POLITECNICO DI TORINO

III Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

Multilingual support to a proposed Semantic Web architecture

Relatori:
prof. Fulvio Corno
ing. Laura Farinetti

Candidato:
Andrea Ferrato

Novembre 2004

INDICE

Pagina

Indice delle tabelle

Pagina

Indice delle figure

Pagina

Parte prima
Testo in italiano

CAPITOLO 1

INTRODUZIONE

La presente tesi ha l'obiettivo di fornire una possibile soluzione per il supporto a più lingue all'interno di una generica architettura Semantic Web. Mentre le soluzioni concettuali proposte puntano alla generalità (al fine di poter risultare interessanti anche per altre, analoghe applicazioni), l'implementazione pratica è obbligatoriamente riferita alla particolare piattaforma preesistente (DOSE), sulla quale tali soluzioni sono state realizzate. Questa introduzione riprende innanzitutto il contesto in cui si è operato (costituito dalla combinazione di Semantic Web e multilingualità, nonché DOSE) e prosegue illustrando più dettagliatamente gli obiettivi, le soluzioni proposte per conseguirli ed infine i risultati ottenuti tramite verifica sperimentale.

1.1 Semantic Web

Il Semantic Web rappresenta l'evoluzione più auspicabile per l'odierno World Wide Web, e si propone come risposta al problema (ormai evidente) di scarsa strutturazione dei contenuti nei documenti disponibili in rete; una limitazione che diventa proibitiva quando si ipotizza che di tali contenuti potrebbero usufruire anche "agenti" non umani, automatizzando così parecchie operazioni finora sempre demandate agli utenti. Di fatto, gli strumenti finora disponibili alla creazione di documenti per il Web hanno offerto potenzialità rivolte soprattutto alla presentazione dei contenuti, e non alla loro strutturazione: in buona sostanza, i dati presenti in rete rimangono (tuttora in gran parte) inaccessibili ed incomprensibili agli agenti informatici. Poiché una nuova stesura in versione strutturata dell'ormai enorme Rete è

sicuramente impensabile, il Semantic Web cerca di proporsi con tecnologie e raccomandazioni che ne permettano la realizzazione semplicemente appoggiandosi ai documenti attualmente disponibili; in pratica, questo intento si risolve nell'arricchire tali documenti con metadati ("annotazioni") che ne descrivano la semantica. Secondo il suo principale teorizzatore Tim Berners-Lee [1], il Semantic Web dovrebbe garantire il rispetto di due princìpi fondamentali, questi ultimi fortemente indicativi dell'ideologia ad esso sottostante:

- *Universalità*. Non deve esistere discriminazione di sorta fra documenti prodotti in diversi contesti culturali o di taglio estetico differente; analogamente, i metadati preposti a completarli dovrebbero annullare le differenze tra risorse rivolte ad utenti umani e risorse rivolte ad agenti informatici;
- *Decentralizzazione*. Le tecnologie a supporto del Semantic Web dovrebbero garantire un compromesso accettabile tra metadati personalizzabili ed interscambiabili.

Per la creazione e l'interpretazione dei suddetti metadati occorrono precise funzionalità, qui riassunte con i formalismi atti a realizzarle:

- *Strutturazione dei contenuti: XML*. Al fine di isolare singoli elementi di rilevanza semantica (testuali e non) contenuti in una pagina Web, è possibile ricorrere ad un particolare linguaggio di marcatura come eXtensible Markup Language [2]. Esso è completamente personalizzabile e permette la definizione di tags su misura per la preparazione di insiemi di entità semantiche, con le quali è possibile organizzare il contenuto dei propri documenti in modo coerente;

- *Identificazione delle risorse: URI.* Qualsiasi entità presente nel Web la cui semantica si dimostri definibile può essere chiamata "risorsa"; la risorsa rappresenta perciò l'unità di dato fondamentale per il Semantic Web. Poiché possono esistere più istanze dello stesso tipo di risorsa, è necessario un sistema di identificazione univoco per distinguerle, che qui è rappresentato da Uniform Resource Identifier [6];
- *Creazione di relazioni logiche: RDF.* Le entità isolate con marcatura XML sono comunque prive di proprietà logiche che permettano ad un elaboratore di effettuare collegamenti ed operazioni fra loro. Resource Description Framework [7] offre un meccanismo estremamente semplice per l'asserzione delle suddette proprietà, che si realizza attraverso le cosiddette "triple RDF". Una tripla si compone di soggetto, oggetto e predicato; quest'ultimo stabilisce una relazione logica tra i primi due. È scontato ricordare che ognuno di questi tre elementi può essere rappresentato da un URI, come dimostra il seguente esempio in cui sono inseriti tutti i formalismi menzionati finora:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/
  1999/02/22-rdf-syntax-ns#"
  xmlns:verbi="http://dizionario.org/verbi/">
<rdf:Description
rdf:about="http://www.homepages.net/dan_hibiki">
<verbs:lavoraPer
```

