidentemente le nuove impaginazioni dei layout di presentazione di questi siti, sono state in grado di mettere in difficoltà le euristiche, cosa che non facevano le versioni precedenti, presumibilmente per il fatto di essere più semplici.

La lunghezza dell'albero del documento generato dal wrapper è proporzionale alla dimensione del documento di input, non è però detto che questo rapporto influenzi la dimensione del documento XML finale e quindi le prestazioni del tool in questa fase.

Si procede ad una breve analisi delle prestazioni temporali del toolkit nelle quattro fasi di cui si compone il processo di estrazione:

- Connessione al server e lettura del documento
- Espansione dell'albero del documento
- Estrazione dei data-object e degli elementi in essi contenuti
- Generazione del codice sorgente del wrapper e successiva precompilazione

I dati relativi alla prima fase non verranno tenuti in considerazione in quanto non è stato possibile utilizzare il toolkit in modalità locale ma si è reso necessario il suo utilizzo mediante connessione remota. Dopo la fase di connessione al server del Georgia Tech e quella di immissione dell'URL del sito di interesse, è stato necessario attendere che il server sul quale risiede il sito mettesse a disposizione le pagine desiderate al server del Georgia Tech, il quale a sua volta le restituiva all'utente dopo averle caricate in memoria locale.

Delle altre tre fasi quella che influisce maggiormente sul tempo totale è la seconda, questo anche a causa di un'invocazione del pacchetto Swing di Java da parte della routine preposta; se da una parte questa soluzione migliora la comprensibilità della struttura della pagina mediante l'utilizzo delle migliori potenzialità grafiche di cui dispone il linguaggio Java, dall'altra rallenta parecchio l'esecuzione dell'operazione. Oltre

questo possiamo immaginare i notevoli problemi di parsing che stanno dietro questa parte del progetto e che non facilitano l'ottimizzazione dei tempi. La terza fase sebbene facente uso anch'essa del pacchetto Swing riesce a sfruttare algoritmi di ottimizzazione che le permettono di pesare meno in termini di tempo sul risultato finale. L'influenza della quarta fase dipende ovviamente dalla dimensione del file XML di output.

2.7 L'Evoluzione delle Pagine HTML come Spiegazione dei Risultati dei Test

Dalla sua infanzia, nei primi anni '90, fino a pochi anni fa, il linguaggio HTML si è continuamente evoluto, introducendo un numero crescente di elementi di complessità progettuale. Fra questi citiamo: tabelle all'interno di tabelle, frames e mappe di immagini, che non erano tra gli elementi iniziali. Più tardi sono venuti i linguaggi di scripting che hanno permesso applicazioni client-side. Per fare un esempio di una delle loro applicazioni più diffuse possiamo citare la gestione degli eventi del mouse (es:il mouse all'interno di un'immagine) o la gestione del bottone relativo al carrello della spesa in molti siti di commercio elettronico. Queste hanno avuto il merito di migliorare l'interattività con i siti Web e hanno permesso a questi ultimi di funzionare come reali applicazioni.

Negli ultimi anni, la parte grafica delle pagine HTML si è stabilizzata e gli sviluppatori del Web hanno spostato la loro attenzione su Web standards più orientati alla programmazione vera e propria, quali XML. Oggi, l'HTML non si evolve più come linguaggio e a parte l'incompatibilità che esiste tra le caratteristiche dell'HTML supportate dai diversi browsers, l'HTML stesso ed i relativi tools di sviluppo e progettazione sono maturi e producono un formato di output universalmente riconosciuto e senza possibiltà di essere male interpretato.

Come risultato di questi sviluppi, è ragionevole pensare che certi paradigmi di progettazione nei siti Web siano ralizzati utilizzando un set

di costrutti HTML consistenti e prevedibili. Per esempio, fatta eccezione per i grafici complessi e gli script client-side, c'è un solo modo di creare menu a tendina su una pagina Web, è cioè usando i tags <select> e <option>. Analogamente caselle per l'inserimento di testo, radio buttons e check box sono specificati utilizzando un set di tags ben definito. La prevedibilità di alcune caratteristiche di HTML non rimuove l'incertezza che accompagna gli aspetti più critici dell'estrazione dei dati che sono in qualche modo collegati alle modalità di disegno della pagina, in termini tecnici al layout della stessa.

Il layout della pagina è definito quasi esclusivamente attraverso l'annidamento dei tags <table>, <tr>, <td> e <frame>. Chiunque abbia osservato l'evoluzione del Web in questi anni, non ha potuto fare a meno di constatare l'aumento di complessità delle pagine Web, che si traduce in termini pratici in intere sezioni di una pagina assegnate a soddisfare vari bisogni di tipo tecnico o economico o quant'altro, un altro aspetto di questo processo di evoluzione è dato dalla ricchezza di annunci, di titoli, di barre di navigazione e di ricerca.

Come conseguenza della crescente strutturazione, quasi sempre il "contenuto principale" di una pagina è spinto sempre più in basso nell'albero HTML. Questo significa che nella maggior parte dei casi i dati di interesse si trovano da qualche parte in una tabella profondamente innestata, rendendo critico il compito di mettere a punto strumenti che siano in grado di recuperare questi dati con un buon grado di robustezza.

Da notare anche che i servers Web attuali rendono semplice combinare l'output di differenti applicazioni e databases in una singola pagina HTML.

Su un tipico portale Web, ogni sezione della pagina (tempo, news, titoli azionari, etc.) possono essere prodotti da una portlet indipendente. In questo approccio distribuito, il processo di progettazione ha l'ulteriore fardello di assicurare la totale compatibilità tra le differenti sezioni. Se il controllo totale non viene mantenuto, c'è la possibilità di generare "aggregati HTML" che sono semplicemente invalidi (es: due

elementi <body>, due forms con lo stesso nome, scripts client-side che entrano in conflitto o caratteri binari generati dall'inclusione di files erroneamente tradotti). Tutto questo non facilità sicuramente la vita agli strumenti automatici di estrazione delle informazioni che come è stato evidenziato nelle sezioni precedenti trovano nell'individuazione del sottoalbero minimo e nel corretto separatore di oggetti due dei loro compiti più gravosi.

2.8 Caratteristiche Utili di XWRAP Elite

Se da una parte il progetto di XWRAP Elite è accomunato a strumenti analoghi, utilizzati all'interno di progetti di integrazione di sorgenti di dati HTML che si basano sull'estrazione della struttura delle pagine contenute in queste sorgenti, dal fatto di utilizzare la codifica delle regole di estrazione in files di descrizione (DTD) e dall'utilizzo di euristiche basate su particolari tag HTML, dall'altro questo toolkit stabilisce un deciso punto di rottura rispetto a tutti gli altri approcci.

- La prima caratteristica distintiva è la sua struttura di generazione del codice a due fasi che presenta tra gli altri vantaggi quello di fornire un'interfaccia user-friendly per permettere agli utenti di generare le loro regole di estrazione dell'informazione con pochi click. Per quanto emerso dalla fase di testing dello strumento, questa è una delle funzionalità più riuscite messe a disposizione dell'utente anche se inevitabilmente ne rappresenta un vincolo alla piena automatizzazione del processo di generazione.

- La seconda peculiarità di questo strumento software è quella di fornire una netta separazione delle fasi di estrazione delle informazioni semantiche e di generazione dei wrappers (realizzati in codice Java), tale separazione permette a nuove regole di estrazione di essere incorporate dal wrapper programmer incrementalmente.

- Terza ma non meno importante distinzione è l'uso dell'approccio micro-feed-back per rivedere e aggiustare i programmi wrappers a run-time.

Degno di nota è anche il fatto che XWRAP Elite separa esplicitamente i compiti di costruzione dei wrappers che sono specifici di una risorsa Web da quelli che sono ripetuti in ogni risorsa e per questi ultimi viene realizzata una libreria di componenti che fornisce blocchi di base per la costruzione di wrappers. Una parte innovativa del progetto è legata agli algoritmi di apprendimento induttivo che derivano o scoprono modelli di wrapper elaborando le informazioni contenute in documenti presi come esempio.

Vale la pena focalizzare l'attenzione su una caratteristica dei wrappers generati da XWRAP Elite che potrebbe essere molto utile in un prossimo futuro al fine di integrare lo strumento all'interno del modulo mediatore MOMIS. Durante la fase di generazione del wrapper è possibile parametrizzare una o più parti dell'indirizzo della pagina Web in esame, in modo da poter fornire una serie di input al wrapper nel momento dell'invocazione. Questi input serviranno al tool per identificare con precisione la pagina da cui estrarre le informazioni. Una volta terminato il processo di generazione del wrapper per una determinata pagina Web è possibile infatti scaricarsi un intero package java contenuto in un archivio .jar che a sua volta contiene i sorgenti ed il bytecode dei componenti del wrapper nonchè la DTD (Document Type Declaration) della pagina esaminata. Lanciando direttamente il wrapper da riga di comando e fornendo il necessario input al programma si riesce ad indicare con esattezza la pagina da cui estrarre i dati ed il file in cui memorizzarli. Questa fuzionalità torna particolarmente utile nel caso in cui i dati restituiti da una ricerca siano memorizzati su più pagine tutte dello stesso tipo, fornendo come input al wrapper il numero della pagina, questo parametro viene sostituito nell'opportuna locazione all'interno dell'URL ed i dati estratti verranno automaticamente memorizzati in un file di output in formato XML. Per fare un esempio pratico consideriamo l'URL di una delle pagi-

ne restituite in seguito ad una ricerca relativa ad attività agricole di Allevamento Animali sul sito www.expomo.it: `http://www.expomo.it/istat_finale.asp?pag=2&codice=125&comune=Tutti...` durante la fase di generazione del wrapper si assegna un parametro all'attributo *"pag"* dell'indirizzo, in modo che una volta creato il wrapper sarà necessario specificare in input un valore da assegnare a tale attributo. Se il package java contenente il wrapper estratto è stato contrassegnato con il nome: WrapperExpomo.jar l'invocazione risulterà del tipo: *"java WrapperExpomo output.xml 1"* dove *1* è il numero della pagina da cui estrarre i dati e *output.xml* è il nome del file in cui i dati restituiti devono essere memorizzati. Ovviamente non sarebbe possibile ricordarsi il formato di invocazione di ciascuno dei wrappers generati, per questo motivo tale formato è restituito invocando il comando: *"jar NomePackage"*. Questa procedura può essere effettuata automaticamente da una macchina senza intervento da parte dell'uomo. Dal momento che il sistema MOMIS crea dinamicamente la vista globale dei dati, sarebbe possibile all'atto di questa creazione far partire una routine che lancia i wrappers sulla prima pagina di interesse e continua automaticamente l'estrazione su quelle successive modificando autonomamente il valore del parametro o dei parametri di invocazione del wrapper.

Vale la pena far notare ancora una volta che tutte le informazioni importanti come il numero di tabelle in una pagina, il numero di attributi in una tabella,etc., in XWRAP Elite vengono calcolate a run-time. E' questo uno dei meccanismi che contribuiscono alla robustezza dei wrappers. Tutto ciò, unito al fatto che le regole di estrazione delle regioni sono definite in un linguaggio dichiarativo indipendente dalla implementazione del codice del wrapper, consente di operare ad un livello di astrazione sufficientemente alto da permettere ai wrappers generati di godere di una buona adattabilità nei confonti dei cambiamenti della sorgente remota.

Le idee ed i risultati di XWRAP Elite appaiono essere efficaci per molte sorgenti Web di dati semistrutturati ma c'è comunque bisogno di

wrappers più avanzati al fine di allargare la portata delle sorgenti per cui possiamo generare wrappers.

Sono allo studio strumenti, uno dei quali anche al Georgia Institute of Technology per la generazione avanzata di data-wrappers con la capacità di maneggiare tabelle complesse espresse su più di due dimensioni e sistemi di query su dati semistrutturati che hanno condizioni di ricerca complesse. Altro filone di ricerca è naturalmente quello che ha come obiettivo l'individuazione di meccanismi che permettano di aumentare l'affidabilità dei wrapper generati.

Aree di miglioramento individuate sono a mio avviso:

- da un lato, fornire migliori strumenti per assistere l'utente nella scelta di pagine del sito Web di interesse che verranno prese come esempio per la generazione del wrapper (al fine di evitare di effettuare la costruzione del wrapper basandosi su pagine troppo semplici che non permetteranno di ottenere buone prestazioni su pagine risultanti da altre ricerche sempre all'interno dello stesso sito (è stato visto che la situazione ottimale per la creazione del wrapper, è quella in cui il programma viene generato sulla base delle pagine più complesse che una ricerca su un sito sia in grado di restituire, in modo che quando lo stesso wrapper si ritrova a dover tradurre pagine più semplici non incontri problemi, cosa che non si può verificare nel caso contrario).

- dall'altro, come precedentemente accennato, l'incorporazione di algoritmi di apprendimento automatico (alcuni attualmente già in fase di studio) per definire regole di estrazione delle informazioni più robuste.

Altro aspetto su cui si può intervenire è quello dell'arricchimento del linguaggio in cui sono espresse le regole di estrazione delle informazioni e della libreria di componenti attraverso l'utilizzo di capacità avanzate di scoperta di modelli e varie tecniche di ottimizzazione (ricordiamo il

grande utilizzo che si è fatto del pacchetto Swing di Java che se da un lato rassicura l'utente con caratteristiche grafiche di sicuro effetto, aumentando l'intuitività di certe scelte, dall'altro costituisce una vera e propria ancora per quanto riguarda le prestazioni).

Sono allo studio ricerche per permettere l'incorporazione di tecnologie di immagazzinamento della Microsoft che permettano la gestione di metadati e per dare una risposta più concreta alla questione del frequente aggiornamento dei siti da cui si vanno a recuperare i dati. Un aspetto su cui si potrebbe effettuare uno studio è quello relativo alle opportunità fornite da XWRAP Elite nel caso di integrazione di funzionalità che gli permettano di seguire hyperlinks, funzionalità che potrebbero essere integrate a livello di information extraction oppure anche di information integration e che sono già in possesso di strumenti concorrenti quali RoadRunner dell'Università di Roma e Andes della IBM e che verranno analizzati rispettivamente nei capitoli 3 e 4.

L'utilità dei wrappers è confermata dal loro utilizzo all'interno di ricerche il cui scopo è quello di trovare una soluzione alla questione delle interrogazioni su dati semistrutturati. L'obiettivo è stato quello di trovare una soluzione simile a quella adottata per le basi di dati tradizionali. Questi sforzi riguardano aspetti quali lo sviluppo di modelli di dati e di query languages per dati semistrutturati, che siano in grado di fornire un valido supporto per la definizione di una semantica formale per tali linguaggi di interrogazione e di implementarli efficentemente.

L'interesse verso questi tipi di problematiche è dimostrato dal fatto che alcuni servizi di wrapping a livello commerciale sono già disponibili su internet, vedi a titolo di esempio Junglee (acquistato da Amazon.com), Jango (adottato da Excite) o mySimon. Questi sono capaci di estrarre informazioni da grandi sorgenti di dati HTML e il loro uso attraverso siti in grado di soddisfare le ricerche di migliaia di potenziali acquirenti ne dimostra la validità del progetto che ne sta alla base.

Capitolo 3

Il Sistema RoadRunner

RoadRunner è un progetto di ricerca sviluppato presso l'Università di Roma Tre [25], il cui intento è quello di sviluppare soluzioni per l'estrazione automatica di dati da sorgenti HTML . Questo capitolo presenta uno studio sul progetto, nella sezione 3.1 vengono presentati gli obiettivi, provando a dare una spiegazione dell'idea chiave che sta alla base del progetto e delle caratteristiche principali rispetto alle altre soluzioni proposte in letteratura. Nella sezione 3.2 sono illustrati gli algoritmi su cui si basa l'approccio. Nella sezione 3.3 viene data una descrizione a livello globale dell'architettura del sistema, all'interno di questa ci si soffermerà su ciascuno dei moduli componenti.

3.1 Descrizone del Progetto

L'obiettivo di questa ricerca, analogamente a quanto visto per XWRAP Elite, sono i siti Web data-intensive, ossia siti che pubblicano una grande mole di dati in strutture complesse correttamente realizzate (aventi cioè una serie di regole di costruzione alla base). I dati sono solitamente memorizzati in DBMS e le pagine HTML vengono dinamicamente generate usando scripts che attingono dal contenuto del data-base. In altre parole questi script effettuano queries sui databases e rendono disponibili i risultati in formato HTML. Ogni script genera una classe di pagine aventi

contenuti e presentazioni omogenee. Questi siti solitamente contengono differenti classi di pagine, corrispondenti ognuna a differenti tipi di contenuto. L'obiettivo è quello di poter effettuare query su queste sorgenti in un formato simile a quello con il quale si interroga un database.

Per fare questo viene seguito un approccio in due passi:

1. viene fornito un livello wrapper che estrae le parti rilevanti delle informazioni contenute nelle pagine HTML originali, presenti sui siti e le restituisce come una collezione di documenti XML aventi la stessa DTD o lo stesso XML Schema

2. quindi, basato su questi wrappers, può essere usato un linguaggio di query che fa uso di XML (come XQuery[15], o un'altro fra quelli recentemente proposti[16]), per esprimere interrogazioni sui documenti XML derivati.

Data la disponibilità di query languages per XML, la complessità maggiore in questo progetto consta nella generazione dei wrappers che effettuano la traduzione da HTML ad XML.

3.1.1 Inferenza di Grammatiche con Ausilio di Supervisore

La generazione di un wrapper per un set di pagine HTML corrisponde ad ottenere una grammatica per il codice HTML della pagina, solitamente una grammatica regolare, e quindi ad impiegare questa grammatica per effettuare il parsing della pagina ed estrarre parti di dati. L'inferenza di grammatiche è un problema già conosciuto e ampiamente studiato [17]. Il fatto che espressioni regolari non possano essere apprese unicamente attraverso esempi positivi, e l'alta complessità dell'apprendimento anche in presenza di informazioni aggiuntive (quali ad esempio un set di esempi negativi), ha limitato l'applicabilità delle tradizionali tecniche di inferenza grammaticale su siti Web. Questo ha motivato la nascita di

una serie di approcci pragmatici alla generazione di wrappers per pagine HTML. Questi lavori si sono avvicinati alla generazione di wrappers inquadrando il problema da diverse prospettive, che vanno dall'apprendimento automatico da parte delle macchine [7, 18, 19, 20, 21], al data mining [1], alla modellizzazione concettuale [22, 23]. Sebbene tutte queste proposte differiscano nel formalismo utilizzato per la specifica del wrappers, esse ereditano tutte l'approccio di "supervisionamento dell'apprendimento" della tradizionale inferenza grammaticale, condividendo così una serie di caratteristiche comuni:

- Selezione manuale di pagine di esempio:

  Viene assunto che il wrapper programmer selezioni una collezione di pagine HTML omogenee dalle quali i dati devono essere estratti, mentre il sistema di generazione di wrappers ottiene il wrapper dal lavoro svolto su una singola pagina alla volta. Quindi il wrapper viene utilizzato per effettuare il parsing delle rimanenti pagine, possibilmente generalizzando la grammatica nel momento in cui questo si rendesse necessario.

- Disponibilità di esempi etichettati:

  Il generatore di wrappers lavora utilizzando informazioni aggiuntive sul contenuto della pagina in esame, tipicamente queste sono costituite da un set di esempi etichettati forniti dall'utente o da qualche altro tools esterno. Il wrapper è ottenuto dalle osservazioni su questi esempi positivi e dal tentativo di generalizzarli.

- Conoscenze a priori sullo schema obiettivo:

  Solitamente viene assunto che il sistema di realizzazione dei wrappers abbia alcune conoscenze a priori sullo schema dei dati nella pagina. Diversi lavori assumono che le pagine obiettivo della traduzione contengano una collezione di record simili tra loro come struttura e in due dimensioni (tabelle semplici che allocano i campi in cui contenere i dati relativi ad un unico oggetto secondo una

semplice distribuzione righe per colonne), in altri casi [1] il sistema è in grado di gestire dati annidati, questo a patto che l'utente specifichi quali sono gli attributi da estrarre e come sono innestati.

Questi lavori essenzialmente si occupano del problema dell'estrazione che possiamo riassumere così: "data una o più pagine HTML, uno schema obiettivo (solitamente una lista di record unici) per queste pagine, ed un set di esempi etichettati, trovare un set di regole di estrazione che permetta di effettuare il parsing di codice HTML e di recuperare i data-object in accordo con lo schema obiettivo".

3.1.2 L'Approccio RoadRunner

Prendendo spunto dalle caratteristiche comuni di diversi di questi sistemi, l'approccio RoadRunner focalizza l'attenzione sul problema della generazione di wrappers, l'obiettivo è quello di raggiungere la piena automatizzazione del processo, eliminando la necessità di un supervisore. Le caratteristiche distintive di questo progetto rispetto agli altri presenti in letteratura sono le seguenti:

- Non verrà assunto che le pagine campioni debbano essere selezionate manualmente dal progettista, al contrario si assume che il sistema sia in grado di raggruppare pagine provenienti da porzioni di siti in classi omogenee.
- Il sistema non si affida ad esempi etichettati e specificati dall'utente, inoltre non viene richiesta nessuna interazione con l'utente durante il processo di generazione del wrappers. Questo significa che il processo di generazione dei wrappers e quello di estrazione dei dati sono effettuati in maniera completamente automatica.
- Per finire, non viene assunta nessuna conoscenza a priori riguardo allo schema delle pagine obiettivo, il sistema non conosce lo schema attraverso il quale sono organizzate le pagine HTML, questo

schema verrà dedotto insieme con il wrapper. Inoltre il sistema non è ristretto unicamente a record bidimensionali, ma può gestire strutture arbitrariamente innestate.

Rispetto agli altri lavori presenti in letteratura, questo sistema si occupa di differenti problemi, quali il ritrovamento dello schema ed il problema dell'estrazione dei dati [24], possiamo riassumere questo obiettivo nel seguente modo: "dato un set di pagine HTML, trovare lo schema (arbitrariamente innestato) per il contenuto di queste pagine, ed un set di extraction rules che permettano di effettuare il parsing del codice HTML e recuperare i dati in accordo con lo schema individuato". Questo cambio di prospettiva sull'intero problema dell'estrazione di informazioni da pagine Web rappresenta la principale sorgente di originalità della ricerca.

Da un lato, questo costringe a considerare il problema dell'inferenza grammaticale per pagine HTML e ad escogitare nuove tecnologie per fronteggiarlo: la principale intuizione dietro le proposte dell'equipe di Roma sta nel fatto che la scoperta di modelli (pattern) può basarsi sullo studio di analogie e differenze tra le pagine; in particolare è stato proposto [25] un nuovo approccio all'inferenza di wrappers da pagine HTML, nel quale, al fine di separare i modelli significativi da quelli non interessanti, vengono utilizzate due pagine alla volta, e le analogie sono utilizzate per individuare strutture di interesse.

D'altro canto, lo studio si è trovato a dover fronteggiare problemi che non erano stati considerati prima in quanto la presenza di input forniti dall'utente li rendeva irrilevanti. Questi problemi vanno da quello della selezione di collezioni di pagine omogenee all'interno di un sito, pagine aventi la stessa struttura, a quello dell'etichettatura di attributi nello schema obiettivo, una volta che questo è stato scoperto (ad esempio indicare che la stringa "$15.99" è un prezzo e che "Java for Dummies" è il titolo di un libro).

Nelle sezioni che seguono verranno descritti i vari moduli che compongono l'architettura del sistema e come questi concorrono alla solu-

zione di questi problemi.

3.2 Algoritmo di Generazione dei Wrappers

Nell'approccio in esame è stato detto che un wrapper viene considerato alla stregua di una grammatica regolare, che può effettuare il parsing su una pagina HTML al fine di identificare i data-object che la compongono. Per essere più precisi, dato un alfabeto di simboli terminali $\sum$, ed un set di simboli non terminali N, verrà considerato un sottoinsieme delle grammatiche regolari, corrispondenti a espressioni regolari costruite sull'insieme $\sum \bigcup N$ usando i seguenti operatori (ξ è la sequenza nulla):

- concatenazione, nella forma a $\cdot$ b, es: la sequenza a AND b
- iterazione, nella forma a^+, es: la ripetizione di a una o più volte;
- opzionalità, nella forma $(a)?$, un'abbreviazione per $a|\xi$. Per esempio, la seguente espressione regolare può definire un wrapper nel formalismo utilizzato dagli sviluppatori ($ denota un elemento non terminale) :

```
<HTML> ...Player: <B>$Name</B>
       $Brand - our price:
<STRONG>$Price</STRONG>
<UL> (<IL> <FONT>$Rating</FONT>
     (<I>$Review</I>)?)+</UL>
... <HTML>
```

Da notare che, rispetto alle più generiche grammatiche regolari, sono permesse un numero molto limitato di disgiunzioni, fondamentalmente solo quelle nascoste in opzioni (zero o uno), e dentro iterazioni (uno o più). Essenzialmente questo tipo di wrapper corrisponde all'unione di grammatiche regolari, introducendo ovviamente limitazioni nel potere

espressivo del linguaggio di wrapping, anche se secondo gli sviluppatori il linguaggio contiene molti dei tipici modelli che si presentano in sorgenti HTML ben strutturate [25].

Poichè i wrappers effettuano il parsing del codice HTML, essi hanno come obiettivo una specifica organizzazione di pagina ed un preciso layout. Per questo motivo, al fine di estrarre i dati da un sito, è necessario derivare un wrapper per ogni classe di pagine all'interno del sito. Nel progetto RoadRunner questa attività di decodifica è basata sulle similitudini che sono presenti tra pagine HTML appartenenti alla stessa classe. Il caso più frequente è quello in cui si vuole costruire una grammatica per una classe di pagine a partire dalle informazioni fornite da alcuni esempi.

Recentemente è stata sviluppata una tecnica originale per ricavare un wrapper da una classe di pagine, basata sull'analisi di similitudini e differenze tra alcune pagine di esempio della classe [25]. In pratica, dato un set di pagine HTML campione, la tecnica utilizzata effettua il confronto tra il codice HTML delle sorgenti, al fine di trovare le corrispondenze tra le parti, e in base a queste, rifinisce progressivamente il wrapper. L'output è una grammatica che può essere utilizzata per il parsing delle pagine della classe al fine di estrarre data-objects. Il componente che si occupa di confrontare le pagine appartenenti ad una stessa classe, alla ricerca di corrispondenze, e di inferire una struttura comune ed un wrapper, prende il nome di **Aligner**. Dato che le tecniche di matching sono basate sul confronto tra pagine dello stesso tipo, un'assunzione critica è data dal fatto che le pagine provenienti dal sito, devono essere in qualche modo raggruppate in base alle differenti classi a cui appartengono. Per questo motivo, al fine di supportare l'algoritmo descritto sopra, è necessario adottare alcune tecniche di clustering per pagine HTML che permettaono una rapida classificazione delle pagine in base al loro tipo. Queste tecniche mirano a dare una buona approssimazione delle classi di pagine presenti nel sito, con l'intento di portare avanti la fase di decodifica.

D'altra parte, i siti di solito contengono anche pagine isolate, come la home page, cioè pagine per le quali non ci sono altre pagine nel sito con la stessa organizzazione e layout. Queste pagine hanno principalmente lo scopo di offrire percorsi di accesso alla navigazione dei dati contenuti nel sito: il componente Aligner descritto precedentemente non può ovviamente essere applicato a questo tipo di pagine in quanto è basato sul confronto tra due o più pagine e sulla presenza di similarità. Nasce quindi l'esigenza di sviluppare tecniche specifiche per effettuare il wrapping di questi tipi di pagine.

Infine, ogni volta che uno schema annidato viene inferito dalle pagine appartenenti ad una classe, questi ha campi anonimi. Per accrescere la semantica dello schema, ad ogni campo deve essere associato un nome significativo, questa operazione dovrebbe essere effettuata manualmente dopo che il set di dati è stato estratto. Per raggiungere l'obiettivo di rendere il RoadRunner completamente automatico, i progettisti hanno sviluppato alcune forme di post-processing del wrapper che permettano autonomamente di ricavare i nomi degli attributi. L'architettura software del sistema RoadRunner, discussa nella prossima sezione, ha lo scopo di fornire soluzioni a tutti questi problemi.

3.3 Architettura del Sistema RoadRunner

All'interno del sistema è possibile identificare quattro moduli principali, ognuno dei quali affronta uno specifico problema:

- il CLASSIFICATORE analizza pagine del sito obiettivo e le colleziona in raggruppamenti aventi strutture omogenee. Questo modulo incorpora un agente che naviga all'interno del sito e sfrutta una serie di euristiche per produrre una buona approssimazione delle classi di pagine presenti all'interno del sito, da notare che alcune di queste classi conterranno pagine candidate ad essere passate al componente Aligner per la generazione del wrapper, mentre altre conterranno pagine individuali che avranno elementi singoli

- la generazione di wrappers per classi con pagine simili è eseguita da Aligner, il quale implementa le tecniche per individuare le corrispondenze. Per ogni classe di pagine, il componente Aligner, confronta le sorgenti HTML di alcune pagine di esempio, al fine di estrarre una grammatica da usare come wrapper per l'intera classe

- classi aventi pagine individuali sono passate ad un modulo Expander, il quale tenta di inferire un wrapper da queste. I wrappers estratti attaverso questo modulo sono basati su tecniche differenti rispetto a quelli ottenuti tramite il modulo Aligner

- l'ultimo componente è il Labeler, il quale si occupa di associare significato semantico ai campi di dati che possono essere estratti eseguendo i wrappers generati dai moduli citati precedentemente sulle pagine Web. Esso attribuisce un nome appropriato ad ogni simbolo non terminale della grammatica del wrapper

Nel seguito verranno brevemente illustrate le principali caratteristiche dei vari componenti.

3.3.1 Il Modulo Aligner

Questo modulo implementa la tecnica ACME, che rappresenta il nocciolo di tutto il sistema in esame. Nel seguito verrà data una sommaria descrizione di questa tecnica, ulteriori dettagli ed esperimenti possono essere trovati in [25].

ACME prende in input il codice sorgente di pagine HTML come lista di tokens (un analizzatore lessicale trasforma le pagine in liste di tokens), dove ogni token non è altro che un tag HTML o una stringa, e lavora su due oggetti alla volta:

- una lista di tokens chiamata il campione

- un wrapper, ad es: un insieme di espressioni regolari.

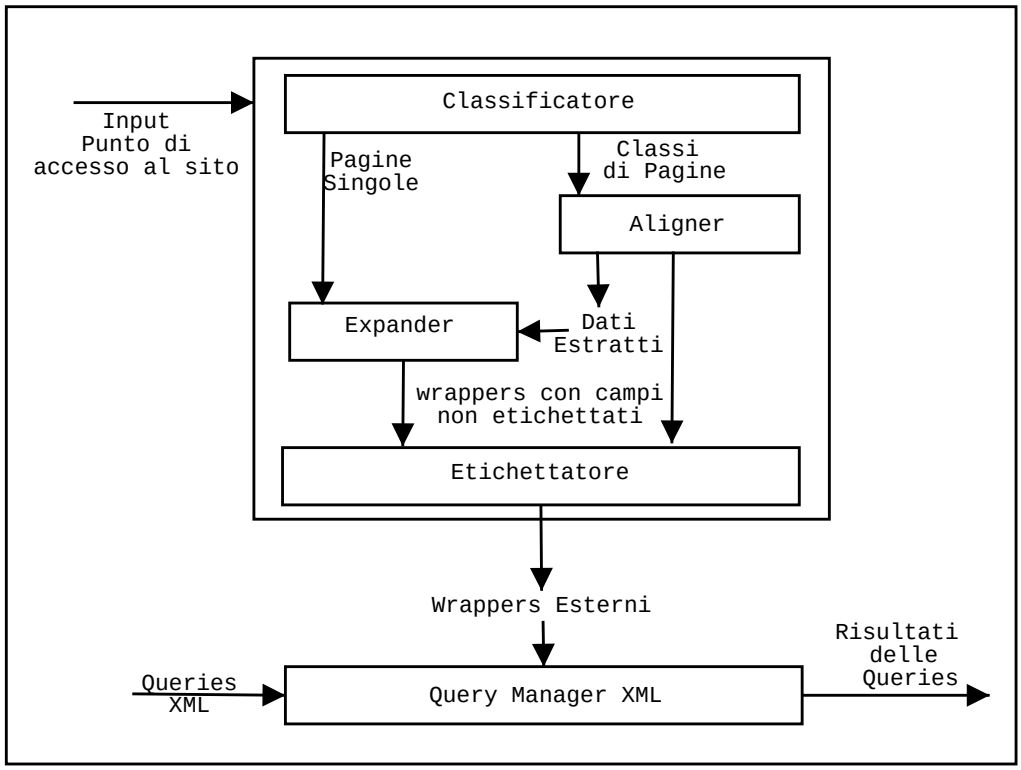

Figura 3.1: Architettura del Sistema

Date due pagine HTML, viene presa la prima come versione iniziale del wrapper, quindi il wrapper viene progressivamente rifinito per trovare una espressione regolare che vada bene per le due pagine. Questo è fatto individuando progressivamente le corrispondenze tra il wrapper e l'esempio. Il wrapper può quindi venire ulteriormente rifinito dall'applicazione iterativa della stessa tecnica sugli esempi di una collezione di pagine HTML della stessa classe. Gli esperimenti condotti dal gruppo di ricerca di Roma dimostrano che un piccolo numero di pagine di esempio (3-6) è sufficiente per permettere l'estrazione di un wrapper per tutte le pagine di una classe di grandi dimensioni.

3.3.2 Il Modulo Classificatore

L'obiettivo del classificatore è quello di identificare efficientemente le differenti classi di pagine all'interno dei siti in esame. Per ogni classe, un certo numero di esempi verrà passato in input al modulo Aligner, al fine di generare i wrappers corrispondenti. Il classificatore si muove all'interno del sito al fine di raggruppare le sue pagine HTML in base alla grammatica a cui obbediscono. Il processo di raggruppamento è basato su tecniche note. Queste tecniche vedono le pagine di esempio da raggruppare come punti in uno spazio n-dimensionale, chiamato spazio caratteristico; i raggruppamenti vengono generati minimizzando la distanza media tra punti all'interno di un cluster [25].

Un punto chiave in questo processo è la scelta del giusto mapping di una pagina di esempio all'interno dello spazio caratteristico. Questo è fatto estraendo alcune caratteristiche rilevanti dalla pagina campione ed usandole come coordinate all'interno dello spazio caratteristico. Nel contesto in cui si muove RoadRunner, determinare caratteristiche utili diventa in realtà un nuovo problema, differente dai classici scenari a cui queste tecniche vengono applicate. Infatti all'interno delle pagine di esempio è presente una nozione di analogia tra esempi rappresentata dalla conformità ad una grammatica regolare comune e questo è un concetto totalmente nuovo. Per dare in'idea di come questa nuova nozione di analogia, renda il problema del raggruppamento differente dai classici problemi di raggruppamento di stringhe, consideriamo le seguenti caratteristiche:

- solitamente due stringhe sono considerate simili se contengono sottoparti comuni. Al contrario, nel caso di pagine Web, richiedere che due pagine contengano sottoparti comuni non è sufficiente per garantire che su di esse possa essere effettuato il parsing utilizzando la stessa grammatica, in quanto è necessaria una forma più profonda di similarità strutturale

- tipicamente stringhe che sono considerate simili dagli usuali meto-

di hanno approssimativamente la stessa lunghezza; nel caso in esame pagine all'interno della stessa classe possono avere dimensioni ampiamente differenti, si pensi a pagine che restituiscono i risultati della ricerca per autore su un sito di commercio elettronico di libri, se si effettuano due ricerche con nomi differenti ed il primo è molto famoso mentre il secondo è un esordiente, la dimensione delle sorgenti HTML restituite sarà molto differente. Nonostante questo, le pagine dell'esempio molto probabilmente apparterranno entrambe alla stessa classe.

Sono state identificate due famiglie di caratteristiche che possono dare informazioni sulla similitudine utile nel contesto nel quale RoadRunner si muove. Nella prima famiglia sono presenti caratteristiche che mirano a dare informazioni sulla struttura interna delle pagine: chiaramente queste proprietà sono legate alla struttura della grammatica comune a cui queste pagine dovrebbero soddisfare. Nella corrente implementazione sono state prese in considerazione le seguenti caratteristiche:

- Tag Probability: è ragionevole assumere che pagine conformi alla stessa grammatica abbiano la stessa distribuzione dei tag. Sono quindi state considerate tag probabilities come possibili protagoniste per il problema del raggruppamento della pagina Web: in pratica è stata calcolata, per ogni esempio, la percentuale di occorrenze di ogni tag rispetto al numero totale di tags, ed i risultati ottenuti sono stati utilizzati come coordinate all'interno dello spazio caratteristico.

- Tag Periodicity: ci sono casi in cui le tag probabilities possono essere ingannevoli, in quanto non forniscono informazioni sulle relative posizioni dei tags. Questo significa che due pagine contenenti approssimativamente gli stessi tags verranno considerate simili anche se in una delle due i tags sono stati completamente risistemati rispetto all'altra e inoltre le pagine sono differenti in termini di grammatica. Per accoppiare la tag probability con qualche altra

caratteristica che restituisca informazioni sulla posizione dei tags, alle pagine HTML verrà applicata una variante del metodo dello spettro di frequenza [27].

- Distanza dalla Home Page: se i percorsi di navigazione all'interno del sito sono ben organizzati, è ragionevole assumere che pagine contenenti informazioni omogenee siano approssimativamente alla stessa distanza dalla home page nel grafo del sito.

- URL Similarity, in molti siti le URL di pagine appartenenti alla stessa classe seguono alcuni percorsi comuni, questo è dovuto al fatto che files HTML sono memorizzati in uno stesso spazio all'interno del server, oppure che le pagine sono generate dallo stesso script. Per tenere conto di questo è stata usata la classica nozione di analogia tra stringhe [28] al fine di associare una serie di caratteristiche numeriche con ogni URL.

Le quattro classi di caratteristiche solitamente garantiscono una buona approssimazione della corretta classificazione.

3.3.3 Il Modulo Expander

Questo modulo è responsabile dell'estrazione di dati da pagine individuali. Vale la pena notare che l'usuale ruolo di queste pagine all'interno di un sito è quello di collezionare percorsi di accesso. Un esempio è dato da links ad altre pagine e l'àncora di questi links è dato da un valore che viene ripetuto nella pagina di destinazione. Considerando un sito di commercio elettronico di libri, una possibile pagina individuale all'interno del sito è quella che presenta la lista di tutti i generi di libri venduti sul sito. Questa pagina offre links che guidano a pagine che presentano libri di un determinato genere. Per migliorare l'usabilità, i nomi dei vari generi compaiono sia nelle pagine individuali che nelle pagine destinazione. Sulla base di queste osservazioni, non c'è motivo per cui si

dovrebbero estrarre informazioni da pagine individuali in quanto queste hanno la stessa funzione degli indici in un database ed i data-objects contenuti sono ridondanti rispetto a quelli forniti dalle altre pagine.

Nel processo di estrazione analizzato è possibile individuare i seguenti passi:

1. scaricare tutte le pagine del sito
2. costruire su queste i relativi wrappers
3. applicare il wrapper per estrarre le informazioni di interesse.

Al fine di tenere i dati a nostra disposizione aggiornati, il processo deve essere ripetuto periodicamente, ogni volta scaricando l'intero sito. Una strategia alternativa è quella di lasciare i dati sul sito di origine e costruire un insieme di dati virtuali. Ogni volta che una query è effettuata sull'insieme di dati, il sistema deve navigare il sito ed estrarre i dati di interesse per la query, in maniera analoga a quanto avviene nel sistema MOMIS per quanto riguarda la creazione del global schema. Le pagine individuali forniscono percorsi di accesso che possono supportare efficientemente questo approccio. L'inferenza di un wrapper anche per pagine individuali può diventare un problema, in quanto non si è interessati all'estrazione di dati da queste pagine, ma semplicemente alla costruzione di un wrapper per permettere al sistema di navigare in tutto il sito.

Questo obiettivo può essere raggiunto applicando tecniche di inferenza di wrappers sviluppate in altri contesti. Il punto cruciale è che abbiamo a disposizione i dati offerti da pagine individuali attraverso l'estrazione da altre pagine del sito, applicando le tecniche di matching. In altre parole abbiamo esempi positivi per inferire la grammatica del wrapper. Questo è il compito di cui si occupano alcune tecniche conosciute in letteratura come NoDoSe [1], Stalker [20]. Così è stato pensato di integrare una di queste tecniche all'interno dell'architettura del sistema RoadRunner al fine di supportare il modulo expander.

3.3.4 Il Modulo Etichettatore

E' stato visto come il componente Aligner sia in grado di inferire un wrapper e lo schema associatogli per una determinata classe di pagine. Come è stato detto precedentemente, ai simboli non terminali che corrispondono ad attributi all'interno dello schema vengono assegnati nomi anonimi.

Il compito del Labeler è quello di assegnare un nome significativo ad ogni attributo dell'insieme di dati estratto. Ovviamente questo step potrebbe essere fatto manualmente, ma al fine di automatizzare ogni parte del processo di estrazione dei dati sono state studiate diverse tecniche per individuare il nome appropriato per ogni campo.

Una possibile soluzione al problema si affida all'adozione di tecniche per la rappresentazione della conoscenza: analizzando i dati estratti e sfuttando le conoscenze gestite da qualche ontologia di dominio è possibile dedurre alcuni significati per i campi. Vale comunque la pena osservare che importanti informazioni sui dati sono disponibili sulle pagine Web stesse. Poichè i siti Web sono costruiti con l'intento di essere navigati da umani, è pratica diffusa che i dati siano accompagnati da descrizioni testuali al fine di aiutare l'utente ad interpretare direttamente e correttamente le informazioni sottostanti. In molti casi queste descrizioni sono indispensabili per presentare correttamente i dati all'utente. Considerando a titolo di esempio la rappresentazione dei prezzi all'interno di un sito Web di e-commerce, senza l'aiuto di stringhe come *"prezzo"* oppure *"sconto"*, i dati relativi ai prezzi sarebbero incomprensibili. Descrizioni testuali sono spesso associate a dati organizzati in tabelle. In questi casi, spesso, la prima riga riporta etichette per descrivere il contenuto delle colonne (l'utilizzo di intestazioni nelle tabelle è caldamente raccomandato anche dal W3C stesso per aumentarne l'usabilità).

Sulla base di queste osservazioni sono stati messi a punto alcuni metodi per analizzare il codice HTML delle pagine di esempio che sono state processate con l'ausilio del modulo Aligner al fine di trovare stringhe che rappresentino buoni candidati per i nomi dei campi dei dati estrat-

ti. I metodi utilizzati sono basati sul concetto generalizzato di chiusura tra i tokens dei wrappers ed i simboli non terminali. In particolare i wrappers HTML sono considerati come grammatiche aventi una natura doppiamente incrociata:

- da una parte il wrapper può essere visto come una sequenza di simboli
- dall'altra, dato che il wrapper mantiene la struttura innestata di un documento HTML, introducendo speciali nodi per denotare iterazioni e opzioni, esso può essere visto come un albero D.O.M.. La Figura 3.3 mostra questo concetto. Nota: il nodo contrassegnato con il simbolo "+" rappresenta iterazione.

```
... <TABLE>
      <TR>
        <TD> Name </TD>
        <TD> Phone </TD>
      </TR>
      (<TR> <TD> $A </TD>
            <TD> $B </TD>
       </TR> )+
    </TABLE> ...
```

Figura 3.2: Frammento di HTML cui si riferisce la Figura 3.3

Quando il modulo Etichettatore è alla ricerca di un nome per un dato campo, esso esplora la rappresentazione sequenziale del wrapper, cercando una stringa di testo che sia adiacente ad un simbolo non terminale. L'"adiacenza" è definita rispetto a questo specifico contesto, ed alcune caratteristiche del formato HTML concorrono a definire queste proprietà.

Nell'analizzare simboli non terminali, che sono inclusi in un modello ripetitivo, l'etichettatore considera la rappresentazione DOM tree del

wrapper. L'idea è quella di verificare se il modello di sottoalbero è adiacente a qualche sottoalbero isomorfico. In questo caso, le foglie dell'albero scoperto possono essere selezionate come nomi per i nodi non terminali dell'albero modello. Anche in questo caso le nozioni di adiacenza e isomorfismo sono definite nel rispetto dello specifico contesto.

La Figura 3.3 dà un'idea di questa strategia. La parte sinistra dell'albero corrisponde ad un'intestazione per il modello iterativo rappresentato dalla parte destra, dato che le due parti sono isomorfiche, le foglie della prima, ossia le stringhe "name" e "phone" sono candidate ad assegnare un nome per i nodi non terminali $A e $B rispettivamente.

Questi metodi assumono che per ogni campo sia presente una descrizione all'interno della pagina (e quindi all'interno del wrapper). Ci sono comunque molte situazioni in cui i dati sono pubblicati nelle pagine Web lasciando implicito il loro significato. Ad esempio, in una pagina che presenta i dettagli di un libro venduto, potrebbe non essere necessario associare esplicitamente al titolo del libro una stringa per descrivere che quello che segue è il titolo di un libro, viene assunto che l'utente sia in grado di interpretare la giusta semantica del data-object. Ovviamente in questi casi la tecnica descritta precedentemente fallisce.

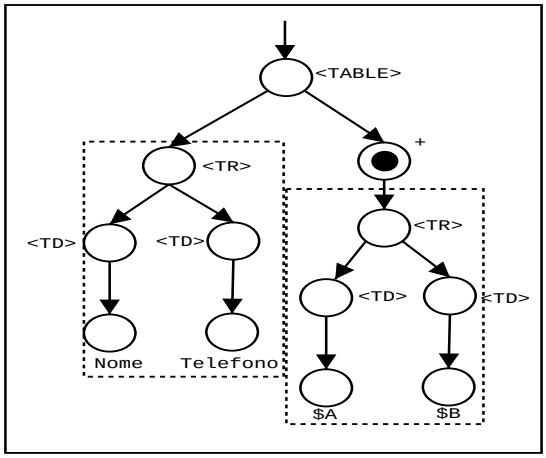

Figura 3.3: Rappresentazione DOM dei wrappers

Per affrontare questi casi, è allo studio un approccio che si affida alla ricchezza del Web stesso, il quale è inspirato dalla tecnica DIPRE di Brin [29]. Il Web può essere considerato come una grande base di conoscenze, dove una particolare parte di informazione può essere pubblicata da migliaia di siti indipendenti. Ovviamente ogni sito può adottare differenti politiche di presentazione, perciò rispetto al nostro specifico problema, è possibile che in alcune pagine, un determinato data-object sia associato ad alcune informazioni che ne descrivono il significato. Intuitivamente, per i simboli non terminali, i cui nomi non sono stati derivati dalla tecnica descritta precedentemente, il sistema può estrarre alcuni data-object di esempio e dare questi come input al motore di ricerca. È ragionevole pensare che in alcune delle pagine trovate dal motore di ricerca, il valore di input sia esplicitamente associato con parti di testo descrittivo.

Capitolo 4

Il Sistema Andes della IBM

Il sistema Andes, è la spina dorsale di alcuni sistemi di estrazione di informazioni in produzione nei laboratori IBM. Il suo obiettivo è quello di estrarre dati da un sito Web realizzato mediante l'utilizzo di HTML ed integrarli con quelli appartenenti ad un dominio dello stesso tipo, ma provenienti da altri siti. L'architettura si basa su navigatori che si occupano di recuperare le informazioni dalle pagine contenenti i dati di interesse.

La parte di interesse per quanto riguarda lo studio specifico di cui questa tesi si occupa è rappresentata dalla conversione dei dati HTML in formato XML, mentre le altre problematiche trattate da questo sistema saranno semplicemente accennate. Nel processo di trasformazione, i documenti HTML vengono per prima cosa normalizzati in documenti equivalenti in formato XHTML [31], quindi mappati nel formato XML mediante l'utilizzo di path expressions (scritte mediante l'utilizzo del linguaggio XPath [41]) ed espressioni regolari. Un metodo la cui diffusione sta rapidamente crescendo per realizzare tali espressioni è quello di usare documenti in formato XSLT (XML Stylesheet Language Transformations) [32].

XSLT è un linguaggio per la trasformazione di documenti XML che utilizza il linguaggio XPath per accedere e riferirsi a parti di un documento XML. XPath non è altro che un set di regole sintattiche per la

definizione di parti di un documento XML.

Affinchè l'estrazione dei dati sia opportunamente trattabile e affidabile è necessario assicurarsi che le espressioni che permettono l'estrazione dei dati continuino a funzionare appropriatamente anche se la struttura o la formattazione di una pagina HTML sorgente cambia. In questo capitolo verrà descritta una metodologia per la creazione di espressioni XPATH che sono in grado di estrarre dati da una qualunque pagina HTML, mantenendo la robustezza di queste espressioni. Le regole di estrazione ottenute da questo sistema verranno suddivise in categorie sulla base della loro dipendenza dal contenuto, dalla struttura o dalle caratteristiche di formattazione. Per finire verranno forniti alcuni suggerimenti pratici su come creare modelli affidabili per l'estrazione di dati dal Web.

4.1 Lavori Correlati

Alcuni gruppi di ricerca si sono concentrati sul problema dell'estrazione di dati strutturati da documenti HTML. Molte delle ricerche riguardano siti che sono costituiti da una serie di pagine denominate templates che vengono originate dinamicamente utilizzando i dati immagazzinati in un database. Il nocciolo di queste ricerche sono i wrappers che permettono di estrarre dati da questi siti mediante l'impiego di linguaggi di query del tutto analoghi a quelli utilizzati nei database tradizionali. Questi software eseguono una traduzione della query in una richiesta Web e scandiscono le pagine HTML risultanti. L'obiettivo del progetto Andes è quello di navigare un certo numero di siti di interesse, estraendo dati strutturati, affinando caratteristiche di estrazione, di ripristino dei dati persi e dei conflitti tra dati, all'interno di uno specifico dominio, rendendo in questo modo disponibili i dati alle applicazioni data-base locali. Il sistema contiene funzionalità per il controllo della qualità dei dati estratti, e la gestione di segnali di allarme al verificarsi di fallimenti, ma più importante, è in grado di sintetizzare dinamicamente gli hyper-

links al fine di recuperare dati anche dai livelli più profondi del Web [29]. La struttura software di Andes unisce capacità di navigazione con le capacità necessarie per l'estrazione di dati basata su XML al fine di formare un processo sulla carta robusto ed affidabile. Il sistema è simile ad altri sistemi di estrazione per quanto riguarda la generazione di un wrapper per siti Web.

Il meccanismo preciso su come i wrappers sono definiti, implementati e usati, è differente in ognuno dei sistemi esaminati. Di seguito verrà effettuata una comparazione dei meccanismi che stanno alla base di Andes con quelli di altre proposte in letteratura.

Il WysiWyg Web Wrapper Factory (W4F) è un toolkit per la generazione di wrappers Web [30]. Esso contiene un linguaggio per identificare e navigare siti Web (regole di recupero) ed un linguaggio dichiarativo per l'estrazione di dati da pagine Web (regole di estrazione). Questo strumento fornisce anche un meccanismo per mappare dati estratti in un modello di struttura determinato a priori. Come suggerisce il nome, W4F fornisce all'utente un'interfaccia grafica per generare le regole di recupero, estrazione e mapping. La principale differenza tra questo approccio e Andes è legato al fatto che W4F usa un linguaggio proprietario per le regole di estrazione e mapping dei dati. Andes, al contrario è basato su XHTML[31] e XSLT [32] e può sfruttare templates, path expressions (anche ricorsivi) ed espressioni regolari per estrazione dei dati, mapping e aggregazione più efficaci. Altra caratteristica peculiare di Andes è data dalla sintesi degli hyperlink che permette ai dati di essere estratti dai livelli più interni del Web.

WIDL è un metalinguaggio che implementa un'architettura all'interno di risorse web basate sul documento. È un'applicazione di XML che permette ad interazioni con servers Web di essere definite come interfacce funzionali che possono essere accedute da sistemi remoti all'interno di protocolli Web standard e fornisce la struttura necessaria per la generazione di codice client-side in linguaggi come Java, C/C++, COBOL e Visual Basic. L'obiettivo di WIDL è quello di definire un'interfaccia di

programmazione per siti Web [33, 34]. Come tale, esso si concentra più sui meccanismi che permettono di risolvere una richiesta su un sito Web, recuperare il risultato e collegare le variabili di input e output ad un linguaggio di programmazione ospite, che sul processo di estrazione dei dati dalla pagina di risultati restituita. WIDL permette ai dati di essere estratti usando path expressions assoluti, ma come verrà spiegato in una delle sottosezioni successive questo non consente di costruire estrattori di dati robusti.

Il Web Language (formalmente WebL) della Compaq è un linguaggio procedurale per la scrittura di wrappers Web [35]. Se da una parte fornisce un potente linguaggio di estrazione dei dati (simile a path expressions ricorsive combinate con espressioni regolari), dall'altra, il linguaggio non è adatto ad avere come input o come output dati in formato XML e mancano i poteri dei templates di XSLT e degli operatori di XPath.

XWRAP [36] è un generatore di wrappers semiautomatico che si basa sul significato semantico di specifici tags HTML (es: titoli e tabelle) e sul come questi tags sono utilizzati per l'impaginazione dei dati. Sono utilizzate delle euristiche per determinare le relazioni padre-figlio tra parti della pagina contenenti dati, per istanziare nomi delle tabelle, nomi dei campi e relativi valori. I wrappers risultanti dipendono dall'annidamento e dall'orientamento di tabelle e altri elementi, inoltre lavorano bene con siti Web con struttura tabellare ma non con siti che hanno struttura povera. Per esempio, alcuni siti Web che concatenano alcune parti contenenti dati in un singolo campo puramente testuale, questi richiedono espressioni regolari o tools di analisi testuale per decomporre il campo nelle sue componenti di dati originali.

Informia [37] è un sistema di mediazione delle informazioni la cui interfaccia di accesso (CAI, common access interface) è configurata con regole di recupero ed estrazione simili a quelle di W4F. Il componente per il recupero è stato progettato principalmente per gestire siti Web che contengono dati ripetitivi come la lista dei risultati di una ricerca. Informia fornisce un toolkit per la produzione automatica di regole di estra-

zione il cui target sono pagine ricche di elementi ripetitivi. Il linguaggio è proprietario e gli estrattori che sono creati manualmente per siti senza dati ripetitivi (es: alcune delle pagine finanziarie di Yahoo), forniscono un basso livello di robustezza.

4.2 Estrazione di dati strutturati da Siti Web

Questo compito richiede di risolvere cinque distinti problemi: trovare le pagine HTML di interesse all'interno di un sito seguendo gli hyperliks (problema della navigazione), estrarre parti di dati rilevanti da queste pagine (problema dell'estrazione dei dati), isolare i singoli data object ed identificare con precisione la loro struttura (problema della sintesi della struttura), garantire l'omogeneità dei dati (problema del mapping dei dati), unire dati provenienti da pagine HTML separate (problema dell'integrazione dei dati). Nelle sezioni successive verrà discusso ognuno di questi problemi.

4.2.1 Navigazione sui Siti Web

Nel sistema Andes le pagine HTML vengono suddivise in due categorie: pagine obiettivo, che contengono dati che vogliamo estrarre e pagine di navigazione, che contengono hyperlinks che puntano a pagine del primo tipo o ad altre pagine di navigazione. Un navigatore automatico (Web crawler) viene utilizzato per ritrovare pagine obiettivo da un sito Web. Il navigatore è guidato da un file di configurazione basato su regole che gli forniscono un punto di partenza, quali hyperlinks seguire (e quali no), ed il livello di profondità desiderato per la navigazione. Tutte queste istruzioni definiscono le regole di navigazione di un dato sito Web.

Il navigatore inizia il suo lavoro ritrovando la pagina o le pagine sorgenti, dal sito Web al quale appartiene l'URL che identifica le pagine di interesse. Determina quindi se la pagina è di tipo obiettivo o di navigazione. Se è del primo tipo viene mandata all'estrattore dei dati per

i processi successivi. Vengono analizzati gli hyperlinks per entrambi i tipi di pagina e viene presa una decisione in base ad una logica link-by-link. Un parametro di profondità della navigazione determina quanto il navigatore può andare lontano, in termini di link, dalla pagina di origine.

Oltre maneggiare hyperlinks statici (come i tags <A>, <FRAME>, <IMG>), l'approccio Andes segue il "deep Web" espresso in [29]. Il deep Web è navigato analizzando forms HTML e codice JavaScript per produrre hyperlinks sintetici che il navigatore può seguire. Questo verrà descritto più nel dettaglio in una delle sezioni successive.

Le regole di navigazione dei siti Web sono tipicamente scritte a mano dopo un'attenta analisi delle pagine obiettivo. Alcuni tools semiautomatici possono assistere nel processo ma la necessità di una messa a punto estremamente rigorosa per le regole di navigazione, rende l'uso di tali strumenti limitato a casi particolarmente semplici.

Il sistema Andes è indipendente dallo specifico navigatore utilizzato in quanto questi strumenti forniscono in generale servizi molto simili tra loro, sono inoltre configurati utilizzando gli stessi principi generali messi brevemente in evidenza nella descrizione fatta precedentemente. L'implementazione attuale di Andes utilizza come navigatore il sistema GCS (Grand Central Station), descritto in [38, 39], anch'esso sviluppato nel centro di ricerca IBM di Almaden. In GCS le regole di navigazione sono espresse in XML, un esempio di regole di navigazione che hanno il compito di recuperare recensioni sui computer di tipo laptop IBM presenti nel sito epinions.com viene riportato in Figura 4.1.

4.2.2 Processo di Estrazione dei Dati

Le pagine HTML obiettivo sono soggette ad una sequenza di step per l'estrazione di dati. Come accennato precedentemente, poichè la maggior parte dei contenuti HTML sul Web sono mal formati, cioè non soddisfano appieno le specifiche HTML, il primo step nel processo di estrazione

```
<gcs-config>
  <group name="Reviews">
    <url-pattern-list>
      <url-pattern recursion-depth="2">
        <seed-list>
          <li>http://www.epinions.com/cmhd_Notebooks-IBM</li>
        </seed-list>
        <include-pattern-list>
          <url-obj-pattern host="www.epinions.com"
           file="/cmhd_Notebooks-IBM*"/>
          <url-obj-pattern host="www.epinions.com"
           file="/cmd-review*"/>
        </include-pattern-list>
      </url-pattern>
    </url-pattern-list>
  </group>
</gcs-config>
```

Figura 4.1: Esempio di file di configurazione di GCS

dei dati è quello di traslare il contenuto in una sintassi XML well-formed, questo ci aiuterà nei passi successivi. L'approccio adottato da Andes è quello di far passare la pagina HTML originale attraverso un filtro che sistema gli errori di sintassi producendo codice HTML well-formed, conosciuto anche come Extensible HTML (XHTML) [31]. Toolkits per questo step esistono già, e anche il pacchetto tidy precedentemente nominato può essere utilizzato [40].

Essendo XHTML basato su XML, ogni tools XML può essere utilizzato per i successivi processi sulle pagine obiettivo in HTML. Dato che lo scopo di Andes è quello di produrre XML come output, il compito di convertire XHTML in XML viene considerato come un problema di trasformazione di XML.

Il meccanismo di trasformazione scelto per Andes è Extensible Stylesheet Language Transformations (XSLT) [32], un linguaggio che fornisce potenti path expressions di XML (XPath) [41] combinate con espressioni regolari attraverso il meccanismo di estensione XSLT.

Come mostrato in figura 4.3, l'URL di un documento XHTML è utilizzato per determinare quale set di files XSLT applicare al documento stesso. Il documento XHTML è passato attraverso il primo file XSLT e l'output viene passato ad altri file XSLT definiti per quell'URL. L'output finale è un file XML la cui struttura e contenuto sono determinate dall'ultimo file XSLT. Questo sarà tipicamente un'applicazione XML. L'approccio "in pipeline", si sposa bene con gli scopi delle applicazioni Andes specifiche di un dominio; il primo file XSLT estrae i dati da una pagina XHTML, il successivo file XSLT nella catena, può rifinire i dati e riempire le informazioni perse con le conoscenze sul dominio (maggiori dettagli sulla struttura di sintesi sono disponibili nella sottosezione 4.2.8).

Uno dei problemi principali dei progetti di estrazione dei dati HTML è legato alla scarsa robustezza in relazione ai cambiamenti di struttura e contenuto dei siti, la strada scelta da Andes per produrre wrappers robusti è quella di affidarsi meno sulla struttura dell'HTML e più sul contenuto.

Alcuni linguaggi per la realizzazione di wrappers (come il linguaggio di estrazione dell'HTML in W4F) richiedono l'uso di percorsi HTML assoluti, che puntino alle sezioni di dati che devono essere estratte. Un percorso assoluto descrive la navigazione all'interno di un albero HTML, iniziando dal primo tag dell'albero (<HTML>) e procedendo verso i nodi figli che contengono i dati che devono essere estratti. Il percorso è assoluto in quanto esso elenca i nomi dei tags che si prevede verranno incontrati nell'albero e le loro posizioni assolute. Per esempio: un percorso assoluto per l'elemento posizionato nella prima riga e seconda colonna della terza tabella contenuta in un documento HTML può essere espresso attraverso l'utilizzo di XPath come: /HTML/BODY/TABLE[3]/TR[1]/TD[2].

L'approccio che considera il percorso assoluto è facile che fallisca quando la pagina HTML obiettivo cambia. Il cambiamento più comune all'interno di una pagina HTML è quello relativo al posizionamento delle sezioni contenenti le informazioni di interesse. L'impaginazione viene solitamente effettuata usando tags come <TABLE>, <TR>, <TD>,

come visto nell'esempio precedente. Quando nuovi contenuti (come ad es. annunci) vengono aggiunti alla pagina o quando viene cambiata la posizione del contenuto esistente, la posizione assoluta dei tags cambia. Per questa ragione è necessario stabilire la posizione delle sezioni contenenti i dati indipendentemente dal loro percorso assoluto.

L'approccio Andes implica l'individuazione di àncore all'interno della pagina che vengono utilizzate come punti di partenza per l'estrazione di dati. Idealmente le àncore sono stabilite sulla base del contenuto delle sezioni di dati, non sul loro percorso HTML.

Per esempio, una pagina che contiene il prezzo di un libro, probabilmente conterrà la parola "Prezzo" da qualche parte vicino al valore che indica il prezzo. Cercando la parola "Prezzo", è possibile stabilire un'àncora per il valore del prezzo ed essere così indipendenti dalla sua locazione assoluta.

Un esempio di codice XSLT che estrae l'ultima quotazione di un'azione da una pagina finanziaria di Yahoo è mostrata in figura 4.2.

```
<xsl:template match="td[contains(.,'Last Trade')]">
  <PRICE><xsl:value-of select="b"/></PRICE>
</xsl:template>
```

Figura 4.2: Esempio di Regola di Estrazione in XSLT

Da notare che ciò che viene cercato è la cella di una tabella contenente la parola "Last Trade" e ciò che viene estratto è il valore contenuto nel tag b (bold). Il processore di XSL inizia dalla radice dell'albero XHTML e ricorsivamente cerca la cella della tabella con la quale effettuare il matching. Una volta che la cella della tabella è stata trovata, le istruzioni contenute nel template vengono eseguite, in questo caso la produzione di un elemento prezzo nel documento XML di output.

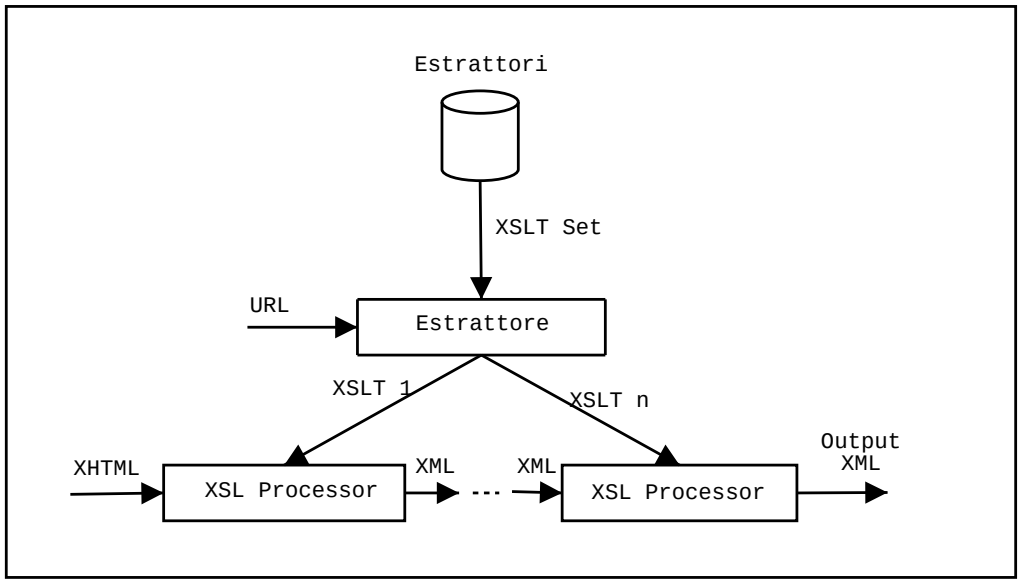

Figura 4.3: L'Estrattore identifica i files XSLT da usare. La sequenza di processori XSLT estrae e rifinisce i dati, fornendo un file XML come ultimo output

4.2.3 Sintesi degli Hyperlinks

Un difetto dei navigatori attuali è dato dal fatto che essi possono seguire solo hyperlinks statici (come quelli contenuti nei tags <A>, <FRAME>, <IMG>) ma non hyperlinks dinamici che sono il risultato di forms HTML e di codice JavaScript. Hyperlinks dinamici sono tipicamente eseguiti in seguito all'input (tipicamente un click sull'ipertesto corrispondente) dell'utente, ma possono anche venire invocati automaticamente in base all'esecuzione di codice JavaScript.

Una grande frazione di contenuti Web viene nascosta in questa maniera. La nozione di "deep Web" [29] è stata utilizzata per descrivere i dati nascosti sul Web. Andes fornisce accesso al deep Web analizzando forms HTML e codice JavaScript ed estendendo i dati a disposizione dei navigatori attraverso "hyperlinks sintetici" o copie statiche di hyperlinks dinamici.

Tutto ciò è realizzato passando ogni pagina Web attraverso uno o più filtri XSLT che analizzano forms HTML e codice JavaScript producendo

da questi una lista di hyperlinks statici che rappresentano una copia delle selezioni effettuate da un immaginario utente. I links sono quindi inseriti all'interno della pagina come semplici tags <A> e la pagina è passata al navigatore. Il vantaggio di questo approccio è che il navigatore stesso non è modificato in alcun modo. In Figura 4.4 è mostrato il processo di arricchimento dell'HTML usato per la sintesi degli hyperlinks.

Un caso particolare si presenta nella gestione dei forms che richiedono l'uso del metodo POST dato che i navigatori non possono utilizzare links diversi da quelli che utilizzano il metodo GET. Il problema viene risolto da un proxy che effettua la conversione del metodo GET-to-POST. Il proxy riceve richieste GET dal navigatore e le converte in hyperlinks marcati in una speciale maniera e che utilizzano il metodo POST. Il sito Web di interesse viene acceduto utilizzando il corretto metodo e la pagina risultante viene restituita al navigatore.

4.2.4 Costruzione di Espressioni di Estrazione Robuste

Verranno ora prese in considerazione le espressioni che consentono l'estrazione, ed i fattori che rendono robusta una particolare espressione. Con il termine robustezza verrà considerata la capacità dell'espressione di continuare ad estrarre dati da una pagina HTML anche quando la struttura o il disegno di una pagina cambiano. La principale fonte di criticità per le soluzioni proposte fino a questo momento è proprio la mancanza di robustezza. Se da una parte l'isolamento totale da questi cambiamenti è difficile da raggiungere, dall'altra è convinzione comune che sia comunque possibile creare regole di estrazione robuste. Questo obiettivo, nel sistema in esame, viene perseguito affidandosi meno alla struttura e più sul contenuto.

Vengono definite due metriche, distinte in base al loro approccio, che possono essere utilizzate per effettuare una valutazione formale della robustezza di un'espressione. La prima è una misura empirica delle prestazioni di un'espressione. La misura più semplice è quella di registrare

la frequenza di fallimento e successiva correzione di un'espressione applicata ad una specifica sorgente Web, all'interno di un lungo periodo di tempo (dai 6 ai 12 mesi). Questa metrica fornisce un controllo con un buon livello di veridicità sull'utilità di un'espressione. Essa può fallire in relazione alla frequenza ed all'intensità dei cambiamenti di un particolare sito Web. Un'espressione particolarmente povera (ad es: dipendente dalla struttura) usata su un sito che non cambia, potrebbe ottenere un tasso migliore di un'espressione più flessibile usata su un sito la cui struttura cambia significativamente e spesso.

Il secondo approccio è quello di esaminare la robustezza fuori dal contesto del suo specifico uso, considerando invece la tendenza ad aggiornamenti sulla struttura della sorgente di dati. Questa metrica esamina il numero delle dipendenze che un'espressione o una catena di espressioni hanno all'interno del documento, la profondità dell'espressione ed il metodo di "ancorare " l'espressione. La prima metrica verrà chiamata robustezza *a posteriori,* mentre la seconda robustezza *a priori*.

Alcuni linguaggi per la realizzazione di wrappers (es: HTML Extraction Language in W4F) richiedono l'uso di percorsi HTML assoluti che puntino alle sezioni di dati che devono essere estratti. Un percorso assoluto descrive la navigazione all'interno di un albero HTML, iniziando dalla radice, il tag <HTML>, e procedendo verso i nodi figli che contengono i dati da estrarre. Il percorso è reso assoluto dal fatto che esso elenca nomi dei tags che ci si aspetta di vedere all'interno dell'albero e le loro posizioni assolute. Un esempio di questa tecnica è stato evidenziato nella sottosezione 4.2.2.

L'approccio del percorso assoluto si dimostra con buona probabilità fallimentare quando il disegno o la struttura di una pagina HTML cambia. In questi casi si dice che la regola di estrazione ha una bassa robustezza *a priori.* Al contrario, un approccio basato sul contenuto o sugli attributi, focalizzandosi sul contenuto attuale che deve essere estratto o su altri contenuti vicini a quest'ultimo, rende i risultati più robusti.

In generale, il metodo basato sul contenuto cerca un'occorrenza di

una parola o di una frase che è noto rimanere costante anche quando il disegno della pagina cambia.

Si consideri l'occorrenza della frase "Last Trade" su una pagina Web contenente informazioni su titoli azionari. Dalla posizione in cui questa frase si presenta all'interno dell'albero HTML, è un compito relativamente facile trovare l'attuale valore del titolo.

A volte però tale testo statico non è dove dovrebbe trovarsi e perciò un'indicazione sulla sua posizione può venire dai valori degli attributi nella pagina. Non è difficile infatti vedere importanti campi di dati messi in evidenza attraverso certe dimensioni del font o altri attributi del testo.

Altre volte il contenuto è mostrato utilizzando precisi parametri di progetto contenuti nell'attributo @class. Si consideri a questo proposito una pagina di articoli di cronaca, in cui il titolo è contenuto in un tag HTML con "title" come valore di @class. L'estrazione dei dati è semplificata e resa più robusta dal fatto che il nome dell'elemento HTML contenente il titolo non sia importante.

Verrà ora introdotto il concetto di robustezza "forward looking" dei modelli di estrazione dei dati. Con questa metodologia, le espressioni sono realizzate tenendo conto in anticipo dei probabili cambiamenti a cui può essere soggetta la pagina HTML nei mesi o negli anni successivi. Questa tecnica richiede la comprensione degli obiettivi della pagina e le sue implicazioni a livello di progetto e di impaginazione. Ci si chiede che cosa il progettista abbia voluto esprimere nella pagina e conseguentemente quali paradigmi di progetto dell'HTML abbia a disposizione per portare a termine il compito. Capire il pensiero che sta dietro il progetto della pagina ed i tools usati per metterlo in pratica rende le espressioni per l'estrazione dei dati immensamente più robuste (robustezza a priori). Per portare un esempio si può pensare al menù a tendina precedentemente menzionato, questo può essere definito in un solo modo, un fatto che è probabile non cambierà in futuro. In base a queste considerazioni la generazione automatica di modelli di estrazione dei dati, basati unicamente sull'apprendimento da esempi e completamente automatico, non

essendo in grado di fornire questo tipo di sensibilità, rischia di non poter essere applicato con successo.

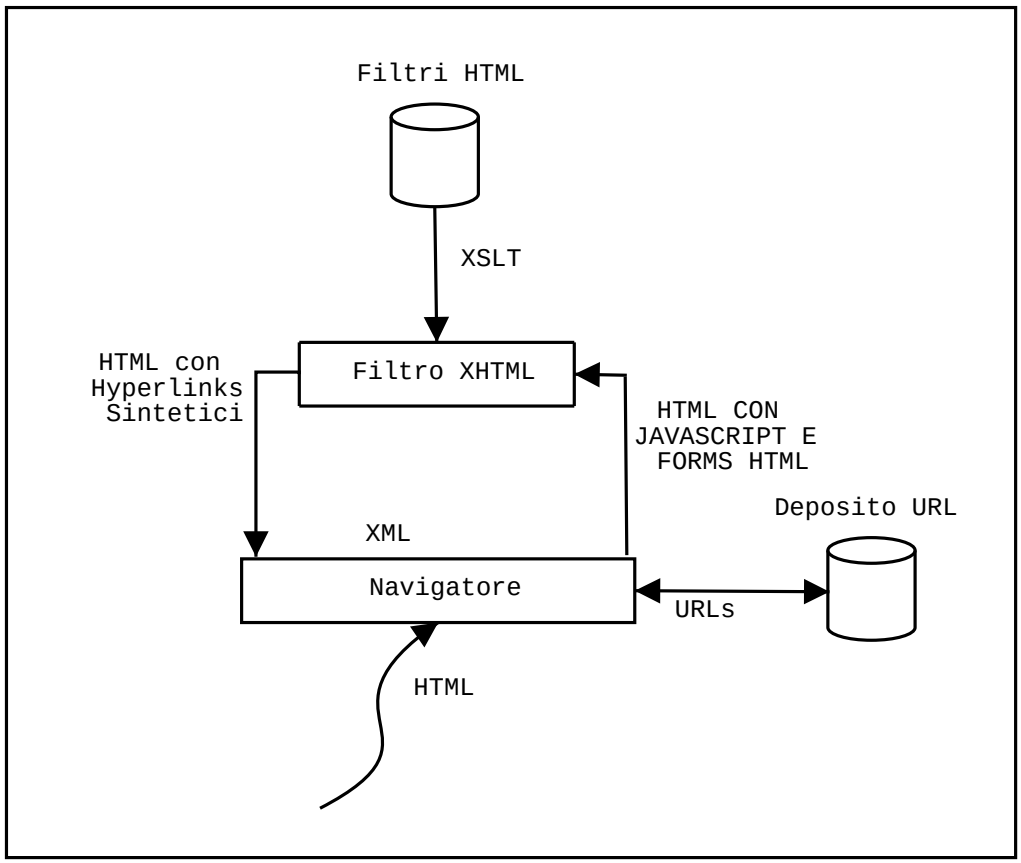

Figura 4.4: Sintesi degli hyperlinks

4.2.5 Àncore e Salti

Il processo di estrazione dei dati viene formalizzato creando un grafo che attraversa il documento sorgente ed è composto da due parti: àncore e salti. Le àncore sono nodi selezionati dall'albero del documento sorgente e rappresentano nodi non foglia del grafo. Un salto è relativo alla sezione di albero che va da un'àncora ad un'altra, o eventualmente, da un'àncora ai dati cercati. Come visto nella discussione sui motivi

che hanno portato i test effettuati su XWRAP Elite ad evidenziare risultati non eclatanti, il contenuto di interesse si trova solitamente innestato in profondità all'interno dell'albero HTML (verso la parte più bassa dell'albero stesso) e quindi il compito principale di un'àncora è quello di individuare la posizione precisa di questo contenuto, in modo che i relativi salti possano essere eseguiti partendo da un punto di riferimento assoluto, e quindi nella maniera più sicura possibile. La costruzione del grafo che attraversa la pagina è mostrata in Figura 4.5.

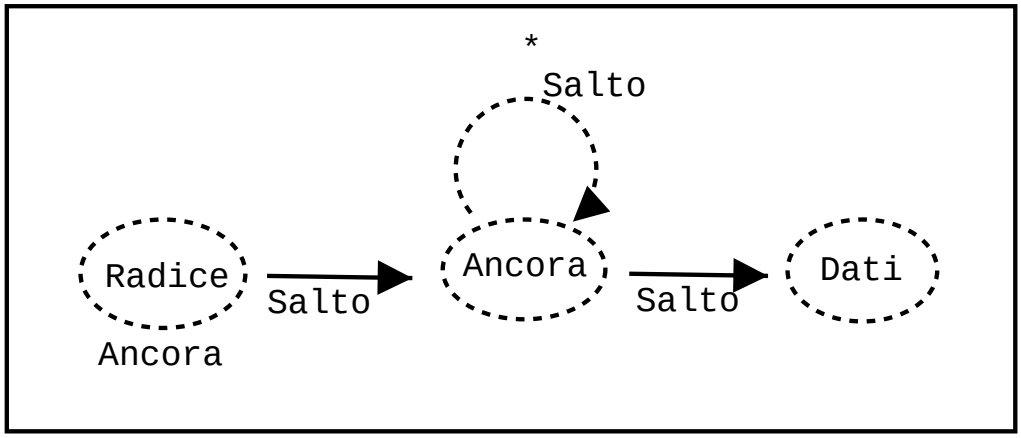

Figura 4.5: Grafo trasversale costruito seguendo àncore e percorsi dei salti verso i dati desiderati

Per mostrare il funzionamento di àncore e salti verranno usate come esempio le pagine finanziarie di Yahoo che si occupano dei valori dei titoli azionari. Consideriamo i prezzi delle azioni IBM alla pagina `http://finance.yahoo.com/q?s=IBM&d=t` (vedi FIGURA 4.6).

Scartando i contenuti relativi alla navigazione, si nota che il contenuto di interesse è all'interno di una tabella posizionata circa al centro della pagina. Ispezionando il codice XHTML per la pagina (mostrato in Figura 4.7), è possibile osservare due cose:

- data la natura XML del codice XHTML è molto facile attraverso browser come Mozilla 6.0 che sono in grado di comprendere l'XML, navigare e ritrovare le sezioni di interesse all'interno del

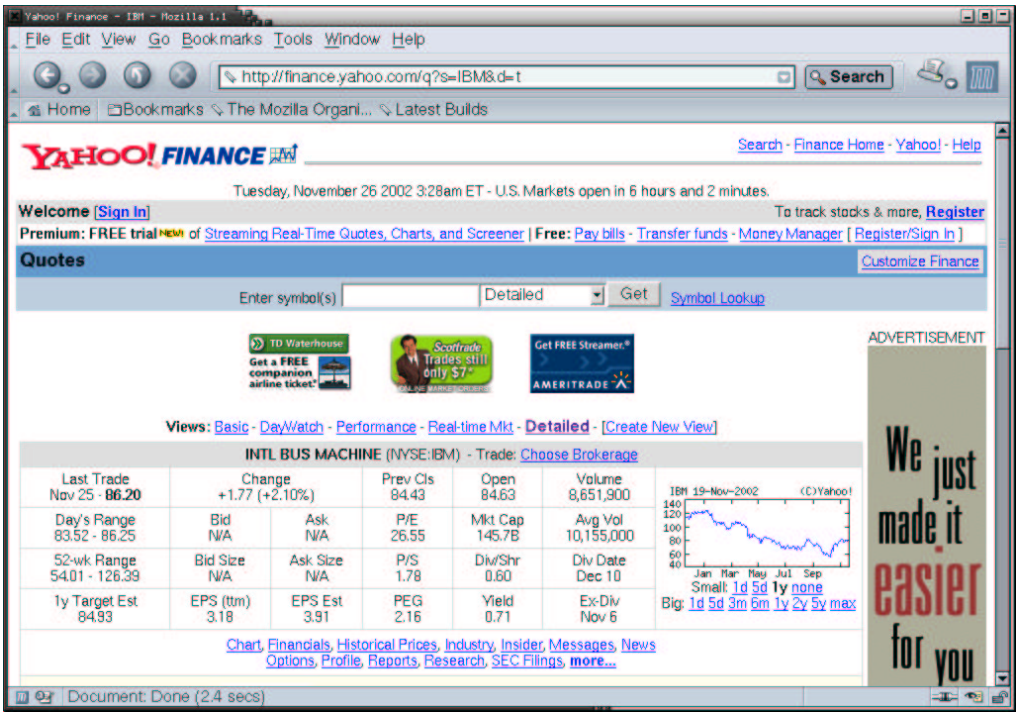

Figura 4.6: Pagina Finanziaria di Yahoo per un titolo azionario IBM

codice XHTML delle pagine. Questo browser infatti consente di espandere o ridurre secondo necessità parte dell'albero (a questo scopo servono i segni "+" e "-" vicini ad ogni elemento e attribuiti in base al fatto che tali elementi contengano o meno figli).