```
rdf:resource="http://www.bose.com/" />
</rdf:Description>
</rdf:RDF>
```

La tripla RDF appena presentata può essere serializzata come "Dan (*soggetto*) lavora per (*predicato*) BOSE (*oggetto*)"; la composizione di più triple RDF come questa permette di creare una fitta rete di relazioni logiche tra risorse, che potrebbero trasformare il Web in un enorme database. In virtù di questa organizzazione dei dati, agenti informatici preposti potrebbero navigare, ricercare, verificare i dati in modo del tutto autonomo;

- *Conoscenza pre-acquisita: Ontologie.* L'impossibilità di controllare il processo di espansione ed evoluzione delle risorse in Rete oppone ulteriori ostacoli agli agenti incaricati di sfruttare le peculiarità del Semantic Web:
  - è decisamente improbabile attendersi sempre la stessa definizione semantica per lo stesso tipo di risorse (decentralizzazione);
  - le risorse in rete subiscono di norma cambiamenti frequenti, complicando il mantenimento di riferimenti stabili;
  - la capacità umana di ragionare contestualmente e sulla base di conoscenze pregresse è totalmente assente in un elaboratore.

  La soluzione più generica (e più gettonata) a queste incongruenze è rappresentata dall'uso delle cosiddette "ontologie" [8]. In questo contesto, un'ontologia si definisce semplicemente come una struttura gerarchica rappresentabile con un grafo, i cui nodi sono costituiti da singoli concetti (tutti

solitamente appartenenti ad un certo dominio di conoscenza), tra i quali gli archi stabiliscono precise relazioni di inferenza. In questo modo, gli agenti possono associare le risorse esistenti ad una o più classi dell'ontologia, così che una navigazione funzionale della stessa permetta loro di compiere (per ora) semplici ragionamenti su di esse. Tra i linguaggi necessari alla costruzione di ontologie, citiamo in ordine di complessità RDF schema [9], DAML+OIL [10], OWL [11].

1.2 Multilingualità

Nell'ambito di applicazioni software come quella qui considerata, si può intendere la multilingualità come "l'estensione a più linguaggi di operazioni già eseguite in un contesto monolingue"; anche a causa di fenomeni di integrazione tra culture a livello globale (di cui Internet potrebbe costituire un esempio), la gestione automatica della multilingualità sembra ormai diventata una necessità comune. L'area di ricerca da cui partire per il suo sviluppo è senza dubbio quella più generica di NLP (Natural Language Processing, o Linguistica Computazionale) [12], anche se questo lavoro si concentra esclusivamente sulla trattazione di "testo in formato elettronico, espresso in due o più linguaggi naturali".

Per meglio comprendere quali sono i tipici problemi da affrontare nel recupero dell'informazione multilingue, è utile introdurli considerando il particolare punto di vista della traduzione automatica: dato un concetto espresso da un vocabolo in una lingua di partenza, potremmo definire il processo di traduzione come "il tentativo di trovare un corrispondente nella lingua di destinazione" (Ingo, citato in [13]). Una

traduzione accettabile richiede di verificare tale corrispondenza sia sul piano del significato, sia del lessico: occorre preservare il significato originale, ma questo è reso più efficacemente anche tramite attente scelte lessicali. In questo passaggio, sono principalmente tre le possibili incongruenze da affrontare:

- *Vuoti lessicali* [14]. Questo inconveniente si presenta nel momento in cui la lingua di partenza esprime un determinato concetto con una singola parola, mentre quella di arrivo lo rende con una combinazione di più vocaboli: si consideri ad esempio la parola italiana "anagrafe", che in inglese sarebbe tradotta come "register of births, marriages and deaths". Lo stesso problema riemerge se si cerca di tradurre un vocabolo che sottende più significati, i quali si esprimono in modo diverso nella lingua di destinazione: per esempio, l'inglese "wood" in italiano può diventare tanto "legno" quanto "bosco";
- *Ruolo del contesto* [12]. La traduzione di una frase non può prescindere dal contesto in cui essa è inserita; parimenti, le parole di tale frase non vanno considerate come entità a sé stanti da processare singolarmente, ma piuttosto come un unico "corpo" che trasmette un certo significato, in virtù della peculiare disposizione ed associazione delle proprie componenti. Ovviamente il "contesto" inizialmente citato si potrebbe anche intendere come culturale: un concetto ben noto in una lingua potrebbe mancare del tutto in un'altra;

- *Mancanza di conoscenze pre-acquisite* (Kay, [12]). L'interpretazione di un testo richiede spesso al lettore umano di riconsiderarlo sulla base di conoscenze preesistenti, al fine di cogliere le opportune implicazioni; poiché non possiamo certamente pretendere un simile innatismo da un elaboratore, occorrerebbe affrontare i problemi legati ad una sua possibile implementazione.

Dato l'attuale disordine della Rete, anche il processo di "recupero dell'informazione in più lingue" deve affrontare i problemi sopra indicati, considerandone in particolare l'influenza sull'interpretazione del testo. Tale attività si dimostra di fondamentale interesse per questa tesi nella particolare accezione di "Multilingual Text Retrieval", che ufficialmente si definisce come "la selezione di documenti rilevanti in collezioni che comprendono più lingue" [15].

1.3 Multilingualità e Semantic Web

Per combinare supporto a più lingue e Semantic Web in modo efficace, è necessario innanzitutto conoscere il modo in cui una generica architettura Semantic Web interagisce con risorse monolingue: le due operazioni fondamentali in questo contesto sono "indicizzazione" e "ricerca". Con la prima, identifichiamo un processo che riceve un certo numero di risorse Web quale input e le analizza, tipicamente basandosi sulla semantica di ontologie a sua disposizione. Ogni occorrenza di concetto noto a tale ontologia produce un'annotazione (collegamento tra contenuto e locazione) che viene conservata all'interno del sistema; l'iterazione di questi passi su un certo numero di documenti produce una banca dati, disponibile all'esplorazione tramite il processo di ricerca. Quest'ultimo (in opposizione al precedente) può essere pilotato direttamente

dagli utenti finali tramite queries semantiche, che permettono la navigazione dei collegamenti appena creati e il recupero delle risorse di interesse. In un simile quadro, l'informazione di linguaggio può essere semplicemente introdotta come parametro addizionale per tutte le operazioni. Risulta dunque importante:

- Riconoscere la lingua di espressione delle risorse da indicizzare;
- Associare contenuti a risorse in modo indipendente dal linguaggio.

Il riconoscimento della lingua deve necessariamente precedere l'indicizzazione, al fine di preparare il sistema alle corrette corrispondenze tra lessico e semantiche (dipendenti dal linguaggio) per la creazione di annotazioni; ogni annotazione conserverà un attributo di linguaggio, ad uso e consumo del motore di ricerca. Due sono le situazioni prefigurabili in questo caso:

- La lingua di espressione è dichiarata dalla risorsa secondo uno degli standards previsti [17] [18], dunque è disponibile al recupero;
- L'informazione di linguaggio non è fornita dalla risorsa: dev'essere quindi ricreata per mezzo di opportune strategie [12].

Per quanto riguarda la creazione di annotazioni indipendenti dal linguaggio, l'operazione è influenzata soprattutto dai meccanismi utilizzati per modellare e gestire la semantica e le rappresentazioni lessicali all'interno del sistema. L'interesse per questo genere di performance deriva da uno dei princìpi cardine del Semantic Web (universalità): in generale, risorse espresse in un determinato linguaggio dovrebbero essere rese disponibili a utenti di qualsiasi altra lingua, superando per quanto possibile eventuali situazioni di incompatibilità. Va investigata la relazione tra forma (dipendente dal linguaggio) e significato (solitamente univoco) dei dati testuali, al fine di proporre il

modello migliore per una gestione dei contenuti indipendente dalla lingua. Poiché ad ora lo strumento più indicato per modellare la conoscenza all'interno di un agente Semantic Web sembra essere l'ontologia, va messo a punto (a partire da prototipi esistenti) un modello di ontologia che ottimizzi la gestione dell'informazione multilingue; sono principalmente tre i canoni da cui trarre spunto per la sua creazione [13]. Nelle ontologie "conceptual", prodotto della ricerca nel campo dell'intelligenza artificiale, vi è separazione totale tra semantica e lessico; una "conceptual ontology" riflette la struttura di una conoscenza intesa a livello globale, senza coinvolgere esplicitamente il linguaggio naturale nella sua descrizione. In diretta opposizione troviamo le ontologie "language-based", che sono fondate proprio sull'evidenza linguistica; un esempio estremizzato di ontologia "language-based" è rappresentato da reti semantiche come English WordNet [19], dove la lingua di riferimento per la definizione dei contenuti è addirittura una sola. Un modello di ontologia che invece combina efficacemente l'attenzione al lessico ed alla semantica è invece quello "interlingua": per la traduzione automatica, un'"interlingua" è una rappresentazione intermedia del significato (ad esempio) di un testo, che funge da "ponte" tra due o più linguaggi. Pur essendo teoricamente perfette, nella pratica tali ontologie si sono spesso dimostrate neutrali solo ad alcuni linguaggi.

Per gli obiettivi in gioco, una buona ontologia multilingue dovrebbe fornire una rappresentazione efficace delle cosiddette "semantiche lessicali" [13], tale per cui i concetti e le definizioni proposte dovrebbero dimostrarsi ugualmente significativi per tutte le lingue. Quest'ontologia dovrebbe perciò garantire tre caratteristiche essenziali:

- *Coerenza*. La validità dell'ontologia dovrebbe essere provata dall'evidenza linguistica;
- *Flessibilità*. I concetti definiti al suo interno dovrebbero avere generalità sufficiente per risultare compatibili con (quasi) tutti i linguaggi supportati;
- *Estendibilità*. La necessità di allargare il supporto ad un'ulteriore lingua non dovrebbe comunque obbligare ad interventi radicali sull'ontologia.

Un modello di ontologia ottimizzata per il supporto multilingue verrà illustrata più avanti, sulla base delle funzionalità preesistenti in DOSE.

1.4 DOSE

Si rende a questo punto necessario introdurre DOSE (Distributed Open Semantic Elaboration platform), la piattaforma Semantic Web che ha costituito il "terreno di prova" per le soluzioni proposte in questo lavoro. DOSE propone un approccio alquanto classico per l'implementazione del Semantic Web, cercando però di recuperare ed ottimizzare tutte le migliori caratteristiche di sistemi analoghi già proposti [20÷31]. Il sistema permette la creazione automatica di annotazioni (metadati a corredo delle risorse esistenti) sulla base di un'ontologia interna che ne modella il/i dominio/i di conoscenza; tali annotazioni possono essere riferite non solo ad interi documenti, ma anche a singoli frammenti od elementi di sottostruttura a loro interni. Questo accorgimento permette in primo luogo di raffinare il processo di indicizzazione ottenendo annotazioni più mirate, ma consente anche di comporre risultati di ricerca personalizzati per l'utente, concatenando più frammenti (ognuno da fonte diversa, ma rispondente ai contenuti cercati) in un singolo documento di output. Le annotazioni

sono costituite da sequenze di triple RDF che permettono la definizione di numerosi attributi in aggiunta a quelli di contenuto (es. URIs ed Xpointers [34] per il tracciamento delle risorse), inclusi quelli per l'organizzazione tassonomica: ne consegue che DOSE permette di strutturare anche le annotazioni come un'ontologia, sfruttando così tutti i vantaggi che ne derivano.

La scelta di collocare tali annotazioni in un deposito esterno al sistema ed universalmente accessibile consente l'eliminazione di informazioni ridondanti; tale deposito rappresenta un ottimo esempio dell'approccio distribuito seguito per implementare l'intero sistema. Sin dalla prima versione, infatti, DOSE è costituita da moduli indipendenti scritti in linguaggio Java (scelto per l'alta portabilità e la disponibilità di diverse APIs utili al progetto) che comunicano tramite messaggi formalizzati secondo il protocollo Apache XML-RPC [35]: quest'approccio dà migliori garanzie per un'implementazione scalabile che consenta di centralizzare più componenti di sistema possibili.

I moduli costituenti l'architettura sono i seguenti:

- *Substructure Extractor*. Insieme a Fragment Retriever, interfaccia il sistema con risorse ed Xpointers; effettua la frammentazione dei documenti secondo insiemi predefiniti di tags che ne delimitano la struttura. Quando un documento viene temporaneamente fornito in ingresso, i singoli frammenti da analizzare sono identificati per mezzo dei tags succitati e tradotti nei corrispondenti Xpointers; in questo modo, se ne individua l'ordinamento gerarchico;
- *Fragment Retriever*. Espone la primitiva `extract(Xpointer,URI)` (accessibile in remoto), che permette di recuperare un preciso frammento in una

risorsa utilizzando una coppia (URI, Xpointer). Restituisce un file XML valido contenente il frammento puntato;

- *Indexer*. Coordina l'intero processo di annotazione sull'insieme di risorse specificato. Partendo dallo URI di ogni risorsa, interagisce con Substructure Extractor e Fragment Retriever al fine di ottenerne una collezione di frammenti descritti da coppie (URI, Xpointer). Ogni frammento viene dunque passato al Semantic Mapper che lo associa (per correlazione semantica) ad un insieme di concetti pesati, producendo altrettante annotazioni;
- *Annotation Repository*. È il deposito destinato a contenere le annotazioni appena descritte; è coinvolto sia nella conservazione, sia nel recupero delle stesse. Il recupero viene compiuto navigando il deposito in cerca di sovrapposizioni (almeno parziali) con i contenuti cercati, utilizzando metodi per il recupero dell'informazione [33] atti a verificarne l'effettiva rilevanza. Viene solitamente considerato il noto sistema di classificazione "tf/idf" [36] in combinazione con un modello vettoriale;
- *Semantic Mapper*. Opera con un'ontologia ed un insieme di entità lessicali. L'ontologia definisce un dominio di conoscenza organizzandone i concetti costituenti; per ogni concetto, viene formato un insieme di entità lessicali e collegamenti ad ontologie correlate (es. WordNet). Un'entità lessicale può essere rappresentata da un vocabolo o da una locuzione che identificano un concetto, od a cui vengono spesso associati; l'insieme di tali entità costituisce un "synset", la cui funzione è proprio quella di fornire un collegamento tra sintassi e semantica. Il compito principale del Semantic Mapper consiste nell'associare un

frammento di risorsa od una stringa di ricerca ad un insieme appropriato di concetti; il modulo mette a disposizione anche particolari primitive per la navigazione dell'ontologia. La sua architettura è indipendente da ontologia ed entità lessicali, dunque è possibile passare ad un diverso dominio di conoscenza semplicemente selezionando un'altra ontologia;

- *Semantic Search Engine*. Permette di navigare Annotation Repository durante una sessione di ricerca per trovare annotazioni che conducano a frammenti di risorsa rilevanti, classificati secondo criteri predefiniti. Il modulo riceve queries espresse in forma di parole chiave o brevi frammenti di testo, ed interagisce con Semantic Mapper per creare una lista pesata dei concetti correlati alla query. Dunque interroga Annotation Repository per recuperare le annotazioni di rilievo, eventualmente raffinando i risultati tramite navigazione dell'ontologia. Una volta ottenuto l'insieme definitivo, viene richiamato Fragment Retriever per recuperare i frammenti corrispondenti e comporre le pagine risultanti.

L'interazione tra moduli del sistema durante due operazioni di alto livello come ricerca ed indicizzazione è illustrata nei diagrammi di Figure 4 e Figure 5, rispettivamente.

1.5 Obiettivi e soluzioni

L'obiettivo primario a cui questa tesi punta è fornire il "supporto multilingue ad architetture di elaborazione semantica" in generale. Nella pratica, l'approccio proposto cerca di sfruttare la peculiare organizzazione della piattaforma DOSE, con lo scopo di estendere alla multilingualità le operazioni comunemente eseguite in un'architettura di

questo genere, tipicamente "indicizzazione" e "ricerca". Nel seguito, si cercherà innanzitutto di tradurre gli obiettivi astratti della tesi in requisiti pratici da soddisfare per implementare la multilingualità (Sezione 1.5.1); verranno dunque descritte le soluzioni adottate, elencando le ipotesi considerate durante la fase di sviluppo quando necessario (Sezione 1.5.2); infine, verranno descritte implementazione e verifica sperimentale delle idee proposte (Sezione 1.6).

1.5.1 Requisiti pratici per l'implementazione corrente

I due scenari principali in cui DOSE opera sono quelli di "indicizzazione" e "ricerca": con il primo processo vengono create annotazioni (collegamenti semantici tra concetti dell'ontologia ed elementi delle risorse), mentre con il secondo tali annotazioni vengono navigate (per recuperare quelle la cui semantica è di interesse per l'utente). Un'implementazione corretta di supporto a più lingue dovrebbe dunque tenere in considerazione soprattutto questi due ambiti operazionali, ed agire per garantire un'appropriata estensione dei suddetti al parametro addizionale di lingua.

L'introduzione del supporto a più lingue influenza radicalmente il modo in cui le annotazioni vengono create: un'annotazione è generata dall'occorrenza di un determinato concetto all'interno di una risorsa. Tale concetto – nell'ambito già inquadrato della trattazione di informazione puramente testuale – può apparire in forma di particolare rappresentazione lessicale fra le tante possibili, e queste ultime solitamente differiscono a seconda del linguaggio di espressione. La lingua in cui un documento è espresso deve dunque essere identificata prima di cominciare ad annotare, al fine di poterne analizzare il contenuto confrontandolo con il corretto insieme di entità

lessicali (proprie del linguaggio individuato) evidentemente presenti al suo interno. Inoltre, l'identificatore di lingua così ottenuto dovrebbe essere conservato in ogni annotazione come attributo addizionale, per consentire finalmente al processo di ricerca (e dunque, per ogni interrogazione alla banca di metadati semantici ottenuta) di discriminare fra annotazioni anche secondo la lingua di espressione della risorsa d'origine.

Per quanto riguarda invece le operazioni di ricerca, possiamo aspettarci una doppia specificazione di linguaggio dall'utente: una per i risultati della ricerca (lingua/e preferita/e), l'altra per la stringa di ricerca (lingua utilizzata). Se in un'operazione di ricerca è possibile sottointendere che ogni annotazione disponibile presenti un'indicazione di lingua sulla risorsa ad essa collegata, è possibile garantire (sotto opportune condizioni) un recupero delle informazioni nella lingua preferita – in sostanza, si tratta semplicemente di gestire un parametro addizionale di ricerca. Rimane il problema di interpretazione delle queries: se la lingua di una query non viene esplicitamente specificata dall'utente (il quale, per ipotesi, potrebbe selezionarla in un insieme di supportate) allora dovrebbe essere implicitamente dedotta dal testo della stringa di ricerca, possibilmente con lo scopo di associare i termini di tale stringa a contenuti effettivamente noti al sistema – i metodi per il rilevamento del linguaggio già menzionati per l'annotazione potrebbero dunque essere riapplicati, questa volta alla query string. Tutte le funzionalità da garantire ad una piattaforma Semantic Web multilingue potrebbero dunque essere riassunte come segue:

Indicizzazione:

- ottenere il linguaggio delle risorse da indicizzare:
  - quando viene dichiarato esplicitamente: identificarlo in conformità agli standards attualmente previsti per la dichiarazione di lingua all'interno di risorse Web, inclusi i casi in cui tale dichiarazione è implicita;
  - quando non viene dichiarato: utilizzare metodi alternativi per riconoscere la lingua di espressione;
- applicare le conseguenti impostazioni al sistema di indicizzazione, per assicurare la coerenza necessaria ad una corretta analisi delle risorse;
- conservare l'informazione di linguaggio come parte dei metadati associati alle risorse di origine, nella prospettiva di riutilizzarla quando necessario;

Ricerca:

- interpretare stringhe di ricerca (idealmente) espresse in linguaggio naturale dall'utente, accompagnate o meno da eventuali preferenze sul linguaggio per i risultati;
  - se per la stringa di ricerca non è presente una dichiarazione esplicita del linguaggio di espressione, attivare opportune strategie per estrarlo dal testo (possibilmente riutilizzando quelle già approntate per la fase di indicizzazione);
- applicare le conseguenti impostazioni al motore di ricerca, per consentire un'eventuale discriminazione dei risultati sulla base del linguaggio.

1.5.2 Principali soluzioni adottate

Le necessità multilinguali dell'utente finale dovrebbero essere state tradotte in requisiti da soddisfare ed estensioni da fornire al sistema: con riferimento a queste ultime, le soluzioni qui proposte si potrebbero etichettare come "estensione al linguaggio" e "riconoscimento del linguaggio". Entrambe sono presentate insieme alle ipotesi generali e particolari prese in considerazione durante l'implementazione sull'architettura.

1.5.2.1 Estensione al linguaggio

Il presente approccio si fonda su un'importante caratteristica di DOSE: l'indipendenza tra le rappresentazioni dei concetti dell'ontologia interna al sistema e dei synsets associati, rispettivamente. Questa peculiarità viene sfruttata per estendere DOSE alla multilingualità, collegando ora ogni concetto (definito come una più generica entità, slegata da particolari istanze lessicali) a un insieme di coppie (definizione, synset) specifiche per ogni lingua (Figure 6). Una definizione di concetto è una breve descrizione (formulata in linguaggio naturale) che lo identifica il più efficacemente possibile ed è espressa in un particolare linguaggio, mentre il relativo synset – come già accennato in Sezione 1.4 – è un insieme di sinonimi ed espressioni correlate che vengono usualmente adottate in linguaggio naturale per identificare quel concetto, oppure vi vengono frequentemente associate. Secondo questo nuovo modello, il synset di un concetto può variare a seconda della lingua considerata; ogni concetto può dunque essere rappresentato come:

```
concept := concept_id, lexical_representation
lexical_representation := (lang_id, description, synset)+
synset := (word)+
```

L'ontologia è fisicamente distinta dalle possibili definizioni di concetti e relative istanze lessicali, permettendo così la gestione separata dei livelli semantico (od "interculturale") e testuale (più spiccatamente dipendente dalla lingua). Specifiche lacune di un particolare linguaggio in una particolare area semantica sono gestite in modo semplice e trasparente: per i concetti che lo richiedano, sono incluse solamente definizioni e synsets nelle lingue che li supportano. In definitiva, l'estensione proposta garantisce sufficiente potenza espressiva per modellare le entità concettuali e lessicali tipiche di ogni linguaggio, ma allo stesso tempo minimizza le ridondanze condensando tutti i concetti comuni all'interno di una singola struttura multilingue.

Per quanto riguarda la creazione (e la manutenzione) del complesso di ontologia e gruppi di definizioni e synsets, appoggiarsi a reti semantiche esistenti come English WordNet [19] o derivate potrebbe evitare molti sforzi implementativi necessari a costruire dal nulla un'ontologia. Tuttavia, l'insieme attuale di reti semantiche derivate da WordNet è costituito da ontologie e synsets costruiti su misura per ogni lingua (e cultura), e pertanto non si accorda alla perfezione con l'approccio centralizzato qui proposto. Sarebbe comunque possibile utilizzarne i synsets per espandere quelli disponibili al sistema, migliorando così l'efficacia dei processi di annotazione e di ricerca.

1.5.2.2 Riconoscimento del linguaggio

Le funzionalità di riconoscimento del linguaggio si dimostrano necessarie soprattutto durante la fase di annotazione, nel momento in cui vengono creati i collegamenti semantici tra gli elementi dell'ontologia di sistema e i frammenti di risorse disponibili al processo di indicizzazione. Le risorse sono analizzate considerando un singolo frammento della loro struttura per volta; poiché ognuno di questi potrebbe teoricamente essere espresso in un linguaggio differente, è necessario analizzarli separatamente. I metodi preposti al riconoscimento del linguaggio dovrebbero dunque essere riapplicati ad ogni nuovo frammento di documento che viene sottoposto al sistema.

L'idea chiave è quella di recuperare la dichiarazione di linguaggio ogni volta che essa è direttamente fornita con le risorse da analizzare, per evitare l'applicazione di euristiche di interpretazione e sfruttare un dato già disponibile. Quando non sono date indicazioni in merito, viene attivato un metodo di analisi appropriato che possa ricreare l'informazione mancante – ovviamente, se anche questo sistema fallisce nel fornire un'indicazione di linguaggio univoca per il frammento correntemente analizzato, ne viene adottato uno predefinito. L'intero procedimento è dettagliato nei punti seguenti, ognuno dei quali descrive una soluzione diversa per ottenere il linguaggio; le soluzioni sono proposte in ordine di applicazione, si intende perciò che il fallimento di una tattica risulti automaticamente nell'invocazione della successiva:

1. *Convalidare una precisa richiesta dell'utente.* Supponendo che l'utente conosca in partenza la lingua di espressione delle risorse da indicizzare, sembra appropriato attribuirvi la massima priorità. Oltre a rappresentare il caso più

banale ed a minor costo per il sistema, una forzatura da parte dell'utente costituirebbe una sorta di "filtro" preliminare sulle annotazioni: sarebbe ad esempio possibile risparmiare tempo di elaborazione annotando solo la metà di interesse in un insieme di risorse bilingue. In ogni caso, non è ancora chiaro se una simile opzione – così come il processo di annotazione stesso – debba essere visibile o trasparente all'utente: quest'ultimo potrebbe ammettere di avere a disposizione un certo numero di annotazioni periodicamente aggiornate in modo autonomo dal sistema, ed avere accesso diretto solamente al motore di ricerca;

2. *Decifrare la dichiarazione di lingua data.* Dopo possibili forzature dell'utente, questo è il metodo più semplice per ottenere il linguaggio. Il sistema dovrebbe infatti essere in grado di riconoscere ed interpretare i formalismi preposti all'indicazione di lingua nelle risorse, al fine di evitare per quanto possibile l'utilizzo di (più onerose) euristiche. Cercando di privilegiare la conformità agli standards, è stato scelto di considerare (almeno per il momento) l'attributo "lang" per tags XHTML [18]. Questa scelta è motivata dalle seguenti ragioni:
   - è specifico per ogni singolo tag: in sostanza, permette di specificare una lingua diversa per ogni singolo frammento (diversamente ad esempio da "Content-language" [17], che fornisce informazioni solo sull'intero documento). Questa peculiarità soddisfa pienamente i requisiti sul livello di raffinamento richiesto per le annotazioni, in quanto per ogni frammento il sistema sarà in grado di distinguere anche il linguaggio;

- è orientato verso applicazioni come quella in questione, anziché essere indirizzato specificamente a particolari agenti, come "Content-language" (pensato per l'interpretazione da parte di un browser).

Ponendo che l'attributo di lingua debba essere rideterminato ogniqualvolta un nuovo frammento viene indicizzato, se il frammento dichiara l'attributo "lang", per semplicità viene preso in considerazione solo il primo valore;

3. *Riconoscere il linguaggio tramite euristiche.* Questa soluzione consiste di una semplice analisi del testo di ogni frammento, volta a dedurne la lingua di espressione. Nella presente implementazione, ciò avviene recuperando entità lessicali denominate "stopwords": tra esse annoveriamo elementi funzionali (come articoli, congiunzioni, preposizioni, ecc.) ma anche vocaboli e lessico di uso comune (come coniugazioni di verbi ausiliari). Le stopwords presentano alcune caratteristiche essenziali che le rendono particolarmente adatte allo scopo:
   - Sono (solitamente) esclusive della lingua a cui appartengono. Ad eccezione di rare sovrapposizioni, queste particelle lessicali sono in assoluto le "meno condivise" fra lingue diverse, tanto da rappresentarne in qualche modo una delle componenti più caratteristiche. Possono presentarsi rari casi di ambiguità, per i quali si rendono necessarie ulteriori distinzioni; segue un esempio, in cui le stopwords appaiono sottolineate:

     <H1>A day in Chicago is never enough.</H1>

La frase è palesemente espressa in inglese, ma se consideriamo il punto di vista di un riconoscitore di linguaggio che analizza la stringa da sinistra a destra in cerca di stopwords, notiamo che incontrerebbe "A", "in", "is" nell'ordine; sia "A", sia "in" potrebbero condurre entrambe a due lingue diverse, inglese ed italiano. Dunque il riconoscitore non potrebbe individuare la lingua di espressione della frase dopo due stopwords su tre, poiché a quel punto i due linguaggi dimostrerebbero uguale occorrenza – la chiarificazione arriva con la terza stopword "is", che porta definitivamente a scegliere "inglese". L'esempio è basato su una frase molto breve (lontana dagli ordini di grandezza di tanti documenti in rete), ma dimostra che "sovrapposizioni" tra lingue pur di origini molto diverse (come inglese ed italiano) possono essere aggirate semplicemente conteggiando le stopwords; nel caso in cui non fosse ancora sufficiente, sarebbe possibile prendere in considerazione l'analisi di un singolo documento secondo più lingue contemporaneamente;

- hanno in media alte occorrenze. A causa del ruolo fondamentale nello strutturare frasi e periodi, occorrono pesantemente anche entro brevi porzioni di testo: spesso l'analisi di una minima parte di ogni frammento è sufficiente allo scopo.

Dopo aver analizzato una porzione ragionevole di testo, otterremo una lista di linguaggi la cui presenza in esso è proporzionale al numero di relative stopwords trovate: il linguaggio con occorrenza maggiore viene scelto per l'annotazione. Questo metodo potrebbe dimostrarsi inefficace solo nei casi (peraltro poco significativi) in cui la natura del testo analizzato non prevedesse la presenza di

stopwords (es. elementi vuoti, liste di parole chiave, ecc.). Euristiche come questa tentano ovviamente di affrontare il problema di scarsa conformità agli standards di dichiarazione del linguaggio nella media delle risorse attualmente reperibili in rete;

4. *Assumere il linguaggio del frammento antenato*. Ogni frammento la cui lingua non è specificata e nemmeno ricavabile, può "ereditarla" dall'antenato più prossimo (nella struttura gerarchica in cui sono organizzati i frammenti di documento) cui è stata assegnata in precedenza. L'accorgimento adottato – che potrebbe sembrare artificioso – si fonda su un presupposto decisamente ragionevole: se un certo frammento di documento è scritto in una certa lingua, è molto probabile che i sottoelementi discendenti facciano uso della stessa lingua. Questa disposizione si limita in realtà ad attuare le specifiche già previste di regola per l'attributo "lang"; l'unica differenza sta nel fatto che qui l'attributo di linguaggio ereditato può avere origini diverse da "lang". L'algoritmo proposto si appoggia ai meccanismi di frammentazione utilizzati dai moduli Substructure Extractor e Fragment Retriever, che nell'estrazione dei frammenti cominciano dal più alto livello gerarchico di documento e poi continuano verso i più bassi recursivamente, utilizzando un approccio top-down [37]: in questo modo, è garantito che la lingua di espressione per i frammenti a più alto livello sia sempre determinata per prima. L'utilizzo di quest'ultima strategia porta comunque ad un risultato: il processo continua a salire nella gerarchia di documento finché non ottiene risultati soddisfacenti, e se ciò non avviene

nemmeno al vertice (condizione decisamente poco probabile) ritorna un linguaggio predefinito originariamente scelto per il sistema.

1.6 Implementazione

1.6.1 Contesto tecnico

Allo stato attuale, è disponibile un prototipo di DOSE che implementa tutte le funzionalità più importanti prospettate nella fase di progetto teorico, anche se la piattaforma è tuttora in continuo sviluppo.

Ogni modulo è in effetti un server XML-RPC indipendente, che incapsula tanti clients quanti sono i servizi esterni ai quali ha necessità di accedere. Substructure Extractor e Fragment Retriever sono per ora sviluppati utilizzando l'API XPath Explorer [38], che permette di estrarre i descrittori XPath dalle risorse da indicizzare per etichettarne i singoli elementi di sottostruttura. La frammentazione si basa sul riconoscimento di tags specifici come <H1,..,Hn> in (X)HTML e custom tags in XML; sono supportati documenti XML ed XHTML (compresi i documenti HTML convertiti in questo formato utilizzando l'API Tidy [39]). Le annotazioni sono composte di diversi campi, uno dei quali punta al frammento antenato espresso come Xpath (Figure 7); se una singola sessione di ricerca restituisce più annotazioni concernenti uno stesso documento, l'annotazione riferita all'elemento antenato viene recuperata tramite generalizzazione. Semantic Mapper è stato implementato utilizzando l'API Jena [40] per la gestione e la navigazione dell'ontologia, e l'API Snowball [41] per la creazione di associazioni tra semantica e lessico, nonché per lo stemming lessicale (componente fondamentale per l'internazionalizzazione della piattaforma). Il Search Engine offre

un'implementazione provvisoria, che punta solamente a dimostrare l'efficacia del processo di annotazione. Non sfrutta ancora le potenzialità della semantica, ma si comporta semplicemente come un'interfaccia verso Annotation Repository: di conseguenza, non sono stati condotti tests sulle queries multilingue. Tale motore può ricevere un breve frammento di testo in input, che trasforma in concetti dell'ontologia utilizzando i servizi offerti da Semantic Mapper; recupera dunque le relative annotazioni da Annotation Repository e le ordina per rilevanza secondo i parametri specificati (es. peso); infine, inoltra solo un sottoinsieme selezionato di esse al Fragment Retriever per il recupero delle relative risorse.

1.6.2 Il problema della consapevolezza del linguaggio

Uno dei primi problemi emersi durante l'implementazione della multilingualità in DOSE ha riguardato la collocazione ed il controllo dell'informazione di linguaggio all'interno del sistema. Non a tutti i moduli della piattaforma è necessaria la conoscenza della lingua: in generale, tale informazione scorre dai moduli di più alto livello a quelli di livello più basso, filtrata ad ogni passaggio intermedio. Per questo motivo, il sistema è stato riorganizzato in tre livelli di profondità, ognuno dei quali esibisce un comportamento diverso nei confronti del linguaggio. Il primo livello, detto "Service layer", include i moduli direttamente a contatto con il mondo esterno, come Indexer e Search Engine; qui ogni modulo dovrebbe essere in grado di trattare l'informazione di lingua. In particolare, Indexer dovrebbe sfruttarla al fine di creare annotazioni coerenti, mentre per Search Engine potrebbe arricchire le funzionalità di ricerca offerte all'utente. In opposizione al precedente, il livello "Kernel Back-end layer" è sostanzialmente

rappresentato dai cosiddetti wrappers (nucleo del sistema) che incapsulano ontologia, synsets e definizioni. Questo livello non necessita della consapevolezza del linguaggio, tuttavia i wrappers di synsets e definizioni dovrebbero incorporare un attributo di lingua per permetterne il selezionamento. Tutti i restanti moduli sono inseriti nel livello intermedio "Kernel Front-end layer": tra questi, solo Annotation Repository e Semantic Mapper necessitano di conoscere l'informazione di lingua. Il primo dovrebbe includerla come nuovo attributo nella creazione di annotazioni multilingue, mentre il secondo in realtà la utilizzerebbe come parametro per selezionare i corretti wrappers sottostanti. Una rappresentazione grafica a livelli di DOSE è data in Figure 8.

1.6.3 Estensione al linguaggio

L'estensione di DOSE a più lingue è stata ottenuta realizzando la soluzione teorizzata in Sezione 1.5.2.1 sull'architettura correntemente disponibile: una singola ontologia (descritta da un file .rdfs) è stata combinata con due insiemi distinti di synsets per le lingue Inglese ed Italiano (due files .rdf), per l'implementazione di un primo supporto bilingue: l'estensione coinvolge diversi moduli di sistema. Il synset wrapper ora contiene un campo `language` che tiene traccia dello stemmer correntemente utilizzato, e permette il caricamento di un nuovo file di synsets ogniqualvolta sia necessario. Il Semantic Mapper aggiunge un parametro di linguaggio alla primitiva `getTopicsOf`, incaricata di individuare le associazioni semantiche tra risorse ed ontologia; il cambio di synset avviene tramite il nuovo metodo privato `setupSyn (language)`. L'Annotation Repository ora inserisce una proprietà addizionale di lingua in ogni annotazione (vedi Figure 9), e fornisce un metodo `getLanguage` per discernere

tra annotazioni sulla base del linguaggio. Infine, anche Indexer gestisce un nuovo parametro di lingua nei suoi metodi, al fine di coordinare i moduli dei livelli sottostanti e fornire loro un'indicazione al riguardo in ogni fase del processo di indicizzazione.

1.6.4 Riconoscimento del linguaggio

Tutte le modifiche appena descritte sono state studiate ed implementate all'interno dell'architettura, e si fondano sul presupposto di esistenza degli identificatori di linguaggio: al fine di ottenere tali identificatori ed assicurare funzionalità trasparenti al cambio di lingua, è stato introdotto un nuovo modulo "Language Detector". Il modulo offre principalmente funzionalità di riconoscimento del linguaggio, e viene interpellato ogniqualvolta si gestisce un parametro del genere; è stato collocato a livello "Kernel Front-end layer", poiché offre servizi soprattutto al "Service layer". Il Detector espone la primitiva principale `findLanguage(text)`, disponibile tramite chiamata XML-RPC, di cui solitamente si serve Indexer per ottenere l'identificatore di linguaggio delle risorse da analizzare. Tale primitiva si appoggia ai metodi privati `lang(text)` e `detect (text)`, che implementano le strategie per il recupero del linguaggio descritte alla Sezione 1.5.2.2, punti 2 e 3 rispettivamente.

La capacità di recupero del linguaggio "antenato" risiede invece nell'Indexer, poiché tale funzionalità richiede di tenere traccia del linguaggio precedentemente attribuito ad ognuno dei frammenti antenati, e DOSE mira a garantire la scalabilità, dunque il Detector non può mantenere alcuna informazione di stato. Indexer è l'unico modulo (insieme a Search Engine) abilitato a gestire singole sessioni utente: viene dunque provvisto del metodo `ancestorLanguage(Xpath)` che recupera il linguaggio

dell'antenato più vicino all'Xpath corrente. Per questa particolare fase sono stati proposti (e implementati) due algoritmi distinti: uno utilizza Annotation Repository, l'altro si serve di uno stack contenuto nell' Indexer.

Il primo algoritmo comincia ricostruendo il corretto Xpath per l'antenato del frammento correntemente analizzato, e poi interroga Repository per recuperarne il linguaggio; se l'operazione fallisce, il procedimento ritenta recursivamente finché non ottiene una lingua valida o raggiunge il livello più alto. Il secondo algoritmo fa invece riferimento ad un "Language Stack", che tiene traccia dei linguaggi determinati lungo il percorso da radice a foglia nella gerarchia di documento, dove la foglia è il frammento correntemente analizzato. L'insieme completo di modifiche e aggiunte ai moduli di sistema è fornito in Table III.

1.6.5 Indicizzazione multilingue

Il diagramma UML di Figure 10 visualizza l'interazione tra moduli di sistema in una sessione d'indicizzazione multilingue. L'unica differenza rispetto al caso monolingue sta nell'intervento di Language Detector tra il recupero di ogni frammento e il suo invio a Semantic Mapper: se il modulo chiamante esterno non forza il linguaggio, Indexer interroga il Detector su tale frammento, inviando un messaggio `findLanguage (fragment)`. Se non gli viene restituito un valore accettabile, Indexer stesso attiva il metodo `ancestorLanguage`. L'attributo di linguaggio così ottenuto viene dunque inviato al Mapper, che selezionerà l'opportuno file di synsets; le annotazioni successivamente create conterranno l'attributo di lingua direttamente fornito da Indexer.

1.6.6 Verifica sperimentale

Per questa preliminare versione multilingue di DOSE è stata approntata una semplice verifica sperimentale, che mira a dimostrare la bontà dell'approccio seguito e sottolineare così il ruolo fondamentale svolto da questa architettura nell'implementazione del supporto a più lingue.