- L'ultimo valore del prezzo dell'azione (86.20) è posizionato vicino alla frase "Last Trade" ed un'ulteriore ispezione del codice mostra che altri valori di dati risiedono all'interno della stessa tabella come ultimo prezzo dell'azione. Si è quindi propensi a definire una espressione di salto del tipo "ricerca globale" nell'albero che cerca le occorrenze della frase "Last Trade". Dall'àncora definita da questo salto, possiamo saltare all'attuale valore dell'azione usando una espressione relativa alla struttura che ci porta fino all'elemento bold <b> in basso a destra.

Si supponga ora di voler estrarre il nome dell'azienda mostrato in questa pagina. L'àncora definita precedentemente serve come punto di partenza anche per questo salto. Dato che il nome della compagnia si trova nella riga di una tabella (elemento <tr>) immediatamente precedente la riga dell'àncora, il salto potrebbe essere definito come segue: trovare la riga della tabella precedente e prendere il primo valore di tipo bold (<b>).

Il grafo per questa estrazione è mostrato in Figura 4.9.

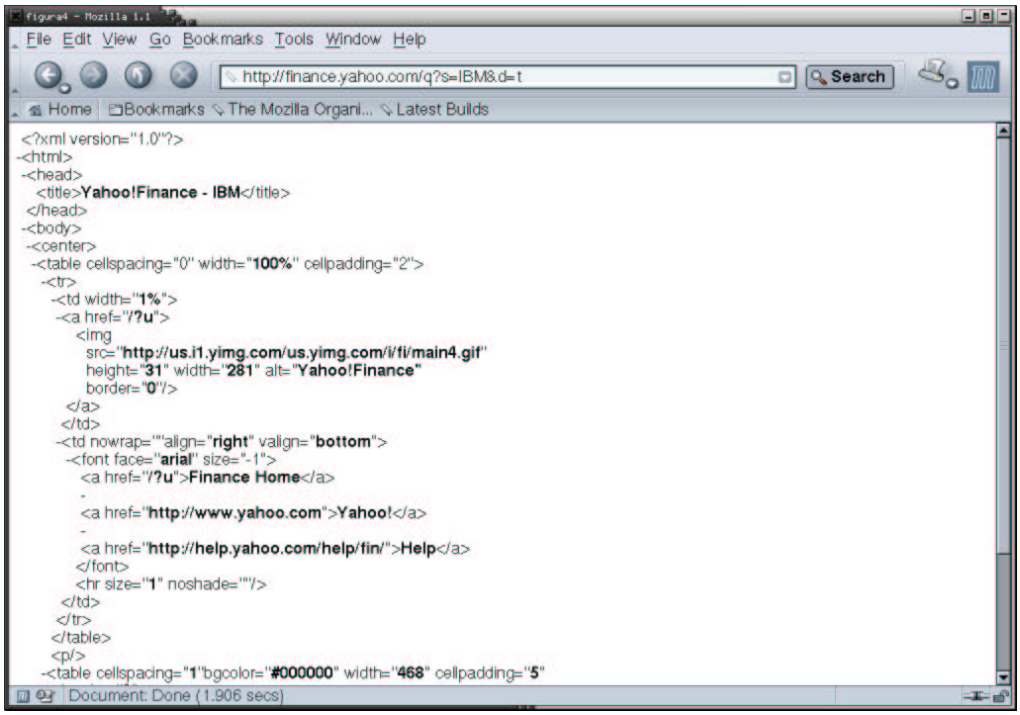

Figura 4.7: Frammento dell'albero XHTML relativo al top del documento di figura 4.6

E' importante notare che mentre i nodi del grafo devono essere presi direttamente dai nodi dell'albero HTML della sorgente, non sono però costretti a rispettare l'ordine gerarchico con cui viene rappresentata la sorgente stessa. Dalla prima àncora definita in questo esempio, un salto attraversa l'albero verso il basso, mentre l'altro risale l'albero per arrivare in una parte differente.

L'eliminazione della dipendenza del salto sulla gerarchia del docu-

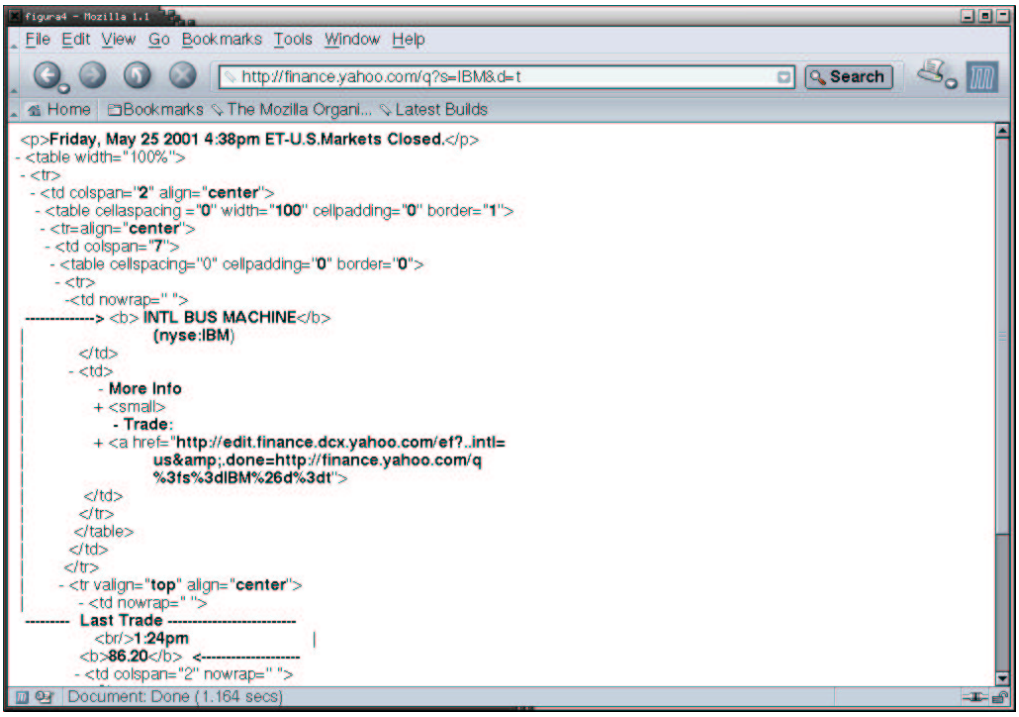

Figura 4.8: Frammento dell'albero XHTML relativo al contenuto di interesse

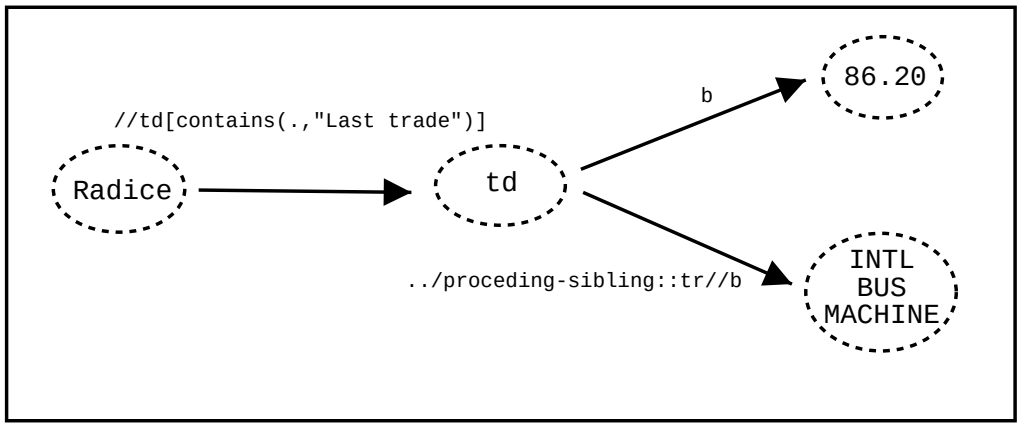

Figura 4.9: Grafo della pagina finanziaria di Yahoo

mento dalla sua àncora sorgente fa sì che si abbia un incremento della capacità del salto, aumentandone quindi la potenziale portata, differenziando questo sistema di estrazione dai suoi pari.

Da notare che sia l'ultimo prezzo dell'azione che il nome dell'azienda, sono stati evidenziati utilizzando il font "bold", questa non è un'anomalia e neanche una coincidenza, nella pratica è abbastanza frequente (anche alcune delle euristiche utilizzate da XWRAP Elite facevano uso di queste relazioni) , queste associazioni contribuiscono ad aumentare la robustezza dell'estrazione. Essenzialmente ciò che si utilizza per catturare l'attenzione dell'utente (evidenziando ad esempio certe parti attraverso un font differente o più grande), rende anche più facile per un estrattore di dati portare a termine il suo compito.

Lo step successivo è quello di inserire l'àncora ed i salti all'interno di un file XSLT che può estrarre i dati da una pagina XHTML. Si assuma che l'output XML desiderato derivi da qualche frammento di linguaggio di mark-up per i prezzi delle azioni in cui questi ultimi sono contenuti in un elemento "PREZZO" ed il nome in uno "NOME". Il codice XSLT mostrato in figura 4.10 fa il lavoro e può essere combinato con altro codice XSLT per una più elaborata estrazione dei dati.

```
<xsl:template match="td[contains(.,'Last Trade')]">
  <PREZZO>
    <xsl:value-of select="b"/>
  </PREZZO>
  <NOME>
    <xsl:value-of select="../preceding-sibling::tr//b"/>
  </NAME>
</xsl:template>
```

Figura 4.10: Esempio di regola di estrazione XSLT per Yahoo!Finance

4.2.6 Struttura di Àncore e Salti

In questa sezione ci si concentrerà sull'espressività di àncore e salti e si introdurrà una loro classificazione che aiuterà a confrontare i relativi poteri e robustezze tra due espressioni. Nella sezione successiva la classificazione verrà utilizzata per uno studio empirico sugli attuali siti

Web e sugli estrattori di dati. Come discusso precedentemente, un salto è utilizzato per trovare un'àncora in un albero XHTML navigando l'albero alla ricerca di un elemento che rispetti un determinato predicato. L'attraversamento può iniziare dalla radice dell'albero, nel qual caso l'espressione del salto verrà chiamata *globale*. Altrimenti l'espressione del salto inizia da un'altra àncora ed è etichettata come *relativa*. Questi due tipi di espressione definiscono la loro *località*.

La richezza dell'espressione definisce quanto profonda deve essere la ricerca all'interno del sottoalbero. Un'espressione può essere una *ricerca* ricorsiva (es: trova un elemento dell'albero innestato arbitrariamente in profondità) oppure, un attraversamento lungo uno specificato *percorso.* La ricerca, le alternative nei percorsi e le loro combinazioni a vari livelli dell'albero, definiscono l'*elasticità* dell'espressione.

Località ed *elasticità* insieme definiscono la *capacità* dell'espressione. Per finire, ad ogni livello dell'albero, la decisione di mettere un sottoalbero o un nodo foglia all'interno di un'altro implica la valutazione di un predicato sulla *struttura,* sul *valore di un attributo,* o sul *contenuto* di un sottoalbero o di un nodo foglia. Un predicato sulla struttura confronta nomi di elementi o posizioni, mentre un predicato sul valore di un attributo usa il valore dell'attributo stesso per prendere una decisione. Un predicato basato sul contenuto è il più potente dei tre e confronta il contenuto dei nodi di testo. Le tre alternative definiscono i tipi di modello della espressione di salto. Se l'espressione di salto combina differenti modelli in una singola espressione, la tendenza è quella di etichettare l'espressione in accordo con il modello più potente all'interno dell'espressione stessa. Come esempio, si considerino le espressioni di salto mostrate in Tabella 4.1. La prima, mostrata nella sezione precedente, verrà etichettata come "ricerca globale del contenuto" poichè essa implica una ricerca ricorsiva sull'intero albero per un'occorrenza di un'istanza di alcune parti puramente testuale. L'espressione 2, d'altra parte , cerca ricorsivamente un elemento font che abbia l'attributo @class con il valore "title" ed è classificata come espressione di " ricerca globale dell'attribu-

to". La terza espressione, ricerca all'interno dell'intero albero XHTML un'àncora che rappresenta una riga all'interno di una tabella, con più di 5 colonne. La sua classe è "ricerca globale della struttura". L'espressione 4 inizia dall'àncora definita dalla 3. Essa cerca un elemento bold (<b>) che inizi con un contenuto testuale "DATE:" e sia l'albero figlio o fratello della riga della tabella. Quest'ultima espressione viene classificata come espressione di "contenuto relativo" poichè inizia da un'àncora esistente e dipende più pesantemente dalle occorrenze della stringa di testo "DATE:" che dalla presenza di un elemento bold.

	Espressione di Salto	Classificazione
espressione 1	//td[contains(.,'Last Trade')]	Ricerca globale del Contenuto
espressione 2	//font[@class='title']	Ricerca globale dell'attributo
espressione 3	//tr[count(td) > 5]	Ricerca globale della Struttura
espressione 4	EXP3/following::b[starts-with(.,'DATE:')]	Contenuto Relativo

Tabella 4.1: Esempi di espressioni di salto e loro classificazione

4.2.7 Risultati Empirici

In questa sezione verrà eseguita una breve recensione di uno studio empirico sui modelli di estrazione dei dati realizzati e utilizzati dagli sviluppatori di Andes, mediante la tecnica illustrata nei paragrafi precedenti, negli ultimi due anni. Sono stati definiti manualmente i files XSLT per alcune sorgenti di dati provenienti dal Web ed altrettanto si è fatto per quanto riguarda la classificazione delle espressioni di salto. In totale i modelli di estrazione dei dati (espressioni di XPath) sono stati circa un centinaio. La stragrande maggioranza è stata definita come àncore intermedie mentre i rimanenti consistevano in referenze dirette ad elementi ben definiti quali ad esempio il titolo di un documento XHTML

(/html/title). Nella Tabella 4.2 è mostrata la distribuzione delle espressioni di salto incontrate. Si può notare che circa la metà delle espressioni di salto individuate riutilizzano un'àncora esistente. Questo è spiegabile dal fatto che in molti dei documenti XHTML è sufficiente trovare una buona àncora, da questa è possibile attraverso piccoli salti raggiungere alcuni campi di dati di interesse. A questo proposito vale la pena ricordare che, nella sezione 4.2.5 è stata usata la stessa àncora sia per trovare l'ultimo prezzo dell'azione che il nome della compagnia.

	Ricerca Globale	Ricerca Relativa	Percorso Globale	Percorso Relativo	Àncora Esistente
Percentuale	36%	9%	0%	4%	51%

Tabella 4.2: Distribuzione delle espressioni di salto

Ora verrà esaminata la distribuzione di tipi di modelli di espressioni di salto usati. La Tabella 4.3 mette in evidenza che circa la metà delle espressioni erano basate sul contenuto o sui valori dell'attributo, mentre le espressioni puramente strutturali contavano per meno del 5% del totale. La categoria "nulla" corrisponde alla "Àncora Esistente" in Tabella 4.2, queste sono àncore che vengono semplicemente riusate. Combinando i risultati si vede che l'espressione più comune è la Ricerca Globale per valori dell'attributo o per contenuto.

Questi tipi di espressioni sono molto potenti e l'equipe dell'IBM è convinta che sia la strada giusta per l'estrazione dei dati dal Web.

	Struttura	Attributo	Contenuto	Nulla
Percentuale	4%	20%	24%	51%

Tabella 4.3: Distribuzione dei tipi di modello di espressioni di salto

4.2.8 Sintesi della Struttura

Estrarre sezioni di dati come quella del valore di un'azione mostrata in figura 4.6, è facile grazie alla sua struttura semplice e alla rappresentazione all'interno della pagina HTML che mappa direttamente nella corrispondente struttura XML (un record di database di tipo piatto).

La sintesi della struttura può essere richiesta in situazioni ben più complesse. Consideriamo il compito di aggregare cataloghi di prodotti provenienti da alcuni siti Web. In questo caso è necessario rappresentare i cataloghi ed i prodotti al loro interno nel maggior dettaglio possibile in modo che l'integrazione dia i migliori risultati. Ciò che rende questo compito così difficile è che la struttura fornita da un sito può non essere sufficiente per poter eseguire direttamente il mapping con una possibile struttura XML. Ad esempio, in molti siti alle caratteristiche dei prodotti sono dedicati paragrafi di puro testo e alcuni dati possono venire omessi perchè recepiti implicitamente dall'utente leggendo il resto o guardando la pagina (es: un computer laptop che ha uno schermo LCD piuttosto che CRT).

Nel sistema Andes, il compito di fornire espressioni regolari che estraggano dati strutturati da frammenti di testo non strutturati è affidato ad un programmatore con conoscenze dello specifico dominio delle pagine obiettivo. Ad esempio, una stringa del tipo "Lunedì 30 Ottobre" può essere facilmente convertita in una struttura attraverso l'uso di espressioni regolari. Questo tipo di analisi è potenzialmente meno curata di un'analisi linguistica approfondita ma è comunque potente e veloce.

I dati persi possono essere completati utilizzando codice XSLT che incapsula conoscenze sul dominio. Per esempio, un file XSLT che è specifico di prodotti di computers, può esaminare la descrizione di un computer e determinare se si riferisce ad uno dei prodotti laptop IBM. In questo caso, una nuova caratteristica del prodotto può essere aggiunta alla struttura dell'output XML.

4.3 Architettura di Andes

Tutte le caratteristiche descritte nella sezione precedente sono state implementate all'interno della struttura di Andes.

Andes è stato scritto in codice Java e consiste in cinque componenti: data retriver, extractor, checker, exporter e scheduler/manager interface.

Come citato precedentemente, il data retriever di default è il navigatore GCS (Grand Central Station). I navigatori sono schedulati per essere lanciati ad intervalli di tempo che dipendono dai siti in esame. Quando il tempo programmato scatta, GCS viene invocato e le pagine HTML obiettivo vengono recuperate dai siti Web. Le pagine vengono consegnate al modulo extractor che si occupa di estrarre i dati, di effettuare le operazioni di sintesi della struttura, del mapping e delle funzioni di integrazione.

Il documento XML prodotto in output viene mandato al data checker e per ultimo all'exporter. Un'interfaccia permette all'amministratore di controllare tutto quanto avviene nel corso delle varie fasi in cui si articola il lavoro di Andes. All'interno del sistema sono presenti funzionalità di monitoraggio sul corretto funzionamento e di allarme in caso di problemi. Dato che i siti Web possono cambiare in qualunque momento, Andes monitorizza la situazione attraverso le informazioni ricevute dai navigatori.

In seguito al cambiamento di un sito Web, anche un file XSLT può fallire nell'intento di estrarre dati dalle pagine di tale sito. Andes monitorizza continuamente la qualità dei dati estratti e avverte l'amministratore quando sono richiesti aggiustamenti ai files XSLT. La validazione dei dati è effettuata sull'output XML o sui filtri XSLT e non sul documento HTML di origine, questo significa che se il sito Web cambia ma il filtro XSLT continua ad estrarre correttamente i dati dalle pagine cambiate, le nuove pagine passano la validazione e non viene generato nessun tipo di allarme.

La validazione dei dati viene effettuata su alcuni step. Per prima cosa

vengono eseguiti controlli sintattici, questi verificano che ogni elemento XML sia presente in output e che il suo valore sia effettivamente del tipo che ci si aspetta. Questo controllo è seguito da una fase di verifica della semantica nella quale vengono individuati valori non corretti. Questo sistema, se da una parte è limitato dal fatto di essere efficace solo su uno specifico dominio, dall'altra è molto potente. Per esempio, se è noto a priori che il prezzo di un'azione deve essere minore di 1000$, questa informazione può essere descritta al data validator, il quale si occuperà di separare i dati buoni da quelli errati. Questi ultimi vengono spostati in una zona di memoria in cui l'amministratore si occuperà di decidere cosa farne. Se l'amministratore accetta i dati così come sono, le condizioni al contorno vengono automaticamente modificate. In alternativa, i dati possono essere ignorati come un errore che è capitato una sola volta o essere corretti manualmente.

4.4 Considerazioni sul sistema Andes

In questo capitolo è stato discusso il problema dell'estrazione di dati da siti Web ed è stato proposto un approccio basato su àncore e salti per risolverlo. Il compito dell'estrazione di dati è stato considerato come un processo con diversi passi in cui l'obiettivo andava ben oltre il semplice "screen-scraping". Lo scopo finale era infatti quello di essere in grado di estrarre dati semistrutturati da siti Web e trasformarli in una rappresentazione ben strutturata e ricca di caratteristiche. La gestione della eterogeneità dei dati recuperati è parte integrante di questo processo.

Come è stato visto nella sezione 2.7 sebbene la sintassi dell'HTML si è stabilizzata negli ultimi anni, il modo in cui i siti Web sono realizzati continua ad evolversi. Potenti tolls per la progettazione rendono più semplice di quanto non lo fosse in passato la creazione di siti complessi, introducendo così un ulteriore livello di difficoltà nell'estrazione dei dati.

Dato che molti siti Web sono realizzati partendo da un database e da una serie di templates costruiti con i dati provenienti da questo, un

estrattore di dati ideale dovrebbe essere in grado di "vedere" attraverso i templates e creare una copia esatta di tale database, anche se ha una visione limitata ai dati. Lo studio del sistema Andes permette di ritenere fattibile la realizzazione di estrattori di dati dal Web in grado di ottenere risultati accettabili e di sostenere che l'incorporazione di conoscenze specifiche del dominio a cui appartengono i siti all'interno del processo di estrazione possa essere determinante nell'assicurare l'alta qualità dei dati estratti. E' stato evidenziato come l'estrazione basata su espressioni di XPath e templates di XSLT come formato compatto per la definizione di tali espressioni stia guadagnando popolarità. E' stato inoltre proposto un metodo per costruire tali XML path expressions attraverso *àncore* e *salti.* Dato che la tendenza generale nella realizzazione di pagine Web è quella di segmentare la pagina sia a livello visuale che a livello sintattico nelle sue parti costituenti (titoli, annunci, aiuti alla navigazione, contenuto principale), è stato proposto un processo in cui per prima cosa viene individuata un'àncora situata approssimativamente nella posizione in cui si trovano i dati desiderati (tabella, lista, frame, etc.). Dall'àncora è poi relativamente semplice effettuare un breve salto alla precisa locazione dei dati.

E' stata effettuata una classificazione dei tipi di espressioni di salto che spaziano lungo tutto l'albero XHTML del documento (espressioni globali) o solo su un sottoalbero (espressioni relative), coprendo matching patterns basati sul contenuto (testo letterale), valori di attributi e strutture. La maggior parte dei tipi di espressioni di salto cade nella categoria "ricerca globale per contenuto", che è anche la più potente. Essa implica la ricerca di un'occorrenza di un testo come "Ultimo Prezzo" che è noto essere unico all'interno del documento e vi rimane anche quando cambia l'impaginazione della pagina Web (robustezza *forward looking*).

Un'altro importante dato emerso è dato dal fatto che molti campi di dati individuali possono essere estratti utilizzando la stessa àncora. Questo è dovuto alla relativa vicinanza di dati di interesse l'uno dall'altro e significa che quando una pagina cambia e l'àncora definita precedente-

mente fallisce, è sufficiente ridefinire una nuova àncora (o comunque un piccolo numero di queste), invece di una nuova àncora per ogni campo di dati che deve essere estratto.

Il lavoro della IBM continua nelle seguenti direzioni. Le regole di navigazione e di estrazione sono attualmente ottimizzate manualmente, un fardello che tools automatici o semiautomatici possono facilitare. Analogamente, l'utilizzo di strumenti automatici può aiutare nella costruzione e nella gestione di regole specifiche di un dominio che consentano di maneggiare le situazioni relative alla perdita o al conflitto tra dati. Non è difficile vedere dati non corretti in un sito Web (nomi sbagliati o valori con errate unità di misura), le regole di validazione dei dati che attualmente sono scritte a mano, verranno rimpiazzate con regole generate da tools semiautomatici, in questo senso tecniche di classificazione dei dati e di apprendimento automatico sembrano essere le soluzioni più probabili. In futuro, le regole di validazione verranno espresse mediante la sintassi di XML Schema.

Capitolo 5

Il Sistema LIXTO

Lo studio di sistemi di generazione di wrappers completamente automatici ha trovato in XWRAP Elite e RoadRunner i toolkits più interessanti in circolazione, l'attenzione si è quindi spostata su strumenti semiautomatici che richiedono l'intervento del progettista per implementare quelle caratteristiche di robustezza che solo la generazione assistita e basata sul contenuto delle pagine piuttosto che sulla loro struttura può garantire: a tale scopo nel capitolo precedente è stato studiato il sistema Andes. L'esigenza di semplificare le procedure di generazione dei wrappers al fine di permettere anche a wrapper designers senza conoscenze specifiche di programmazione in HTML di ottenere buoni risultati ha portato alla ricerca di strumenti più intuitivi nel loro uso ma che fossero in grado di garantire risultati di livello accettabile. Tra quelli in circolazione i più interessanti sono Lixto e Lapis. In questo capitolo ci si concentrerà sul primo, il secondo verrà analizzato nel capitolo successivo.

Lixto Visual Wrapper è un tool visuale e interattivo per la generazione di programmi wrappers che vengono utilizzati per estrarre le informazioni da pagine Web in formato HTML e convertirle in XML. Mediante l'utilizzo di tali wrappers Lixto permette di rendere l'estrazione di informazioni da pagine Web completamente automatica e di tradurre il contenuto estratto in formato XML. Dopo che un programma wrapper è stato scritto per una classe di pagine, il wrapper può ripetutamente esse-

re usato per estrarre dati anche da pagine il cui contenuto cambia di frequente. Uno dei punti di forza di questo strumento è rappresentato dalla sua interfaccia grafica, completamente visuale ed interattiva; la procedura di selezione e marcatura delle regioni è sufficientemente intuitiva da permettere anche agli utenti con poca familiarità con l'HTML di lavorare con il generatore di wrappers. Bisogna però precisare che in questo caso come nell'utilizzo di strumenti simili i risultati migliori si ottengono con la conoscenza della struttura della pagina HTML in esame, come si evince dai risultati della sperimentazione diretta illustrata in una delle sezioni successive. Lixto permette l'estrazione di modelli basati su punti di riferimento circostanti ai contenuti stessi, su attributi HTML, sulla sequenza di apparizione e su concetti semantici e sintattici. Sono presenti caratteristiche avanzate come definizioni di modelli disgiuntivi, navigazione verso altre pagine durante l'estrazione, e wrapping ricorsivo. Al progettista è assegnato il compito di definire interattivamente e visivamente modelli di estrazione dell'informazione sulla base di pagine di esempio. Questi modelli di estrazione sono collezionati in una gerarchia di conoscenze che costituisce un programma wrapper dichiarativo. La conoscenza estratta è internamente rappresentata dal linguaggio Elog. L'utente di Lixto non deve preoccuparsi di conoscere questo linguaggio, in quanto attraverso l'ausilio di interfacce grafiche sarà in grado di costruire un wrapper senza occuparsi dei dettagli implementativi. Lixto permette al progettista di definire regole di trasformazione in XML che specificano come i contenuti estratti possano essere traslati in XML.

5.1 Architettura del Sistema Lixto

L'architettura di sistema di Lixto comprende due blocchi principali: il VISUAL BUILDER ed il PROGRAM EVALUATOR. Il visual builder permette al wrapper designer di creare e visualizzare un wrapper, nonchè di specificare come i dati estratti possano essere convertiti in XML. Il program evaluator esegue automaticamente un programma di estrazio-

ne su pagine Web e mediante uno schema di traduzione trasforma i dati estratti in formato XML. Il program evaluator è utilizzato anche durante la fase di progettazione del wrapper al fine di testare completamente o parzialmente il programma. Riassumendo, un wrapper designer crea un programma wrapper interagendo con il visual builder e specificando quali sono le informazioni da estrarre da una pagina di esempio. Il wrapper creato mediante queste indicazioni viene quindi applicato ad una classe di pagine strutturate in maniera simile. Nell'architettura Lixto, i modelli specificano come estrarre sezioni di dati da pagine Web. Un modello comprende un numero di filtri che sono interpretati indipendentemente l'uno dall'altro. Alcune condizioni sono utilizzate per restringere il numero di istanze di un modello di interesse, queste ultime non sono altro che sezioni elementari di dati che soddisfano uno o più modelli. Le condizioni sono interpretate congiuntivamente.

5.2 Estrazione di modelli in Lixto

Lixto permette al wrapper designer di costruire modelli basati su una o più pagine Web. Un wrapper viene costruito formalizzando, collezionando e memorizzando la conoscenza su modelli di estrazione desiderati, questi ultimi descrivono singoli data-object o sezioni di dati che devono essere estratti da pagine Web. I modelli di estrazione sono generati e rifiniti interattivamente e semi-automaticamente con l'aiuto di un wrapper designer (un progettista), vengono costruiti in maniera gerarchica su pagine di esempio mediante la marcatura di sezioni di dati rilevanti o regioni, attraverso click del mouse o azioni simili come selezioni dal menu o semplici input testuali. Mentre i modelli sono descrizioni dei dati che devono essere integrati, le istanze dei modelli sono data-objects o sezioni generiche di dati contenute in pagine Web che soddisfano tali descrizioni e quindi devono essere estratti. Su alcune pagine zero, una o alcune istanze di un modello possono soddisfare lo stesso modello.

I modelli che costituiscono un wrapper formano una struttura ge-

rarchica. Ogni modello comprende almeno un filtro e può avere alcuni modelli figli. I filtri specificano quale parte della pagine Web dovrebbe essere estratta. I filtri possono essere resi più selettivi mediante l'impiego di condizioni. Tutte le condizioni di un filtro devono essere soddisfatte dagli elementi estratti dal filtro ma solo uno dei filtri di un modello deve essere soddisfatto per poter permettere l'estrazione del dato dal modello. Il processo di scrittura di un wrapper consiste nella creazione di modelli, nell'aggiunta di filtri a questi ultimi e nel testaggio delle istanze che soddisfano i filtri e solo quando non se ne può fare a meno nell'aggiunta di condizioni. Un programma wrapper può essere applicato a classi di pagine strutturalmente simili.

I modelli sono i construtti di base di un programma Lixto e definiscono la sua struttura gerarchica. Ogni modello viene successivamente mappato in un tag XML ed un modello figlio corrisponde a tags innestati all'interno del tag padre.

5.2.1 Categorie di modelli

Il tipo di modello da utilizzare dipende dal tipo di informazione che si vuole estrarre, è possibile scegliere una delle seguenti tre possibilità:

- Modelli ad albero, servono ad estrarre parti di documenti corrispondenti ad elementi HTML o liste di documenti. Questi possono essere ad es: tabelle, paragrafi, una lista di righe di tabelle e anche elementi di contenuto. Nel modello ad albero possono essere identificate le seguenti caratteristiche: il nome del modello, la categoria, il modello padre di default, i filtri e condizioni opzionali assegnate al modello. I filtri di un modello sono mostrati direttamente sotto al modello al quale appartengono. Le condizioni sono mostrate sotto il filtro a cui a loro volta appartengono (ad eccezione delle condizioni di Range che sono mostrate nella riga del filtro stesso). Tutte le altre informazioni sono mostrate nella stessa linea del nome del modello.

- Modelli stringa, servono ad estrarre stringhe testuali da parti visibili e invisibili di un documento. Queste parti possono essere un indirizzo mail, o un codice postale ma anche valori di attributi come il nome di un'immagine.

- Modelli di documento, servono ad estrarre un'intera pagina Web e sono utilizzati per navigare tra le pagine. Il modello iniziale di un programma Lixto, chiamato rootDocument, è un esempio di modello di documento, ed altri tipi di modelli di documento possono essere aggiunti nel seguito per poter estrarre dati da pagine Web collegate mediante hyperlinks.

5.3 I filtri in Lixto

L'organizzazione logica di un modello di estrazione è la seguente: ogni modello di estrazione ha un nome e contiene un insieme di filtri: ogni filtro fornisce una definizione alternativa dei dati che devono essere estratti ed associati con il modello. Per esempio, assumiamo di voler estrarre parole chiave da una pagina Web, che si presentano sia in colore rosso che in grassetto. Un modello chiamato "keyword" potrebbe essere definito per mezzo di due filtri: uno per le parole chiave rosse e l'altro per quelle in grassetto. Il set di filtri di un modello è interpretato disgiuntivamente (connessi in logica OR). Gli sviluppatori di Lixto hanno individuato tre tipi di base di filtri: quelli ad albero, quelli di stringa e quelli di documento. I modelli ad albero sono specificati attraverso filtri ad albero (che a loro volta definiscono regioni ad albero), i filtri di testo che sono specificati attraverso filtri di stringa (che definiscono stringhe testuali che devono essere estratte). Un filtro ad albero contiene una rappresentazione di un albero di parsing generalizzato per pagine Web strutturate, che venga soddisfatta da un set di sezioni di dati all'interno di tale pagina ed un set di condizioni che restringano queste sezioni di dati a quelle di interesse. La restrizione di soddisfare un albero generalizzato e le con-

dizioni di un filtro sono interpretate congiuntivamente (logica AND) nel senso che un elemento di una pagina Web soddisfa un filtro se e solo se esso soddisfa il suo albero generalizzato e tutte le condizioni del filtro simultaneamente. In maniera analoga, un filtro stringa contiene in un linguaggio formale (ad es: attraverso espressioni regolari) le specifiche da cui estrarre testo. I costituenti di un filtro stringa sono interpretati congiuntivamente, nel senso che una stringa, per poter essere estratta, deve soddisfare tutti i requisiti contemporaneamente.

5.3.1 Filtri ad albero

Se più di una istanza soddisfa il modello padre, deve essere scelta quella che vuole essere utilizzata per la creazione del filtro. A questo punto deve essere selezionata la regione che il filtro vuole estrarre (questa regione è costituita da un insieme di elementi), e con questa i percorsi del nodo padre, del primo e dell'ultimo dei nodi figli. Il sistema individua automaticamente il percorso HTML verso l'istanza selezionata, questo percorso può essere modificato dal progettista selezionando uno dei tre seguenti modi:

- default: Il sistema crea automaticamente il percorso di default, il quale consiste in una sequenza di elementi del percorso ad albero separati da asterischi.
- manuale: Per ogni elemento del percorso individuato, il progettista può selezionare se aggiungere o meno un asterisco come prefisso ad uno degli elementi che compongono il path.
- minimale: Il sistema prenderà solo il percorso dell'ultimo componente con un asterisco prima di esso.

Una volta specificato il percorso occorre indicare il modo di costruzione.

I differenti tipi di costruzione sono:

- Default: Il sistema seleziona automaticamente tutti gli attributi disponibili per gli elementi selezionati ma non considera i loro valori.
- All exactly: Il sistema prende tutti gli attributi ed i loro valori esattamente come trovati nell'istanza selezionata.
- None: Il sistema non prende alcun attributo.
- Categories: Il progettista può specificare quali attributi della pagina HTML considerare per i suoi scopi. Solo le categorie applicabili all'elemento selezionato sono abilitate. Una volta selezionate le categorie occorre assegnarvi uno dei seguenti tipi di valori:
  - Ignore value: L'attributo deve esistere ma il suo valore viene ignorato.
  - Exactly: Il valore dell'attributo deve essere uguale a quello dell'elemento selezionato.
  - Contains: Il valore dell'attributo deve contenere quello dell'attributo selezionato. È anche possibile aggiungere restrizioni al contenuto. A questo scopo sono presenti più di un elemento nella listbox da cui effettuare la scelta uno dei quali è "matches regexp", che permette di inserire un'espressione regolare.
- Custom: Si ha il pieno controllo sulla selezione degli attributi. Devono essere selezionati tutti gli attributi che vogliono essere utilizzati, ed i relativi tipi dalle listboxes corrispondenti. Nel modo custom sono presenti quattro tipi di attributo aggiuntivi:
  - is sintactical concept
  - contains syntactical concept
  - is semantical concept
  - item contains semantical concept

Tra i concetti sintattici a disposizione per poter effettuare la scelta del tipo da assegnare ad un attributo sono presenti: “date”, “mail”, “number”. È prevista la possibilità per il progettista di definire nuovi concetti di questo tipo. Un discorso analogo lo si può fare per quanto riguarda i concetti semantici.

5.3.2 Filtri Stringa

Sono utilizzati sia per estrarre parti di un elemento testuale che per estrarre parti di testo invisibili come ad esempio valori di attributi. Per quanto riguarda il primo scopo, può essere raggiunto utilizzando espressioni regolari o concetti semantici o sintattici, mentre il secondo, inserendo il nome dell’attributo. Molto spesso durante la fase di test sono stati usati per separare il nome di un hyperlink dal valore dell’URL relativo.

5.3.3 Filtri Documento

Sono usati per concatenare informazioni provenienti da alcune pagine HTML in un singolo file XML. Ci sono due situazioni tipiche in cui sono utilizzati filtri documento:

- Quando è possibile trovare informazioni dettagliate su pagine collegate alla pagina originale. Solitamente queste pagine hanno struttura differente rispetto alla pagina principale e sono scandite con un sottoalbero del programma.

- Liste molto lunghe sono spesso spezzate in alcune pagine connesse attraverso un collegamento "next". Queste pagine solitamente hanno tutte la stessa struttura e viene utilizzato un programma ricorsivo per scandirle. Questo viene fatto aggiungendo un nuovo filtro di documento al modello rootDocument che segue il link "next".

5.4 Condizioni in Lixto

Lixto offre la possibilità di esprimere vari tipi di condizioni per restringere il numero di data object restituiti da un filtro. I principali tipi di condizioni sono: *inerent conditions* (condizioni interne), *contextual condition* (condizioni esterne) e *range conditions*.

- Una condizione interna di un filtro specifica che alcune caratteristiche devono apparire (o non apparire) all'interno dei data objects restituiti dal filtro per poter essere estratti. Per esempio, una condizione interna può definire che un URL deve presentarsi all'interno di testo affinchè questo possa essere estratto, oppure che una tabella per poter essere estratta non dovrebbe contenere alcun elemento di colore rosso.

- Una condizione contestuale di un filtro impone alcune restrizioni sul contesto nel quale ad un data object che soddisfa il modello è consentito di apparire. Per esempio, una condizione "before" potrebbe specificare che nel testo che precede un modello obiettivo, devono essere presenti alcuni elementi caratteristici (una parola, la dimensione di un font o il colore). In maniera analoga Lixto fornisce una condizione "after" che segue lo stesso principio di funzionamento della condizione "before". È anche possibile qualificare ulteriormente tali condizioni con indicatori di distanza. Le condizioni "before" ed "after" possono essere negate. Es: la condizione "notbefore" specifica che qualche elemento caratteristico non deve presentarsi prima di un modello obiettivo.

- Una condizione range specifica che tra i data object figli di un data object padre che soddisfano tutte le condizioni, solo quelli in un certo intervallo (es:i primi tre) dovrebbero essere estratti.

In aggiunta a questi tre tipi base di condizioni, Lixto permette al progettista di esprimere condizioni ausiliarie. Le più importanti tra queste sono:

- Condizioni per referenziare i modelli, esprimono il fatto che alcuni elementi devono essere un'istanza di modelli definiti (solitamente un modello differente da quello associato al filtro corrente).
- Condizioni concettuali, esprimono che qualche valore di attributo appartiene ad una classe "ontologica" predefinita (per esempio, che una stringa corrisponde al nome di una città o ad una data).
- Condizioni comparative, sono relazioni predefinite per classi ontologiche di elementi che a loro volta sono stati definiti a priori (es: le date possono essere confrontate per verificare se una data viene prima di un'altra data specificata, oppure se una stringa rappresenta il nome della città "Modena", tutto ciò indipendentemente dalla lingua in cui è scritta). Questa caratteristica è già supportata da alcuni concetti predefiniti. Gli utenti possono aggiungere nuove condizioni comparative.

L'uso di condizioni concettuali e comparative presuppone che un'implementazione di Lixto contenga in sè la possibilità di conoscenze ontologiche attraverso la costruzione di un dizionario interno oppure abbia accesso ad un database ontologico esterno. Il sistema è aperto ad entrambe le soluzioni.

5.5 Il processo di progettazione del wrapper

I modelli di estrazione sono definiti dal progettista in maniera gerarchica. Al modello che descrive un intero documento ci si riferisce come al modello del documento. Alcuni modelli di documenti sono disponibili come modelli predefiniti automaticamente creati dal sistema. In particolare il modello di documento corrispondente alla pagina Web di partenza, chiamato "home document pattern" è disponibile come modello preesistente. Altri modelli sono definiti interattivamente dal progettista. Filtri o modelli possono essere definiti nel contesto di altri modelli (chiamati modelli sorgenti). Per esempio, si può definire il modello *nome*,

quindi solo successivamente si possono definire i modelli *nome_proprio* e *cognome* all'interno del modello sorgente *nome*. Per i compiti di estrazione più usuali, sarà sufficiente definire modelli semplici o una stretta gerarchia di modelli. Non è necessario che la definizione del modello segua un approccio strettamente gerarchico (come un albero), è permesso che essa possa essere ricorsiva, come nella definizione di tipi ricorsivi all'interno dei linguaggi di programmazione. Ad esempio, un modello "table" potrebbe essere definito mediante l'uso di due filtri, dove uno si riferisce al modello del documento mentre l'altro si riferisce al modello "table" stesso. In questo modo è possibile definire un wrapper che automaticamente estrae tutte le tabelle da una gerarchia di tabelle innestate. È questo il sistema utilizzato per l'estrazione delle informazioni dalle pagine del sito expomo.it (vedi figura 5.1), in questo caso il con-

Figura 5.1: Pagina dinamica del sito www.expomo.it

cetto di livello più esterno veniva dettagliato mediante l'utilizzo di una

tabella di livello più interno che racchiudeva i dati relativi ad ogni data object di interesse (vedi figura 5.4). Questo consente di affermare che la possibilità di isolare graficamente ogni data object come simple element è una garanzia di corretta estrazione della struttura dei dati e dei relativi valori. Un simple element viene inteso come una sezione della pagina

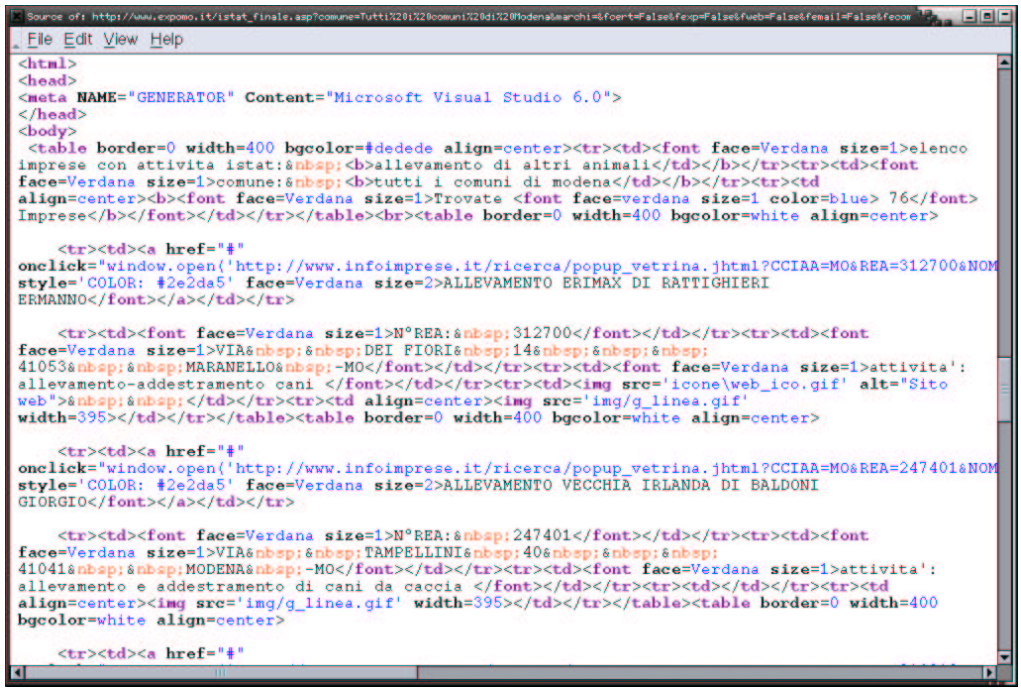

Figura 5.2: Pagina dinamica del sito www.expomo.it

nella quale il sistema riconosce una struttura dati elementare (come nel caso di una tabella semplice o un modello ripetitivo di tabelle contenenti un determinato tipo di struttura). I problemi nascono nel momento in cui il concetto più esterno viene dettagliato mediante l'uso di uno o più concetti di livello più interno racchiusi in tabelle dello stesso livello, unitamente a concetti allo stesso livello del concetto esterno. In questo caso il data object di interesse, non essendo riconosciuto come simple element, dovrebbe essere ricostruito mediante l'unione dei concetti a lui relativi, indipendentemente dal loro livello di profondità. In questa situazione è possibile la costruzione di una serie di filtri ognuno dei quali

dedicato all'estrazione di una specifica sezione del data object di interesse. Seguendo questa strada si raggiunge un punto critico nel momento di ricollegare i dati estratti dai filtri al singolo data object. Allo stato attuale Lixto non è in grado di intervenire per ricostruire un data object partendo dai componenti elementari estratti, al fine di fornire la giusta rappresentazione XML della struttura delle pagine fornite in input. Quest'ultimo problema si è presentato nel tentativo di costruire wrappers per alcune voci della sezione "Servizi CNA Interpreta" del sito interpreta.it. Per portare un esempio pratico, nelle pagine relative alla voce "Affari Generali" i data object sono costituiti da una un campo "Area" , dalla data, da un campo "Oggetto" e da due immagini rappresentanti il formato in cui è possibile scaricare il documento. Le informazioni relative a tutti i

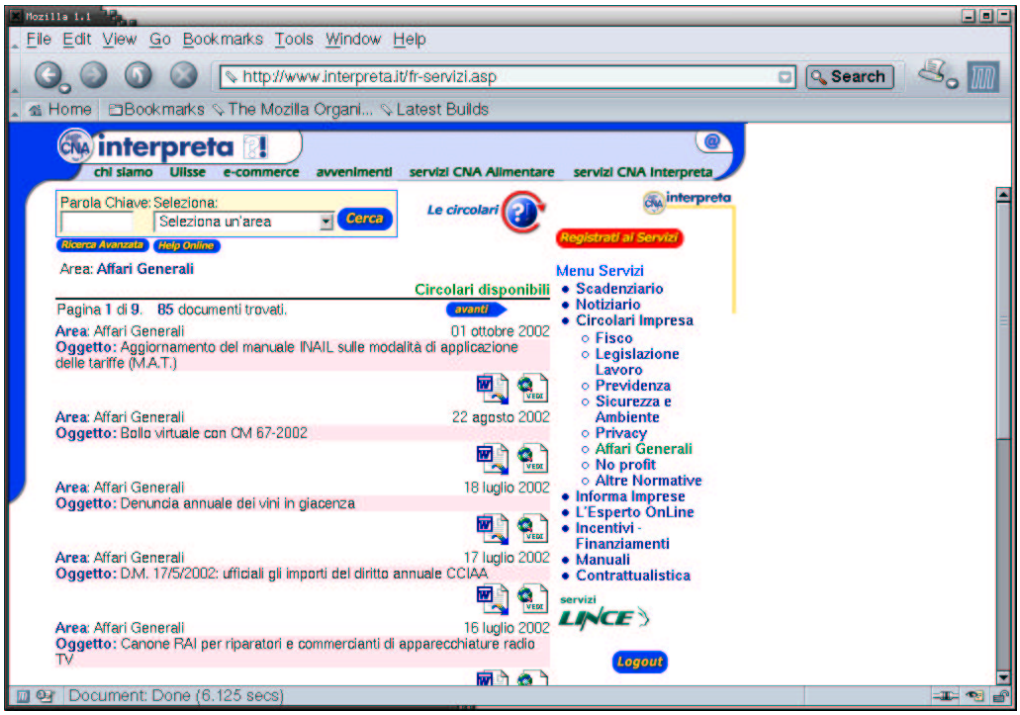

Figura 5.3: Pagina dinamica del sito www.expomo.it

data objects sono contenute in una tabella all'interno della quale ne viene allocata un'altra ogni qualvolta si presenta un nuovo data object per

definirne il contenuto del campo "Area" e la data, questa seconda tabella viene quindi chiusa e viene riportata la parte centrale dell'informazione che è costituita dal contenuto del campo "Oggetto" e dalle immagini. Questi ultimi dati sono espressi come righe e colonne della prima tabella definita ossia quella più esterna. In questo caso il data object è quindi costituito da informazioni aventi diversi livelli di profondità nell'albero HTML relativo alla pagina.

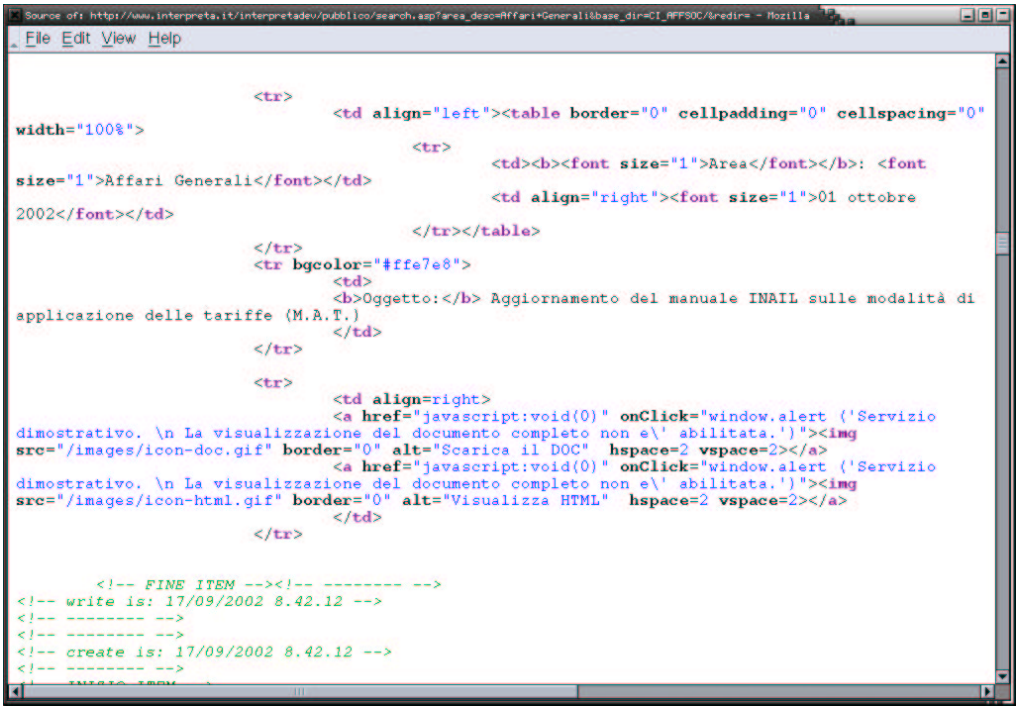

Figura 5.4: Pagina dinamica del sito www.expomo.it

Al contrario di quanto accade per i filtri, ai modelli non è richiesto di formare una gerarchia ad albero, le istanze dei modelli, formano sempre una gerarchia di tale tipo. Queste istanze sono infatti costituite da dati (una regione ad albero o una stringa) più una referenza all'istanza di un modello padre (eccetto ovviamente per l'istanza del modello "home", dal quale Lixto è partito).

Il progettista è in grado di definire un programma di estrazione ed

uno schema associato alla traduzione in XML mediante il metodo di definizione del modello. Il visual builder di Lixto permette di definire modelli e filtri con l'aiuto di una o più pagine di esempio e di modificare e memorizzare i modelli. A vari passi intermedi, il progettista ha la possibilità di testare un modello o un filtro, indipendentemente dal fatto che sia stata costruita solo una parte del wrapper piuttosto che l'intero programma. Questo test può venire effettuato sia sulle pagine di esempio usate per costruire il modello o il filtro che su ogni altra pagina Web. Il risultato di tale test è un set di istanze del modello, le quali sono mostrate da un browser come elementi evidenziati. Un modello è inizialmente progettato mettendo a punto un filtro, quindi nel caso in cui questo non sia sufficiente (è questo il caso in cui i test mostrano che non tutti i data object di interesse sono stati estratti dalle pagine di prova), il procedimento viene ripetuto mettendo a punto un altro filtro e così via, fino a che ogni data object di interesse è coperto da almeno un filtro associato a quel pattern. Al fine di progettare un determinato modello, il progettista ha la possibilità di realizzare graficamente alcuni filtri per quel modello, testare ogni filtro separatamente e quindi testare l'intero modello. A grandi linee la descrizione del processo di realizzazione di filtri ad albero può essere descritta come segue:

- Il progettista contrassegna un elemento iniziale su una pagina di esempio (per esempio una tabella).
- Il sistema associa a questo elemento un percorso ad albero generalizzato appartenente all'albero di parsing che possibilmente corrisponde ad alcuni elementi simili (per esempio alcune tabelle).
- Il progettista a questo punto testa il filtro. Se vengono estratte altre informazioni oltre i data object di interesse, il progettista aggiunge condizioni restrittive al filtro e ripete il test.

Questa procedura viene ripetuta fino a quando il filtro realizzato rende possibile l'estrazione dei dati desiderati. Una procedura analoga viene

utilizzata per mettere a punto filtri per stringhe. Degna di nota è la disponibilità di metodi visuali per la definizione e l'aggiunta di condizioni di filtro. Per prima cosa il progettista può scegliere attraverso un menù

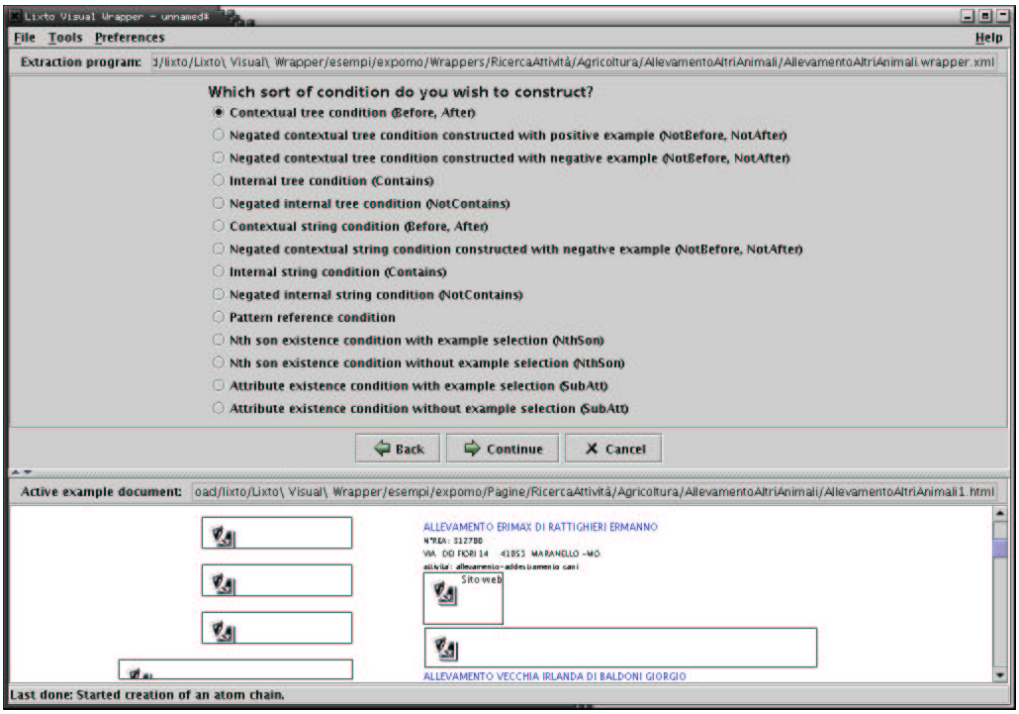

Figura 5.5: Definizione di condizioni in Lixto

il tipo di condizione che vorrebbe aggiungere al filtro che si sta costruendo. In secondo luogo, ci sono metodi visuali e interattivi per definire una condizione del tipo richiesto. Per esempio, una condizione di tipo "before" può essere specificata innanzitutto evidenziando (con il mouse) un elemento con le caratteristiche richieste che si presenta prima del precedente esempio identificato come istanza del modello, quindi selezionando gli attributi rilevanti (es, colore, dimensione del font, contenuto testuale, etc.), dell'elemento "before" attraverso il menu. Il supporto visuale ed interattivo per la costruzione del modello offerto da Lixto, include un ausilio specifico per l'organizzazione gerarchica di modelli e filtri. Per esempio, quando un progettista crea un nuovo filtro ha biso-

gno di contrassegnare il modello sorgente (chiamato modello padre) del filtro, attraverso un click del mouse su di un modello all'interno della gerarchia dei modelli. Il sistema a questo punto aiuta il progettista a scegliere una istanza del modello sorgente adatto sulle pagine di esempio il quale dovrà essere appropriato per definire il filtro subordinato. Il processo di definizione del wrapper non è limitato ad un singolo documento web di esempio e nemmeno alle pagine Web di esempio aventi lo stesso tipo o struttura. Durante la definizione del wrapper, un progettista può utilizzare altre pagine di esempio (ad esempio caricandole all'interno del visual builder) e continuare da queste la definizione del wrapper. Gli URLs di tali ulteriori pagine possono essere forniti esplicitamente dal progettista (scrivendoli direttamente o mediante le azioni di navigazione) oppure sono ottenute come istanze di modelli definiti precedentemente. Ci sono due principali ragioni per cui può risultare utile spostarsi verso altre pagine.

- Primo, un progettista potrebbe voler definire un modello comprendente filtri che descrivono obiettivi di estrazione per differenti tipi di pagine. Per un semplice esempio, assumiamo che un progettista voglia scrivere un wrapper che estragga prezzi da pagine Web sia degli Stati Uniti che dell'Inghilterra. Dove l'unità di misura degli elementi contenenti i prezzi sarà data da "$" per pagine US e "£" per quelle UK. Assumiamo inoltre che la pagina di esempio corrente sia una pagina US. Il progettista crea un modello chiamato "prezzo" dopodichè definisce per questo due filtri, il primo si occupa delle pagine US, il secondo di quelle UK. Dopo che è stato creato con gli strumenti di ausilio visuale un appropriato filtro per prezzi in dollari su una pagina di esempio US già caricata, il progettista si sposta sulla pagina di esempio UK e definisce il secondo filtro per il modello "prezzo" su quest'ultima pagina. A questo punto il wrapper sarà in grado di lavorare su entrambi i tipi di pagine.

- Secondo, un progettista può voler definire un modello ed i realtivi

sotto-modelli che raggruppino elementi da alcune pagine. Il progettista a questo punto ha bisogno di "insegnare" al sistema sulla base di pagine di esempio come seguire URLs e collezionare gli elementi da differenti pagine. Per esempio, un sito di aste elettroniche come eBay pubblica cataloghi sottoforma di pagine Web contenenti data object con le informazioni di interesse. Ogni data object è descritto da una riga di sommario. Ognuna di queste righe contiene un link ad una pagina Web con informazioni più dettagliate sui rispettivi articoli (come ad esempio l'ubicazione del venditore etc.). Il progettista per prima cosa definisce in maniera visuale un modello chiamato "elemento" sulla pagina di sommario, questo viene fatto mettendo in evidenza un elemento di esempio e specificando per esso appropriate condizioni. Quindi il progettista aggiunge modelli figli per tutti gli elementi di dati rilevanti di un data object che sono disponibili sulla pagina di sommario (come ad esempio il numero dell'articolo, il prezzo e il numero delle offerte), aggiunge inoltre un modello di documento chiamato "link" corrispondente al link a pagine più specifiche che descrivono l'articolo. A questo punto il progettista procede seguendo questo URL, caricando la corrispondente pagina e definendo i modelli di interesse rimanenti (come l'ubicazione del venditore), come modelli figli di questo modello di documento.

Dopo che il wrapper è stato definito, i dati estratti devono essere convertiti in XML. L'XML translation builder, è un modulo del visual builder responsabile delle operazioni di supporto al progettista durante la generazione del programma di traduzione in XML. Una questione chiave è quella dei nomi di default dei modelli che sono stati scelti dal progettista durante il processo di realizzazione dei modelli stessi, questi nomi infatti sono presi come valori di default per i tags XML di output. Inoltre la gerarchia delle istanze di modello estratte (che ha sempre struttura ad albero) determina la struttura del documento XML di output. In questo modo, nel caso il progettista non esegua azioni specifiche, viene comun-

que realizzata una traduzione standard delle istanze di modello estratte in formato XML, tutto ciò senza bisogno di ulteriori interazioni. Lixto offre al progettista la possibilità di eseguire la traduzione XML standard nei tre modi seguenti:

- Il progettista può rinominare alcuni modelli con l'effetto che un nuovo nome al posto di quello attuale appare come nome del tag nella traduzione XML.

- Il progettista può sopprimere alcuni modelli dalla traduzione. In questo caso istanze di modelli non soppresse che sono figlie di istanze di modello soppresse S, appariranno nella traduzione XML come figlie dell'antenato non soppresso di S. Per esempio, assumiamo che un progettista voglia costruire un wrapper che estragga tutti i records dalla terza tabella di qualche pagina Web. Dovrebbe per prima cosa costruire un modello tabella che identifichi precisamente la terza tabella. Quindi dovrebbe definire un modello record usando un filtro il cui modello padre sia il modello tabella (con l'effetto che solo records appartenenti alla terza tabella del documento sarebbero identificati come istanze del modello record). In questo caso la gerarchia di modelli è della forma:

  $$documento \leftarrow tabella \leftarrow record$$

  dove ogni freccia sta a significare una referenza ad un modello padre. Mentre il modello tabella ha essenzialmente un ruolo nell'estrazione di data object, il progettista potrebbe decidere di sopprimerlo nell'output XML. Allora l'output XML del documento mostrerà una gerarchia nella forma:

  $$documento \leftarrow record$$

  dove ogni istanza del modello record è figlia dell'istanza di documento.

- Per ogni modello il progettista può scegliere il set di attributi che dovrebbero essere inclusi all'interno del documento XML. Per esempio font, colore o attributi di posizione di elementi HTML possono essere inclusi nel documento XML di output oppure soppressi nel caso in cui non interessino.

- Il tool XML offre la possibilità di settare vincoli, in modo ad esempio che "il modello di prezzo occorra esattamente una volta all'interno di ogni record". Se questi vincoli non sono soddisfatti, viene lanciato un messaggio di allarme.

Le modalità desiderate della traduzione in XML sono determinate durante il processo di progetto del wrapper mediante l'uso di interfacce grafiche e sono memorizzate nella forma di uno "schema di traduzione in XML" che codifica il mapping tra i modelli di estrazione e lo schema XML in una forma opportuna.

5.6 Editors

Una delle possibilità concesse da lixto ai suoi utenti è quella di aggiungere nuovi contenuti semantici e sintattici. Per soddisfare questa esigenza, vengono forniti due tipi di editor concettuali: uno per la creazione di concetti semantici ed uno per quelli sintattici. Di questi due strumenti verrà data una breve descrizione nel seguito.

5.6.1 Editor dei concetti semantici

Questo strumento mette a disposizione dell'utente un'interfaccia grafica che permette di occuparsi di ontologie, contesti, termini, concetti e relazioni tra termini provenienti dal database ontologico costruito con MySQL. Vengono mostrate le ontologie ed i corrispondenti contesti con i quali l'utente sta lavorando in un determinato momento. Quando l'editor concettuale viene avviato per la prima volta, l'utente può vedere

l'elemento radice "ontologie", il quale è un elemento astratto e non ha un corrispondente all'interno del database. Per ottenere dal database le ontologie con le quali l'utente vuole lavorare, è predisposto un'apposito bottone all'interno dell'interfaccia, questo permette di mostrare sull'albero le corrispondenti ontologie. Una delle opportunità fornite da questo editor è quella di aggiungere concetti provenienti da WordNet.

5.6.2 Editor dei concetti sintattici

Il meccanismo di funzionamento di questo editor è analogo a quello dello strumento visto nel paragrafo precedente. Sulla parte sinistra è presente un albero contenente i concetti già creati. Se l'utente mette in evidenza un elemento, il nome ed il valore del concetto selezionato sono mostrati nella parte destra dell'interfaccia. I nomi dei tag stanno a simbolizzare i nomi dei concetti. Il valore del concetto è un'espressione regolare. L'utente può aggiungere nuovi concetti, cancellarne di vecchi o cambiare i valori di concetti esistenti. Le espressioni regolari fornite devono essere scritte mediante l'utlizzo del linguaggio di scripting Perl5. I concetti possono anche essere aggiunti in maniera innestata, anche se Lixto non gestisce in modo particolare i concetti innestati ma li tratta tutti come se fossero allo stesso livello. Gli elementi XML non sono altro che i vari tipi di concetti. I tags XML specificano i nomi dei concetti ed i valori XML sono espressioni regolari le quali definiscono i vincoli sintattici per i concetti.

Questa funzionalità è stata molto utile nella fase di testing dei siti di prova, per tutta una serie di concetti che all'interno delle pagine HTML in esame non trovavano un modo di espressione standard, come nel caso dei numeri di telefono che potevano presentare come separatore tra il prefisso ed il numero vero e proprio una lunga serie di caratteri quali: "/","."," "," "-" etc., attraverso la definizione di una espressione regolare è stato possibile crearsi un opportuno concetto che fosse in grado di tenere conto di tutti questi casi, così come dei casi in cui il prefisso era

composto da tre o quattro cifre ed il numero da sei o sette. Altro caso in cui si è utilizzata questa funzionalità di Lixto è stato la definizione di un apposito concetto che contenesse l'insieme dei valori di un determinato campo. Nella sezione "Bacheche" del sito "tessilmoda.com" , era presente una voce "Banca Dati Dell'Usato" in cui ogni data object conteneva un campo di nome "Compro/Vendo", il valore di questo campo era presente all'interno del seguente insieme: "V","C","v","c". Attraverso l'uso dell'editor dei concetti sintattici è stato possibile definire un'apposita espressione che fosse in grado di estrarre solo il giusto valore dell'attributo "Compro/Vendo" senza rischiare di selezionare altre parti di testo circostanti che non erano di interesse. Nota dolente di questa funzionalità è data dal fatto che i nuovi concetti inseriti non rimangono a disposizione per la successiva sessione di lavoro, questo è probabilmente dovuto ad un bugs del software che tiene in memoria solo i concetti predefiniti dagli sviluppatori mentre al momento della chiusura dello strumento perde quelli aggiunti dagli utenti.

5.7 Espressioni Regolari

Dal momento che il termine è stato usato in più di un'occasione nei capitoli precedenti viene data una breve definizione: un'espressione regolare è una stringa di caratteri che permette di definire modelli di tipo stringa. Con questo tipo di espressioni viene semplificata la ricerca di quelle stringhe che soddisfano il modello. Nella sua forma più semplice, un'espressione regolare è solo una parola o una frase da cercare. Per esempio se consideriamo l'espressione regolare costituita dalla parola foot, ogni argomento con la stringa foot al suo interno soddisfa l'espressione, e quindi parole come foot, football, barefoot, Bigfoot rientrano in questa categoria. In maniera analoga anche parole inframezzate da spazi possono far parte di un'espressione regolare, quindi se viene cercata la stringa "Top ten", è possibile utilizzare l'intera stringa per la ricerca. Questo modello verrà soddisfatto anche dalla stringa "how to stop tension". Un

ultimo accenno è riservato alla proprietà case insensitive delle espressioni regolari, così le espressioni LiXto, LIXto, LiXtO... soddisferanno la stessa stringa. Non ci si dilungherà oltre nella spiegazione di questo argomento anche se questo tipo di espressioni regolari sono state un valido supporto durante la fase di testing di Lixto.

5.8 Risultati della sperimentazione diretta

Analogamente a quanto fatto nella fase di testing di XWRAP Elite, al fine di mettere alla prova Lixto sono stati realizzati wrappers per una serie di siti relativi alla realtà industriale Modenese. I risultati delle prove sono stati più che soddisfacenti, ogni pagina di questi siti ha avuto modo di essere tradotta in XML mediante la realizzazione di un opportuno wrapper che ne ha consentito un arricchimento semantico ad hoc, fornendo le basi per la successiva integrazione mediante l'utilizzo del mediatore MOMIS. Quando il processo di selezione dei dati all'interno delle pagine ha evidenziato dei limiti è comunque stato possibile mettere a punto un modello per l'estrazione parziale dei dati contenuti nella pagina.

Uno dei problemi con cui ci si è dovuti scontrare, prima ancora di arrivare al processo di generazione di wrappers è stata la mancanza di supporto da parte di Lixto ai siti strutturati utilizzando una divisione in frames, mancanza rilevante dal momento che questi rappresentano una buona parte dei siti in circolazione. Il problema è stato risolto, al fine di poter procedere nel lavoro di estrazione dei dati, attraverso il download dei siti test sul server sparc20.ing.unimo.it e l'utilizzo, per il processo di generazione dei wrappers, di quella parte di pagina costituita dal frame contenente i dati di interesse.

Di seguito verranno riportati alcuni dei casi più significativi in cui si sono riscontrati problemi nel processo di generazione di wrappers e le soluzioni che sono state adottate volta per volta per risolverli.

Nella pagina che si raggiunge selezionando la voce "Ricerca Attività" dalla home page del sito expomo.it, troviamo una lista delle principali

Figura 5.6: Voce "Ricerca Attività" del sito www.expomo.it

attività di cui il sito si occupa catalogate in base al settore a cui appartengono (vedi figura 5.6). La pagina è strutturata mediante un elenco di settori di interesse, ognuno dei quali seguito da una lista delle attività che gli competono. Ad ognuna di queste voci è associato un hyperlink che porta ad un'altra pagina di selezioni relative agli ambiti di interesse dell'attività in esame. Volendo mettere a punto un wrapper per la pagina descritta, si è tentato di racchiudere in un simple element ognuno dei data objects contenuti (questi ultimi erano costituiti dal nome del settore di attività e dalle voci relative). Nel momento in cui si è cercato di concretizzare questa idea ci si è però dovuti scontrare con il fatto che Lixto non riconosceva come simple element ogni data object contenuto nella pagina (vedi figura 5.7), il primo dei quali è costituito dal settore "AGRICOLTURA" e dalla voce "Agricoltura Allevamento, pesce e caccia". Andando ad analizzare il codice HTML si è potuto scoprire che la lista dei settori e delle relative attività era stata costruita mediante l'impiego di una tabella e che tutti gli elementi contenuti erano innestati allo stesso livello. Quindi sia i settori che le attività erano strutturate come

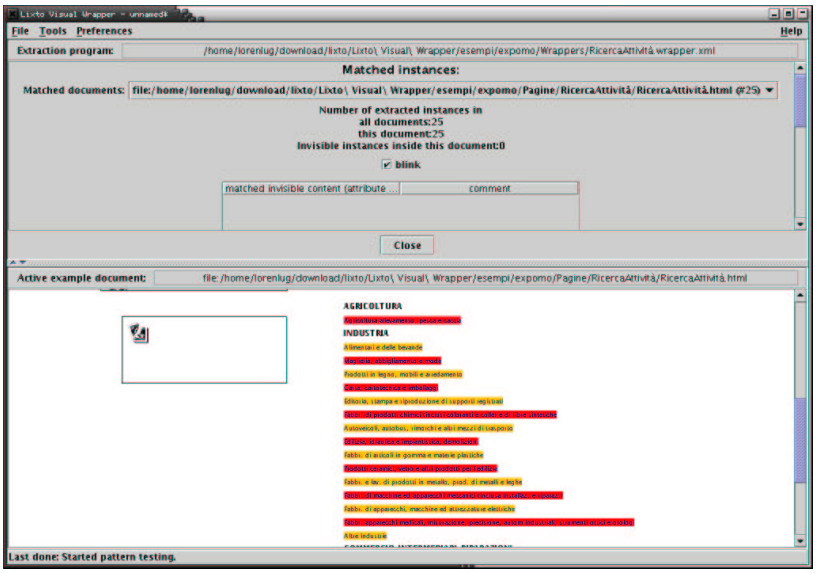

Figura 5.7: Elementi riconosciuti nella pagina "Ricerca Attività" del sito www.expomo.it

righe di questa tabella, l'unica differenza era rappresentata dai font utilizzati e dal fatto che le attività fossero contenute all'interno di un tag àncora (<a>), dal momento che erano ipertesti. Giunti a questo punto, non potendo estrarre tutte le righe della lista come se fossero allo stesso livello, in quanto alcune presentavano differenze concettuali, si è deciso di estrarre unicamente quelle relative alle attività. I dati sui settori sono stati mantenuti in una lista separata, in questo caso però, come evidenziato nella sezione 5.5, Lixto non possiede funzionalità in grado di ricostruire un simple element partendo dai suoi componenti e quindi si mantiene un'informazione che se non collegata alle attività pertinenti perde gran parte del suo valore ai fini dell'integrazione. Nel caso ideale in cui fosse stato possibile identificare come simple element un settore e le relative attività, si sarebbe potuta innestare all'interno del tag "Settore" il cui valore era rappresentato dal nome del settore di interesse, il tag "Lista.Attività" contenente l'insieme degli elementi "Attività" il cui valore era rappresentato dal nome dell'attività. A sua volta l'insieme

dei tag "Settore" sarebbe stato contenuto all'interno di un tag di lista di nome "Lista.Settore". Ad ognuno degli elementi "Attività" figli dell'elemento "Lista.Attività" è possibile associare una coppia di elementi contenenti il testo dell'hyperlink e l'URL corrispondente. Questo viene realizzato mediante l'aggiunta di due modelli di tipo stringa al modello che estrae i singoli item della lista. Per quanto riguarda l'indirizzo URL se esso referenzia una pagina contenuta all'interno del sito stesso è necessario aggiungere l'indirizzo URL relativo della pagina in esame, il sistema provvederà poi a completarlo con la parte mancante.

La notazione adottata per i nomi degli elementi segue due semplicissime regole: si utilizza il carattere underscore "_" per unire parole che hanno un significato se collegate tra loro in base al loro ordine di apparizione e non sempre mantengono significato se prese singolarmente, mentre si utilizza il carattere "." per separare parole che forniscono significati se prese singolarmente e forniscono significati anche se considerate insieme, indipendentemente dal loro ordine di apparizione. Per fare un esempio il tag "Lista.Attività" indica che ciò che è contenuto all'interno del tag è una lista delle attività. In questo modo un semplice motore di ricerca in grado di esaminare i nomi dei tags di una pagina, potrebbe fornire un importante ausilio al dizionario lessicale "WordNet" nella fase di integrazione dei dati estratti in MOMIS.

Proseguendo nella navigazione e selezionando la voce "Agricoltura allevamento, pesce e caccia" si arriva ad una pagina in cui la situazione precedente si ripete (vedi figura 5.8), la differenza sta nel fatto che qui abbiamo un'unica voce: "Agricoltura allevamento, pesce e caccia" seguita da un elenco delle attività relative, questo ha consentito di mettere a punto un modello che riconosca come simple element l'intero contenuto della pagina (vedi figura 5.9). All'interno di questo modello, poco importa se la sequenza: titolo, relative voci, vengono riconosciute come simple element o meno dal momento che è l'unica struttura presente, è stato sufficiente mettere a punto un modello che estraesse il titolo ed un altro per la lista degli ipertesti, quest'ultima non poteva che essere relati-

Figura 5.8: Voce "Agricoltura Allevamento, Pesce e Caccia" del sito www.expomo.it

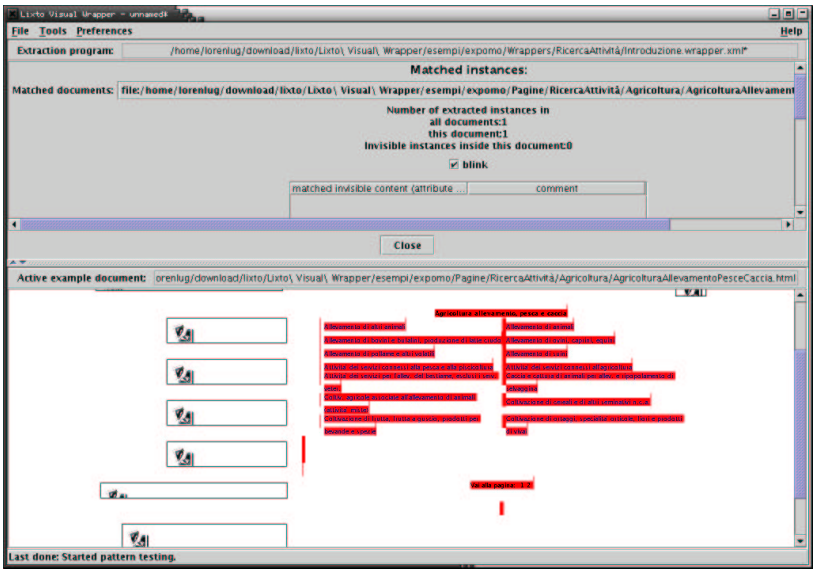

Figura 5.9: Elementi riconosciuti "Agricoltura Allevamento, Pesce e Caccia" del sito www.expomo.it

va all'unico titolo presente. La parte restante del processo di costruzione è analoga a quanto visto per il caso precedente.

Figura 5.10: Voce "Allevamento altri animali" del sito www.expomo.it

Andando ancora avanti nella navigazione di questo sito, selezionando ad esempio la voce "Allevamento altri animali" dalla pagina precedente, si arriva ad una delle pagine costruite mettendo insieme il vero e proprio contenuto informativo del sito (vedi figura 5.10), i suoi componenti sono una serie di data object contenenti ciascuno i campi necessari ad ospitare i dati relativi ad ogni allevamento, per la precisione: il nome dell'azienda, il numero del codice REA, l'indirizzo e per finire l'attività o le attività principali. In questo caso lo strumento è stato in grado di selezionare ogni data object come un simple element (vedi figura 5.11). Andando a vedere il codice HTML, si constata che la pagina è strutturata mediante l'utilizzo di una tabella che ospita al suo interno una lista di tabelle, ognuna delle quali dedicata a contenere i dati relativi ad un data object, possiamo ritenere questa situazione una delle più adatte a favorire l'estrazione della struttura del documento rappresentato dalla pagina, si ha infatti un legame diretto tra la strutturazione dei dati e quella della pagina HTML destinata a contenerli.

Il wrapper messo a punto per questa pagina potrebbe essere utilizza-

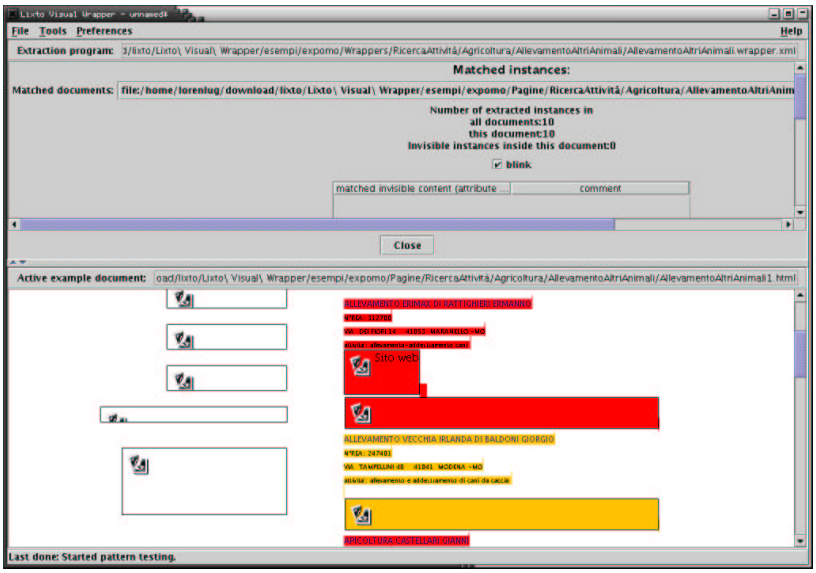

Figura 5.11: Riconoscimento di elementi dalla pagina "Allevamento altri animali" del sito www.expomo.it

to anche per altre parti dello stesso sito strutturate in maniera analoga, in questo caso però le informazioni semantiche introdotte, al fine di poter essere il più generali possibili, dovrebbero essere relativamente povere. Si è preferito perciò mettere a punto un wrapper differente per ogni classe di pagine, il cui principio di funzionamento è sempre lo stesso ma attraverso una differente nomenclatura dei modelli è stato possibile effettuare quell'arricchimento semantico che permetterà un buon grado di integrazione ai dati estratti attraverso i wrappers generati. Per citare solo un esempio di ciò che differenzia un wrapper da un'altro, si mostra il caso dell'elemento che contiene la lista degli item relativi a ciascun data object della pagina in esame, è stato utilizzato a questo scopo il nome "Allevamento_Altri_Animali.Lista", questo consentirà ad un motore di ricerca di riconoscere il fatto che il tag contenga una lista di elementi ognuno dei quali contiene i dati relativi ad un differente allevamento di altri animali (animali che non rietrano nelle classificazioni effettuate in altre voci della maschera precedente). Cambiando la pagina in esa-

me sarà possibile riferirsi all'attività a cui fanno riferimento i data object contenuti, attraverso un differente nome del tag di lista.