Il sistema utilizza un'ontologia centrata sul dominio delle disabilità (circa 500 concetti), sviluppata in collaborazione con il servizio Passepartout della Città di Torino (un servizio pubblico per l'integrazione e l'assistenza alle persone disabili in Torino), a cui sono stati affiancati – come già detto – tutti i synsets necessari nelle lingue Inglese ed Italiano. Poiché Passepartout al momento non offre una versione bilingue del proprio sito, la prima difficoltà è consistita nel reperire un'adeguata collezione di documenti da sottoporre a questa nuova versione di DOSE. In più, avendo scelto un dominio semantico così definito – sia nel lessico sia nei contenuti – come quello della disabilità per l'ontologia di prova, solo un altro sito dedicato come quello di Passepartout avrebbe potuto fornire una base dati consistente. I seguenti punti riassumono l'insieme di requisiti delineati per una collezione di documenti appropriata alla sperimentazione:

- *direttamente o indirettamente correlata alla disabilità.* I documenti scelti dovrebbero sfruttare il dominio semantico dell'ontologia utilizzata, allo scopo di massimizzare il numero di associazioni semantiche e dunque di annotazioni;
- *multilingue (od almeno bilingue).* Ovvio per le finalità della sperimentazione; poiché al momento sono disponibili i synsets di supporto all'ontologia solo in

Italiano ed Inglese, l'insieme di risorse scelto dovrebbe includere almeno queste due lingue;

- *con struttura parallela.* Potremmo definire due "documenti paralleli" come documenti aventi uguale struttura, e che presentano gli stessi contenuti espressi con lingue diverse in ogni versione parallela dello stesso frammento. Posto che si trovi un sito multilingue, in cui copie multiple dello stesso documento condividono la stessa struttura e lo stesso layout, differenziandosi solo per il linguaggio di espressione, tale sito potrebbe definirsi adeguato allo scopo. Questo requisito è indispensabile per verificare la capacità di DOSE di annotare in modo omogeneo gli stessi contenuti espressi in più lingue;
- *sufficientemente estesa.* Maggiore è il numero di risorse analizzate, maggiore dovrebbe risultare la porzione di ontologia coperta dalle relative controparti lessicali ritrovabili in tali risorse;
- *contenente testo descrittivo ed organicamente strutturato.* I documenti dovrebbero essere organizzati gerarchicamente rispetto ai contenuti presentati; ogni paragrafo dovrebbe risultare il più descrittivo possibile, al fine di costituire una rappresentazione verosimile del genere di informazioni che un utente cerca. Elenchi di links e parole chiave dovrebbero essere evitati, anche perché mancano di elementi lessicali importanti come le stopwords (vedere Sezione 1.5.2.2, punto 3).

Trovare una collezione di documenti comprendente tutte queste caratteristiche si è dimostrato non facile, specialmente in merito alla multilingualità. Spesso infatti, l'informazione riguardante il mondo delle disabilità concerne particolari disposizioni

legali od iniziative locali, che sono proprie di ogni singola giurisdizione (paese): in rete questo fenomeno si riflette nella disponibilità di parecchi siti monolingua sull'argomento, ma pochi in versione multilingue (o addirittura bilingue).

Tutte le prove sono state condotte su venti documenti provenienti da www.asphi.it [42], il sito di un'associazione italiana impegnata nel promuovere tecnologie informatiche indirizzate al supporto di persone disabili; il sito è disponibile nelle versioni (parallele) inglese ed italiana. La verifica sperimentale, così come l'implementazione, è stata divisa in due parti: la prima si occupa di testare la strategia proposta per l'integrazione semantica (estensione a più lingue) di DOSE tramite "annotazione parallela", mentre la seconda è una semplice valutazione delle prestazioni offerte dal nuovo modulo Language Detector per il riconoscimento della lingua.

1.6.6.1 Annotazione parallela

Questo test sfrutta la recente estensione di DOSE alla multilingualità, valutandone la capacità di annotare risorse in modo indipendente dalla lingua: ciò risulta dimostrato verificando che due o più insiemi di documenti fra loro paralleli sono (quasi) simmetricamente associati agli stessi concetti. I documenti in questione sono quelli già menzionati in precedenza [42], e la loro indicizzazione ha prodotto quasi 5100 annotazioni (sia per la versione inglese sia per quella italiana), totalizzando 24 Mb circa in files .rdf. Tale indicizzazione è stata eseguita specificando esplicitamente il linguaggio delle risorse (Sezione 1.5.2.2, punto 1), al fine di evitare un possibile inquinamento dei risultati dovuto a eventuali defezioni di Language Detector (i cui test specifici sono discussi nella successiva Sezione 1.6.6.2). A livello semantico, la piattaforma dovrebbe

(idealmente) rappresentare ogni coppia di frammenti con lo stesso insieme di concetti; il raggiungimento di tale risultato è stato valutato calcolando il fattore di correlazione ottenuto tra frammenti paralleli e non, previa analisi delle annotazioni ottenute sia per i documenti inglesi, sia per quelli italiani. In questo contesto, sono stati presi in considerazione i livelli di frammentazione <BODY> ed <Hx> .

La misura di correlazione adottata è direttamente derivata dalla metodologia "Vector Space Model" [33] – tradizionalmente utilizzata per misurare il grado di correlazione tra una stringa di ricerca ed i relativi risultati, questa tecnica esprime la correlazione tra due frammenti paralleli come il coseno dell'angolo tra i vettori che li rappresentano. Tale correlazione è definita considerando ogni classe dell'ontologia come una dimensione indipendente: maggiore è il peso che un concetto assume all'interno di un frammento, maggiore sarà la componente corrispondente nella sua rappresentazione vettoriale. Per motivi pratici, qui è riportata la tabella contenente i valori di correlazione risultanti per tutte le possibili coppie di frammenti al solo livello <BODY> (Figure 11), dove righe e colonne sono etichettate considerando l'origine di tali frammenti, secondo la sintassi "lingua/pagina/frammento".

I valori di correlazione ad ogni livello di frammentazione sono stati raggruppati in due insiemi, uno riferito alle sole coppie "parallele", l'altro inclusivo di tutte le altre possibili coppie. Gli elementi sulla diagonale rappresentano i valori di correlazione ottenuti per coppie di frammenti paralleli: questi valori dovrebbero risultare nettamente superiori a tutti gli altri. Figure 11 e Figure 12 visualizzano le classi di correlazione per le coppie di frammenti a livello <BODY> e <Hx>, rispettivamente; tali classi distinguono fra le due differenti distribuzioni per frammenti paralleli e non, che danno una

rappresentazione grafica rispettivamente di "hit ratio" e "miss ratio" totalizzate dalle annotazioni. Per semplicità, tali distribuzioni potrebbero essere approssimate da curve gaussiane; considerando per esempio il livello <BODY>, le curve potrebbero essere centrate approssimativamente attorno ai valori medi di 0.3 (non paralleli) e 0.55 (paralleli). Proprio questi valori attestano una delle prime risposte positive del sistema: ci si attende che i frammenti paralleli detengano un grado di correlazione maggiore, a causa della comunanza di gran parte del contenuto – DOSE sembra in grado di soddisfare le aspettative (0.55 > 0.3). D'altra parte, tali coefficienti in valore assoluto non rappresentano ancora elevati livelli di performance: una constatazione che diventa piuttosto evidente quando ne consideriamo il significato pratico, poiché una correlazione media di 0.55 tra frammenti con (più o meno) gli stessi contenuti significa aver mancato quasi la metà dei concetti in comune. Quello che più importa è comunque il comportamento complessivo delle due distribuzioni approssimate: infatti, i diagrammi dimostrano che le distribuzioni del primo e del secondo insieme sono chiaramente separate. Questo significa che frammenti paralleli in lingue diverse sono annotati come corrispondenti (sostanzialmente associati allo stesso insieme di concetti), mentre i frammenti non paralleli sono mantenuti distinti tramite associazione a concetti ragionevolmente differenti: la piattaforma sembra in grado di compiere questa distinzione. Il fenomeno diventa ancora più evidente considerando una rappresentazione tridimensionale di tutti i possibili campioni di correlazione, come mostrato in Figure 13 e Figure 14, dove i picchi sulla diagonale si distinguono abbastanza nettamente dal resto del grafico.

Questa discussione dei risultati a livello <BODY> potrebbe essere opportunamente riapplicata al caso <Hx>: la differenza principale sta nel fatto che qui il ventaglio di concetti compresi in ogni frammento è solitamente ridotto rispetto al caso precedente, poiché qui si considerano frammenti più brevi e focalizzati su argomenti presumibilmente più specifici. Come visibile in Figure 12, qui i risultati sono migliori per frammenti non paralleli e peggiori per frammenti paralleli: questa è probabilmente una conseguenza naturale della maggiore specializzazione semantica di ogni frammento. Infatti, sovrapposizioni casuali su alcuni concetti tra frammenti scorrelati qui sono più improbabili; d'altra parte, le corrispondenze tra frammenti paralleli sono attestate da un più ristretto insieme di concetti, divenendo così più difficili da evidenziare.

1.6.6.2 Riconoscimento del linguaggio

La seconda verifica sperimentale si propone di valutare le prestazioni del modulo Language Detector. Tutti i documenti analizzati nella precedente sezione sono stati nuovamente sottoposti al sistema, questa volta per il solo riconoscimento del linguaggio di ogni singolo frammento della loro sottostruttura. È importante riconsiderare a questo proposito l'utilizzo dell'identificatore di linguaggio predefinito: esso viene aggiornato ogniqualvolta se ne determina uno di valore differente per il livello di gerarchia più alto della risorsa correntemente analizzata. Al contrario, se nessuno dei metodi proposti in Sezione 1.5.5.2 è in grado di individuare un linguaggio, alla risorsa viene proprio attribuito il valore predefinito. Se in generale questa scelta di progetto appare ragionevole (poiché è piuttosto probabile imbattersi in sequenze di documenti espressi in un'unica lingua), in questa sede è sembrato opportuno alternare le pagine

inglesi ed italiane (in sequenza per "batch processing") al fine di evitare casi di falso riconoscimento dovuti a semplice riassegnazione del valore predefinito di lingua.

Al fine di testare separatamente le due strategie per il recupero del linguaggio antenato (vedi Sezione 1.6.4) sono state condotte due sessioni di valutazione distinte; in ogni caso, qui sono presentati solo i risultati ottenuti con il metodo che si appoggia ad Annotation Repository, dal momento che la soluzione alternativa su "Language Stack" non ha ancora evidenziato prestazioni accettabili. Il diagramma presentato in Figure 15 visualizza, per ogni pagina, la percentuale di frammenti per i quali il linguaggio è stato interpretato correttamente.

Per la quasi totalità dei documenti presenti nella collezione, il riconoscimento del linguaggio ai diversi livelli di struttura ha dato risultati praticamente perfetti, salvo cali decisamente occasionali di prestazioni: simili risultati dimostrano quindi che le strategie per il riconoscimento del linguaggio qui proposte non debbono essere accantonate, ma piuttosto raffinate e/o integrate con soluzioni addizionali.

1.7 Conclusioni

Questa tesi rappresenta probabilmente il primo specifico tentativo di introdurre il supporto a più lingue in una piattaforma Semantic Web. L'approccio proposto evidenzia le facilitazioni offerte dalla particolare architettura in questione, sfruttandole direttamente attraverso una semplice implementazione preliminare.

Da un punto di vista pratico, le funzionalità preesistenti di DOSE sono state estese alla gestione di informazione in più lingue. Il nuovo sistema di integrazione semantica permette di gestire l'informazione multilingue in modo potente e flessibile; il

processo di indicizzazione è ora in grado di riconoscere il linguaggio delle risorse e creare annotazioni multilingue; l'attributo di lingua memorizzato in ogni annotazione può essere riutilizzato per compiere ricerche specifiche. Il sistema è stato dotato di un nuovo modulo indipendente per la gestione del linguaggio, accessibile in remoto.

Nonostante i risultati incoraggianti, il collaudo dell'intero sistema è ancora agli inizi: l'approccio proposto va comunque ottimizzato e testato su una più vasta (nonché linguisticamente eterogenea) base di conoscenza, con particolare attenzione alla stesura di ontologia e synsets multilingue. Un eventuale raffinamento dei risultati ottenuti potrebbe garantire maggiore affidabilità da parte dell'implementazione proposta, possibilmente gettando le basi per un'adozione su più larga scala di questa architettura.

In particolare, questa prima versione di Language Detector può essere senz'altro migliorata, sia lavorando sugli algoritmi già proposti, sia introducendo nuove funzionalità, come riassunto nei seguenti punti:

- offrire servizi specifici per le queries semantiche;
- garantire la piena compatibilità con tutti gli standards previsti per la dichiarazione di lingua;
- supportare la gestione contemporanea di più lingue, al fine di massimizzare il numero di annotazioni create;
- ottimizzare le euristiche fornite, eventualmente integrandole con alternative che si dimostrino applicabili al maggior numero possibile di casi.

Il lavoro svolto sul supporto a più lingue è già stato presentato alla SAC Conference 2004 [43]; per quanto riguarda l'intera piattaforma DOSE, essa è ora stata resa disponibile presso una delle più popolari comunità informatiche a promozione del

software libero (SourceForge, [44]), con l'obiettivo di raccogliere contributi esterni al suo sviluppo.

Parte seconda
Testo in inglese

CHAPTER 2

SEMANTIC WEB

The present chapter will try to give a general overview of the Semantic Web, by briefly describing its general intentions, core data structures and technologies. This should only give a general idea of how critical the role of aspects like resource naming and knowledge representation in this context is – the problem of multilinguality, along with that of ontology design, will be tackled more directly in Chapter 3.

2.1 Introduction

The Web could be considered as an impressive success story, both in terms of available information and of growth rate of human users. It now penetrates most areas of our lives, and its success is mainly based on simplicity – the linearity underlying HTTP and HTML gave Web publishers and users an easy access to this new media, helping to reach high levels of utilization. Unfortunately, this simplicity has so far represented an obstacle in better structuring and referencing the huge amount of Web contents now available, from the particular point of view of their semantics. Semantic Web comes as an answer to this need of "upgrading" WWW management.

Semantic Web has been thought up by Tim Berners-Lee (already inventor of the WWW) as a generic effort that "will bring structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users" [1]. To date, the languages used to write documents for the Internet have almost exclusively focused on layout management, giving more and more effective means to refine their aesthetics, but nothing similar has

been done for semantics. The primary aim of Semantic Web is to provide an effective method to enrich the contents of Web pages (or "resources", more generally) with semantic metadata that make them effectively interpretable by user agents. In order to provide some feasibility, Semantic Web has to guarantee two essential features:

Universality. This innovative way of processing hypertexts cannot make differences on their characterizations – again, Berners-Lee states that "Web technology [...] must not discriminate between the scribbled draft and the polished performance, between commercial and academic information, or among cultures, languages, media and so on" [1]. This should work not only on the side of data but also on that of metadata, meaning that no difference between human-oriented and computer-oriented data should exist anymore;

Decentralization. In order to be widely deployed, techniques for modeling the Semantic Web on top of existing WWW resources should not impose a unique, narrow methodology to express meaning, but instead allow for a good compromise between free customization of semantic tagging and interchangeability of the resulting metadata.

2.2 Foundations

A possible realization of this new form of Web has to cope with the current status of available network resources, whose actual organization is so uncontrolled to make a complete tracking of them almost impossible. What follows is a list of all the basic principles and technologies designed to support its practical implementation.

2.2.1 Content structure: XML

The first step to make is to give a structure to Web contents. The language nowadays used by a vast majority of Web pages does provide facilities to structure the page layout only, exclusively affecting the appearance of pages – no additional information can be given about the nature of contents.

A way to provide description of single elements (words, paragraphs, images, etc.) in Web pages is given by XML ("eXtensible Markup Language" [2]): it can be considered as the father of all markup languages, as it allows Web writers to create customized tags to label the contents of their documents. A number of custom XML tags can be created by anyone, to isolate the so-called single "XML entities" (usually single words or short statements) that are of interest for individual retrieval. Sets of coherent entities can be used to organize the whole structure of a Web document, upon previous declaration as a DTD ("Document Type Definition"), or an XML schema [3]; such document can be declared "valid" (if it follows the DTD it declared) or simply "well-formed" (in case it only respects generic XML syntax, but not the specific DTD). To clarify the use of XML, we can consider a generic, raw statement contained in a Web page as HTML text:

"Dan works for BOSE"

to a computer, this merely represents a meaningless sequence of characters – no further implications can be automatically argued. By applying XML, we could change the statement expression so that it becomes:

```
<sentence>
<person>Dan</person> works for <company>BOSE</company>
</sentence>
```

The added markup can now help the machine identify "Dan" as an element of "person" type, likewise "BOSE" is correctly identified as a "company"; more than that, the whole string is defined as a "sentence". All these single, finally isolated pieces of information can be reused for reference and processing.

It is quite evident that markups defined by different people can easily collide (i.e. refer to different data types with the same label) and generate confusion. In order to avoid this, XML introduces "namespaces": custom tags created by the same author (or, in reference to a common context) can be grouped under a single namespace, that delimits an area of the Web ("space") from which the meaning of such tags can univocally be drawn [4] [5].

2.2.2 Resource identification: URI

The whole WWW can be seen as a web of "resources" – a "resource" can be anything that is described somewhere on the Web (from HTML pages to real objects), and represents the basic data unit that can be manipulated by Semantic Web agents.

As it has been just detailed, single resources within Web documents can be structured using XML tags; however, this kind of description does not help to disambiguate between different instances of the same data type. A standard naming

system, capable of identifying single resources with a unique label is then required, and this is when URI (Uniform Resource Identifier) comes in. A URI [6] is basically a "Web identifier for resources"; it can be used to distinguish between multiple resources that share the same XML markup. Reconsidering the previous example, we could complete it as follows:

```
<sentence>
<person href="http://www.homepages.net/dan_hibiki">
Dan</person>
 works for <company href=http://www.bose.com">BOSE</company>
</sentence>
```

The new markup helps the computer to map univocally the two resources within the Web, by associating both "Dan" and the company identified by the string "BOSE" with the WWW resources that actually represent them. Other examples of resources that can be named this way are XML namespaces, or Web pages – as a matter of fact, URLs ("Uniform Resource Locators") represent just a particular kind of URI, but while the former follows a fixed scheme of expression (depending on http naming conventions) the latter ones are totally decentralized. A common issue called "The Semantic Web Identification Problem" occurs in this connection, by recognizing that a URI often delivers both the name of a resource and the location of its related Web page, but the true aim of URIs remains the mere identification of Web resources.

The good flexibility allowed by URI naming does also bring high risk of redundancies, as many resources of the same type will probably be named with a multitude of identifiers by different people – such a tradeoff is however necessary to pave the way for easy Semantic Web deployment.

2.2.3 Logic: RDF

The technologies detailed so far provide for structured, univocal markup of contents, but at this point the new resources thus obtained still lack the most important feature: meaning. A set of elementary relations needs to be established between resources, in order to make them somehow understandable by computers. This is what RDF (Resource Description Framework) provides, via a basic mechanism that allows for the assertion of "statements".

A "statement" is an elementary relation composed of three parts: a subject, a predicate and an object. The predicate establishes a logical link between the subject and the object in a way very similar to that of simple sentences expressed in natural language. Such a predicate can be chosen between a set of predefined ones: usually simple logical connectors are represented (like "subclass of", "inverse of", etc.). There is total freedom both on what can be asserted and how it can be asserted: this is the main powerful principle underlying RDF, "anything can say anything about anything". Needless to say, each of the three elements of a statement can be a URI – the statement itself can also become a URI, and this also reveals the object-oriented nature of RDF.

The official RDF specification [7] defines the representation of statements through an XML-based syntax, exemplified in the following expression:

```
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:verbs="http://dictionary.org/verbs/">
<rdf:Description
rdf:about="http://www.homepages.net/dan_hibiki">
<verbs:worksFor
rdf:resource="http://www.bose.com/" />
</rdf:Description>
</rdf:RDF>
```

According to previous examples, the statement can be decomposed as follows:

Subject: Dan

Predicate: works for

Object: BOSE

The details given on RDF can testify its suitability for the representation of databases on the Web. With the work of intelligent programs, such databases could be cross-queried in order to obtain composite results in a totally automated way. Suppose for example that a hi-tech company needs an Hi-Fi expert for external counseling: their Semantics-enabled web agent could start browsing the list of major Hi-Fi research laboratories, then stop at BOSE's website and scan the database of their employees; the

search could eventually end at Dan's homepage, with the agent possibly collecting useful data about curriculum vitae and contact information.

2.2.4 Knowledge: ontologies

The cross-database scenario outlined just above seems a really versatile one, however such a perspective has to undergo the randomness of WWW. It is very unlikely to expect always the same identifier for the same concept when browsing a certain amount of resources (databases). More than that, Internet resources (definitions, references, etc.) frequently change, thus seriously hardening the task of mantaining stable references (both in terms of reachability and consistency) to data banks. Worse, the gap between human and computer reasoning has to be taken into account. A lot of human understanding is based on context reasoning, and a computerized device inevitably lacks such capability. In other words, URIs have to be described in meaning - to face all these problems, Semantic Web generally makes use of ontologies [8].

The American Heritage Dictionary defines "ontology" as "the branch of metaphysics that deals with the nature of being". The term has officially been adopted in Artificial Intelligence to refer to a set of concepts that can be used to describe some area of knowledge, or build a representation of it. To us, an ontology is a hierarchical structure built with a set of concepts, all belonging to a particular knowledge domain, that establishes inference relations between these concepts. Existing resources (URIs) can be associated to one or more concepts if they retain any lexical representation of them; by navigating these semantic associations previously created, a user agent can distinguish between resources basing on their semantic labeling. Research in this field

has produced several languages to describe ontologies; other than RDF schema [9] and DARPA Agent Markup Language With Ontology Inference Layer (DAML+OIL) [10], which constitute the base standard for ontology description, worth mentioning is OWL (Web Ontology Language) [11].

2.3 The role of agents

Apart from all technologies aimed at making Web contents semantically accessible, the other big players in the development of Semantic Web are – of course – agents able to take advantage of all the metadata thus produced.

Semantic Web will show its true power only with the creation of applications that are capable of collecting Web contents from diverse sources, process them and exchange results with other programs. Beyond the "basic" requirements planned for these intelligent agents are some important services that should be expected by fully-deployed versions. Such services were already prospected by Berners-Lee in his early visions of the Semantic Web [1].

Firstly, there should be full support to logic. To date, Semantic Web agents have been created to deal with semantic data by means of simple inference rules (such as subclass, inverse, etc.) that already allow a good deal of processing for the existing resources. The final goal is however to make agents intelligent enough to actually understand every possible logic assertion: this should ideally allow users to develop semantic resources by approaching more and more the use of natural language.

Secondly, Semantic Web applications should be able to verify the correctness of data just remotely recollected with respect to semantics – that is to say, after having

retrieved information requested directly or indirectly by users, agents should check for its validity. Verification could require for example logical search of the same target data through parallel ways, or backwards search starting from the result obtained. We could think of smart "heuristic engines", sorts of intelligent agents that can follow millions of links on the Web in order to create proofs for well-defined scenarios, and place them in a provided repository for convenient access and reuse [2].

Finally, the great deal of freedom allowed by Semantic Web core principles and related technologies could potentially favour the creation of incorrect data. As we have seen for RDF, virtually any statement can be asserted using RDF triples – this means that, concerning a particular domain, two people could potentially state exactly the opposite about something. A trusting system that uses digital signatures should therefore be provided, allowing to certify semantic data and choose which sources to trust during the information retrieval process. From a first impression, such an approach could seem hard to deploy, as it would require each user accessing the Web to make discriminations on every possible source of information: a clearly impossible achievement. But a so-called "web of trust" could solve the problem, by enabling some sort of "transitive trust" (i.e. person A trusting another person B is automatically trusting all of B's "friends"). The whole process could be made transparent or hidden to the user, allowing him/her to make informed decisions on Semantic Web contents.

CHAPTER 3

MULTILINGUALITY

The primary subject targeted by this thesis is multilinguality - this chapter tries to give some background information on it. After a brief introduction, main issues for automated, cross-language data management in general will be first given from the point of view of machine translation. Then the most popular techniques for multilingual text retrieval will be presented, along with their advantages and disadvantages. Finally, the chapter will try to better focus on their application to Semantic Web, with particular attention to ontology design and language ID retrieval.

3.1 Introduction

The concept of multilinguality could seem quite a blurred one, in that its name does not really define something concrete. One possible description of multilinguality could be "the extension to multiple languages of tasks already performed in a monolingual context". Even if such a definition could well sum up all the realizations of multilinguality, somebody prefers to identify it concretely with the whole number of practical operations that are commonly performed when dealing with data coming in several languages: these could include interpretation of speech, as well as translation.

Today's need for multilinguality roots back to the origin of world cultures, that from the early beginnings started to evolve in different directions. Of course, this is reflected in the development of human expression, producing what is nowadays recognized as a multitude of languages, each one retaining its peculiar graphic symbols, features and idioms. Such a separation in language has always hindered the free

exchange of information between cultures throughout the years, and now that the era of information technology – together with globalization – is opening up new perspectives, the requirement for automated management of multilinguality problems seems really a pressing one.

All issues involved by the automation of multilinguality take part in the larger research area of NLP ("Natural Language Processing"). This field actually spans a great variety of processing modes for natural language in general (i.e. the language naturally used for communication between human beings), be it written, spoken, or represented with other means than traditional ones [12]. To this purpose, it is important now to delimit what will constitute the object of manipulations discussed throughout this thesis, that is "electronic text (in multiple languages)" only.

3.2 Multilingual issues from machine translation

This paragraph will quickly approach the main issues that occur in cross-language mapping in general, but starting from the particular ground of Machine Translation. Even if this is not the actual target of the thesis (as we will see, it will be closer to translingual information retrieval), it is a good starting point to illustrate the difficulties involved in the process of rendering contents in multiple languages. After this overview, it should be easier to zoom to our particular task and face the issues related to text retrieval strategies.

Machine Translation is one of the oldest challenges posed by the need for multilinguality. Considering a single word in a source language, we could at best define the process of translating it as the attempt to find a correspondent in the target language

(Ingo, cited in [13]). The accomplishment of an acceptable translation requires to consider the connection between source and target word(s) both from the point of view of meaning and of lexicon: translation should primarily preserve the original meaning, but this is best conveyed with careful lexical choices in the target language. Considering this "dual mapping" problem, we could outline three incongruences that mainly show up during the process of translation:

Lexical gaps. This kind of incongruence occurs whenever a language expresses a concept with a lexical unit whereas the other language expresses the same concept with a free combination of words (Hutchins and Somers, cited in [14]). Usually this is overcome by rephrasing the original concept with more words, with the risk of distorting the original meaning: an example could be the italian word "anagrafe", that in English is rendered as "register of births, marriages and deaths".

The same gap problem occurs when a word in the source language groups a certain range of variants in meaning, that map to distinct words in the target one: the English "wood" in Italian becomes either "legno" or "bosco". One solution to this problem, given that these distinctions are known between the languages, is to create pseudowords such as wood (material), and wood (environment), that identify a semantic domain for each variant (Fluhr, [12]).

Role of the context. Translation cannot be carried out by considering words in a sentence as separate entities to be individually processed, but rather as a unique body that conveys a certain meaning based on the peculiar positioning and association of its

components. Such a fact becomes evident when evaluating the meaning of the English word "spring" in the two following expressions:

"A spring of fresh water";
"After Winter comes Spring".

But context could be intended as cultural background as well: it is natural to expect a different organization of knowledge from languages related to different cultures, so that a concept well known (and expressed) in a certain culture could be definitely missed in another one. The occasional solution to this problem could be referred to as "borrowing": the concept missing in a certain language is donated by another one of major influence in the field (i.e. words originated from Latin).

Lack of pre-acquired knowledge. Another important point about Machine Translation issues concerns the knowledge that should be somehow pre-fetched by the machine. The interpretation of a text often requires the human reader to reconsider it on some pre-acquired knowledge basis, in order to catch the proper implications; this cannot be clearly expected from a device (Kay, [12]). It has often been argued that high-quality translation is impossible without systems capable of understanding the meaning of texts, implicitly suggesting that some human assistance would be required anyway. A convincing example in this case has been readapted from Carbonell (cited in [13]): when attempting a correct translation of sentences like

1. Morocco seized large quantities of chocolate bars from Switzerland;
2. Hawaii seized territory from Alaska;

the translator needs to know background facts like

1.
   a) The chocolate bars were not seized from Switzerland but probably from another African country;
   b) The chocolate bars were made in Switzerland;
2. The territory was really seized from Alaska.

This knowledge enables the translator to choose the right prepositions when translating into Russian for instance, which uses two distinct prepositions for the two senses of "from" in 1. and 2..

3.3 Multilingual Text Retrieval

Having listed – very basically – all the possible problems that can show up when trying to perform cross-language contents mapping within Machine Translation, we can now approach the area of NLP that most closely relates to the aim of this thesis, that is Multilingual Text Retrieval: this one could be well summarized as "the selection of useful documents from collections that may contain several languages" [15]. When dealing with large, unstructured and heterogeneous flows of information of multilingual nature (such as the Internet), the process of Information Retrieval cannot be managed

assuming a monolingual perspective or any keyword matching approach; all the problems given above are then back again in this scenario. What follows is a summary of all the main strategies that research has designed to face the problem.

3.3.1 Popular approaches

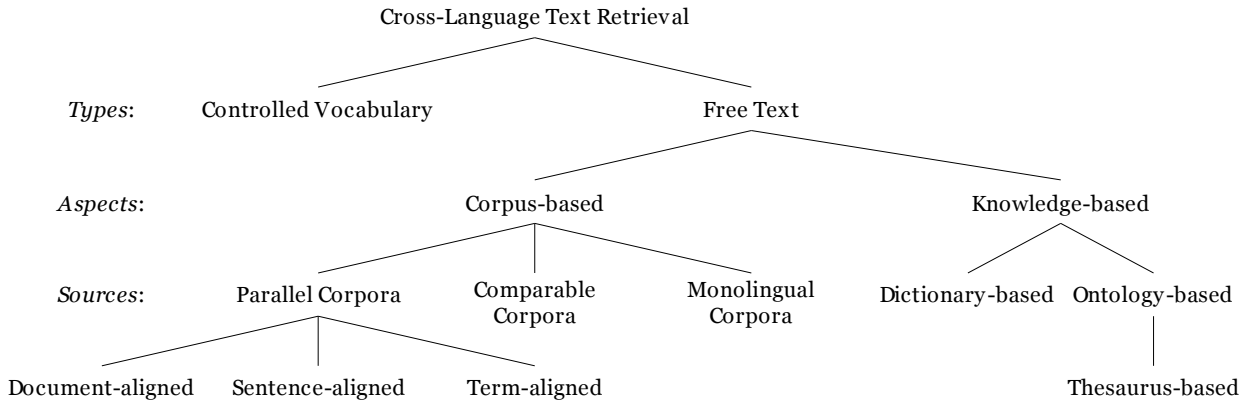

Figure 1: multilingual text retrieval strategies.

The cathegorization of Multilingual Text Retrieval techniques given here is drawn by Oard's summary [15] – Figure 1 illustrates them along with related sources of information. The next chapter will take it up again to explain which strategy has been chosen for the present work.The oldest and simplest approach is referred to as "controlled vocabulary": it requires all documents in a collection to be manually indexed using a predetermined vocabulary, and the user to express queries using terms drawn

from the same vocabulary, too. The system utilizes a multilingual thesaurus to match query words coming in multiple languages to a common set of language-independent concept identifiers, and document selection is based on concept identifier matching. Apart from work overload caused by the creation of large vocabularies and (manual) document indexing, the use of a limited vocabulary poses an upper limit on query precision. More than that, it has proven fairly difficult to train users in effectively selecting search terms and exploiting thesaurus relationships.

The main alternative to "controlled vocabulary" consists in the use of words directly appearing in documents as they were a vocabulary. Strategies of this kind are grouped under the label "free-text retrieval" (or "full-text"), and are further distinguished in "corpus-based" and "knowledge-based". While with controlled vocabulary one was forced to translate both the query and the documents into a common language (the vocabulary), "free-text" does actually limit such process to the document or the query string only. In reality, only the second possibility is taken into account for evident performance advantages [16], so that it will be implicitly assumed from now on.

The "dictionary-based" approach tries to re-apply the use of multilingual thesauri (already a staple of "controlled vocabulary") to the randomness of free text in documents. The methodology most exploited in this context relies on bilingual dictionaries (of the kind used for human translation): each word in a user query is replaced with one or more from the vocabulary, both expressing the same original meaning. Two are the issues that usually limit the performance of dictionary-based techniques: first, the recurring lack of univocal translations that causes a source word to

be mapped into many target ones from different contexts, thus consistently blurring search results (a common effect of cryptic query strings). Second, dictionaries will always lack some lexical correspondent for the words inserted by users, so that documents indexed through that missing concept (keyword) will inevitably be lost.

Another knowledge-based possibility comes from the use of ontologies ("ontology-based"), especially designed to model structures of concepts along with internal semantic relationships.

Each concept has (or actually, "is") its lexical counterpart, which will be the one determining its occurrence in analyzed documents. Search queries refer to concepts instead of words, actually giving higher improvement and precision to the query. Even this approach suffers from some typical limitations: these are the restricted domain extension of ontologies (as the process of building an ontology is always an expensive one) and the (easily defective) relationship between form and meaning exploited by concept-term pairs, that often fails to guarantee exact matching in texts.

While the techniques discussed until now could be seen as top-down ones – since they search for textual evidence of some general, pre-acquired knowledge of things – we could identify "corpus-based" strategies (the other big branch in the whole set of text retrieval strategies) as based on example. Their purpose is to optimize the translation of queries by analyzing large collections of documents and automatically extracting the information needed to construct automatic application-specific translation techniques. The collections which are analyzed may contain existing translations and the original documents ("parallel corpora"), or they may be composed of documents on similar subjects which are written in different languages ("comparable

corpora"). In this connection, it is probably worth-mentioning what could be seen as state-of-the-art, the so-called Cross-Language Latent Semantic Indexing developed by Landauer and Littman (evaluated in [16]). The main drawback for such approaches is the requirement for a large collection of documents that suits the need to extract translation samples of good quality, and the work overload in translating monolingual sets of documents in order to obtain parallel corpora.

3.4 Multilinguality and the Semantic Web

We have already discussed the possibility of implementing Semantic Web on top of the pre-existing WWW infrastucture, and our task now is to think about a flexible management of resources coming in different languages. The two main capabilities on which any semantic architecture relies are those of indexing and retrieval. These ones are of course generic definitions that mainly take into account primary inputs and outputs of the whole system, thus justifying the use of such "end-user" oriented terminology. With "indexing" we identify a mainstream process that receives a certain number of existing web resources as an input and parses them based on the knowledge of hierarchically structured concepts. Each occurrence of a different (known) topic in a certain portion of an analyzed resource produces a content-location match that is stored within the system. The iteration of this basic analysis on a certain amount of documents produces a data repository which is then available for exploration through the "search" process. This task (opposite to the previous one) can be directly carried out by end-users via conceptual queries that allow for navigation of the just-created set of references (and resource retrieval if possible). From this quick summary of the basic mechanisms

activated within a generic Semantic Web platform, we can easily understand when and where the support for many languages can come into play. Very simply, language represents just an additional parameter for all operations - single (semantic) data units that before were expected to come with no language attribute (or, at least, a default one) now must retain one. Two basic goals are then to be achieved in such a context:

- recognize language information (language ID) of available resources;
- map resource contents independently from language.

Both topics are specifically addressed in the following Sections, 3.4.1 and 3.4.2 respectively.

3.4.1 Language ID in resources

When mapping documents to concepts, the indexing process has to realize the presence of such concepts through their lexical entities, and since these ones witness considerable change when switching from one language to another, it is necessary to identify a resource language before its conceptual analysis takes place. We could then outline two scenarios that differ consistently from the point of view of language determination:

- language information is explicitly stated, meaning that is already available for use. This reduces "language recognition" to a mere process of "language information retrieval", thus requiring minor resource overhead for the system;

- language information is not given: this requires its actual creation, based on some trial mechanism like the ones outlined in [12].

Of course, the situation in which a standard Web specification can help is evidently the first one: if language attributes come together with documents to be indexed, they can be directly retrieved. Developers and standardizers dealing with the internationalization of web resources (such as XHTML documents) have actually supplied a set of standard forms to allow for language specification: below is a list of all them, so that it will be easier to make future reference to the ones this approach has taken advantage of.

1. The "Content-language" field [17]: it may be specified in document headers (it is mainly provided as an XHTML extra META tag) and it indicates which is the language (or set of languages) used in the document. This description is usually added by servers at the HTTP protocol level, and is primarily targeted to browsers; they must return an "Accept-Language" message with the language chosen for processing.
2. The HTML "lang" attribute [18]: this is a tag attribute specially designed to label single XHTML elements with language. It can be added to the set of attributes listed within a generic XHTML opening tag: all contents enclosed by this markup will retain the language attribution specified by its value. A set of possible primary values for "lang" is derived directly from the list of country-codes used for Internet domain name extensions - the most part is actually just

a lowercase version of the original ones, even if in many cases a subcode extension can be added. An example of acceptable value could be