Figura 5.12: Strumenti di salto per strutturazioni multipagina nel sito www.expomo.it

I dati relativi ad ognuna delle attività vengono nella maggior parte dei casi restituiti mediante l'impiego di più di una pagina, è stato visto nelle sezioni precedenti che Lixto contiene al proprio interno le funzionalità necessarie ad effettuare un salto da una pagina all'altra, seguendo ad esempio i link relativi ad un bottone "next". Nel nostro caso non era presente un bottone "next" ma si aveva a disposizione nella parte finale di ogni pagina una lista di numeri, ciascuno dei quali contenente un link che consente di saltare alla relativa pagina (vedi figura 5.12), questo implica che volendo permettere al nostro wrapper di seguire ogni link sarebbe stato necessario mettere a punto un modello ricorsivo per ogni possibilità presente, questo approccio è possibile quando il numero delle pagine è inferiore alla decina ma diventa molto dispendioso in termini di tempo quando questo numero sale fino a dimensioni dell'ordine del centinaio, come accade in alcune delle sezioni del sito in esame.

Il sito in esame era quello con struttura più semplice tra quelli a di-

sposizione, e fatta eccezione per qualche maschera di navigazione, tutte le pagine del sito erano generate dinamicamente e con la stessa struttura per i data object. La sua modalità costruttiva, semplice ed omogenea per tutti i tipi di pagine, ha consentito di ottenere ottimi risultati dalle fasi di costruzione dei wrappers e di successiva estrazione dei dati, questo sia a livello qualitativo che in termini di tempo totale dedicato alla generazione dei wrappers. Si passa ora ad esaminare il risultato dell'applicazione di Lixto agli altri siti.

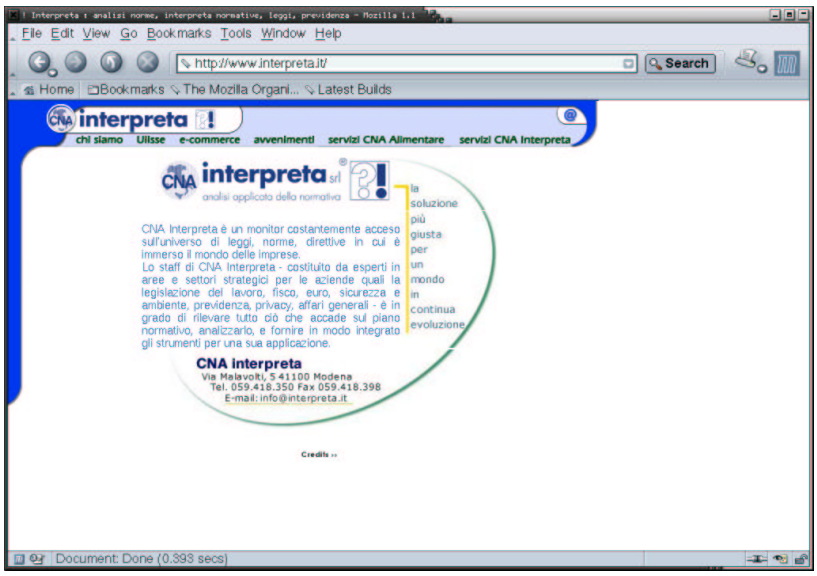

Figura 5.13: Home page del sito www.interpreta.it

Per quanto riguarda il sito interpreta.it (vedi figura 5.13), esso si articola nelle seguenti parti principali: "ChiSiamo", "Avvenimenti", "Ulisse, E-commerce", "Servizi CNA Interpreta", "Servizi CNA Alimentare". La voce "ChiSiamo" permette l'accesso ad una serie di pagine, ognuna delle quali contenente informazioni su un argomento differente (vedi figura 5.14), è stato quindi messo a punto un wrapper in grado di estrarre i contenuti testuali di queste pagine e questo è stato utilizzato per ricavare wrappers differenti per ogni tipo di pagina, in grado di aggiungere in-

formazioni sul significato delle informazioni contenute in ognuna delle pagine.

Figura 5.14: Sezione "Chi Siamo" del sito www.interpreta.it

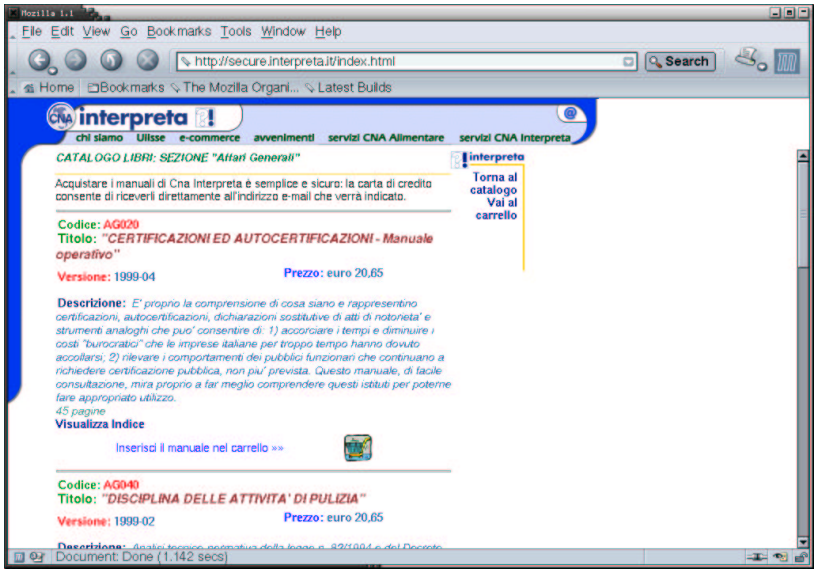

Figura 5.15: Sito www.interpreta.it, voce "E-commerce/AffariGenerali"

Per la sezione “E-commerce” , selezionando a titolo di esempio la voce “Affari generali”, si ottiene una pagina di data objects costituiti da un set di informazioni di vario genere relative ad un documento (vedi figura 5.15). Lixto ha riconosciuto ogni data object come un simple ele-

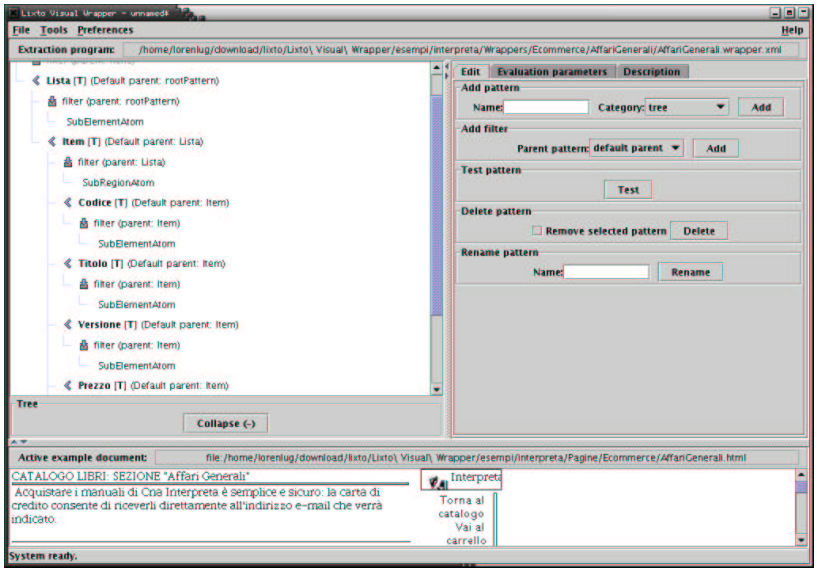

Figura 5.16: Sito www.interpreta.it sezione del wrapper per la voce “E-commerce/AffariGenerali”

ment, consentendo così di creare la giusta gerarchia di modelli in grado di riportare nella rappresentazione XML la strutturazione delle informazioni della pagina. Una sezione del wrapper estratto è visibile in figura 5.16). Guardando l’HTML salta all’occhio che all’interno delle tabelle principali, i dati contenuti in ogni data object sono rappresentati mediante l’impiego di una tabella, che ne ospita un’altra per i dati relativi all’immagine del bottone utilizzato per l’aggiunta del documento al carrello. Una volta selezionata con il mouse l’area relativa ad un data object, Lixto ha riconosciuto come simple element tutto ciò che era rappresentato dalla tabella più esterna, quindi anche ciò che riguarda l’immagine del carrello e la relativa tabella. Se le informazioni contenute nella tabella più interna fossero state rilevanti ai fini dell’estrazione, si sarebbe potuto costruire un modello apposito per questa, nel nostro caso i dati

relativi alle immagini del carrello non sono di interesse e quindi sono stati trascurati.

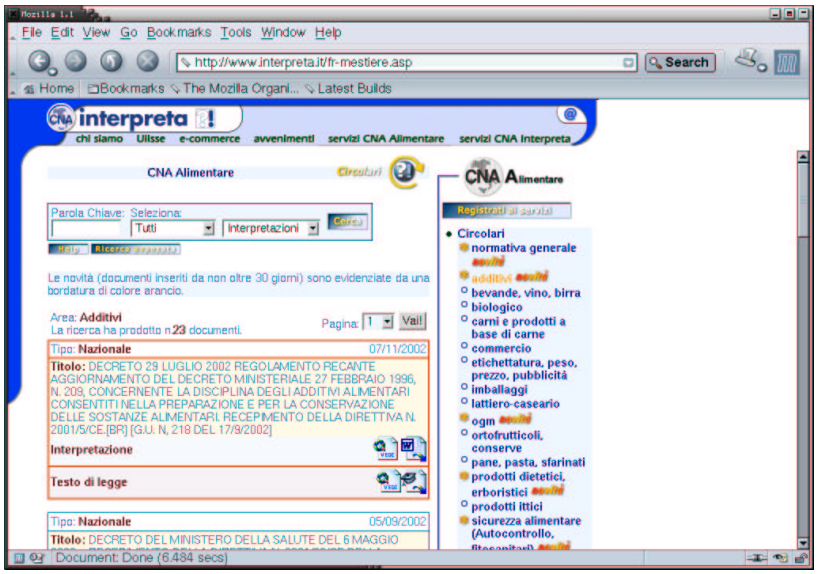

Figura 5.17: www.interpreta.it, "Servizi CNA Alimentare/Additivi"

Per la sezione "Servizi CNA Alimentare", selezionando la voce "Additivi", la pagina che viene restituita (vedi figura 5.17) è composta da data objects ognuno dei quali strutturato mediante una tabella contenente altre tre tabelle di livello più interno che ospitano ciascuna dati relativi ad una sezione differente del data object. Mediante la selezione con il mouse è stato possibile far riconoscere al toolkit la tabella più esterna come simple element, per poi creare un modello differente da associare ad ognuna delle tabelle più interne (come si può vedere in figura 5.18).

Per quanto riguarda la sezione "Servizi CNA Interpreta", selezionando la voce "Affari generali" , si ottiene una pagina che contiene una lista di data object in apparenza simili a quelli visti precedentemente (vedi figura 5.19), dopo ripetuti tentativi di selezione però si è scoperta l'impossibilità di Lixto di individuare un simple element che contenga l'intero data object, a partire dal quale costruire il modello di estrazione. Analizzando il codice HTML si scopre la situazione descritta nella sezione

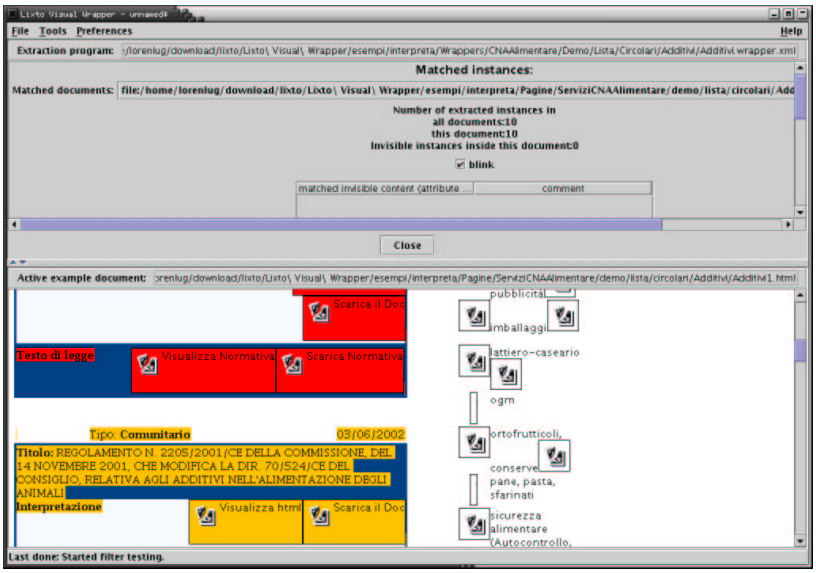

Figura 5.18: Corretto mapping tra data-objects e simple element "Servizi CNA Alimentare/Additivi" del sito www.interpreta.it

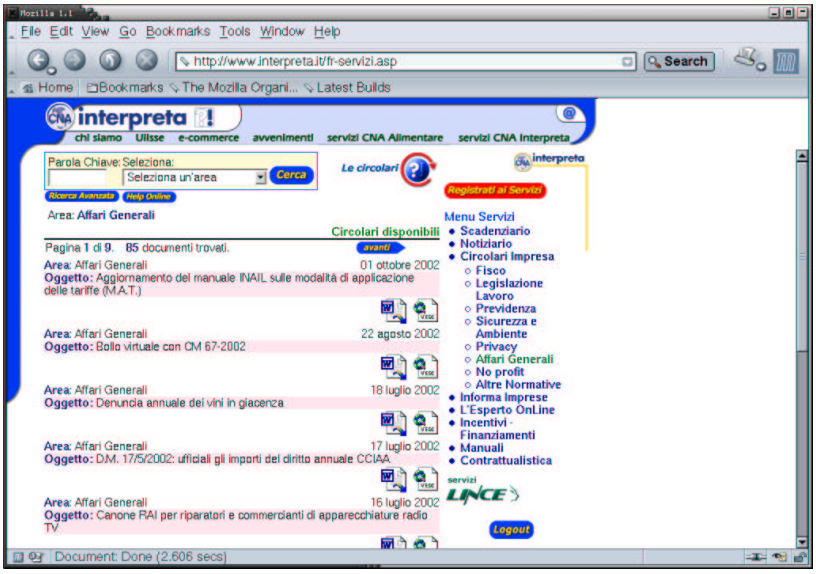

Figura 5.19: www.interpreta.it, "Servizi CNA Interpreta/AffariGenerali"

5.5, qui infatti il concetto più esterno viene dettagliato mediante l'uso di uno o più concetti di livello più interno racchiusi all'interno di tabelle

dello stesso livello unitamente a concetti allo stesso livello del concetto esterno. Il data object di interesse, non essendo riconosciuto come simple element, dovrebbe essere ricostruito mediante l'unione dei concetti a lui relativi, indipendentemente dal loro livello di profondità. In questa situazione è possibile la costruzione di una serie di filtri ognuno dei quali dedicato all'estrazione di una specifica sezione del data object di interesse. Seguendo questa strada si raggiunge un punto critico nel momento di ricollegare i dati estratti dai filtri al singolo data object. Come detto, allo stato attuale Lixto non è in grado di intervenire per riscostruire un data object partendo dai componenti elementari estratti. I data object sono costituiti da un campo "Area" , dalla data, da un campo "Oggetto" e da due immagini rappresentanti il formato in cui è possibile scaricare il documento. Al contrario di quanto accadeva nei casi precedenti in cui è

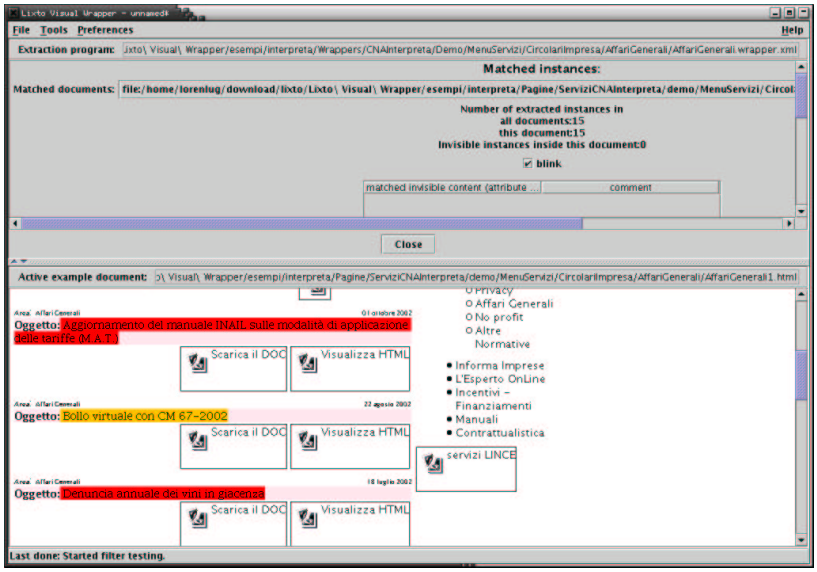

Figura 5.20: Modello di estrazione per la voce "Servizi CNA Interpreta/AffariGenerali" del sito www.interpreta.it

stato possibile isolare ciascun data object in un simple element grazie al fatto che ad ogni data object era riservata una determinata tabella, anche se composta a sua volta da altre tabelle, in questo tipo di pagina alcune

sezioni di dati di ogni data object sono contenute in righe della stessa tabella (i valori del campo oggetto e delle immagini relative ai formati di download), altre, come i valori del campo "Area" e della data sono al contrario memorizzati in una tabella di livello più interno. In questa situazione si è reso necessario scegliere il livello al quale effettuare l'estrazione, ed essendo l'area e la data assai poco significative se prese singolarmente, la scelta è caduta sulla selezione del valore degli oggetti (vedi figura 5.20).

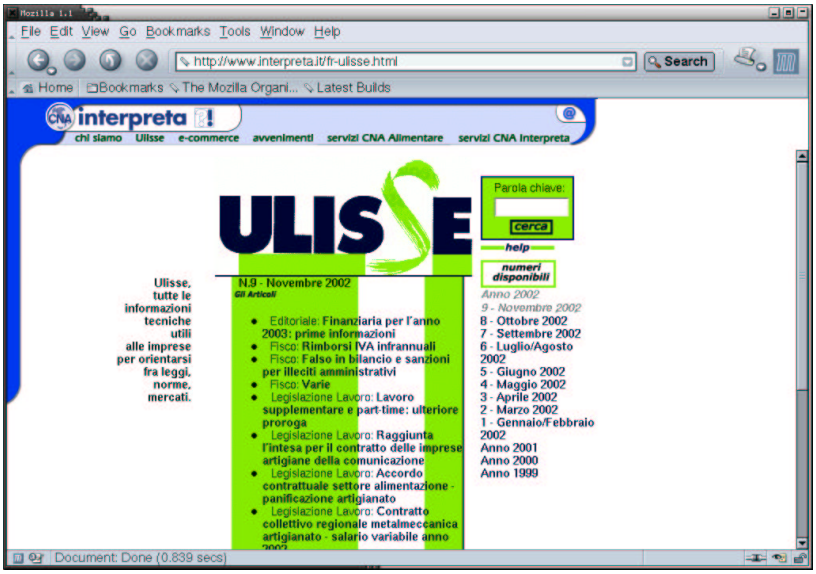

Figura 5.21: Voce Ulisse del sito www.interpreta.it

L'ultima voce del sito analizzata è stata "Ulisse". Le pagine di queste sezioni erano costituite da liste numerate i cui elementi contenevano l'argomento trattato (vedi figura 5.21) e l'ipertesto contenente il nome dell'articolo e ovviamente il link alla pagina vera e propria. Le selezioni effettuate hanno prima messo in evidenza la lista, dopodichè è stato selezionato un singolo elemento che contenendo un ipertesto è stato identificato senza difficoltà come un simple element. Una volta che si è stati in grado di identificare il singolo elemento della lista, è stato messo a punto un modello per la parte iniziale ossia quella riguardante l'argomento ed

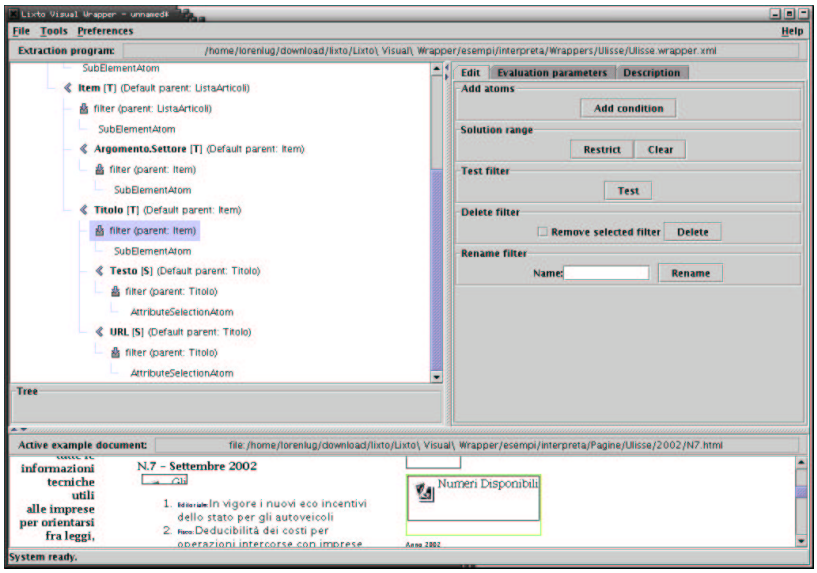

Figura 5.22: Sezione del wrapper generato per il settore "Ulisse" del sito www.interpreta.it

un secondo per quella riguardante la stringa di hyperlink vera e proprio. All'interno di quest'ultimo, così come è stato fatto precedentemente, si sono poi creati due modelli di tipo stringa per estrarre la parte testuale e l'URL (la sezione del wrapper che estrae questi dati è mostrata nella parte destra di figura 5.22).

L'ultimo sito su cui sono stati eseguiti test è tessilmoda.com. Per la HomePage (vedi parte sinistra di figura 5.23) sono stati messi a punto una serie di wrappers, ognuno con il compito di estrarre una parte differente delle informazioni della pagina, che si presentava composta da una serie di liste. L'unica anomalia riscontrata in questo lavoro è stata legata al fatto che pur essendo le liste tutte composte da hyperlink ad altre sezioni del sito o ad altri siti, non sempre gli elementi venivano riconosciuti come tali e quindi non sempre è stato possibile estrarre i relativi URL. Un esempio di anomalia in questo senso è costituito dai menù "Bacheche" e "Documenti", pur essendo entrambi strutturati allo stesso modo, con tag <a> contenenti al loro interno un attributo href al

Figura 5.23: Home page del sito www.tessilmoda.com

quale viene attribuito l'URL al quale si vuole saltare, nel primo caso gli elementi della lista vengono riconosciuti come hyperlink ma tentando di eseguire lo stesso procedimento sulla sezione Documenti, si scopre che gli elementi vengono riconosciuti come semplice testo, senza la possibilità di poterli caratterizzare ulteriormente attraverso il loro indirizzo URL. Il motivo del problema non è da ricondursi al proccesso di generazione del wrapper quanto piuttosto al browser interno a Lixto, è questo infatti che ignora gli hyperlink presenti nelle sezioni "Documenti", "New Entry", "Link Utili", "Informazioni" ed in alcune parti di "Settori e Attività", "Novità del sito".

Ognuna delle liste citate fornisce una strutturazione propria ai dati restituiti, alcune organizzano i dati mediante l'utilizzo di tabelle, altre mediante strutture ad albero, nelle cui foglie sono presenti le informazioni. Uno dei casi di strutturazione del primo tipo è dato dalle pagine della voce "Immobili/Attività" nella sezione "Bacheche" (vedi figura 5.24). In questo caso le informazioni relative a ciascun data object sono allocate all'interno di una tabella che a sua volta contiene altre due tabelle, la pri-

Figura 5.24: Voce "Immobili e Attività" del sito www.tessilmoda.com

ma per l'intestazione verticale, contenente la ragione sociale, la data e il numero dell'inserzione, mentre la seconda contenente i parametri veri e propri dell'inserzione, quali: numeri di telefono, referente, articolo in vendita etc.. Selezionando con il mouse la zona corrispondente alla tabella più esterna, è stato possibile individuare un simple element, ripetendo le selezioni per le zone corrispondenti alle tabelle più interne sono stati identificati altri due simple element, uno per l'intestazione della tabella più esterna ed uno per il contenuto. Si è riusciti, in questo modo, ad ottenere un'ottima immagine di quella che è la struttura dei dati, che facilita non poco l'arricchimento semantico necessario per una buona trasformazione dell'HTML in XML. Una sezione del wrapper generato è riportata in figura 5.25. Lavorando su questa pagina così come con quelle restituite dalle voci presenti in "Documenti", si è registrata un'anomalia nell'uso dello strumento: all'intestazione delle tabelle che contengono i data object, come è stato detto precedentemente, è riservata una tabella. Questa tabella è composta da due righe e tre colonne, nella prima riga sono presenti i nomi dei campi: "Ragione Sociale", "Data del

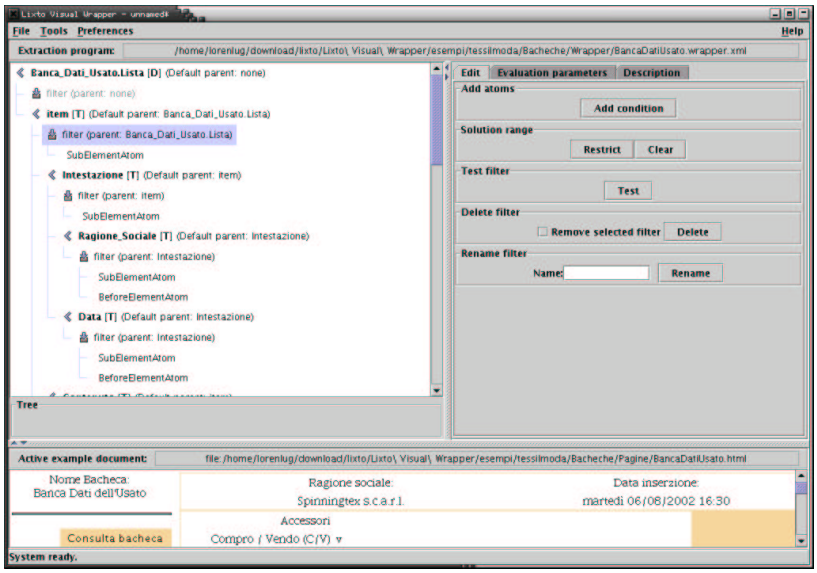

Figura 5.25: Sezione del wrapper generato per la pagina "Servizi CNA Interpreta/AffariGenerali" del sito www.interpreta.it

Documento" e "Numero del Documento", nella seconda riga i rispettivi valori. Il valore del terzo campo dell'intestazione, ossia il numero del documento non viene caricato dal browser di Lixto e non è quindi possibile creare un modello adatto all'estrazione. Dal confronto tra le due maschere in figura 5.24 e 5.25 è possibile farsi un'idea del problema.

Per quanto riguarda la sezione "New Entry", analizzando ad esempio la pagina che si ottiene dalla voce "Alan Maglificio" (vedi parte figura 5.26), si vede che è presente un singolo data object avente struttura ad albero. Leggendo il codice HTML che sta dietro la pagina, è possibile constatare che questo tipo di struttura è ottenuto mediante l'uso di una tabella, nella quale i vari rami sono ottenuti mediante l'utilizzo di liste non ordinate. In queste condizioni è stato possibile selezionare come simple element iniziale tutto l'albero, quindi solo uno degli elementi della lista più esterna ed il suo contenuto, fino ad arrivare alla selezione di uno degli elementi della lista più interna che rappresenta una foglia dell'albero. Lavorando su queste strutture, con l'ausilio degli strumenti di

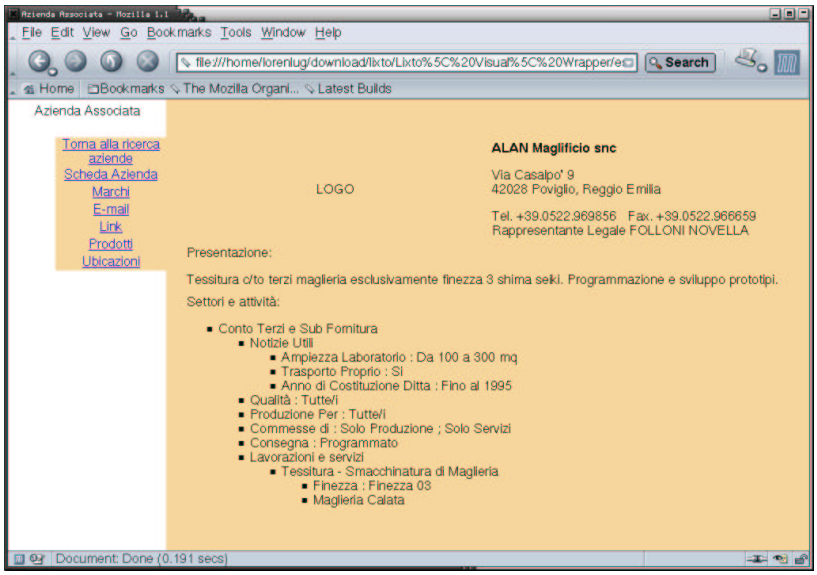

Figura 5.26: Voce "Alan maglificio" del sito www.tessilmoda.com

raffinamento della selezione a disposizione dell'operatore, è stato sempre possibile individuare il simple element desiderato. Uno dei motivi del successo del toolkit su questo tipo di strutturazione della pagina è legato al fatto che le liste sono contenute nella stessa riga della tabella nella quale è contenuto il nome del campo che da inizio alla strutturazione ad albero. Questo permette al tool di ignorare l'annidamento delle liste e dei loro elementi per arrivare al riconoscimento di un simple element (vedi figura 5.27). A giustificazione di questo successo vale inoltre la pena ricordare che la pagina in esame era costituita da un solo data object, la situazione si complica se viene usata la strutturazione ad albero per i dati anche quando i data object restituiti sono più di uno, come nel caso della pagina restituita in seguito alla selezione della voce "Commercio" nella sezione "Settori e Attività" (vedi parte di sinistra di figura 5.29). In questo caso ogni data object è composto da un'intestazione rappresentata da un collegamento ipertestuale al nome dell'azienda e dalla relativa struttura ad albero che ne rappresenta il contenuto. Il problema in questo caso è che sia le intestazioni che i contenuti sono allocati come righe

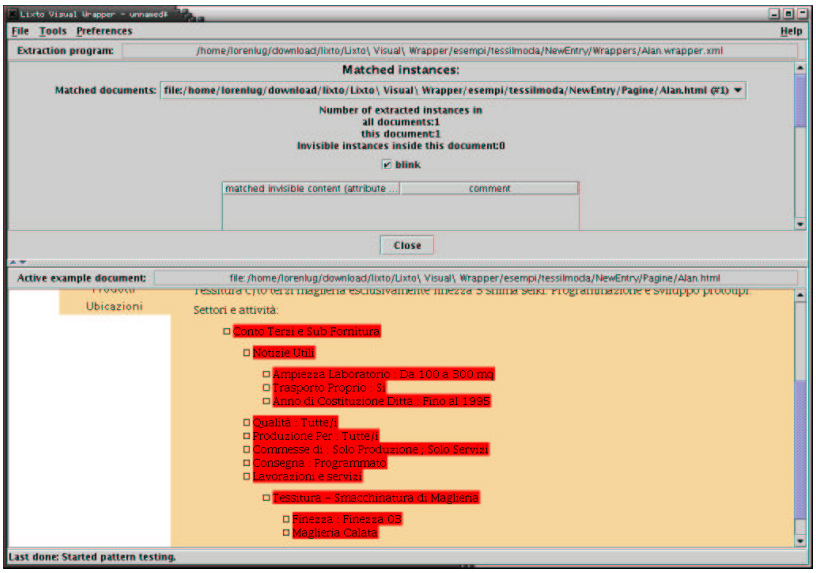

Figura 5.27: Simple element evidenziato nella pagina "Servizi CNA Interpreta/AffariGenerali" del sito www.interpreta.it

Figura 5.28: Voce "Commercio" del sito www.tessilmoda.com

di un'unica tabella e allo stesso livello (vedi parte destra di figura 5.29). Questo comporta l'incapacità di Lixto di individuare un simple element

Figura 5.29: www.tessilmoda.com, codice sorgente della pagina "Commercio"

costituito dall'intestazione e dal relativo contenuto. Vale il discorso fatto precedentemente per cui si può considerare l'opportunità di mettere a punto filtri che recuperino separatamente i due tipi di elementi, senza però avere la possibilità di collegarli, oppure si può mettere a punto un modello ed il relativo filtro per uno dei due elementi, in questo caso la scelta cadrebbe sul nome dell'azienda e sull'URL corrispondente, dal momento che l'insieme dei contenuti senza un'azienda a cui collegarli oltre ad essere ripetitivi non avrebbero alcun significato (la prima è stata la soluzione adottata).

I problemi descritti non sono ovviamente gli unici che si sono incontrati nelle operazioni di "wrapping" dei siti, ma forniscono interessanti spunti su quelle che possono essere le situazioni tipo nelle quali ci si viene a trovare nel momento in cui si analizza una pagina con l'intenzione di generare un wrapper adatto all'estrazione dei dati contenuti.

Capitolo 6

Il Sistema LAPIS

L'analisi di questo toolkit viene motivata dalla volontà di verificare quanto di vero ci sia nelle critiche mosse agli strumenti di estrazione dei dati la cui logica di funzionamento si basa su espressioni regolari [42]. Alcune di queste critiche nascono dalle difficoltà incontrate nelle fasi di lettura e di successiva comprensione della sintassi delle espressioni regolari e delle grammatiche che vengono utilizzate per processare automaticamente i testi. Sarebbero questi alcuni dei motivi per cui, secondo gli sviluppatori di Lapis, gli strumenti di questo tipo non si sono guadagnati il consenso del mercato. Per le stesse ragioni sono stati sviluppati programmi che forniscano procedure intuitive per la manipolazione automatica del testo. Seguendo queste convinzioni il sistema LAPIS è stato sviluppato con l'intento di offrire un ampio intervallo di possibilità per la manipolazione del testo. Essendo le pagine HTML niente più che semplici documenti di testo, LAPIS può essere utilizzato per estrarre dati mediante la generazione di wrappers. Questo toolkit è attualmente uno dei pochissimi sistemi di generazione di wrappers non commerciali e open source che sia ancora sviluppato e supportato.

6.1 Caratteristiche del programma LAPIS

LAPIS è essenzialmente un browser Web con un editor di testo integrato che può essere programmato per processare e manipolare documenti di testo caricati all'interno del programma [43]. I singoli passi di manipolazione manuale sono contenuti in uno script eseguibile scritto in un linguaggio di scripting proprietario. Per iniziare il processo di manipolazione automatica dei dati, lo script creato con LAPIS è attivato all'interno del toolkit. Usato come browser Web, LAPIS è in grado di scaricare tutti i files che hanno un URL valido, attraverso i protocolli FTP o HTTP. Le pagine HTML possono anche essere mostrate come documento di testo mostrando l'intero codice sorgente. In alternativa, la pagina può essere presentata come una pagina Web standard come in un normalissimo browser [44]. All'interno di LAPIS è presente un linguaggio per il riconoscimento di modelli chiamato "*text constraints*", la cui sintassi risulta più intuitiva rispetto a quella delle espressioni regolari e che consente all'utente di specificare e circoscrivere i dati richiesti in un determinato documento.

Con l'aiuto degli strumenti incorporati all'interno del programma, è possibile manipolare regioni precedentemente messe in evidenza all'interno del documento. I dati possono essere cancellati, filtrati, ordinati o estratti [44]. Ci sono alcune possibilità per la definizione dei dati di interesse all'interno di un documento. L'utente può applicare manualmente il linguaggio text contraints per definire uno specifico modello per tutti i dati richiesti, oppure può evidenziare solo poche regioni di testo interessanti e LAPIS tenterà di generare automaticamente un modello adatto [45]. Per documenti HTML complessi verrà utilizzata una combinazione di entrambi i metodi. Dopo che LAPIS ha generato un modello, l'utente può intervenire manualmente se necessario, per raffinare il risultato. Ulteriore assistenza è fornita dal toolkit attraverso il testing dei modelli sviluppati al fine di individuare eventuali errori. Questo compito viene svolto da LAPIS evidenziando separatamente le aree di testo che si

trovano all'interno della definizione valida di un determinato modello, ma differiscono nella struttura da quelle regioni che sono più comunemente frequenti [46]. Inoltre LAPIS ha una "*command shell*" che permette al programma di processare comandi nel linguaggio di scripting Tcl. Questa caratteristica consente l'invocazione tramite riga di comando di programmi esterni direttamente dall'interno di LAPIS e in maniera analoga di trasferire dati estratti direttamente ad un programma esterno per ulteriori manipolazioni.

6.2 Text Constraints

Lo sviluppo di wrappers all'interno di LAPIS è basato sulla definizione di regioni di testo all'interno di un documento. I modelli che sono stati inferiti dalle parti di testo o dati evidenziate, sono definiti utilizzando il linguaggio *text constraints* appositamente implementato. Uno degli obiettivi che gli sviluppatori di questo linguaggio si sono prefissi è stata la facilità di utilizzo, raggiunta mediante l'impiego di termini semplici della lingua inglese e la trasparenza dai dettagli implementativi del linguaggio, in modo che l'utente non debba essere a conoscenza delle complesse procedure di programmazione. L'esempio in figura 6.1 mostra l'effetto della messa in evidenza del semplice modello text constraints:
Number.ScientificNotation just before " Euro"
su un codice HTML estratto da una pagina Web di confronto di prezzi per uno specifico articolo, la parte evidenziata costituisce il prezzo degli articoli restituiti dalla ricerca.

In LAPIS è installato più di un parser, tra gli altri anche uno HTML con il quale si può accedere direttamente agli elementi in un documento HTML. É prevista la possibilità di specificare delle restrizioni all'interno dei modelli text constraints, in modo che le regioni contenenti dati di interesse possano essere ridotte utilizzando termini come *contains, first, last, just, before* e *after*. Per esempio, la prima colonna della prima tabella

```
<TD> Sony Cybershot DSC-P 50 <BR> Inc. Memory Stick </TD>
<TD align=right> <B> 429.99   ;Euro </B></TD>
<TD align=middle> Primustronix.de</TD>

<TD>Sony DSC-P50 PC/MAC <BR> 2.1 Megapixel, 3x Zoom </TD>
<TD align=right> <B> 452.55   ;Euro </B></TD>
<TD align=middle>avitos.com </TD>
```

Figura 6.1: Evidenziazione di regioni mediante l'utilizzo di Text Constraints

all'interno di una pagina HTML dovrebbe essere evidenziata utilizzando il modello:
text in first [td]
In questo modo possono essere sviluppati e aggiunti nuovi modelli alla libreria LAPIS permettendo all'utente di identificarli con un nome [46]. Per LAPIS non fa differenza se le regioni contenenti testo di interesse sono evidenziate attraverso i parsers, le espressioni regolari o mediante selezione con il mouse, o ancora attraverso una combinazione di questi tre metodi.

6.3 Estrazione Automatica dei Dati

Una volta che è stato creato un modello adatto ad un determinato documento HTML utilizzando *text constraints* e che i passi individuali di manipolazione del testo o dei dati sono stati definiti, è necessario combinare questi passi in una procedura automatica. LAPIS offre un tools accessorio per questo scopo. Il toolkit registra i passi individuali di un utente durante l'esecuzione della processazione e della manipolazione dei dati per poi generare uno script eseguibile. Uno script di questo tipo può contenere, tra le altre cose, indirizzi URL, istruzioni sul completamento e l'accettazione di forms HTML, modelli *text constraints* , invocazioni di programmi esterni oppure comandi di manipolazione del testo. Attraverso l'invocazione di tale script, la manipolazione di uno o più do-

cumenti, può essere direttamente trattata da LAPIS, il quale processa le istruzioni contenute nello script sequenzialmente [43].

6.4 Esempio di Applicazione

Per dimostrare la possibile applicazione di LAPIS per la generazione di wrappers, in questa sezione verrà presentato un esempio nella forma di un confronto tra prezzi. Questo confronto si basa sull'estrazione di informazioni sui prezzi di uno specifico prodotto ottenute da vari meta comparatori di prezzi presenti sul Web, uno dei quali è WebDeShopping, e combina queste informazioni in un nuovo confronto tra prezzi. La procedura è identica a quella seguita normalmente da un meta motore di ricerca. Per raggiungere l'obiettivo, verranno scaricate le pagine di risultati provenienti da tre differenti meta comparatori di prezzi online e verranno estratti i dati di interesse, questi ultimi verranno nuovamente combinati in una nuova tabella HTML che viene appositamente creata e ordinata e infine presentata all'interno di un browser standard. I modelli o wrappers per le pagine di risultato individuali devono essere per prima cosa creati usando *text constraints* e successivamente devono essere memorizzati nella LAPIS parser library (vedi Figura 6.2).

Lo script per la processazione automatica è sviluppato utilizzando LAPIS (vedi figura 6.3). Ognuna delle pagine risultanti dei comparatori di prezzi online è scaricata nella finestra del browser LAPIS. Vengono applicati i wrappers sviluppati precedentemente per ognuno dei comparatori di prezzi, quindi i dati estratti vengono memorizzati provvisoriamente all'interno di file temporanei separati. Per processare ulteriormente i dati estratti, uno speciale simbolo unico ("#") viene usato per distinguere i data objects all'interno dei files temporanei.

Quando tutte le pagine di risultato sono state processate, LAPIS invoca il *mediatore*, in questo caso un programma Java esterno. Il mediatore ha il compito di combinare tutti i risultati estratti in una tabella all'interno di una nuova pagina HTML. Questa tabella viene restituita a

```
prefix Wrapper
|  #
|  # WebDeShopping
|  #
 ---prefix WebDeShopping
    |--PriceWebDe is Business.Number.ScientificNotation
    |  just before “ Euro”
    |--ProductWebDe is Text in [td] just before [td]
    |                            contains PriceWebDe
    |--ShopWebDe is Link contains “Haendler”
    |              in row contains PriceWebDe
     --WrapperWebDe is either PriceWebDe
                       or ProductWebDe
                       or ShopWebDe
```

Figura 6.2: Wrapper per il comparatore di prezzi WEB.DE

```
doc -type text
[http://shopping.web.de/Suche/Fotografie+%26+
 Optik/Fotografie/Digitale+Fotografie/
 Digitalkameras/?pricef=&prict=&sort=&search=
        sony+dscp50&id=-L0Sm10**sga700]
extract -separatedby “ # “{Wrapper.
                             WebDeShopping.
                             WrapperWebDe}
save /home/lorenlug/LapisFiles/ExtractedData/WebDe_1.ref
...
doc -type html [java -cp /home/lorenlug/LapisFiles/
                JavaFiles/Mediator]
sort row -by {4th cell in row} -order numeric
browse
```

Figura 6.3: Frammento dello script LAPIS utilizzato per automatizzare il processo di estrazione

LAPIS ridirezionando lo standard output sulla riga di comando. La caratteristica di poter costruire tabelle HTML ordinate consente a LAPIS di riordinare le righe della tabella in ordine di prezzo, riducendo il tempo necessario per la conversione dei prezzi in un tipo di dati adatto all'ordinamento all'interno del mediatore. Per finire, LAPIS invoca un browser

internet standard per trasferire i documenti HTML da lui prodotti. Cliccando sui links contenuti nelle pagine risultanti, l'utente può navigare all'interno dei comparatori di prezzi o, in alcuni casi, può andare dritto verso i negozi online con le offerte più attraenti.

6.5 Considerazioni sul sistema Lapis

LAPIS è stato originariamente sviluppato per automatizzare l'editing e la manipolazione di documenti testuali. Nelle sezioni precedenti è stato visto come l'uso e la combinazione di alcune delle caratteristiche implementate in questo strumento consenta la generazione di wrappers per documenti HTML. È un fatto però che manchi di certe funzionalità che sono implementazioni standard nei generatori di wrappers commerciali. Queste includono, per esempio, la processazione di comandi JavaScript, i quali vengono usati attualmente in una grande quantità di documenti HTML. A causa dell'assenza di questa caratteristica, i documenti sono a volte presentati in maniera incompleta o inadeguata nella finestra del browser di LAPIS. Per questo motivo, l'utente è spesso costretto a lavorare con il codice sorgente HTML all'interno della finestra per l'editing del testo di LAPIS, al fine di generare modelli *text constraints* adatti. Altri problemi sono creati dal mancato supporto della gestione dei *cookies*, requisito fondamentale per poter scaricare alcuni tipi di pagine Web. In aggiunta a tutto ciò è quasi impossibile per un utente non possedere alcune capacità di programmazione per integrare correttamente i wrappers sviluppati e gli scripts per l'automazione all'interno delle librerie di LAPIS.

Capitolo 7

Analisi Comparativa di Generatori di Wrappers

L'analisi di XWRAP Elite ed i risultati dei test sullo strumento hanno orientato la ricerca verso soluzioni in grado di continuare nella stessa direzione di totale automazione dei processi di generazione dei wrappers. Fra i sistemi individuati lo studio si è concentrato su quello che sulla carta è il più promettente: RoadRunner sviluppato dall'Università di Roma. Questo approccio garantisce una completa automatizzazione di tutte le fasi di generazione del wrapper e della relativa estrazione dei dati.

Non potendo testare lo strumento sui siti di riferimento, così come era stato fatto con XWRAP Elite, non è stato possibile smentire le critiche che vengono mosse a questo tipo di soluzione, la maggior parte delle quali riferite al fatto che l'estrazione completamente automatica è costretta a basarsi sulla struttura del documento, creando wrappers molto vulnerabili nei confronti dei cambiamenti delle pagine su cui viene effettuata l'estrazione. Per questo motivo lo studio si è rivolto a quella che fondamentalmente è l'alternativa a questo filone di ricerca: la generazione assistita da un supervisore. Tra gli strumenti presi in considerazione, quello che ha suscitato maggior interesse è stato il sistema Andes della IBM che si basa sull'utilizzo di espressioni regolari e grammatiche. Come ogni soluzione che si rispetti, non si può pensare che l'approccio

adottato dal sistema Andes sia esente da critiche, secondo alcuni infatti [42] l'uso delle espressioni regolari per realizzare un generatore di wrappers introduce difficoltà di utilizzo troppo grandi per poter sperare in una buona diffusione di queste soluzioni. Sono quindi state valutate le alternative a questo approccio ed è stato individuato LAPIS per quanto riguarda i sistemi non commerciali, la cui logica di funzionamento si basa sull'utilizzo di un linguaggio di scripting dedicato di nome text constraints. Tra i sistemi commerciali un'alternativa è stata trovata in Lixto della Lixto Software. Esistono diverse altre soluzioni in circolazione per l'estrazione dei dati, ognuna basata su un differente modello teorico, che per brevità non sono state riportate estesamente in questa tesi anche se oggetto di studio. Ci si limiterà quindi ad una breve analisi comparativa.

7.1 Toolkits per la Generazione di Wrappers

Un wrapper potrebbe essere sviluppato manualmente, in un determinato linguaggio di programmazione ed usando espressioni regolari. Questo approccio entra inevitabilmente in crisi nel momento stesso in cui è richiesto un numero elevato di wrappers, circostanza nella quale viene motivato l'uso di quelli che vengono chiamati toolkits, strumenti in grado di generare wrappers completi sulla base di parametri forniti dall'utente, per una determinata sorgente di dati. Una delle caratteristiche più importanti dei wrappers generati è il formato nei quali i dati estratti possono essere esportati. Nel caso in cui sia possibile effettuare la conversione dei dati estratti in formato XML, questi potranno essere importati e processati da un grande numero di applicazioni software.

I toolkits per la generazione di wrappers possono differenziarsi tra loro per una lunga serie di caratteristiche. Possono essere classificati in base ai loro formati di output, ai tipi di interfaccia, alle capacità di navigazione del Web, all'uso di interfacce grafiche (GUI) e molto altro ancora.

Esulando dalle caratteristiche implementative, in questo capitolo si

effettuerà una semplice divisione in base alla disponibilità di tipo commerciale piuttosto che puramente accademica.

I programmi di generazione di wrappers all'interno di entrambe queste categorie presentano alcune differenze per quanto riguarda il tipo di interazione con l'utente. Alcuni di questi strumenti consentono l'interfacciamento unicamente tramite la riga di comando e richiedono routine sviluppate in un linguaggio di scripting predeterminato. Questi linguaggi di scripting per lo sviluppo di wrappers sono utilizzati all'interno di editors di testo standard e possono essere visti come alternative specifiche all'applicazione di linguaggi general-purpose come Perl o Java. La maggior parte dei toolkits offre una GUI (Graphic User Interface), e alcuni prevedono la possibilità di evidenziare attraverso il mouse i dati di interesse all'interno di un documento HTML, a partire da queste selezioni il programma genererà un wrapper. Alcuni toolkits combinano la GUI con la riga di comando, in modo da concedere al progettista la possibilità di modificare il wrapper generato automaticamente dal sistema in base agli esempi etichettati forniti dall'utente. Il fatto che siano necessarie frequenti correzioni o meno, dipende dagli algoritmi sottostanti e dalla maturità funzionale del toolkit.

Nelle tabelle che seguono sono elencati in ordine alfabetico i toolkits per la generazione di wrappers attualmente in circolazione, alcuni di questi sono stati sviluppati e ne viene fornito il codice sorgente per poter effettuare prove e migliorie, altri sono utilizzabili unicamente online, altri ancora non vengono più supportati ed è quindi impossibile sottoporli ad una fase di testing. Nella Tabella 7.1 viene fornita una lista di quelli non commerciali, nelle Tabelle 7.2 e 7.3 di quelli commerciali. Nella sezione successiva sarà possibile trovare una brevissima descrizione per ognuno dei toolkits non commerciali, per poi concludere mettendo in evidenza alcune delle caratteristiche più sofisticate dei vari programmi.

7.2 Breve descrizione delle caratteristiche di ognuno dei toolkits non commerciali

- Araneus: Comprende al suo interno strumenti quali Minerva ed EDITOR, fa uso di un modello dei dati chiamato ADM (Araneus Data Model) derivato direttamente da ODMG. Minerva combina un approccio dichiarativo basato su una grammatica definita in EBNF, con caratteristiche tipiche di linguaggi di programmazione procedurali. Per ogni documento viene messo a punto un set di definizioni per la struttura dei simboli non terminali della grammatica. Minerva viene completato da un linguaggio per la ricerca e la ristrutturazione di documenti chiamato EDITOR. Viene utilizzato all'interno di RoadRunner. Non ci sono informazioni sul fatto che il tool venga ulteriormente sviluppato o meno, ultima versione Giugno 99.

- BYU: Le regole di estrazione sono basate su "ontologie", questa soluzione consente di generare wrappers estremamente stabili grazie alla loro indipendenza dalla struttura del documento. Le ontologie sono però complesse e molto difficili da mettere a punto. Ultima versione Ottobre '98.

- Commix: Toolkit cinese basato su ontologie e data base con templates. Sulla base delle parole chiave, in una parte dell'HTML viene utilizzato il template appropriato. Non ci sono informazioni sul fatto che il toolkit venga sviluppato ulteriormente. Ultima versione Gennaio 2002.

- DEByE: Il tool è ancora in fase di sviluppo. È disponibile una versione BETA. La generazione dei wrappers avviene attraverso una serie di esempi forniti dall'utente, questi identificano una struttura per gli oggetti di interesse quindi una pagina Web viene creata attraverso porzioni di dati che implicitamente sono conformi a tale

struttura. La struttura è fornita in accordo con un set di primitive di modellazione (es: tuple, liste, etc.) che sono conformi al modello dei dati sottostante. Viene utilizzata una tabella innestata come paradigma visuale. Ultima versione Marzo 2002.

- InfoExtractor: Integrazione di vari Tools in un programma Java al fine di ottenere la struttura del documento di una parte dell'HTML. Non ci sono informazioni sul fatto che il tool sia sviluppato o meno, ultima versione Giugno '98.

- Jedi: Java API, è possibile la compilazione e l'esecuzione di scripts Jedi da parte di programmi Java. Ultima versione Giugno '98.

- LAPIS: Tool dotato di un browser integrato ed espandibile con parser esterni, quindi relativamente versatile. Sviluppato originariamente come editor di testo è stato adattato alla generazione di wrappers. Consente l'esecuzione di script in codice Tcl. Ultima versione Aprile 2002.

- NoDoSe: Estrazione automatica dell'HTML da parti ugualmente strutturate e generazione del relativo wrapper. Fornisce all'utente un'interfaccia per etichettare gli esempi. Non ci sono informazioni se il wrapper sia ancora sviluppato, ultima versione Marzo '98.

- RoadRunner: Il progetto è basato sul toolkit Araneus per la generazione di wrappers. Il toolkit non è ancora disponibile.

- Scout: Progetto derivante da una tesi di laurea terminata nel '97.

- SoftMealy: Toolkit Taywanese per la generazione di wrapper. Questo tool così come STALKER e VIENNA si affida a caratteristiche di formattazione che implicitamente delineano la struttura delle sezioni di dati trovate. Sono ancora disponibili solo moduli individuali del toolkit. Ultima versione Gennaio 2000.

- TSIMMIS: Sviluppato tutt'ora dalla IBM in collaborazione con l'Università di Stanford. Possibile solo l'installazione su Unix. Disponibili solo i wrappers realizzati.
- WebL: Linguaggio di scripting messo a punto dalla Compaq che rende possibile la gestione dei forms, generazione di Web robots e di regole di estrazione. Ultima versione Giugno '99.
- WebSphinx: Navigatore Web (crawler) con estrattore dei dati. Non ci sono informazioni sul fatto che il tool sia stato ulteriormente sviluppato, l'ultima versione risale al Gennaio 2000. La stessa equipe si occupa del progetto LAPIS.
- VIENNA (WIEN): La generazione dei wrappers si basa sugli esempi forniti dall'utente in cui sono state etichettate le informazioni da estrarre, l'induzione dei wrappers è ottenuta attraverso euristiche, a differenza della maggior parte degli altri approcci questo sistema non usa un albero dei tags HTML. Non si hanno informazioni sulla situazione dello sviluppo.
- XWRAP: Toolkit disponibile solo online. I wrappers generati sono disponibili accompagnati dal codice sorgente Java e possono essere implementati localmente. Gli sviluppatori stanno tutt'ora lavorando sul codice, entro la fine del prossimo anno è prevista una versione scaricabile per i test in locale.

La libera disponibilità di toolkits proviene quasi senza eccezioni da progetti sviluppati in ambito universitario. WebL è l'unico che viene fornito liberamente da una casa di produzione di software. Fatta eccezione per DEByE (Data Extraction by Example), LAPIS, RoadRunner e XWRAP, gli atri progetti sono stati completati e manca qualunque tipo di supporto tecnico e di ulteriore sviluppo dei programmi. Alcuni di questi strumenti non forniscono formati di output diffusi e quindi sono limitati nel loro utilizzo e richiedono l'implementazione di speciali interfacce per l'interazione con applicazioni esterne.

XWRAP è un'eccezione nella categoria dei toolkits non commerciali. A causa delle regole sulle licenze infatti, può essere utilizzato solo online e fornisce unicamente il codice sorgente Java dei wrappers generati, al fine di poterli utilizzare come applicazioni locali. Il formato di output dei dati estratti è XML [36]. DEByE è anch'esso capace di esportare i dati estratti in XML. Utilizzando solo un piccolo numero di esempi sui dati forniti dall'utente, DEByE prova a riconoscere automaticamente dati strutturati in maniera simile all'interno del documento HTML [47].

Passando ai toolkits commerciali, quasi tutti quelli analizzati sono in grado di fornire in output i dati estratti in formato XML, mentre tutti, senza eccezione, forniscono un'interfaccia grafica GUI e sono utilizzati attraverso il mouse.

Uno dei primi toolkits commerciali è stato W4F della Tropea Inc., dotato di un linguaggio di scripting dedicato ed offriva solo piccole interfacce grafiche chiamate wizard, come supporto aggiuntivo. Le regole di estrazione dei programmi erano basate sulle definizioni di percorsi all'interno della struttura di una pagina HTML. Queste definizioni dei percorsi portavano alla posizione dei dati di interesse [48].

Toolkits più sofisticati come quelli offerti da *Caesius Software, Crystal Software, Fetch Technologies, Item Field, Kapow Technologies, Lixto e Wisosoft,* non richiedono di evidenziare tutti i dati che devono essere estratti. Un piccolo numero di esempi sono sufficienti per questi progammi per generare regole di estrazione adatte allo scopo. Lixto, per esempio, può essere utilizzato dagli utenti anche se questi ultimi non hanno nessuna conoscenza di programmazione in HTML, dato che non richiede all'utente di lavorare con il codice sorgente HTML [49]. In questo modo l'utente può ottenere efficientemente e velocemente un wrapper completamente funzionale. *Fetch Technologies* va un passo avanti e genera wrappers che entro certi limiti sono in grado di rimanere al passo con cambiamenti strutturali dei siti Web, questo è consentito da una serie di controlli sulle funzionalità dei wrappers generati e sulla capacità di implementare automaticamente correzioni se necessario [50]. Un toolkit simile a que-

st'ultimo e in grado di integrarne tutte le funzionalità, è *RoboSuite* della *Kapow Technologies.*

La stabilità e l'affidabilità dei wrappers dipende fortemente dai metodi di estrazione dei dati applicati dai diversi toolkits. Per esempio, i toolkits che si affidano unicamente alla struttura dell'HTML per identificare dati di interesse sono molto vulnerabili ai cambiamenti dei siti Web e richiedono quindi frequenti aggiustamenti ai wrappers. Un modo per ottenere maggiore robustezza è quello di combinare diversi metodi, un esempio può essere dato dalla combinazione delle tecniche di riconoscimento della struttura dell'HTML e dei modelli basati sul contenuto. L'affidabilità di un wrapper non necessariamente dipende dal modo in cui l'utente interagisce con il toolkit ossia dal fatto che nel processo di generazione il progettista tenga in considerazione i criteri costruttivi della pagina. Se, per esempio, il toolkit genera wrappers basati sul riconoscimento del modello o del linguaggio naturale, la messa in evidenza di esempi di dati all'interno di un browser può portare alla realizzazione di wrappers molto robusti, senza che l'utente sia costretto a lavorare con il codice sorgente HTML. Questo è particolarmente vero se l'utente non dipende completamente dall'applicazione automatica di metodi statistici del toolkit ed è in grado di realizzare o rifinire la messa a punto delle regole di estrazione risultanti.

La maggior parte dei toolkits possono generare wrappers che hanno la capacità di ritrovare un'unica pagina. Alcuni, hanno la capacità di navigare automaticamente le pagine Web (è questo il caso di Andes della IBM) o seguire un ristretto numero di links (è stato visto come Lixto implementi questa possibilità). Questa funzionalità, abilita il wrappers a ritrovare tutte le pagine in una richiesta di ricerca, anche se i dati desiderati sono contenuti in alcuni documenti HTML. In generale, la capacità di seguire automaticamente links è utilizzata per ritrovare tutti i dati richiesti contenuti nelle pagine di un particolare sito Web e non è necessariamente destinata all'implementazione di Web robots che seguano anche links verso siti sconosciuti. Questo studio si è concentrato princi-

palmente sulla fase di estrazione dei dati e solo brevemente si è occupato della fase di ritrovamento del documento.

7.3 Conclusioni

In confronto alla programmazione manuale, l'uso di toolkits per la generazione automatica o semi-automatica di wrappers permette di risparmiare tempo nello sviluppo di wrappers per l'estrazione di dati da pagine HTML.

Toolkits basati unicamente sulla riga di comando possono generare wrappers robusti ma non sono facili da usare e hanno poche possibilità di essere accettati dalla comunità informatica. Le possibili applicazioni di altri toolkits non comerciali sono limitate anche dal ristretto numero di caratteristiche. Solo due dei toolkits non commerciali attualmente disponibili, sono in grado di fornire i dati estratti in formato XML: DEByE e XWRAP. Da quanto emerso dalla sperimentazione diretta sui siti di prova, si è giunti alla conclusione che nel momento in cui sono richiesti un numero limitato di wrappers, toolkits non commerciali sono una buona alternativa alla programmazione manuale.

Per quanto riguarda i tools commerciali, focalizzando l'attenzione sulla facililtà d'uso, le caratteristiche più importanti sono una facile amministrazione dei wrappers generati ed il supporto dei formati di output moderni. Anche in questo caso come in quello dei toolkits non commerciali la robustezza dei wrappers rispetto ai cambiamenti strutturali dei siti dipende dalle regole di estrazione impiegate. Se il wrapper dipende fortemente dalla struttura standardizzata di un documento all'interno di un sito Web e si affida principalmente alla rappresentazione DOM senza combinarla con altri metodi, allora anche piccoli cambiamenti possono causare malfunzionamenti del wrapper.

Al giorno d'oggi una grande quantità di dati è processato attraverso linguaggi di programmazione per il Web e non è contenuto in pagine di puro codice HTML. Per fare un esempio, gli hyperlinks sono spesso ge-

nerati dinamicamente da codice JavaScript. La maggior parte dei toolkits correnti non sono in grado di superare automaticamente questo tipo di difficoltà e richiedono l'assistenza dell'utente per navigare correttamente all'interno dei siti Web. La soluzione di questo problema porterebbe ad un grande miglioramento dell'usabilità dei toolkits.

Passi avanti nel campo dell'intelligenza artificiale possono fornire ulteriori opportunità per lo sviluppo di questi strumenti. Per esempio la possibilità di incorporare applicazioni di individuazione della conoscenza e funzionalità di classificazione automatica, possono aumentare fortemente l'affidabilità e le possibilità di automazione dei wrappers per l'estrazione di dati. Scenari nei quali l'utente definisce semplicemente criteri specifici per le informazioni desiderate e invia agenti intelligenti che si occupino della ricerca di siti precedentemente sconosciuti ed estraggano i dati richiesti, stanno diventando possibili e non sono più fantasia. Per mezzo di questi strumenti sarà possibile raggruppare tutti i prodotti ed i relativi prezzi provenienti da tutti i negozi online al fine di effettuare un confronto dei prezzi su scala mondiale completamente automatico, utilizzando per questo fine uno sforzo relativamente piccolo. I wrappers attualmente generati potrebbero evolvere in più sofisticati agenti di estrazione che permetterebbero la realizzazione di nuovi e più complessi tipi di applicazioni. Per esempio, potrebbe essere possibile ottenere documenti direttamente da varie sorgenti di ricerca in modo che all'utente rimanga solo il compito di selezionare semplicemente un tipo di contenuto a cui è interessato, saranno poi gli agenti ad occuparsi di collezionare automaticamente e di riassumere tutte le informazioni disponibili online relative al contenuto della ricerca. Questo potrebbe drasticamente ridurre il tempo attualmente speso per effettuare manualmente tali ricerche. Le uniche incognite in tutto ciò rimangono da una parte il fatto che non sempre è nell'interesse dei proprietari di siti Web che i propri dati vengano estratti ed utilizzati da terze parti e dall'altra il grande interesse commerciale relativo all'uso di banner e altri tipi di annunci pubblicitari, che vengono trascurati da questi sistemi.

Toolkit	Dati di Output	Java Api	Open Source	Codice Sorgente	Web Crawling	GUI	Editor	Linguaggio di Scripting	Supporto del Tool
Araneus	XML, Text	Sì	solo Api	Java	-	-	-	EDITOR	-
BYU	Text, Tabular	Sì	-	Java	-	-	-	ontologie	-
COMMIX	HTML, XML, Text	No	-	Java	Sì	Sì	-	-	Sì (strumento cinese)
DEByE	XML, Text	No	-	Java	Sì	Sì	-	-	Sì
Info Extractor	struttura di tipo DOM	Sì, JDBC	-	Java	-	-	-	espressioni regolari, regole grammaticali	-
Jedi	XML, Text	Sì	-	Java	Sì	-	-	JEDI	-
LAPIS	HTML, XHTML, XML, Text	Sì	Sì	Java	Sì	Sì	Sì	vincoli testuali	Sì
NoDoSe	XML, TabODL, OEM	Sì	Sì	Java	-	Sì	-	-	-
Road Runner	XML, HTML,Text	Sì	-	Java	Sì	-	-	espressioni regolari	toolkit non ancora disponibile
Scout	HTML, Text	Sì	Sì	Java	-	-	-	espressioni regolari	-
SoftMealy	HTML,Text	Sì	Sì	Java	Sì	Sì	-	-	-
TSIMMIS	HTML,Text	-	Sì	C/C++	-	-	-	-	-
WebL	XML, HTML,Text	Sì	Sì	Java	Sì	-	Sì	WebL	Sì
Web Sphinx	HTML,Text	No	Sì	Java	Sì	Sì	Sì	ORO Matcher regular expression	Sì
WIEN	HTML,Text	No	Sì	Java	-	Sì	-	-	-
XWRAP	XML (con DTD)	Sì	Sì,solo wrappers	Java	Sì	Sì	-	-	Sì

Tabella 7.1: Toolkits non commerciali

Azienda	Toolkit	Versione Demo	Dati di Output	Web Crawl-ling	Interfaccia pro-grammi esterni	GUI	Editor	Linguaggio di Scripting
Caesius Software	WebQL (web query language)	-	Text, HTML	Sì	Sì	Sì	Sì	WebQL
Connotate Technolo-gies	vTag	-	XML	Sì	Sì	Sì	-	-
Crystal Software	TextPipe	Sì	XML,Text	-	Sì	Sì	Sì	Perl, VBScript, JScript, Py-thon
Data Junc-tion	Content Extractor	Sì	XML,Text	Sì	Sì	Sì	Sì	CXL
Extradata Technolo-gies	Unwwwrap	Sì	HTML, Text, Tabelle	Sì, book-marks	Plug-in in MS Int Exp	Sì	-	-
Fetch Tec-nologies	Agent Builder	–	XML,Text	Sì	ODBC, Database	Sì	-	-
Fire Spout	ETL Engi-ne	Sì	HTML, XML, Text	Sì	Sì	Sì	-	-
firstRain	firstRain Studio	-	HTML, XML, Text	Sì	Sì	Sì	-	-
IBM	Intelligent Miner	Sì	HTML, Text	Sì	Sì	Sì	-	-
ItemField	Parser Studio	-	XML	Sì	Sì	Sì	Sì	Parser Script Language
Kapow Technolo-gies	RoboSuite	Sì	HTML, XML, Text, SQL	Sì	XML, SOAP, java Beans	Sì	Sì	espressioni regolari, Text Expressions
Knowmadic	KM Studio	-	XML,Visual Basic Object	Sì	Sì	Sì	-	-
Lencom Software	Visual Web Task	Sì	HTML,Text	Sì	Sì	Sì	-	-

Tabella 7.2: Toolkits commerciali

Azienda	Toolkit	Versione Demo	Dati di Output	Web Crawlling	Interfaccia programmi esterni	GUI	Editor	Linguaggio di Scripting
Lixto	Lixto, InfoPipes	-	HTML, XML, Text, WML	Sì	Sì, Java API	Sì	Sì	ELOG
Loton Tech	WebDataKit	Sì	Tabella	Sì	Sì, Java API	Sì	Sì	SQL for HTML and XML
NQL Inc.	Network Query Language	Sì	HTML, XML, Text	Sì	Sì	Sì	Sì	Network Query Language
Orsus Solution	UnoStudio	-	HTML, XML, WAP, Text	Sì, solo URLs predef	Sì	Sì	-	-
Republica	X-Fetch Wrapper	Sì	XML, WML, XHTML	-	Sì, Java API	Sì	Sì	DEL
Temis Group	Online Miner, Insight Discoverer Extractor	-	HTML, XML, Text	Sì	Sì	Sì	-	-
Thunderstone	Webinator	Sì	HTML, Text	Sì	Sì	Sì	Sì	Texis Webscript, espressioni regolari
Tropea Inc.	W4F Wrapper Factory	-	HTML, XML, Text	Sì	Sì, Java API	Sì	Sì	HEL
WhizBang!	WhizBang! Extraction Framework	-	HTML, Text	Sì	Sì	Sì	-	-
Wisosoft	Info Scanner	Sì	XML	Sì, Macro predef	HTTP, COM, J-API, ODBC	Sì	Sì	VBScript, JScript, espressioni regolari

Tabella 7.3: Toolkits commerciali

Appendice A

Glossario I^3

Questo glossario ed il vocabolario sul quale si basa sono stati originariamente sviluppati durante l'I^3

Architecture Meeting in Boulder CO, 1994, sponsorizzato dall'ARPA, e rifiniti in un secondo incontro presso l'Universitá di Stanford, nel 1995. Il glossario é strutturato logicamente in diverse sezioni:

- Sezione 1: Architettura
- Sezione 2: Servizi
- Sezione 3: Risorse
- Sezione 4: Ontologie

Nota: poiché la versione originaria del glossario usa una terminologia inglese, in alcuni casi é riportato, a fianco del termine, il corrispettivo inglese, quando la traduzione dal termine originale all'italiano poteva essere ambigua o poco efficace.

A.1 Architettura

- Architettura = insieme di componenti.

- architettura di riferimento = linea guida ed insieme di regole da seguire per l'architettura.
- componente = uno dei blocchi sui quali si basa una applicazione o una configurazione. Incorpora strumenti e conoscenza specifica del dominio.
- applicazione = configurazione persistente o transitoria dei componenti, rivolta a risolvere un problema del cliente, e che puó coprire diversi domini.
- configurazione = istanza particolare di una architettura per una applicazione o un cliente.
- collante (glue) = software o regole che servono per per collegare i componenti o per interoperare attraverso i domini.
- strato = grossolana categorizzazione dei componenti e degli strumenti in una configurazione. L'architettura I^3

  distingue tre strati, ognuno dei quali fornisce una diversa categoria di servizi:

  1. Servizi di Coordinamento = coprono le fasi di scoperta delle risorse, distribuzione delle risorse, invocazione, scheduling...
  2. Servizi di Mediazione = coprono la fase di query processing e di trattamento dei risultati, nonché il filtraggio dei dati, la generazione di nuove informazioni, etc.
  3. Servizi di Wrapping = servono per l'utilizzo dei wrappers e degli altri strumenti simili utilizzati per adattarsi a standards di accesso ai dati e alle convenzioni adoperate per la mediazione e per il coordinamento.

- agente = strumento che realizza un servizio, sia per il suo proprietario, sia per un cliente del suo proprietario.

- facilitatore = componente che fornisce i servizi di coordinamento, come pure l'instradamento delle interrogazioni del cliente.
- mediatore = componente che fornisce i servizi di mediazione e che provvede a dare valore aggiunto alle informazioni che sono trasmesse al cliente in risposta ad una interrogazione.
- cliente (customer) = proprietario dell'applicazione che gestisce le interrogazioni, o utente finale, che usufruisce dei servizi.
- risorsa = base di dati accessibile, server ad oggetti, base di conoscenze...
- contenuto = risultato informativo ricavato da una sorgente.
- servizio = funzione fornita da uno strumento in un componente e diretta ad un cliente, direttamente od indirettamente.
- strumento (tool) = programma software che realizza un servizio, tipicamente indipendentemente dal dominio.
- wrapper = strumento utilizzato per accedere alle risorse conosciute, e per tradurre i suoi oggetti.
- regole limitative (constraint rules) = definizione di regole per l'assegnamento di componenti o di protocolli a determinati strati.
- interoperare = combinare sorgenti e domini multipli.
- informazione = dato utile ad un cliente.
- informazione azionabile = informazione che forza il cliente ad iniziare un evento.
- dato = registrazione di un fatto.
- testo = dato, informazione o conoscenza in un formato relativamente non strutturato, basato sui caratteri.

- conoscenza = metadata, relazione tra termini, paradigmi..., utili per trasformare i dati in informazioni.
- dominio = area, argomento, caratterizzato da una semantica interna, per esempio la finanza, o i componenti elettronici...
- metadata = informazione descrittiva relativa ai dati di una risorsa, compresi il dominio, proprietá, le restrizioni, il modello di dati,...
- metaconoscenza = informazione descrittiva relativa alla conoscenza in una risorsa, includendo l'ontologia, la rappresentazione...
- metainformazioni = informazione descritiiva sui servizi, sulle capacitá, sui costi...

A.2 Servizi

- Servizio = funzionalitá fornita da uno o piú componenti, diretta ad un cliente.
- instradamento (routing) = servizio di coordinamento per localizzare ed invocare una risorsa o un servizio di mediazione, o per creare una configurazione. Fa uso di un direttorio.
- scheduling = servizio di coordinamento per determinare l'ordine di invocazione degli accessi e di altri servizi; fa spesso uso dei costi stimati.
- accoppiamento (matchmaking) = servizio che accoppia i sottoscrittori di un servizio ai fornitori.
- intermediazione (brokering) = servizio di coordinamento per localizzare le risorse migliori.
- strumento di configurazione = programma usato nel coordinamento per aiutare a selezionare ed organizzare i componenti in una istanza particolare di una configurazione architetturale.

- servizi di descrizione = metaservizi che informano i clienti sui servizi, risorse...
- direttorio = servizio per localizzare e contattare le risorse disponibili, come le pagine gialle, pagine bianche...
- decomposizione dell'interrogazione (query decomposition) = determina le interrogazioni da spedire alle risorse o ai servizi disponibili.
- riformulazione dell'interrogazione (query reformulation) = programma per ottimizzare o rilassare le interrogazioni, tipicamente fa uso dello scheduling.
- contenuto = risultato prodotto da una risorsa in risposta ad interrogazioni.
- trattamento del contenuto (content processing) = servizio di mediazione che manipola i risultati ottenuti, tipicamente per incrementare il valore delle informazioni.
- trattamento del testo = servizio di mediazione che opera sul testo per ricerca, correzione...
- filtraggio = servizio di mediazione per aumentare la pertinenza delle informazioni ricevute in risposta ad interrogazioni.
- classificazione (ranking) = servizio di mediazione per assegnare dei valori agli oggetti ritrovati.
- spiegazione = servizio di mediazione per presentare i modelli ai clienti.
- amministrazione del modello = servizio di mediazione per permettere al cliente ed al proprietario del mediatore di aggiornare il modello.

- integrazione = servizio di mediazione che combina i contenuti ricevuti da una molteplicitá di risorse, spesso eteorgenee.
- accoppiamento temporale = servizio di mediazione per riconoscere e risolvere differenze nelle unitá di misura temporali utilizzate dalle risorse.
- accoppiamento spaziale = servizio di mediazione per riconoscere e risolvere differenze nelle unitá di misura spaziali utilizzate dalle risorse.
- ragionamento (reasoning) = metodologia usata da alcuni componenti o servizi per realizzare inferenze logiche.
- browsing = servizio per permettere al cliente di spostarsi attraverso le risorse.
- scoperta delle risorse = servizio che ricerca le risorse.
- indicizzazione = creazione di una lista di oggetti (indice) per aumentare la velocitá dei servizi di accesso.
- analisi del contenuto = trattamento degli oggetti testuali per creare informazioni.
- accesso = collegamento agli oggetti nelle risorse per realizzare interrogazioni, analisi o aggiornamenti.
- ottimizzazione = processo di manipolazione o di riorganizzazione delle interrogazioni per ridurne il costo o il tempo di risposta.
- rilassamento = servizio che fornisce un insieme di risposta maggiore rispetto a quello che l'interrogazione voleva selezionare.
- astrazione = servizio per ridurre le dimensioni del contenuto portandolo ad un livello superiore.

- pubblicitá (advertising) = presentazione del modello di una risorsa o del mediatore ad un componente o ad un cliente.
- sottoscrizione = richiesta di un componente o di un cliente di essere informato su un evento.
- controllo (monitoring) = osservazione delle risorse o dei dati virtuali e creazione di impulsi da azionare ogniqualvolta avvenga un cambiamento di stato.
- aggiornamento = trasmissione dei cambiamenti dei dati alle risorse.
- instanziazione del mediatore = popolamento di uno strumento indipendente dal dominio con conoscenze dipendenti da un dominio.
- attivo (activeness) = abilitá di un impulso di reagire ad un evento.
- servizio di transazione = servizio che assicura la consistenza temporale dei contenuti, realizzato attraverso l'amministrazione delle transzioni.
- accertamento dell'impatto = servizio che riporta quali risorse saranno interessate dalle interrogazioni o dagli aggiornamenti.
- stimatore = servizo di basso livello che stima i costi previsti e le prestazioni basandosi su un modello, o su statistiche.
- caching = mantenere le informazioni memorizzate in un livello intermedio per migliorare le prestazioni.
- traduzione = trasformazione dei dati nella forma e nella sintassi richiesta dal ricevente.
- controllo della concorrenza = assicurazione del sincronismo degli aggiornamenti delle risorse, tipicamente assegnato al sistema che amministra le transazioni.

A.3 Risorse

- Risorsa = base di dati accessibile, simulazione, base di conoscenza,... comprese le risorse "legacy".
- risorse "legacy" = risorse preesistenti o autonome, non disegnate per interoperare con una architettura generale e flessibile.
- evento = ragione per il cambiamento di stato all'interno di un componente o di una risorsa.
- oggetto = istanza particolare appartenente ad una risorsa, al modello del cliente, o ad un certo strumento.
- valore = contenuto metrico presente nel modello del cliente, come qualitá, rilevanza, costo.
- proprietario = individuo o organizzazione che ha creato, o ha i diritti di un oggetto, e lo puó sfruttare.
- proprietario di un servizio = individuo o organizzazione responsabile di un servizio.
- database = risorsa che comprende un insieme di dati con uno schema descrittivo.
- warehouse = database che contiene o dá accesso a dati selezionati, astratti e integrati da una molteplicitá di sorgenti. Tipicamente ridondante rispetto alle sorgenti di dati.
- base di conoscenza = risorsa comprendente un insieme di conoscenze trattabili in modo automatico, spesso nella forma di regole e di metadata; permettono l'accesso alle risorse.
- simulazione = risorsa in grado di fare proiezioni future sui dati e generare nuove informazioni, basata su un modello.

- amministrazione della transazione = assicurare che la consistenza temporale del database non sia compromessa dagli aggiornamenti.
- impatto della transazione = riporta le risorse che sono state coinvolte in un aggiornamento.
- schema = lista dedlle relazioni, degli attributi e, quando possibile, degli oggetti, delle regole, e dei metadata di un database. Costituisce la base dell'ontologia della risorsa.
- dizionario = lista dei termini, fa parte dell'ontologia.
- modello del database = descrizione formalizzata della risorsa database, che include lo schema.
- interoperabilitá = capacitá di interoperare.
- eterogeneitá = incompatibilitá trovate tra risorse e servizi sviluppati autonamamente, che vanno dalla paittaforma utilizzata, sistema operativo, modello dei dati, alla semantica, ontologia,...
- costo = prezzo per fornire un servizio o un accesso ad un oggetto.
- database deduttivo = database in grado di utilizzare regole logiche per trattare i dati.
- regola = affermazione logica, unitá della conoscenza trattabile in modo automatico.
- sistema di amministrazione delle regole = software indipendente dal dominio che raccoglie, seleziona ed agisce sulle regole.
- database attivo = database in grado di reagire a determinati eventi.
- dato virtuale = dato rappresentato attraverso referenze e procedure.
- stato = istanza o versione di una base di dati o informazioni.

- cambiamento di stato = stato successivo ad una azione di aggiornamento, inserimento o cancellazione.
- vista = sottoinsieme di un database, sottoposto a limiti, e ristrutturato.
- server di oggetti = fornisce dati oggetto.
- gerarchia = struttura di un modello che assegna ogni oggetto ad un livello, e definisce per ogni oggetto l'oggetto da cui deriva.
- network = struttura di un modello che fa uso di relazioni relativamente libere tra oggetti.
- ristrutturare = dare una struttura diversa ai dati seguendo un modello differente dall'originale.
- livello = categorizzazione concettuale , dove gli oggetti di un livello inferiore dipendono da un antenato di livello superiore.
- antenato (ancestor) = oggetto di livello superiore, dal quale derivano attributi ereditabili.
- oggetto root = oggetto da cui tutti gli altri derivano, all'interno di una gerarchia.
- datawarehouse = deposito di dati integrati provenienti da una molteplicitá di risorse.
- deposito di metadata = database che contiene metadata o metainformazioni.

A.4 Ontologia

- Ontologia = descrizione particolareggiata di una concettualizzazione, i.e. l'insieme dei termini e delle relazioni usate in un dominio, per indicare oggetti e concetti, spesso ambigui tra domini diversi.

- concetto = definisce una astrazione o una aggregazione di oggetti per il cliente.
- semantico = che si riferisce al significato di un termine, espresso come un insieme di relazioni.
- sintattico = che si riferisce al formato di un termine, espresso come un insieme di limitazioni.
- classe = definisce metaconoscenze come metodi, attributi, ereditarietá, per gli oggetti in essa istanziati.
- relazione = collegamento tra termini, come *is-a*, *part-of*,...
- ontologia unita (merged) = ontologia creata combinando diverse ontologie, ottenuta mettendole in relazione tra loro (mapping).
- ontologia condivisa = sottoinsieme di diverse ontologie condiviso da una molteplicitá di utenti.
- comparatore di ontologie = strumento per determinare relazioni tra ontologie, utilizzato per determinare le regole necessarie per la loro integrazione.
- mapping tra ontologie = trasformazione dei termini tra le ontologie, attraverso regole di accoppiamento, utilizzato per collegare utenti e risorse.
- regole di accoppiamento (matching rules) = dichiarazioni per definire l'equivalenza tra termini di domini diversi.
- trasformazione dello schema = adattamento dello schema ad un'altra ontologia.
- editing = trattamneto di un testo per assicurarne la conformitá ad una ontologia.

- algebra dell'ontologia = insieme delle operazioni per definire relazioni tra ontologie.
- consistenza temporale = é raggiunta se tutti i dati si riferiscono alla stessa istanza temporale ed utilizzano la stessa granularitá temporale.
- specifico ad un dominio = relativo ad un singolo dominio, presuppone l'assenza di incompatibilitá semantiche.
- indipendente dal dominio = software, strumento o conoscenza globale applicabile ad una molteplicitá di domini.

Appendice B

Omini: Un Sistema Automatico di Estrazione di Oggetti

B.1 Cenni Preliminari

Questo capitolo presenta un sistema di estrazione di oggetti completamente automatico chiamato Omini. L'obiettivo di Omini è quello di fornire il supporto tecnico necessario a strumenti quali XWRAP Elite in un passo particolarmente delicato del processo di estrazione dei dati. Il software Omini estrae content object (conosciuti anche come data-object) ed ignora parti irrilevanti della pagina. Una delle caratteristiche del progetto su cui hanno orientato i loro sforzi gli sviluppatori, è la *robustezza*, termine che in questo contesto sta a significare la capacità del tool di funzionare anche in seguito a modifiche delle pagine Web dalle quali estrae i dati, eliminando così la necessità per il programmatore di dover identificare con esattezza la posizione degli oggetti all'interno della pagina, ogni qualvolta si verifichi un cambiamento, anche lieve, nei documenti di interesse.

Omini scandisce pagine Web con l'intento di estrarre la struttura ad albero e realizza l'estrazione di oggetti in due passi. Primo, usa un set di algoritmi di estrazione del sottoalbero, per individuare il più piccolo sottoalbero che contiene tutti gli oggetti di interesse (evitando ad esempio

di prendere in considerazione gli annunci publicitari). Secondo, impiega una suite di algoritmi per l'estrazione di oggetti, al fine di trovare i tags separatori di oggetti corretti che possono effettivamente separare un'oggetto dall'altro. Entrambi questi passi sono completamente automatizzati. La fase di estrazione del sottoalbero consente di ridurre considerevolmente il numero di possibilità considerate nella successiva fase di estrazione degli oggetti. In questo capitolo si focalizzerà l'attenzione principalmente sulle regole automatiche di estrazione del sottoalbero minimo e su quelle sempre automatiche di identificazione dei confini degli oggetti. Da notare che ogni volta ci si riferirà ad un oggetto si prenderà in considerazione un data-object ossia la struttura dati elementare contenuta nella pagina Web, nel caso ad esempio di una pagina costruita dinamicamente dopo una ricerca attraverso una parola chiave, si considera un oggetto ogni singolo risultato della ricerca, quindi se si stanno cercando libri, un data-object sarà rappresentato dall'insieme di informazioni costituite dal titolo del libro, dall'autore, dal prezzo e quant'altro verrà restituito. L'approccio Omini è stato testato e valutato dagli utenti all'interno del sistema di generazione di wrappers XWRAP Elite. Sono stati condotti una serie di esperimenti su pagine Web provenienti da diversi siti. I risultati secondo gli sviluppatori sono stati soddisfacenti e verranno riportati in una delle sezioni successive.

Nella prossima sezione saranno rivisti brevemente un set di concetti preliminari, quindi nelle seguenti verrà presentata l'architettura del sistema e saranno descritti gli algoritmi di estrazione dei separatori di oggetti. Nell'ultima sezione verrà descritto un algoritmo combinato che mette insieme i cinque algoritmi indipendenti di estrazione di oggetti. Attraverso un'analisi dei risultati degli esperimenti effettuati sfruttando sia gli algoritmi individualmente che la loro combinazione, verrà messo in evidenza il grado di efficacia dell'approccio di estrazione degli oggetti adottato in Omini.

B.1.1 Documento Web Well-Formed

I documenti Web presi in considerazione in questo studio sono di tipo HTML o XML. Un documento Web consiste in tags e testo. Un tag in un documento Web è composto da un nome del tag e da una lista opzionale di attributi del tag racchiusi in una coppia di parentesi angolari aperte e chiuse ("<" e ">"). Il testo è una sequenza di caratteri in mezzo a due tags. Come indicato nelle specifiche standard di HTML ed XML, molti dei tags in un documento Web appaiono in coppia. Un tag il cui nome non inizia con una forward slash (es: "/") viene chiamato uno *start tag*; in caso contratrio esso viene chiamato *end tag* ed il nome di un end tag è il nome del suo corrispondente start tag preceduto da una "/".
Un documento Web è definito well formed se soddisfa le seguenti condizioni:

- Non ci sono parentesi angolari di apertura o di chiusura, < e >, nel testo del documento che non siano tags. Al posto di questi caratteri, quando utilizzati nel testo, si utilizzano le codifiche < e >

- Tutti i tag devono essere accoppiati, ogni start tag nominato ha un corrispondente end tag.

- Tutti i valori degli attributi in un tag devono essere inseriti tra virgolette (es:).

- Tutti i tags che normalmente non hanno un end tag (come <IMG>, <HR> e
) sono immediatamente seguiti dal corrispondente end tag. Per esempio:
 verrà denotato con
</BR>.

- Coppie di tags devono essere innestate una dentro l'altra senza sovrapposizione. Per esempio il frammento di documento "<a> ... ... " non è well-formed. Il corretto innesto per questo frammento di esempio è dato da "<a> ... ... ".

Documenti che non sono well-formed possono essere convertiti al fine di soddisfare il suddetto requisito. A tale trasformazione ci di riferirà come document normalization [40].

B.1.2 Rappresentazione ad Albero di Documenti Web

Un documento Web well-formed può essere modellato come un tag tree. Tutti i nodi interni di un tag tree sono *tag nodes* e tutti i nodi foglia sono *content node* (numeri, stringhe, o altri tipi di dati come quelli codificati MIME). Un tag node denota la parte del documento Web identificata da uno start tag, dal suo corrispondente end tag e da tutti i caratteri contenuti tra questi. Un tag node è etichettato con il nome dello start tag. Un nodo foglia denota i dati contenuti (testo) tra uno start tag e il suo corrispondente end tag oppure tra un end tag ed il successivo start tag nel documento. Un nodo foglia è etichettato con il suo contenuto. Un esempio di tag node in un documento HTML è <Title> Home Page </Title>, dove <Title> è il nome del tag node mentre la stringa di testo Home Page è un nodo foglia.

C'è un percorso (path) dal nodo radice (root) ad ogni altro nodo nell'albero. Per un dato nodo, l'espressione che descrive il percorso dal nodo radice dell'albero al nodo in esame può unicamente identificare quel nodo. Nelle sezioni che seguiranno, l'espressione del percorso potrà essere utilizzata per riferirsi ad un determinato nodo. Consideriamo Figura B.1. Il percorso dal nodo radice HTML al nodo Title passa attraverso il nodo Head. Tale percorso può essere espresso attraverso una dot notation come HTML[1].Head[1].Title[1]. I numeri tra le parentesi dopo ogni nodo denotano l'ordine del figlio nel tag tree. Analogamente il percorso da HTML a Body è HTML[1].Body[2].

Definiamo Γ=(V, E) il tag tree di un documento Web D, dove $V = V_t \bigcup V_c$, V_t è in insieme finito di tag nodes (nodi interni) e V_c è un insieme finito di content nodes (nodi foglia); $E \subset (V \times V)$, rappresenta l'insieme

dei collegamenti diretti tra due nodi. Chiamiamo sottoalbero ancorato al nodo u, $u \in V$, il sottoalbero minimo con la proprietà P, se questo è il più piccolo sottoalbero che soddisfa le seguenti condizioni: non ci sono altri sottoalberi, chiamati $sottoalberi(w)$, $w \in V$, che soddisfino sia la proprietà P che la condizione che u sia un antenato di w. In Figura B.1ci sono due sottoalberi che contengono tutti i nodi hr, il sottoalbero ancorato ad HTML e quello ancorato a Body, Quest'ultimo, come mostrato in Figura B.2, è il minimo sottoalbero che contiene tutti i nodi hr.

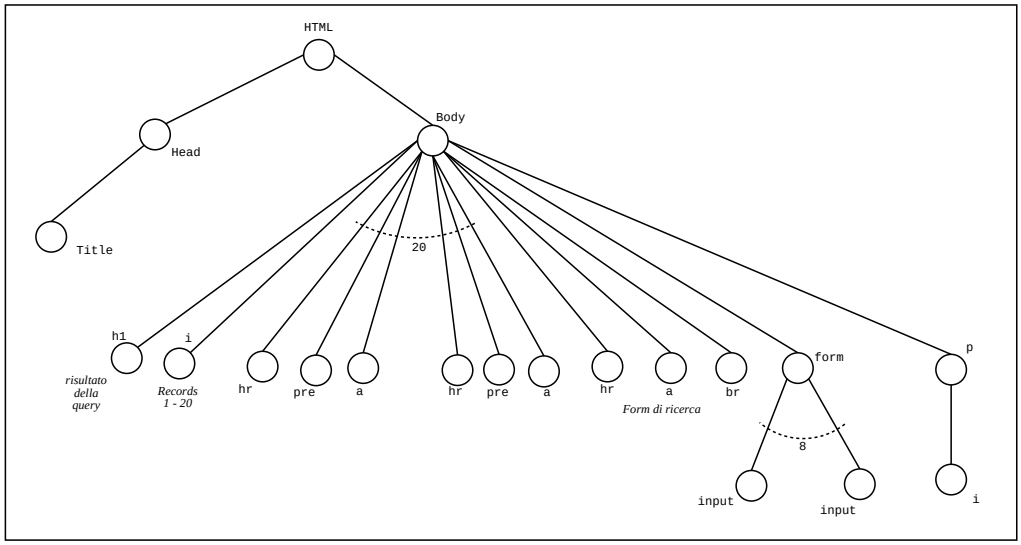

Figura B.1: Rappresentazione ad albero della pagina di ricerca della Library of Congress (loc.gov)

In aggiunta alla nozione di sottoalbero minimo, i seguenti concetti sono usati di frequente nelle sezioni che seguiranno per descrivere gli algoritmi di estrazione di oggetti di Omini.

$Padre(u)$**:** Il nodo padre di u è definito come $Padre(u) = w|w \in V, (w, u) \in E$. Il nodo radice di un albero Γ è l'unico nodo che non possiede un nodo padre.

$Figlio(u)$**:** Il nodo padre di u si riferisce ad un insieme di nodi figli di u. $figlio(u) = w|w \in V, (u, w) \in E$. Questa definizione dice che un

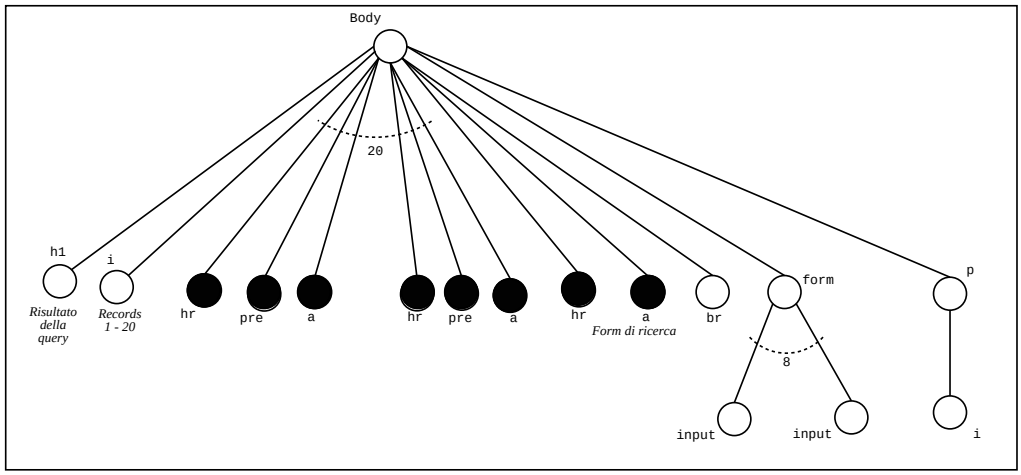

Figura B.2: Sottoalbero minimo della Figura B.1, in evidenza i tags separatori di oggetti candidati

nodo w è un nodo figlio di u se e solo se esiste un collegamento diretto (u,w) ∈ E.

$Fanout(u)$**:** Per ogni nodo u ∈ V, usiamo fanout(u) per denotare la cardinalità dell'insieme di figli di u. $Fanout(u) = ||Figlio(u)||$ se $u \in Vt$ e $Fanout(u) = 0$ se $u \in V_c$

$DimensioneNodo(u)$**:** Per ogni nodo $u \in V$, se $u \in V_c$, es: u è un nodo foglia, allora $DimensioneNodo(u)$ denota la dimensione in bytes del contenuto del nodo u. Altrimenti, se u è un nodo foglia, es: $u \in Vt$ e $Fanout(u) > 0$. Definiamo $DimensioneNodo(u)$ come la somma delle dimensioni dei nodi di tutti i nodi foglia raggiungibili dal nodo u, es: $DimensioneNodo(u) = \sum_{\forall V_i \in figlio(u)}(DimensioneNodo(V_i)$. Per ogni nodo $u \in V$, definiamo la dimensione del sottoalbero ancorato al nodo u, denotata da $DimensioneSottoalbero(u)$, come la dimensione del nodo u. es: $DimensioneSottoalbero(u) = DimensioneNodo(u)$.

*NumeroTagContenuti(u)***:** Per ogni nodo $u \in V$, se $u \in V_c$ è un nodo foglia, allora $NumeroTagContenuti(u) = 0$. Altrimenti $u \in V_t$ è un tag no-

de e $NumeroTagContenuti(u) = 1+\sum_{\forall V_i \in figlio(u)}(NumeroTagContenuti(V_i))$. $NumeroTagContenuti(u)$ si riferisce la numero totale di tag nodes dei quali u è un antenato.

B.2 Architettura del Sistema

Figura B.3 mostra l'architettura di sistema di Omini. Un utente o un'applicazione può fornire un URL al sistema Omini al fine di iniziare il processo di estrazione degli oggetti. I risultati restituiti da Omini sono costituiti da una lista di oggetti estratti dalla pagina Web locata all'indirizzo fornito dall'URL.
Il processo di estrazione degli oggetti di questo sistema si articola in tre fasi:

1. **Preparazione di un documento Web per l'estrazione**
   In questa fase viene ricevuto un URL dall'utente finale o da un'applicazione e vengono eseguiti i seguenti tre compiti:

   - la pagina Web specificata dall'URL viene recuperata da un sito remoto
   - la pagina letta viene "ripulita" dalle imperfezioni utilizzando l'algoritmo di normalizzazione sintattica, il quale trasforma la pagina Web data in un documento equivalente well-formed .
   - il documento Web well-formed verrà convertito nell'equivalente rappresentazione ad albero (tag tree), basata sulle strutture innestate di start ed end tag.

2. **Individuazione degli oggetti di interesse in una pagina Web**
   Il processo realizzato in questa fase è diviso in due step consecutivi: estrazione del sottoalbero ricco di data-object ed estrazione dei separatori di oggetti. Il primo step ha il compito di individuare il sottoalbero minimo che contiene gli oggetti di interesse

in una pagina. Il secondo trova i tags separatori di oggetti che possono effettivamente separare un oggetto dall'altro all'interno della pagina.

Step1: Estrazione del sottoalbero ricco di data-object

Le pagine Web sono progettate per la lettura da parte degli uomini. In aggiunta alle regioni che contengono i dati principali, molte pagine Web spesso contengono altre informazioni come ad esempio gli annunci, i link per la navigazione e altro. Perciò dato un documento Web D, il primo compito nella fase di scoperta degli oggetti è quello di identificare quale parte del documento sia la regione che racchiude il contenuto principale. Sia T il tag tree del documento D, il problema di identificare tale regione è ridotto a quello di individuare il sottoalbero di T che contiene tutti i data-object di interesse. Ci si riferirà a questo compito come: Individuazione del Sottoalbero Ricco di Oggetti. Ovviamente, potrebbe essere presente più di un sottoalbero che contiene tutti gli oggetti di interesse. Il principale obiettivo della Individuazione del Sottoalbero Ricco di Oggetti sarà quindi quello di individuare il sottoalbero minimo di T che contiene tutti gli oggetti di interesse. Nel primo prototipo di Omini sono state implementate tre regole individuali di scoperta del sottoalbero ed un metodo per combinare queste regole tra loro. Al fine di scegliere il sottoalbero corretto verranno confrontati *Fanout*, *DimensioneNodo*, *NumeroTagContenuti* (così come sono state definite queste grandezze precedentemente), di tutti i sottoalberi nel documento Web in esame. Si consideri il documento Web in Figura B.4, il cui tag tree è stato omesso per ragioni di spazio, gli oggetti di cui siamo interessati all'estrazione sono ovviamente i risultati della ricerca, ossia gli otto items di libri relativi alla ricerca attraverso la parola chiave "Java" incapsulati in altrettante tabelle nella parte destra dell'albero. Applicando le euristiche per

l'individuazione del subtree minimo otteniamo che il sottoalbero minimo che contiene tutti gli oggetti, di interesse è quello ancorato al tag node HTML[1].body[2].form[4]. La descrizione dettagliata degli algoritmi di individuazione del sottoalbero minimo sarà per brevità omessa in questa tesi, a questo proposito si possono trovare maggiori informazioni in [11].

Step2: Estrazione dei separatori di oggetti
Una volta che è stata individuata la regione che racchiude il contenuto principale, il compito successivo è quello di decidere come separare i data objects l'uno dall'altro, e da ogni altra informazione all'interno della pagina Web. Ci si riferirà a questo compito come Individuazione di Separatori di Oggetti. Uno degli obiettivi di tale funzionalità del tool è quello di sviluppare un metodo che possa automatizzare il processo di scoperta del corretto separatore di oggetti, il quale effettivamente separerà oggetti nella regione principale e consentirà l'estrazione. Per raggiungere questo obiettivo, sono stati sviluppati un set di algoritmi individuali, ognuno dei quali può autonomamente, identificare una lista ordinata di separatori di oggetti, viene quindi fornito un meccanismo per combinare questi algoritmi indipendenti in un approccio metodico per la scoperta di separatori di oggetti.

3. **Estrazione di oggetti di interesse da una pagina**
Possiamo individuare due attività di base: Costruzione di Oggetti Candidati e Rifinitura del Processo di Estrazione di Oggetti.

   - Costruzione di Oggetti Candidati:
   è il processo di estrazione degli oggetti dalle righe di dati testuali del documento Web usando il tag separatore di oggetti identificato nello Step 2. Dopo che il tag separatore di oggetti è stato scelto, gli oggetti devono essere estratti dal sottoalbero scelto nello Step 1. A volte il tag separatore è situato tra

un oggetto e l'altro, altre volte esso è la radice dell'oggetto o una parte dell'oggetto stesso. Occasionalmente, un oggetto può essere separato in due o più parti dal tag separatore scelto. Quando questo accade, si rende necessario un meccanismo per ricostruire l'intero oggetto di partenza unendo tali parti insieme.

- Rifinitura del Processo di Estrazione di Oggetti:
  viene effettuata al fine di eliminare oggetti candidati che non sono conformi al set di criteri minimi, i quali sono derivati dal processo di estrazione di oggetti e soddisfatti dalla maggior parte degli oggetti estratti. Più concretamente, nel processo di costruzione degli oggetti, quelli estranei, come ad esempio i titoli di liste, possono essere estratti occasionalmente. Lo step di rifinitura del processo di estrazione rimuoverà quegli oggetti che strutturalmente non sono dello stesso tipo della maggior parte degli oggetti, così come quelli che non includono un set di tag comune a tutti gli altri oggetti, o che hanno molti tags unici. Indipendentemente dal fatto che tali oggetti siano molto piccoli o molto grandi, essi verranno rimossi.

Nel resto dello studio ci si concentrerà sulle euristiche per l'estrazione di separatori di oggetti e sulla valutazione delle loro performances.

B.3 Estrazione di Separatori di Oggetti

Dopo il processo di estrazione del sottoalbero ricco di data-object, il problema dell'estrazione del tag separatore di oggetti all'interno di una pagina Web si riduce a quello di trovare il corretto tag separatore di oggetti all'interno del sottoalbero minimo. Il problema può essere affrontato in due passi:

- bisogna decidere quali tags del sottoalbero minimo scelto possono essere considerati tags candidati.

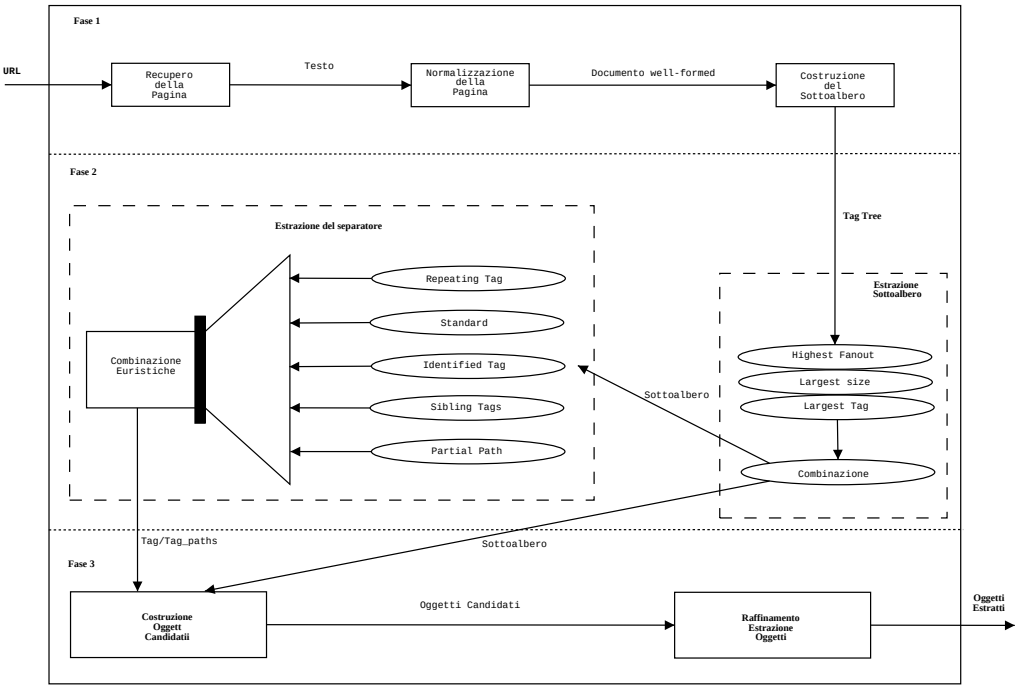

Figura B.3: Architettura del Sistema Omini

- bisogna trovare un metodo per identificare il giusto separatore di oggetti all'interno dell'insieme dei tags candidati, il quale separerà effettivamente tutti gli oggetti

Per quanto riguarda le tecniche che permettono di individuare l'insieme di partenza dei tags candidati sono disponibili diversi approcci. I più comuni si dividono tra quelli che valutano ogni nodo del sottoalbero minimo come tag candidato e quelli che prendono in considerazione solo i nodi figli del sottoalbero scelto. Omini ritiene sufficiente considerare i nodi figli nel sottoalbero minimo al fine di farne nodi candidati per essere separatori di oggetti.

Il sistema supporta cinque euristiche atte all'individuazione del tag separatore tra quelli candidati, queste permettono di avere a disposizione un intervallo sufficientemente ampio di possibili meccanismi per la determinazione di separatori di oggetti. Ognuna delle cinque euristi-

che indipendentemente dalle altre ordina i tags candidati. L'euristica *Standard Deviation* (SD) e la *Repeating Pattern* (RP) sono state le prime proposte in [5], la prima ordina i tags candidati in base alla deviazone standard della distanza in termini di numero di caratteri tra due tags, la seconda ordina in base alla differenza tra il numero di occorrenze di una coppia di tags e quello di ognuno dei due tags considerati separatamente. Le seguenti due euristiche sono state introdotte in Omini: *Partial Path Heuristic (PP)*, si basa sulla considerazione che istanze multiple di oggetti dello stesso tipo spesso hanno i tags strutturati alla stessa maniera, *Sibling Tag Heuristic (SB)*, si basa sulla considerazione che oggetti contenenti un alto numero di coppie di tags è più facile che siano oggetti dello stesso tipo rispetto a quelli che contengono un alto numero di tags singoli. Per finire l'ultima a disposizione è l'*Identifiable Path Separator Heuristic (IPS)*, è un'estensione della IT (Identifiable Path) proposta in [5]. Invece di usare la stessa lista di tags candidati ordinati e predeterminati per tutti i tag trees, è utilizzata una lista differente in base al sottoalbero che si è scelto.

Nelle seguenti sezioni verrà fornita una descrizione di ognuna delle 5 euristiche individuali e quindi verrà descritto il metodo per combinare nel modo migliore gli ordinamenti delle cinque, al fine di selezionare l'object separator corretto.

B.3.1 Standard Deviation Heuristic (SD)

L'euristica SD misura la deviazione standard della distanza (in termini di numero di caratteri) tra due occorrenze consecutive di un tag candidato, ordina quindi la lista di tags candidati in modo ascendente in base alla loro deviazione standard. Essa è motivata dall'osservazione che istanze multiple dello stesso tipo di oggetto sono poste alla stessa distanza.

Consideriamo il tag tree per la pagina Web della Library of Congress mostrata in Figura B.1. Dalla fase di estrazione del sottoalbero, quello

ancorato al nodo HTML[1].Body[2] è il sottoalbero minimo scelto, come mostrato in Figura B.2. All'interno dell'insieme dei tags relativi a nodi figli, alcuni tags hanno un numero di occorrenze più alto di quanto non abbiano altri. Per portare un esempio possiamo citare i tags *hr* e *a* che appaiono 21 volte, ed il tag *pre* che appare 20 volte. D'ora in poi sarà possibile fare riferimento a questi tags come a quelli aventi il numero di occorrenze più alto. La deviazione standard in termini di distanza è calcolata tra due occorrenze consecutive del tag *hr*, tra due occorrenze consecutive del tag *pre*, tra due occorrenze consecutive del tag *a* e così via.

La Tabella B.1 mostra i tre migliori tags candidati ottenuti applicando l'algoritmo SD al sito della Libreria del Congresso Americano riportato in Figura B.2. I tags candidati sono stati elencati in ordine ascendente rispetto alla deviazione standard calcolata sulla base della distanza tra un'occorrenza e l'altra, con il tag avente valore risultante minore in alto.

Rango	Tag	Deviazione Standard
1	hr	114
2	pre	117
3	a	122

Tabella B.1: Deviazione Standard dei tags del sottoalbero minimo

B.3.2 Repeating Pattern Heuristic (RP)

L'euristica RP sceglie i separatori di oggetti contando il numero di occorrenze di tutte le coppie di tags candidati che non contengano testo al loro interno. L'algoritmo calcola il valore assoluto della differenza tra il numero di occorrenze di una coppia di tags quando compaiono insieme e il numero di occorrenze di ognuno dei due tags quando appare da solo, quindi viene costruita una lista dei tags candidati in ordine ascendente in base a questo valore assoluto. L'intuizione dietro questa euristica è legata al fatto che l'utilizzo del singolo tag può significare molte cose,

ma una coppia di tags ha probabilmente un unico significato. Quando non ci sono coppie di tags nel sottoalbero scelto, l'euristica RP produce una lista vuota di tags candidati che viene interpretata dal sistema come l'impossibilità dell'euristica di fornire una risposta al problema di determinare un tag separatore di oggetti tra quelli candidati.

Coppie di Tag	Numero Occorrenze	Differenza
table, tr	13	0
img, br	2	0
map, table	1	0
form, table	1	0
br, img	1	1
br, table	1	1

Tabella B.2: Ordine dei Tag Ripetuti per il sottoalbero minimo di Figura B.4

B.3.3 Identifiable Path Separator Tag Heuristic (IPS)

L'euristica IPS costruisce una lista dei tags candidati per il sottoalbero scelto in accordo con la lista dei tags IPS fornita da sistema. I tags IPS sono quelli identificati dal sistema come i più comunemente utilizzati per svolgere la funzione di tags separatori di oggetti per i differenti tipi di sottoalbero nei documenti Web. L'idea dietro questa euristica nasce dall'osservazione che i documenti Web, generati sia manualmente che attraverso programmi sul server su cui risiede il sito, spesso contengono diversi layout di presentazione anche all'interno di una singola pagina, ognuno dei quali è definito da qualche tipo specifico di tags HTML. Per esempio, una pagina Web può contenere una tabella marcata attraverso il tag di tabella specifico *table*, una lista marcata con il tag di lista *ul* o *ol* ed un paragrafo marcato con il tag *p*. Inoltre ognuno di tali layout di presentazione tende ad usare una struttura regolare. Per esempio, una tabella tende ad usare il tag di riga *tr* ed il tag di colonna *td* per definire righe e colonne all'interno della tabella; così come una lista tende ad

usare i tag per item di lista per definire la struttura di una lista. Perciò, per ogni layout di presentazione (es. un tipo di sottoalbero), ci sono pochi tags che sono usati coerentemente per separare oggetti all'interno di un sottoalbero.

Sulla base di queste osservazioni e di test effettuati direttamente su documenti Web dallo staff del Georgia Tech, è stata creta una lista di tags separatori di oggetti per ogni tipo di sottoalbero. L'intera lista di separatori di oggetti è composta da tutti i tags identificati per ogni tipo di sottoalbero elencato in Tabella B.3, eliminando i duplicati.

Sottoalbero	Lista dei Tags
body	table, p, hr, ul, li, blockquote, div, pre, b, a
table	tr, b
form	table, p, dl
td	table, hr, dt, li, p, tr, font
dl	dt, dd
ol	li
ul	li
blockquote	p

Tabella B.3: Tabella dei Tags Separatori di Oggetti

Lo step successivo consta nel determinare l'elenco di questi tags separatori di oggetti. La Tabella B.4 elenca la distribuzione di tutti i tags separatori di oggetti sulla base dei risulatati ottenuti dai test effettuati su un grosso numero di pagine Web appartenenti a diversi siti. La lista ordinata così ottenuta verrà d'ora in poi chiamata IPSList:
{tr,table,p,li,hr,dt,ul,pre,font,dl,div,dd,blockquote,b,a,span,td,br,h4,h3,h2,h1,strong,em,i}.

Per tags aventi lo stesso rango l'ordine è arbitrario.

B.3.4 Sibling Tag Heuristic (SB)

L'euristica SB conta le coppie di tags che sono vicine nel sottoalbero minimo e crea una lista di tutte queste coppie di tags in ordine discendente per numero di occorrenze della coppia. Per quelle coppie di tags che

Tag	% di volte usata come separatore di oggetti
tr	34
table	18
p	10
li	8
hr	6
dt	6
ul	2
pre	2
font	2
dl	2
div	2
dd	2
blockquote	2
b	2
a	2

Tabella B.4: Probabilità di essere un Object Separator

hanno uguali occorrenze, il rango segue l'ordine della loro apparizione nel documento Web. Per esempio, nel frammmento * <a> ... ... <c> ... </c> *, le coppie vicine sono: *(a,b)* e *(b,c)* ed il numero delle occorrenze per entrambe è 1. La Tabella B.5 elenca coppie vicine dal tag tree della Libreria del Congresso vista in Figura B.1 e da quello estratto dalla pagina di BarnesandNoble.com vista in figura B.4.

La motivazione alla base di quest'euristica è legata all'osservazione che dato un sottoalbero minimo, ci si aspetta di trovare il tag separatore di oggetti un numero di volte uguale al numero degli oggetti.

B.3.5 Partial Path Heuristic (PP)

L'euristica Partial Path crea una lista di tutti i percorsi da un nodo candidato ad ogni altro che sia raggiungibile all'interno del sottoalbero scelto e conta il numero di occorrenze di ogni percorso identificato. I tags candidati sono elencati all'interno di una lista ordinati in base al numero di

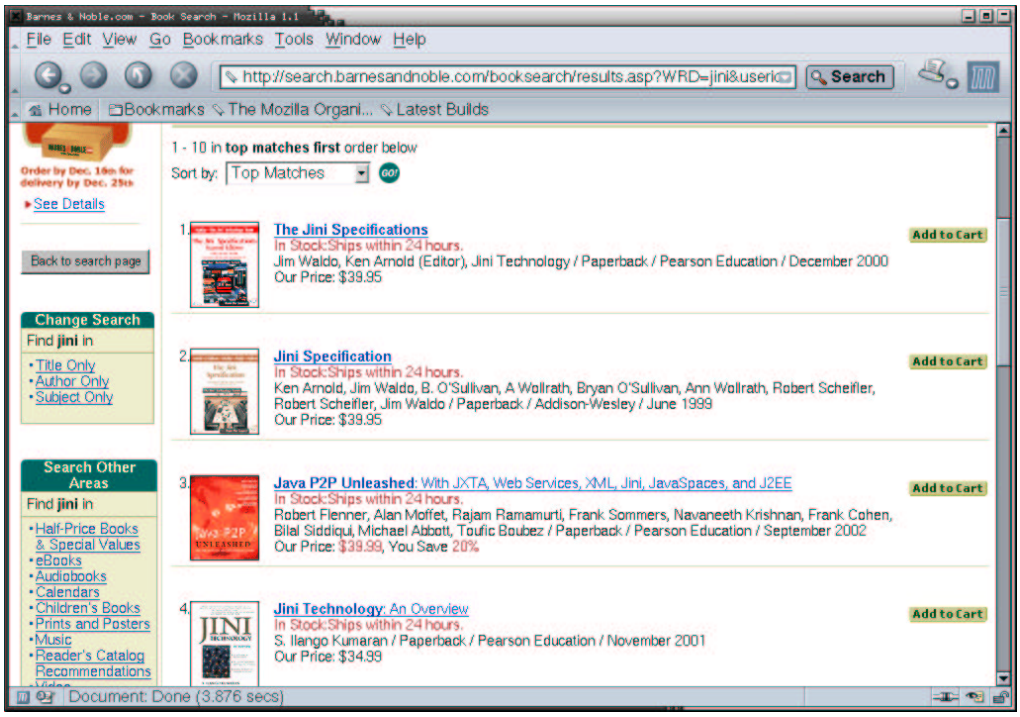

Figura B.4: Pagina di ricerca di BarnesandNobles.com

	BarnesandNoble.com		Library of Congress	
Rango	coppia	occorrenze	coppia	occorrenze
1	table, table	11	hr, pre	20
2	img, br	2	pre, a	20
3	br, img	1	a, hr	20
4	br, table	1	h1, i	1
5	table, map	1	i, hr	1
6	map, table	1	hr, a	1
7	table, form	1	a, br	1
8			br, form	1
9			form, p	1

Tabella B.5: Ordinamento dei tags per l'euristica SB applicata a Figura B.1 e Figura B.4

occorrenze di tutti i percorsi identificati. Se due percorsi hanno uguali occorrenze, allora il percorso più lungo verrà inserito con un rango mag-

giore rispetto a quello più corto in quanto esso indica la presenza di un maggior grado di struttura. E' interessante notare, che se non ci sono percorsi aventi lunghezza maggiore di 1, come nel caso di Figura B.1, questa euristica si riduce alla semplice scelta del tag avente numero di occorrenze maggiore.

La principale idea dietro questo algoritmo è legata dall'osservazione che istanze multiple dello stesso tipo di oggetto spesso hanno i tags strutturati allo stesso modo.

La Tabella B.6 mostra l'ordinamento ritornato dall'euristica Partial Path per i documenti di esempio in Figura B.1 ed in Figura B.4, rispettivamente le pagine del sito della Libreria del Congresso Americano e quelle del sito BarnesandNoble.com.

	BarnesandNoble.com		Library of Congress	
Rank	tag	occorrenze	tag	occorrenze
1	table	26	hr	21
2	form	2	a	21
3	img	2	pre	20
4	br	2	form	8

Tabella B.6: Ordinamento dei tags per l'euristica PP applicata a Figura B.1 e Figura B.4

B.4 Determinazione del Corretto Tag Separatore di Oggetti: L'Algoritmo Combinazione

Nella sezione precedente è stata discussa ogni euristica individuale ed il relativo algoritmo per produrre una lista ordinata di tags candidati. Tutti e cinque gli algoritmi individuali lavorano autonomamente allo scopo di raggiungere lo stesso obiettivo: trovare il giusto separatore di oggetti all'interno dell'insieme dei tags candidati. Come risultato ogni euristica sceglie il tag avente rango più alto come "corretto" separatore di oggetti. Comunque, come osservato dalle discussioni nelle precedenti sezio-

ni, i risultati dell'elaborazione di queste cinque euristiche possono non sempre essere concordi tra loro.

Euristica	Rango 1	Rango 2	Rango 3	Rango 4	Rango 5
SD	0.78	0.18	0.10	0.00	0.00
RP	0.73	0.13	0.00	0.00	0.00
IPS	0.40	0.46	0.13	0.07	0.00
PP	0.85	0.17	0.12	0.06	0.00
SB	0.63	0.17	0.12	0.06	0.03

Tabella B.7: Probabilità di successo del rango ottenuto da euristiche per il separatore di oggetti

Per comprendere la portata delle prestazioni offerte da queste euristiche individuali su differenti tipi di pagine Web, sono state condotte una serie di prove su un campione omogeneo di documenti (vedi [11] per ulteriori dettagli). Di questi test saranno riportati i valori, questo non tanto per l'importanza dei dati in sè, quanto piuttosto per facilitare, attraverso un esatta comparazione delle prestazioni offerte dai vari algoritmi e dalle diverse tecniche in esame, la piena comprensione dei motivi che hanno portato i progettisti del Georgia Tech ad effettuare determinate scelte implementative. In Tabella B.7 è elencata una distribuzione di probabilità empirica per il tasso di successo di ogni euristica individuale. Per ogni euristica, è stato per prima cosa calcolato il numero di volte in cui l'euristica ha scelto il separatore di oggetti corretto, questo relativamente ad ogni rango. Quindi per ogni sito Web, è stato calcolato il tasso di successo dell'euristica in esame normalizzando il numero di volte in cui il tag separatore di oggetti trovato era corretto, con il numero di pagine testate di un particolare sito Web. Il tasso di successo per ogni euristica all'interno dei siti è stato calcolato facendo la media dei numeri normalizzati provenienti da ogni sito Web.

Un modo ovvio per migliorare l'accuratezza della ricerca del corretto separatore di oggetti in un documento Web è quello di considerare il modo migliore di combinare queste euristiche indipendenti. Un ap-

proccio spesso utilizzato per combinare prove provenienti da due o più osservazioni indipendenti è quello di usare le leggi base delle probabilità [12]. Sia $P(A)$ la probabilità associata al risultato dell'applicazione di un'euristica A all'interno di un documento Web. La formula

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

fornisce la probabilità composta $P(A \cup B)$ per individuare un corretto tag separatore di oggetti in un documento.

Per combinare le 5 euristiche individuali, si hanno 26 possibili combinazioni in aggiunta al caso banale in cui nessuna delle euristiche sia in grado di restituire un risultato. La Tabella B.8 elenca i risultati di tutte le combinazioni. Per rappresentare convenientemente una combinazione, il nome di ogni euristica è abbreviato utilizzando un acronimo di una sola lettera: HC con H, SD con S, RP con R,ISP con I, PP con P e SB con B. Così, RSIPB sta a significare la combinazione delle euristiche RP, SD, IPS, PP ed SB. Anche grazie ai test eseguiti sul set di documenti campioni, si è in grado di provare che la combinazione delle cinque euristiche è quella in grado di offrire risultati migliori.

Combinazione	Successo	Combinazione	Successo	Combinazione	Successo
IB	0.61	RB	0.73	RI	0.75
RS	0.78	SI	0.78	SB	0.78
RIB	0.80	RSB	0.84	SIB	0.84
RPB	0.85	RP	0.85	SP	0.85
IPB	0.85	IP	0.85	PB	0.85
RSI	0.86	RIPB	0.86	RIP	0.86
RSP	0.88	SIPB	0.89	SIP	0.89
SPB	0.89	RSIP	0.92	RSIB	0.92
RSPB	0.92	RSIPB	0.98		

Tabella B.8: Tassi di Successo per Combinazioni di Euristiche

B.5 Risultati degli esperimenti

In questa sezione saranno riportati i risultati degli esperimenti effettuati dall'equipe di Georgia Tech sul sistema Omini. I risultati ottenuti saranno poi utilizzati al fine di confrontare l'approccio Omini con quello BYU citato in [5]. Verrà quindi discusso l'algoritmo combinazione delle euristiche per l'estrazione di oggetti e l'approccio sperimentale adottato a Georgia Tech per determinare la migliore combinazione. Per validare l'efficacia dell'approccio Omini per l'estrazione di oggetti, l'algoritmo è stato eseguito su una grande quantità di documenti. Per ognuno di questi, sono state applicate le cinque euristiche e la combinazione delle cinque RSIPB. La Tabella B.9 elenca i risultati degli esperimenti. Sebbene quattro delle cinque euristiche individuali non raggiungano un tasso di successo più alto del 90%, per la combinazione delle cinque il tasso di successo si attesta intorno al 94%. I risultati sperimentali indicano che, mentre le euristiche prese singolarmente non sono estremamente stabili, è molto remunerativo combinarle.

Euristica	Rango 1	Rango 2	Rango 3	Rango 4	Rango 5
SD	0.77	0.15	0.06	0.05	0.00
RP	0.77	0.10	0.07	0.02	0.00
IPS	0.88	0.08	0.07	0.00	0.00
PP	0.93	0.05	0.00	0.00	0.00
SB	0.71	0.13	0.06	0.07	0.04
RSIPB	0.94	0.05	0.02	0.01	0.00

Tabella B.9: Probabilità dei vari Ranghi ottenute da Eutistiche per Separatori di Oggetti

Il tasso di successo non è la sola misura della qualità degli algoritmi per l'identificazione del separatore di oggetti. Ne sono stati individuati altri due: **precisione** e **richiamo**. Il richiamo, per definizione, è la percentuale di istanze positive del concetto obiettivo (nel nostro caso il tag separatore di oggetti) che sono correttamente identificate. La precisione è la percentuale di estrazioni corrette che sono state portate a termine.

Entrambi questi numeri sono definiti in termini di *false positive* (FP), *false negative* (FN), e *true positive* (TP).

$$Richiamo = \frac{TP}{TP + FN}$$

$$Precisione = \frac{TP}{TP + FP}$$

Un true positive è un'istanza in cui un separatore di oggetti esiste ed è correttamente identificato dagli algoritmi. Un false negative è un'istanza in cui il separatore di oggetti esiste ma non è individuato dagli algoritmi. Un false positive è un'istanza in cui il separatore di oggetti non esiste, ma un tag è erroneamente identificato come separatore di oggetti.

La Tabella B.10 mostra le misurazioni di precisione e richiamo ottenute da una serie di esperimenti su pagine Web campioni (vedi [11] per ulteriori dettagli). I risultati arrivano a raggiungere, per quanto riguarda il richiamo, tassi prossimi al 98% e per quanto riguarda la precisione, il massimo valore possibile, il 100% su tutti i documenti esaminati.

Euristica	Successo	Precisione	Richiamo
SD	0.78	1.00	0.78
RP	0.73	0.84	0.73
IPS	0.71	0.82	0.71
PP	0.85	0.92	0.85
SB	0.62	0.89	0.62
RSIPB	0.98	1.00	0.98

Tabella B.10: Probabilità dei ranghi ottenuti da Euristiche per Separatori di Oggetti sui documenti testati

B.5.1 Comparazione dei Risultati con l'Approccio BYU

L'approccio di estrazione degli oggetti di Omini è stato motivato da alcuni risultati di ricerche riportati in letteratura [5, 8, 13]. Tra i risultati di queste ricerche, quello che ha ispirato maggiormente gli uomini

del Georgia Tech è BYU [5] sviluppato dal Brigham Young University Extraction Group. Per citare solo uno dei punti in comune, due delle cinque euristiche utilizzate per l'individuazione del corretto separatore di oggetti, SD ed RP, sono state prese da [5], senza alcun cambiamento. L'euristica IPS è un'evoluzione della IT di [5]. IT sceglie tags basati su liste predefinite di separatori di oggetti comuni, soluzione considerata troppo inflessibile quando l'obiettivo è quello di prendere in considerazione una grande varietà di siti Web. Perciò invece di usare la stessa lista di separatori candidati per tutti i tipi di siti Web, sono state utilizzate liste differenti in relazione al tipo di tag node al quale il sottoalbero minimo è ancorato. Questo permette, ad esempio, di mettere il tag *tr* al top dell'elenco per le *tabelle* così come nel caso di una *lista* al top dell'elenco metteremo *li*, e così *table* per tag *body* etc. etc.. L'euristica IPS sembra essere quella in grado di fronteggiare meglio delle altre il problema della velocità di evoluzione del Web. Non è stata inclusa un'euristica Highest Count (HC), la quale assegna un rango ai tag in base al numero di volte in cui appaiono. Questa scelta è stata motivata dal basso tasso di successo ottenuto negli esperimenti a cui è stata sottoposta. Inoltre, questa euristica non faceva parte di alcune delle combinazioni di euristiche più performanti; questo perchè è stato osservato che le combinazioni che la includevano avevano spesso prestazioni più scadenti rispetto alle stesse combinazioni senza l'euristica HC. Al posto di questa è stata inserita invece l'euristica PP, che ne rappresenta la naturale evoluzione, tanto che la HC diventa un caso particolare della stessa PP, quando applicata ad alcuni tipi di siti Web (vedi nel caso in esame l'applicazione al sito della Libreria del Congresso in Figura B.1).

Nel momento di dover scegliere le soluzioni da integrare in Omini è inoltre stata respinta l'ipotesi di utilizzare un'euristica che sfrutti le conoscenze contenute in un'ontologia. Questi tipi di euristiche si affidano infatti alle conoscenze su un determinato sito Web e sono in possesso di una dettagliata ontologia per quel dominio. Un esempio di tale ontologia è quello di usare una specifica parola o frase che sappiamo appare

solo una volta in ogni oggetto di interesse. E' facile immaginare che l'utilizzo di questo tipo di euristiche comporti una serie di vincoli troppo stringenti al fine di poter essere utilizzate all'interno di uno strumento, come quello in esame, che si prefigge tra gli obiettivi di poter interagire con il maggior numero di tipi di siti possibile, indipendentemente dalle conoscenze specifiche su ognuno di loro. Inoltre, come riportato in [5], lo sviluppo di un'ontologia per un dominio comporta un costo di circa 2 settimane/uomo di lavoro, troppo alto per pensare di poter tenere il passo con l'evouzione dei contenuti sul Web. Anche per questo motivo l'obiettivo del progetto Omini è quello di sviluppare un sistema di estrazione di oggeti completamente automatizzato. La scelta di un approccio completamente automatico sembra essere la strada migliore per sviluppare un sistema scalabile che sia in grado di rimanere agganciato ai ritmi di crescita del Web. Lo stesso livello di scalabilità non si potrebbe ottenere con euristiche come ad esempio la OM riportata in [5], la quale richiede un intervento da parte dell'operatore.

Al fine di confrontare l'approccio Omini e quello BYU citato in [5], sono state implementate dai progettisti di Georgia Tech tutte le euristiche contenute in quest'ultimo, ad eccezione di quella basata sull'ontologia, quindi sono stati condotti esperimenti su una grande varietà di pagine Web. Ci sono casi in cui sia le prestazioni ottenute da BYU che quelle ottenute da Omini sono soddisfacenti e più o meno allo stesso livello. Ci sono però anche casi in cui le 4 euristiche in [5] si comportano peggio, ad esempio su siti quali ebay.com, goto.com, bookpoll.com, powells.com, signpost.org, raggiungendo un tasso di successo del 59% contro quello del 93% raggiunto da Omini.

B.6 Considerazioni sul sistema Omini

In questo capitolo sono stati presentati una serie di algoritmi per l'estrazione del separatore di oggetti. Questi sono stati testati dagli sviluppato-

BYU		Omini	
Euristica	Successo	Euristica	Successo
RP	19	RP	19
SD	23	SD	23
IT	40	IPS	76
HC	40	SB	56
		PP	78
HTRS	59	RSIPB	93

Tabella B.11: Tassi di successo per differenti euristiche e combinazioni, sui siti web precedentemente citati

ri di Georgia Tech e valutati usando un insieme di documenti provenienti da un paniere di siti Web (principalmente di commercio elettronico).

L'estrazione completamente automatica di oggetti è una componente importante e necessaria nella costruzione di una nuova generazione di servizi di ricerca ed aggregazione sul Web, che siano scalabili ed affidabili. L'utilità dell'approccio Omini è dimostrata solo integrandolo in un sistema di generazione di wrappers (come quello realizzato a Georgia Tech e la cui analisi precede questo capitolo della tesi, XWRAP Elite [14]) al fine di automatizzare la generazione di wrappers e fornire funzionalità aggiuntive. Altre potenziali direzioni di ricerca futura includono il lavoro per rendere automatici i processi di valutazione dei risultati delle singole fasi e l'incorporazione degli step di rifinitura basata sul feed-back dell'estrazione degli oggetti, così come l'integrazione con ottimizzatori di query e sistemi software per la gestione dell'arricchimento semantico.

In maniera analoga ad XWRAP Elite, gli sviluppatori di Georgia Tech rendono disponibile il toolkit Omini mediante collegamento remoto. In seguito ai problemi che si sono riscontrati durante la sperimentazione diretta di XWRAP Elite sui siti di prova, in buona parte legati alla non corretta individuazione del giusto separatore di oggetti, è stata effettuata una serie di prove su Omini, utilizzando le pagine che avevano precedentemente creato problemi. L'utilizzo di Omini, ha reso possibile l'individuazione del corretto separatore di oggetti su pagine come quel-

le restituita dalla voce "Immobili/Attività" nella sezione "Bacheche" del sito "tessilmoda.com". In questo caso le pagine restituite erano strutturate mediante l'utilizzo di una tabella per ogni inserzione, questa era poi composta a sua volta da altre due tabelle di livello più interno, una per l'intestazione ed una per il contenuto dell'annuncio. La struttura complessa così creata consentiva di restituire graficamente le informazioni relative a ciascuna inserzione mediante l'uso di un'unica tabella. È questo uno dei casi in cui nell'utilizzo di XWRAP Elite ci si era dovuti confrontare con il fatto che non fosssero gestite tabelle in più di due dimensioni. Mediante l'utilizzo di Omini questo tipo di tabelle sono state riconosciute correttamente. Questo fa sperare che a breve sarà possibile sfuttare questa ulteriore potenzialità anche all'interno del sistema XWRAP Elite. Come accennato nel paragrafo 2.7, la tendenza nella costruzione di pagine HTML è quella di spingere sempre più in profondità i contenti di interesse all'interno del tag tree, è naturale quindi che la fase di individuazione del corretto separatore di oggetti rappresenti uno scoglio non piccolo alla completa automazione dei generatori di wrappers e che si continui la fase di messa appunto di tecniche sempre più efficaci nel perseguimento di questo obiettivo.

Appendice C

XWRAP Elite: Passato e Futuro

C.1 XWRAP Original, il passato

Al fine di fornire una visione globale delle problematiche presenti nell'integrazione di sorgenti semi-strutturate, seguirà una breve analisi di quello che si può definire il progetto che ha fornito le basi per lo sviluppo di XWRAP Elite, dal quale quest'ultimo eredita buona parte della logica di funzionamento: XWRAP Original.

La caratteristica distintiva di XWRAP Original è quella di fornire un costruttore di wrappers che si basa sul continuo feed-back richiesto all'utente, questa peculiarità, se da una parte ne limita fortemente la completa automatizzazione del processo, dall'altra garantisce ottimi risultati dal punto di vista della qualità della rappresentazione finale che non lascia spazio così agli errori in cui le euristiche possono incappare in casi che si discostano da quelli per cui sono state messe a punto. Da notare che tali euristiche si basano su supposizioni che hanno un fondamento statistico e quindi sono lontane dal poter essere applicate con successo in tutte le situazioni. È possibile definire colui che si occuperà di dare questo continuo feed-back durante la costruzione il "wrapper manager".

La filosofia dietro questa metodologia è quella di sviluppare meccanismi che forniscono una netta separazione tra la conoscenza semantica estratta dalle informazioni e la generazione del codice del wrapper.

In altre parole per prima cosa vengono sfruttate le informazioni sulla formattazione delle pagine Web per fare ipotesi sulla struttura della semantica che sta dietro la pagina stessa, quindi viene codificata l'ipotetica struttura e le conoscenze sull'estrazione delle informazioni. Dal set di regole di estrazione dell'informazione e dagli XML-templates derivati dagli step in cui si articola il processo di estrazione, il sistema costruisce un programma wrapper che facilita sia i compiti di query su di una sorgente Web semistrutturata che l'integrazione della sorgente stessa con altre sorgenti di informazione provenienti dal Web.

Il processo di generazione è stato partizionato in una serie si sottoprocessi chiamati fasi. Ogni fase è un'operazione logica che prende in input una rappresentazione del documento sorgente e produce in output una rappresentazione di tipo diverso.

La prima fase è chiamata analisi della struttura, essa esegue la codifica delle conoscenze sull'estrazione delle informazioni in un set di regole di estrazione (extraction rules). Questo implica tre passi:

- Leggere una pagina Web da un sito remoto e generare una struttura ad albero per la pagina.
- Identificare regioni e tokens che sono componenti o elementi chiave per l'estrazione del contenuto informativo della pagina.
- Dedurre l'annidamento gerarchico delle sezioni che rappresenta la struttura sintattica della pagina. Per una pagina Web HTML questo richiede la costruzione di un parser HTML specifico per la sorgente che estragga la struttura ad albero della pagina, inserisca i contenuti in un comma-delimited file e converta la gerarchia in un file XML-templates (per quest'ultima parte vedere fase successiva).

La seconda fase è quella di generazione dell'XML specifico per la sorgente, riconosciamo due passi:

- Per prima cosa la generazione di un file XML-template basato sul contenuto dei tokens e sulle specifiche della gerarchia di annidamento ottenuto nella fase di analisi della struttura.

- Come secondo passo costruire un generatore di XML specifico per la sorgente usando il generatore di XML basato sui templates. Un XML-template è un documento XML well formed. Il file XML ottenuto contiene oltre i dati estratti anche delle processing instruction, vedremo nella sezione successiva in cosa consistono queste ultime.

La routine di bilancio del wrapper colleziona informazioni su tutti i data object che appaiono nel documento sorgente, tiene traccia dei nomi usati dal programma e registra informazioni essenziali su ognuno. Ad esempio un wrapper ha bisogno di sapere quanti argomenti un tag si aspetta o quando un elemento rappresenta una stringa o un intero.

Il gestore degli errori ha il compito di rilevare e riportare gli errori trovati nel documento sorgente. Questo viene automaticamente invocato quando viene trovato un difetto nel documento sorgente e deve avvertire lo sviluppatore del wrapper emettendo una diagnosi del problema e aggiustare l'informazione, in modo che ogni fase possa procedere a partire da un documento corretto. I messaggi di errore dovrebbero permettere allo sviluppatore di wrapper di determinare esattamente in quale punto si sono verificati gli errori.

Sia la routine di bilancio che il gestore degli errori interagiscono in tutte le fasi di generazione del wrapper.

C.1.1 Estrazione dell'Informazione

Il codice HTML di una pagina Web viene scandito attraverso un parser che ragiona seguendo criteri appositamente messi a punto per quella sorgente (il linguaggio Perl sembra essere in questo senso la scelta migliore per implementare questo tipo di strumenti). Ne viene quindi estratta la struttura ad albero che in futuro chiameremo HTML tree, da questa

vengono individuati i tipi di tag che all'interno dell'albero rappresentano dei nodi ed un numero di semantic tokens che all'interno dell'albero sono rappresentati dai nodi foglie.

A questo punto lo sviluppatore di wrapper aiuta il **region extractor** del sistema ad individuare le regioni semanticamente interessanti. Nella maggior parte dei casi queste regioni sono rappresentate da tabelle o da liste o da qualche altro tipo di struttura.

Il **semantic tokens extractor** effettua i collegamenti tra i contenuti delle foglie che sono legate da significati comuni e stabilisce le prime associazioni tra il contenuto delle foglie e la struttura del documento. Attraversando l'intero albero il semantic tokens extractor produce come output un set di regole di estrazione delle informazioni (extraction rules) al fine di riunire tutti i semantic tokens di interesse in un unico **comma delimited file** e d'altro canto di inserire la struttura gerarchica del documento in un XML template file.

In breve, l'algoritmo sviluppato, data una pagina con tutte le regioni identificate (tabelle, sezioni di testo, intestazioni), manda in output l'XML template file corrispondente.

A causa del fatto che le euristiche utilizzate per identificare sezioni e intestazioni possono avere eccezioni per alcune sorgenti di informazioni, è possibile che il sistema commetta degli errori quando tenta di individuare la struttura gerarchica di una nuova pagina. Ad esempio basandosi sull'euristica della dimensione del font, il sistema può identificare alcune parole o frasi come titoli o intestazioni quando non lo sono, o vice versa non riconoscere parole o frasi che lo sono ma non soddisfano nessuna delle espressioni regolari predefinite. Questi problemi sono stati previsti nella fase di progettazione di XWRAP Original e sono quindi stati forniti strumenti che permettono al wrapper's programmer di correggere interattivamente le assunzioni effettuate dal sistema quando queste non si rivelassero esatte o comunque opportune alla situazione.

C.1.2 Generatore di XML specifico della sorgente

Consiste in un XML template engine ed in un parser XML. Il primo genera statements XML usando sia il comma delimited file che l'XML template scandito attraverso l'uso del parser XML. I documenti XML templates sono file XML ben formati che contengono processing instruction e accompagnano i normali documenti XML. Tali istruzioni sono utilizzate per indirizzare il template engine nella posizione in cui i campi contenenti i dati dovrebbero essere inseriti nel template. In parole povere ciò significa che viene cercato all'interno del comma delimited file un campo con un nome che è stato specificato ed una volta trovato i dati contenuti in questo campo vengono inseriti nel punto in cui la processing instruction è stata invocata mediante il seguente statement

$$<?XG - InsertField - XG''NomeCampo''?> .$$

Un XML template spesso contiene parti ripetitive, l'inizio e la fine di queste parti viene indicata da un altro tipo di processing instruction avente il seguente statement

$$<?XG - Iteration - XG''NomeIterazione''?> .$$

Una ripetizione può essere considerata come l'equivalente di un ciclo in un classico linguaggio di programmazione. Dopo che il template engine è arrivato alla fine di una ripetizione esso prende un nuovo record dal comma delimited file e torna indietro alla posizione di start per creare lo stesso set di tags XML visto nei precedenti passi. I nuovi dati sono inseriti nel file XML risultante.

C.1.3 Considerazioni su XWRAP Original

La necessità di un tool quale XWRAP Original è stata messa in luce durante gli esperimenti condotti sul paniere di siti della realtà industriale modenese su cui si è testato XWRAP Elite, quando si è tentato di tradur-

re la home page di uno dei siti in esame: tessilmoda.com, trascurando il fatto che non fosse una pagina ricca di data-object. Per riuscire in questo intento sarebbe stato necessario far riconoscere ad XWRAP Elite ogni area in cui è suddiviso il sito alla stregua di un oggetto, fatto questo sarebbe stato necessario, per ogni oggetto individuato, identificare e dare significato semantico a ciascuno degli elementi componenti. Il problema è che anche ponendosi per assurdo nel caso in cui XWRAP Elite fosse in grado di separare correttamente gli oggetti all'interno della pagina, i nomi e i significati degli elementi di uno di questi oggetti dovrebbero essere attribuiti dall'operatore, e questo oggetto verrebbe poi utilizzato come riferimento per tali valori, dagli altri oggetti della pagina, oggetti che però non sono dello stesso tipo. La situazione di errore che si viene a creare è legata al fatto che XWRAP Elite è stato sviluppato per poter estrarre automaticamente il codice XML da pagine che contengono oggetti tutti dello stesso tipo, per pagine come quella in esame sarebbe stato più adatto XWRAP Original. Altro caso in cui si è sentita la necessità di un tool come XWRAP Original è stato nel momento in cui si sono incontrati problemi nell'utlizzo di XWRAP Elite, come nei casi di mancato riconoscimento del corretto separatore di oggetti. In questi casi, il fatto che lo strumento fosse completamente automatico rappresentava un'arma a doppio taglio, l'operatore infatti con alcuni input (come quelli necessari ad XWRAP Original) sarebbe potuto intervenire per guidare manualmente il toolkit verso il giusto riconoscimento, possibilità che viene preclusa dalla sua logica di funzionamento.

L'equipe di Georgia Tech ha orientato i propri sforzi verso la ricerca di tecniche di estrazione completamente automatiche, di cui si parlerà nella prossima sezione, abbandonando lo sviluppo di XWRAP Original.

C.2 Il Futuro: OminiSearch, Un Metodo Per la Ricerca di Contenuti Dinamici sul Web

E' un fatto assodato, messo in evidenza anche nei capitoli precedenti, che la velocità di evoluzione dei cambiamenti sul Web rappresenti un ostacolo non piccolo nella messa a punto di strumenti capaci di integrare informazioni da pagine Web, basati su programmi wrappers. Indipendentemente dal fatto che i wrappers siano costruiti in maniera manuale o semiautomatica attraverso l'ausilio di tools appositamente sviluppati al fine di facilitare e velocizzare il lavoro del programmatore di wrappers, l'integrazione delle informazioni è ancora troppo laboriosa e lenta, senza considerare il numero di errori introdotti. Questa scarsità di risultati degli estrattori si accentua lavorando con quei siti che cambiano molto di frequente il loro layout di presentazione sul Web. Le problematiche legate alla lentezza dell'integrazione si ripercuotono sull'efficacia delle ricerche effettuate attraverso questi sistemi, che a conti fatti rappresentano l'obiettivo principale degli studi presi in considerazione. Allo stato dei fatti quindi i vantaggi che sulla carta dovrebbe dare un sistema di integrazione dell'informazione si trovano a scontrarsi con la lentezza con cui le informazioni vengono aggiunte o aggiornate in questi sistemi e con i relativi alti costi in termini di settimane/uomo.

Sono queste alcune delle motivazioni che hanno spinto il team di sviluppo di XWRAP sulla strada dell'estrazione completamente automatica. Il sistema OminiSearch, in questo senso, rappresenta sulla carta un balzo in avanti nell'integrazione di nuove sorgenti dinamiche all'interno di un sistema di integrazione nel quale i data object di interesse sono estratti da pagine Web dinamiche senza l'intervento del programmatore. Non solo, viene inoltre garantita la capacità di estrarre informazioni anche in seguito all'evolvere della struttura dei siti, qualità questa che conferisce una certa robustezza a questo software. OminiSearch si avvicina per certi aspetti del suo funzionamento ad uno strumento messo a punto nei laboratori IBM di Almaden, il cui nome è Andes e che verrà

analizzato nel capitolo 4.

C.2.1 Architettura di OminiSearch

L'architettura di OminiSearch consiste in quattro principali componenti:

- Un motore di esecuzione delle ricerche
- Il motore di estrazione degli oggetti
- Un agente autonomo in grado di navigare il Web al fine di ricercare e categorizzare nuovi siti
- Un catalogo contestuale nel quale memorizzare i siti individuati, i contesti di interesse nei quali questi siti si vanno ad inserire e le regole di estrazione delle informazioni associate.

Gli utenti inseriscono una richiesta di ricerca e scelgono da una lista di contesti per la loro ricerca (es. ricerca generale sul Web, ricerca di libri, ricerca di notizie,etc.). Il motore di esecuzione della ricerca per prima cosa richiede al catalogo contestuale una lista di siti Web che soddisfi il contesto scelto per la richiesta. Quindi per ogni sito Web viene costruito l'URL appropriato basato sulla richiesta di ricerca dell'utente. Insieme a questo viene inoltre costruita la descrizione dell'interfaccia di ricerca del sito. Gli URL sono passati al motore di estrazione di oggetti di Omini, il quale recupera le pagine Web e restituisce gli oggetti estratti al motore di esecuzione della ricerca per la formattazione. I risultati provenienti da ogni sito Web sono raggruppati insieme ed inviati all'utente. Nel caso in cui vengano rilevati cambiamenti nel formato di una pagina o nell'interfaccia di una query ed il processo di estrazione degli oggetti fallisce, vengono automaticamente trovate nuove regole di estrazione e viene inviato un aggiornamento al catalogo contestuale.

Il componente finale del sistema OminiSearch è l'agente autonomo di navigazione (autonomous Web crawler). Esso naviga nel Web costantemente alla ricerca di nuove sorgenti che abbiano interfacce di ri-

cerca e che siano collegate ai contesti contenuti nel catalogo del sistema. In aggiunta gli utenti possono suggerire pagine statiche da integrare nel catalogo o anche definire nuovi contesti da incorporare nella meta-ricerca.

Il nucleo di OminiSearch è costituito dal sistema di generazione di wrappers XWRAP Elite, esaminato nel capitolo 2, la cui implementazione prevede quattro step (viene omessa la fase di tagging):

1. la pagina Web è preparata per l'estrazione, viene infatti normalizzata in un documento well-formed e convertita in un tag tree
2. è individuata la regione del documento in cui è allocato il contenuto principale, ed il relativo sottoalbero minimo che la include
3. si individuano i confini tra un oggetto e l'altro
4. gli oggetti sono estratti dal documento, trascurando le parti irrilevanti

Il sistema OminiSearch è scritto come un'applicazione Java a 3 livelli. Gli utenti sottopongono delle queries attraverso una maschera Web ad una servlet la quale incorpora il motore di esecuzione delle ricerche (search execution engine). La servlet usa JDBC per richiedere informazioni sul contesto al catalogo di OminiSearch (context catalogue). L'agente navigatore (Web crawler) agisce come entità indipendente, fornendo nuove sorgenti al catalogo in maniera asincrona. Le effettive capacità di OminiSearch, in termini di precisione e richiamo del componente estrattore di oggetti, sono state testate su su una serie di siti Web campioni. Pur non essendo il toolkit ancora disponibile ne è stato effettuato uno studio teorico che ne ha messo in luce interessanti prospettive.

Bibliografia

[1] B. Adelberg. Nodose - a tool for semi-automatically extracting structured and semi-structured data from text documents. *ACM SIGMOD, 1998.*

[2] N. Ashish and C.A. Knoblock. Semi-automatic wrapper generation for internet information sources. *In Proceedings of Coopis Conference, 1997.*

[3] P. Atzeni and G. Mecca. Cut and paste. *Proceedings of 16th ACM SIGMOD Symposion on Principles of Database Systems, 1997.*

[4] R. Doorenbos, O. Etzioni, and D. Weld. A scalable comparison-shopping agent for the world wide web. *Proceedings of Autonomous Agents, pagine 39-48, 1997.*

[5] D.W. Embley, Y. Jiang, and Y.-K. Ng. Record-boundary discovery in web-documents. *In Proceedings of the 1999 ACM SIGMOD, Philadelphia, Pennsylavania, USA, June 1999.*

[6] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, and A. Crespo. Extracting semi-structured data from the web. *Proceedings of Workshop on Management of Semi-structured Data, pagine 18-25, 1997.*

[7] N. Kushmerick, D.Weil, and D. Doorembos. Wrapper induction for information extraction. *In Proceedings of Int. Joint Conference on Artificial Intelligence (IJ-CAI), 1997.*

[8] L. Liu, C. Pu, and W. Han. XWRAP: An XML-enabled Wrapper Construction System for Web Information Sources. *Proceedings of the International Conference on Data Engineering, 2000.*

[9] A. Sahuguet and F. Azavant. WysiWyg Web Wrapper Factory (W4F). *Proceedings of WWW Conference, 1999.*

[10] S. Soderland. Learning to extract text-based information from the world wide web. *Proceedings of Knowledge Discovery and Data Mining, 1997.*

[11] D. Buttler, L. Liu, and C. Pu. Omini: An Object Minig and Extraction System for the Web. Technical Report, Sept. 2000. Georgia Tech, College of Computing.

[12] J.J. Higgins and S. Keller-McNulty. Concepts in Probability and Stochastic Modeling. Duxbury, 1995.

[13] O. Etzioni and D. Weld. A softbot-based interface to the internet. *Communications of the ACM, 37(7):72-76, 1994.*

[14] G. Tech. XWRAP Elite Project. `http://www.cc.gatech.edu/projects/disl/XWRAPElite`*Maggio 2000.*

[15] D. Chamberlin. Xquery 1.0: An xml query language. *W3C Working Draft, Giugno 2001.*

[16] A. Bonifati and S. Ceri. Comparative analysis of five XML query languages. *ACM SIGMOD Record, 29(1):68 79, 2000.*

[17] L. Pitt. Introductive inference, DFAs and computational complexity. *In K.P. Jantke, editor, Analogical and Inductive Inferrence, Lecture Notes an AI 397, pagine 18-44. Springer-Verlag, Berlin, 1989.*

[18] T. Goan, N. Benson, and O. Etzioni. A gammar inference algorithm for the world wide web. *An AAAI Spring Symposium on Machine Learning in Information Access, 1996.*

[19] C. Hsu and M. Dung. Generating finite-state transducers for semistructured data extraction from the web. *Informations Systems, 23(8):521-538, 1998.*

[20] I. Muslea, S. Milton, and C.A. Knoblock. A hierarchical approach to wrapper induction. *In Proceedings of the Third Annual Conference on Autonomous Agents, pagine 190-197, 1999.*

[21] S. Soderland. Learning information extraction rules for semistructured and free text. *Machine Learning, 34(1-3):233-272, 1999.*

[22] D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, Y.Ng, D. Quass, and R.D. Smith. A conceptual-modeling approach to extracting data from the web. *In Proceedings of 17th International Conference on Conceptual Modeling (ER'98), pagine 78-91, 1998.*

[23] B.A. Ribeiro-Neto, A.H.F. Laender, A. Soares da Silva. Extracting semistructured data through exemples. *In Proceeding of the 1999 ACM International Conference on Information and Knowledge Management (CIKM'99), pagine 94-101, 1999).*

[24] S. Grumbach and G. Mecca. In search of the lost schema. *In Seventh International Conference on Data Base Theory, (ICDT'99), Jerusalem (Israel), Lecture Notes in Computer Science, Springer-Verlag, pagine 314-331,1999.*

[25] V. Crescenzi, G. Mecca, and P. Merialdo. ROADRUNNER: Towards automatic data extraction from large Web sites. *In International Conf, on Very Large Data Bases (VLDB'2001), Rome, Italy, September 11-14, pagine 109-119, 2001.*

[26] A.K. Jain, N. Murty, and P.J.Flynn. Data clustering:A review. *ACM Computing Surveys, 31(3):264 323, 1999.*

[27] A.V. Oppenheim, R.W.Schafer, and J.R. Buck. Discrete-Time Signal Processing. *Prentice Hall, second edition, 1999.*

[28] R. Agrawal, C. Faloutsos, and A.N. Swami. Efficient similarity search in sequence databases. *In International Conference od Foundations of Data Organization (FODO'93), pagine 69-84, 1993.*

[29] BrightPlanet.com DeepWeb White Paper. `http://www.completeplanet.com/Tutorials/DeepWeb/index.asp`

[30] Arnaud Sahuguet and Fabien Azavant. Building Light-Weight Wrappers for Legacy Web Data-Sources Using W4F. *In Proceedings International Conference on Very Large Data Bases (VLDB), Edimburgh, Scotland, September 1999.*

[31] XHTML: The Extensible HyperText Markup Language, *W3C Recommendation, Gennaio 2000.* `http://www.w3.org/TR/xhtml1.`

[32] XML Extensible Stylesheet Language Transformations (XSLT), *W3C Recommendation, Novembre 1999.* `http://www.w3.org/TR/xslt.html.`

[33] Charles Allen. WIDL: Application Integration with XML. *World Wide Web Journal 2(4), Novembre 1997.*

[34] Web Interface Definition Language, *W3C Note, Settembre 1997.* `http://www.w3c.org/TR/NOTE-widl.`

[35] Compaq,s Web Language, Compaq Computer, `http://www.research.digital.com/SRC/WebL/index.html`

[36] Ling Liu, Calton Pu, and Wei Han. XWRAP: An XML-Enabled Wrapper Construction System for Web Information Sources. *Proc. International Conference on Data Engineering (ICDE), San Diego, California, Febbraio 2000.*

[37] Maria Luisa Borja, Tore Bratvold, Jussi Myllymaki, and Gabriele Sonnenberger. Informia: a mediator for Integrated Access to Heterogeneous Information Sources. *Proc. ACM Conference on Information and Knowledge Management (CIKM), Washington, DC, Novembre 1998.*

[38] Bruce Schechter. Information on The Fast Track, *IBM Research Magazine, 35(3): 18-21, 1997.*

[39] Marc Songini. IBM: All Searches Start at Grand Centrak Network World, *11 Novembre 1997.*

[40] HTML Tidy. `http://www.w3.org/People/Raggett/tidy/.`

[41] XML Path Language (XPath), *W3C Recommendation, Novembre 1999.* `http://www.w3.org/TR/xpath`

[42] R. Miller. Lightweight Structured Text Processing, *PhD Thesis Proposal, Computer Science Department, Carnegie Mellon University, USA, Aprile 1999,* `http://www-2.cs.cmu.edu/~rcm/papers/proposal/proposal.html`*Agosto 2000.*

[43] R. Miller, and B. Myers. Integrating a Command Shell Into a Web Browser. *In Proceedings of USENIX 2000 Annual Technical Conference, San Diego, pagine 171-182, Giugno 2000.*

[44] R. Miller, and B. Myers. Lightweight Structured Text Processing. *In Proceedings of 1999 USENIX Annual Technical Conference, Monterey, CA, pagine 131-144, Giugno 1999.*

[45] R. Miller, and B. Myers. Outlier Finding: Focusing User Attention on Possible Errors. *In Proceedings of UIST 2001, Orlando, FL, pagine 81-90, Novembre 2001.*

[46] R. Miller, and A. Myers. Multiple Selections in Smart Text Editing. *In Proceedings of IUI 2002, San Francisco, CA, pagine 103-110, Gennaio 2002.*

[47] A. Laender, B. Ribeiro-Neto, A. Silva, and E. Silva. Representing Web Data as Complex Objects. *In Proceedings of First International Conference on Electronic Commerce and Web Technologies (EC-Web 2000), pagine 216-228, Greenwich, UK, 2000.*

[48] A. Sahuguet, and F. Azavant. Building Intelligent Web Applications Using Lightweight Wrappers, *Paper, Giugno 2000,* `http://db.cis.upenn.edu/DL/dke.pdf`

[49] R. Baumgartner, S. Flesca, and G. Gottlob. Visual Web Information Extraction with Lixto. *Paper for the 27th International Conference on Very Large Data Bases (VLDB 2001), Roma, Italia, Settembre 2001.*

[50] Fetch Technologies. Technology Overview - Reliably Extracting Web Data, *White Paper, Novembre 2001* `http://www.fetch.com/whitepapers/FetchWhitePaper.doc`

[51] S. Montanari. Un approccio intelligente all'integrazione di Sorgenti Eterogenee di Informazione. Tesi di Laurea, Università di Modena, Facoltà di Ingegneria, corso di laurea in Ingegneria Informatica, 1998.

[52] A. Rabitti. Architettura di un Mediatore per un Sistema di Sorgenti Distribuite ed Autonome. Tesi di Laurea, Università di Modena, Facoltà di Ingegneria, corso di laurea in Ingegneria Informatica, 1998.

[53] S. Bergamaschi, S. Castano, S. Capitani di Vimercati, S. Montanari, and M. Vincini. An intelligent approach to information integration. *In Proceedings of the International Conference in Formal Ontology in Information Systems (FOIS'98)*, Trento, Italy, june 1998.

[54] S. Bergamaschi, S. Castano, S. De Capitani di Vimercati, S. Montanari, and M. Vincini. Exploiting schema knowledge for the integration of heterogeneous sources. *In Sesto Convegno Nazionale su Sistemi Evouti per Basi di Dati - SEBDB98*, Ancona, pages 103-122, 1998.

[55] S. Bergamaschi, S. Castano, D. Beneventano, and M. Vincini. Semantic integration and query of heterogeneous information sources. *Journal of Data and Knowledge Engineering 1*, 1999.

[56] R. Hull and R. King et al. Arpa i^3 reference architecture, 1995. Available at `http://www.isse.gmu.edu/I3_Arch/index.html`

[57] S. Chawathe, Garcia Molina, J. Hammer, K. Ireland, Y. Papakostantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneou information sources. *In IPSJ Conference, Tokio, Japan*, 1994. `ftp://db.stanford.edu/pub/chawathe/1994/tsimmis-overview.ps`

[58] H. Garcia-Molina et al. The TSIMMIS approach to mediation: Data models and languages. *In NGITS workshop*, 1995. `ftp://db.stanford.edu/pub/garcia/1995/tsimmis-models-languages.ps`

[59] Y. Papakonstantinou, H. Garcia-Molina, and J. Ullman. Medmaker: A mediation system based on declarative specification. *Technical report, Stanford University*, 1995. `ftp://db.stanford.edu/pub/papakonstantinou/1995/medmaker.ps`

[60] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediation systems. *In VLDB Int. Conf., Bombay, India*, September 1996

[61] Gio Wiederhold et al. Integrating Artificial Intelligence and Database Technology, volume 2/3. *Journal of Intelligent Information Systems*, June 1996.

[62] F. Saltor and E. Rodriguez. On intelligent access to heterogeneous information. *In Proceedings of the 4th KRDB Workshop, Athens, Greece*, August 1997.

[63] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25:38-49, 1992.

[64] N. Guarino. Semantic matching: Formal ontological distinctions for information organization, extraction and integration. *Technical Report, Summer School on Information Extraction*, Frascati, Italy, July 1997

[65] N. Guarino. Understanding, building and using ontologies. A commentary to 'Using Explicit Ontologies in KBS Development', by van Heijst, Schreiber, and Wielinga.

[66] M.J. Carey, L.M. Haas, P.M. Scwarz, M. Arya, W.F. Cody, R. Fagin, M. Flickner, A.W. Lunieski, W. Niblack, D. Petkovic, J. Thomas, J.H. Williams, and E.L. Wimmers. Object exchange across heterogeneous information sources. *Technical report, Stanford University*, 1994.

[67] M.T. Roth and P. Schwarz. Don't scrap it! a wrapper architecture for legacy data sources. *In Proc. of the 23rd Int. Conf. on Very Large Databases*, Athens, Greece, March 1995.

[68] Y. Arens, C.Y. Chee, C. Hsu, and C.A. Knoblock. Retrieving and integrating data from multiple information sources. International Journal of Intelligent and Cooperative Information Systems, 2(2):127-158, 1993.

[69] Y. Arens, C.A. Knoblock, and C. Hsu. Query processing in the sims information mediator. *Advanced Planning Technology*, 1996.

[70] A. Zanoli. Si-designer, un tool di ausilio all'integrazione di sorgenti di dati eterogenee distribuite: progetto e realizzazione. Tesi di Laurea, Università di Modena, Facoltà di Ingegneria, corso di laurea in Ingegneria Informatica, 1998.

[71] G.P. Grifa. Analisi di Affinità Strutturali fra classi ODL_{I^3} nel Sistema MOMIS. Tesi di Laurea, Università di Modena, Facoltà di Ingegneria, corso di laurea in Ingegneria Informatica, 1999.

[72] A. Zaccaria. Momis: Il componente query manager di momis: utilizzo della conoscenza estensionale. Tesi di Laurea, Università di Modena, Facoltà di Ingegneria, corso di laurea in Ingegneria Informatica, 1998.

[73] M. Franceschi. Il componente query manager di momis: utilizzo della conoscenza estensionale. Tesi di Laurea, Università di Modena e Reggio Emilia, Facoltà di Ingegneria, corso di laurea in Ingegneria Informatica, 2000.

[74] Domenico Beneventano, Sonia Bergamaschi, Claudio Sartori, and Maurizio Vincini. ODB-QOPTIMIZER: A tool for semantic query optimization in oodb. *In Int. Conference on Data Engineering - ICDE97*, 1997. `http://sparc20.ing.unimo.it`

[75] D. Beneventano, S. Bergamaschi, C. Sartori, and M. Vincini. Odb-tools: a description logics based tool for schema validation and semantic query optimization in object oriented databases. *In Sesto Convegno AIIA* - Roma, 1997.

[76] A.G. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39-41, 1995.

[77] S. Castano and V. De Antonellis. Deriving global conceptual views from multiple information sources. In *preProc. of ER'97 Preconference Symposium on Conceptual Modeling, Historical Perspectives and Future Directions*, 1997.

[78] F. Guerra. MOMIS: il wrapper per sorgenti di dati XML. Tesi di Laurea, Università di Modena e Reggio Emilia, Facoltà di Ingegneria, corso di laurea in Ingegneria Informatica, 2000.