```
lang = "en-US"
```

where "en" is the primary code (made up of two letters by constraint) that conveys the base language specification (English), while the "US" subcode refines it, leading to a particular localized slang variation (U.S. English). When this attribute is omitted, the default value to assume is "unknown".

3. Language inheritance [18]: this property can rely on the well-formed hierarchical rules that underlie the use of XHTML tags. As a matter of fact, any page element whose lang attribute is left unspecified can receive its value from the closest parent element that declares it. This is to say that if a document – which for instance is monolingual and structured on several levels – retains language definition at the HTML or BODY level only, this suffices for a complete language denotation of all embedded elements.
4. CSS-level declarations: the ways for language definition listed so far have been referred to elements of XHTML documents, but they could be included in their stylesheets as well. This can lead to situations where language definition for a certain portion of document becomes substantially "overloaded" by two coexisting values (one in the document, another one in the stylesheet): in such cases, proper merging techniques should be applied in order to accept the language choice most meaningful for the rendering of contents.

3.4.2 Multilinguality and ontology modeling

As we have seen, one of the main strong points of the Semantic Web is its universality: no discrimination exists between resources. Of course, this involves also language: a resource expressed in a certain language should be made available to users of any other language. This means that the representation of knowledge within a semantic architecture has to retain independence from language or culture, while the system should be able to identify its factual representations in all possible languages. Hence, the relationship between form (varying on language) and meaning (univocal as possible) has to be correctly investigated in order to provide the best model for language-independent knowledge management. In the following, existing kinds of ontologies will be first presented, so as to underline their advantages and disadvantages for multilinguality; a definition of ontology for multilinguality will then be presented.

3.4.2.1 Available ontology models

The concept of ontology should be clear from the definition given in Chapter 2. Our interest is to make a distinction between different ontology models to see how they interact with language representations. In [13], Agnesund makes a proper classification of the three main kinds of ontologies, using two related, yet distinct terms to clarify their different relationship to language. The term "language independent" is used to characterize an ontology that is not directly dependent on language in its representation of knowledge, while "language neutral" refers to an ontology which does not favour the peculiarities of any particular language.

The first discussed model is that of "conceptual ontologies" – these are the ones most closely related to formal ontology. They are rooted in artificial intelligence research, and are based on the total separation of ontology and language. A conceptual ontology reflects the structure of the world, without explicitly involving natural language in the description of this knowledge – therefore it claims to be maximally language independent, motivating this on the will to give a representation as generic as possible. This also implies that an ontology of this kind is language neutral.

In direct opposition to conceptual ontologies are "language-based ontologies": these ones rely on the assumption that knowledge is a linguistic construct that cannot be seen as neutral with respect to language. According to this principle, ontologies should be linguistically motivated and based on linguistic evidence. Then it follows that a language-based ontology is derived from language and can be neither language independent nor language neutral. An extreme accomplishment of this conception is represented by semantic networks like the English WordNet [19] – however, instead of considering many languages from which to draw knowledge representation, such networks use only one.

An ontology model that well merges the attention to lexicon and meaning is represented by "interlingua ontologies". To explain its basic functioning, it is necessary to go back to the Machine Translation scenario, where ontologies of this kind serve as the basis of an interlingua. An interlingua is an intermediate representation of the meaning of a text, acting as a bridge between languages in a Machine Translation system. A true interlingua should ideally represent a universal interchange format between all languages. Assuming that to be possible, we would define interlingua

ontologies as truly language neutral, but in practice most of them are "restricted interlinguas" (neutral only to a few languages). The knowledge representation in an interlingua ontology is usually language independent but could also be more or less influenced by language.

A brief summary for ontology classification is presented in Table I.

3.4.2.2Requirements for multilinguality

For our purposes, a good multilingual ontology must provide a valid structure for what we could call "lexical semantics" [13], i.e. the representation of the meaning of words or other lexical items, in a way equally meaningful to all languages. Such an ontology should therefore be able to avoid the typical idiosyncrasies pointed out for cross-language mapping. In short terms, a multilingual ontology should provide three essential features:

Kind	*Language independent*	*Language neutral*
Conceptual	Yes	Yes
Language-based	No	No
Interlingua	Yes / No	Yes / No

Table I: ontology classification.

Consistency. The validity of concepts included in the ontology should be backed by the so-called linguistic evidence, i.e. the real occurrence of lexical representations for such concepts. This should prove the effectiveness of defining a certain concept with respect to the concrete world.

Flexibility. The ontology should provide concepts general enough to map to natural language expressions in most languages, helping to match different, language-dependent terms of the same meaning.

Extendibility. The addition of support for a new language should not require to modify the ontology, which should be considered virtually independent from language.

The composition of these three requirements is effectively reformulated by Agnesund [13] as the need to find a way of "making it possible to stick with the demand for linguistic evidence, while still enabling all concepts to be meaningful for all languages". A model of "multilingual ontology" will be proposed in the next chapter, embedded within the DOSE platform.

CHAPTER 4

DOSE

This chapter is dedicated to the description of DOSE (Distributed Open Semantic Elaboration platform), the ground for the thesis implementation, as a bridge between the need for Semantic Web and multilinguality. After a brief introduction on the typical problems of Semantic Web integration and implementations, the chapter gives a generic description of the system, by taking up the crucial points just underlined. A brief overview of the modular architecture of DOSE is then presented, along with a step-by-step description of two typical usage scenarios. Finally, the proposed architecture is reconsidered with respect to multilinguality.

4.1 Background

The promise of the Semantic Web to innovate the way we design and use the web is slowly progressing, as proposed standards settle down and semantic applications are developed. One of the major problems for a wide deployment of semantic information processing on the web is a scale factor: until a significant fraction of web-accessible information is not semantically accessible, semantic-oriented applications will not be able to unveil their full potential. For this reason, several proposals are being put forward to help adding semantic information to existing or newly developed Web resources.

In this connection, one interesting goal that can be achieved by the Semantic Web is the automatic creation of documents by collecting and concatenating all relevant paragraphs from available resources found through a search query, something that

requires automatic extraction of semantic information and infrastructures for external "annotation" storage. Annotations represent the way to achieve better search performances on the web using semantic metadata separated from resources, as the most important and well-known disadvantage of semantics is the heavy work of resource structuring and description required to make semantic processing available. What is still unclear and highly debated is how annotations could be created automatically, since manual annotation of the whole web resources is clearly impossible; many recent works involved Natural Language Processing, Visual Processing and Ontology Learning techniques. On a large scale deployment of Semantic Web technologies, however, a second problem would arise, related to the heterogeneity of content structure: the single semantic index would need to point both to smaller pieces of information and to more extended documents.

Systems such as Yawas [20] and Annotea [21] [22] allow to create and share annotations attached to web documents; however, these systems do not use a structured ontology for the metadata associated to annotations. Other systems are designed to build annotations using ontologies [23] [24] [25] [26], and others even try to build frameworks in which semi-automatic tagging is possible through knowledge extraction rules [27] [28] [29] [30] and machine learning techniques [31]. However, these ones all allow annotation at the document level, only.

The DOSE approach tries to cover all these aspects, providing an architecture for creating and managing annotations using previously defined ontologies, and allowing the tagging of documents at different granularity levels, ranging from the whole document to the single paragraph. This framework also includes the search

functionalities able to exploit the semantic annotations for retrieving and composing new documents starting from the existing ones. The system is based on an external annotation repository, which stores the relationships between ontology concepts and web resources – these ones being indexed down to the document substructure level. Standard W3C technologies are adopted (such as URI and XPointer on top of RDF annotations) in order to increase the interoperability of the proposed system. Communication among the various submodules is achieved through a distributed service infrastructure (XML-RPC/SOAP) to allow further growth and integration with other systems by using standard and open technologies.

4.2 Overview

DOSE (Distributed Open Semantic Elaboration platform) [32] has been designed in the direction of embodying an architecture that can be exploited by the forthcoming Semantic Web technologies but that – at the same time – can be used today with existing technologies. The main principles underlying its design are then modularity, scalability and of course semantic integration. Modularity and scalability can be easily obtained with existing technologies: while modularity guarantees third part module integration, scalability is the capability to work contemporarily with a wide number of users (either humans or machines) and is usually achieved by adopting modular architectures, where all modules can be progressively substituted by more effective ones.

About semantic integration, a simple strategy for automatic annotation is exploited in the primary version of the system, while more complex approaches could be easily plugged in as new modules. The proposed methodology uses a standalone

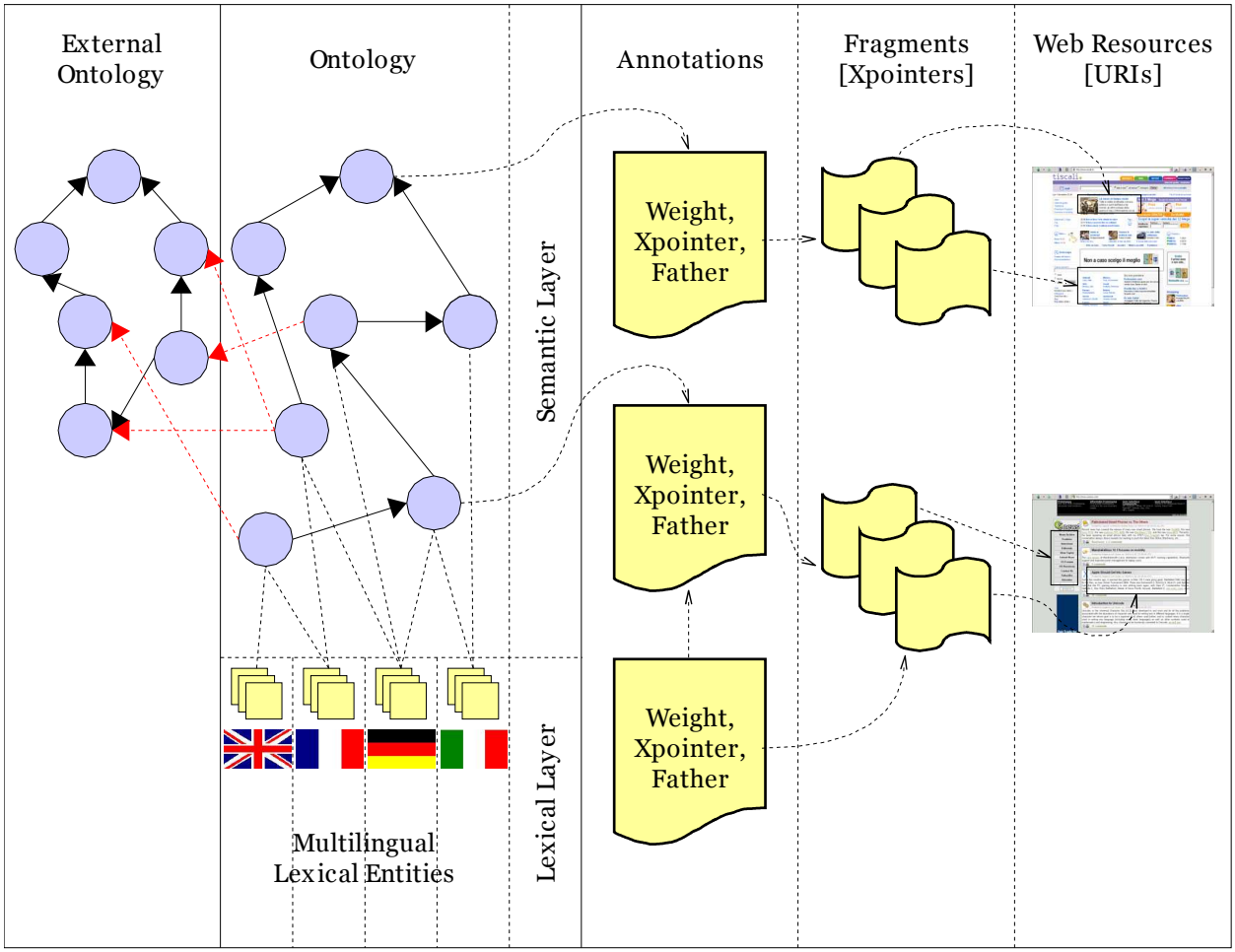

Figure 2: conceptual organization of the DOSE platform.

annotation repository, in which annotations are stored independently from annotated resources. Resources are related to annotations by means of XPointers and URIs. Automatic annotation uses a module called Semantic Mapper which basically takes an ontology and a group of lexical entities as working components, and for each input resource returns the collection of ontology concepts the resource is related with. Each concept is connected to a group of lexical entities called synset (as each set of lexical

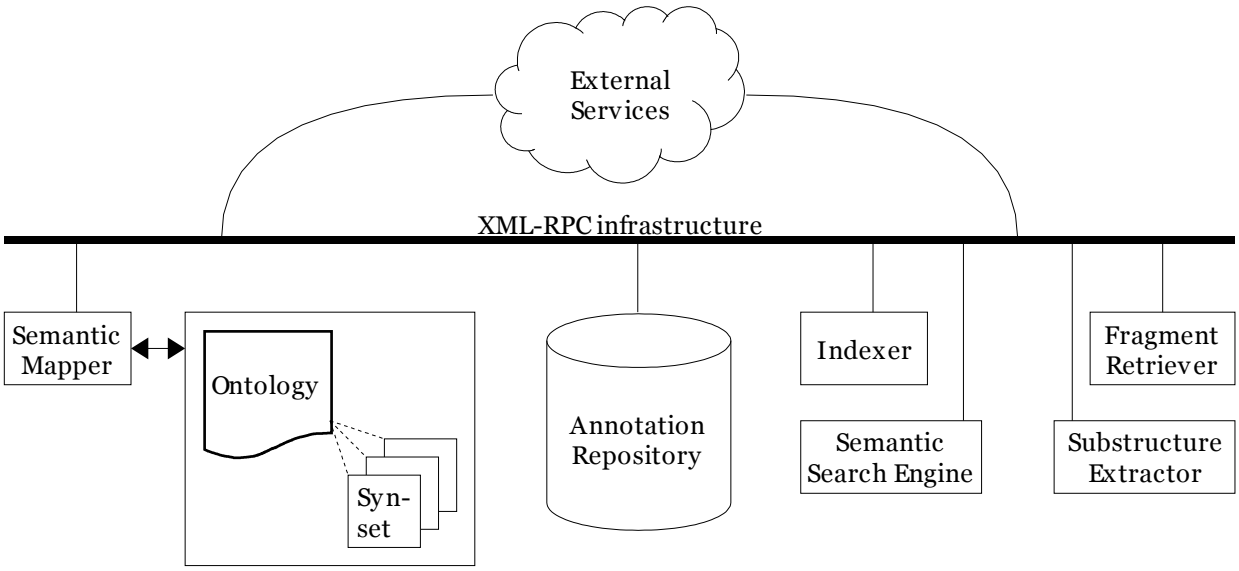

Figure 3: architecture of the DOSE platform.

entities is mainly composed of synonyms) that is used by a classical information retrieval technique [33] to classify resources and to identify most reliable associations with the ontology concepts.

The DOSE architecture also allows the annotation of relevant fragments of web resources in order to achieve better search performances: the architecture offers a single, interchangeable module that can syntactically split a web resource into its basic components and produce unique identifiers using XPointer syntax [34], the standard proposed by W3C to identify document substructures.

Another innovation proposed in the architecture design is the semantic annotation repository, where the word "semantic" is referred in particular to semantic relationships that occur between different annotations. DOSE introduces relationships

also between semantic annotations, obtaining a taxonomy in a first instance and an annotation ontology as foreseeable result. The organization of annotations into a taxonomic structure allows for the application of many useful operations defined for taxonomies in the annotation repository – in particular, the proposed annotation repository will be able to identify appropriate Level Of Detail (LOD) for annotations by means of generalization relationships. During a query, more relevant results will therefore be obtained by focalizing or generalizing annotation search according to the query. More than that, annotations referred to fragments coming from the same resource will be collapsed into a more general one referred to the entire resource.

Annotations stored in the repository are serialized as sequences of triples in RDF format, and their physical storage is totally independent of the annotation repository capabilities. Annotations are referred to single fragments of web resources by means of a (URI, XPointer) pair.

A semantic search module has also been provided, in order to leverage the full potential of semantics, that will enable translation of text queries into conceptual ones. The ontology structure and in particular the relationships between concepts will allow automatic search refining. Search results will be composed by many fragments coming from different web resources and will be accommodated into one or more result pages using relevance criteria.

The whole conceptual organization of DOSE is shown in Figure 2.

4.3 Architecture

The DOSE platform described above is organized as a collection of loosely coupled services (Figure 3). Architecture modules can act as server or as client depending on required tasks, however some modules have only one typical behavior. In general, core modules (the ones interacting with raw data, like the Fragment Retriever, the Substructure Extractor and the Semantic Mapper) act as server while middle level ones (like the Indexer, the Semantic Search Engine and the Annotation Repository) behave both as client and as server. The following paragraphs describe the intended functionalities of each module in detail, while all methods exposed by DOSE modules via XML-RPC [35] are summarized in Table II.

Substructure Extractor

The Substructure Extractor isolates, together with the Fragment Retriever, the architecture from XPointers and from contents. Resource fragmentation is done by following directives for fragment identification that specify the set of tags at which the document must be broken (e.g. H1, H2, P). Resources to be fragmented are temporarily fetched, fragments are identified by applying fragmentation rules and corresponding XPointer strings are created, taking into account hierarchy relationships between fragments. Indexed fragments are not locally stored, but a set of identifiers pointing at specific substructures of resources are generated.

Fragment Retriever

The Fragment Retriever exposes via XML-RPC the `extract(Xpointer,URI)` primitive that uses an input pair constituted by a URI and an XPointer and retrieves a specific fragment of a web resource. The output is a valid XML file containing the specified resource.

Indexer

The Indexer coordinates the entire process of document substructure annotation. Given the URI of a specific resource, the indexing service interacts with the Substructure Extractor and the Fragment Retriever, in order to obtain a collection of fragments associated with (URI, XPointer) pairs. Each retrieved fragment is then sent to the Semantic Mapper which "converts" this fragment in a weighted set of concepts semantically related, stored as semantic annotations in the Annotation Repository.

Annotation Repository

The Annotation Repository stores semantic annotations related to a fragment of a specific resource, with respect to a given ontology, by means of the Semantic Mapper; it is used in both annotation storage and retrieval. This double capability is reflected by the provided primitives (Table II).

Annotations are stored in a taxonomic way, and are related to each other by the taxonomical relationship of inheritance (generalization induced by the "subset-of" relationships of embedded XPointers). The resulting structure of the Annotation Repository is therefore a forest: a tree for each URI starting from more generic

annotations and branching as long as the level of detail increases, until annotations related to specific substructures are reached.

Module	*Accessible methods*	*Description*
Annotation Repository	`addAnnotation (concept, URI, Xpointer)`	Creates a new annotation
	`searchByTopic (concept)`	Searches annotations by topic
Substructure Extractor	`fragment(URI, cutPoints)`	Fragments a Web resource according to a given set of cut points, and preserves semantic relationships between fragments
Fragment Retriever	`extract(Xpointer, URI)`	Extracts a fragment of a Web resource identified by an (Xpointer, URI) pair
	`compose(Xpointers, URIs)`	Composes a result page, merging all retrieved fragments
Semantic Mapper	`getTopicsOf(text)`	Returns the concepts associated to a fragment
	`getOntoFather (concept)`	Returns the concept parent of the given concept
	`getOntoSons(concept)`	Returns the concepts children of the given concept
Indexer	`index(URI)`	Creates annotations for the specified URI
Semantic Search Engine	`search(queryText)`	Searches Web resources conceptually related to the queryText

Table II: XML-RPC accessible methods, grouped by module.

Annotations are retrieved searching the repository for a partial match with specified concepts, using information retrieval methods [33] for asserting the relevance of retrieved annotations. A basic matching rule based on the well-known vector space model "tf/idf" ranking [36] is used by default, enriched by heuristics taking into account taxonomic relationships between annotations and language information.

Semantic Mapper

The Semantic Mapper operates on an ontology and a set of lexical entities. The ontology specifies a knowledge domain using interrelated concepts, for each concept a set of lexical entities and links to related ontologies (e.g. WordNet) is defined.

A lexical entity is constituted by a word or a multi-word string that usually identifies a concept, or that is often used in conjunction with the same concept; entities referred to a single concept are usually grouped in synsets. Synsets allow the mapping of fragments of textual resources to semantics-rich definitions (concepts), bridging the gap between the syntactic and the semantic level.

The main function of the Semantic Mapper is to take a resource fragment or a query string, and find a suitable mapping with a set of ontology concepts. The Semantic Mapper could also be used for ontology navigation, thanks to a specific set of primitives ("navigation primitives") which are used for ontology navigation, i.e. for the identification of more general concepts from a given one, and so on. The Semantic Mapper architecture is totally independent of ontologies and lexical entities, thus it is applicable to different knowledge domains, simply by selecting a new ontology with a set of associated lexical entities. Given an ontology and a group of related lexical entities,

the semantic mapper can reliably identify mappings between documents/substructures and ontology concepts, offering the bases for a semantic annotation service.

Semantic Search Engine

The Semantic Search Engine provides facilities for querying the Annotation Repository in order to obtain semantically relevant resource fragments, ranked according to predefined criteria.

The engine accepts queries expressed as keywords, short texts or concepts and interacts with the Semantic Mapper by creating a weighted set of concepts semantically related to the original query. The Annotation Repository is then queried to retrieve relevant annotations bound to the concepts – a relevance feedback based on ontology navigation eventually refines the set of annotations.

Once a set of ranked annotations is found, the search engine uses the Fragment Retriever to fetch the corresponding fragments and to compose result pages.

4.4 Operational scenarios

The two main application scenarios of DOSE are given, in order to assess the platform viability: one relates to a much more effective search on the web, while the other concerns automatic indexing of web resources.

In the first scenario, some sample high level module like a semantic information retriever interacts with DOSE in order to obtain a set of fragments related to a specific topic. The corresponding sequence diagram is shown in Figure 4.

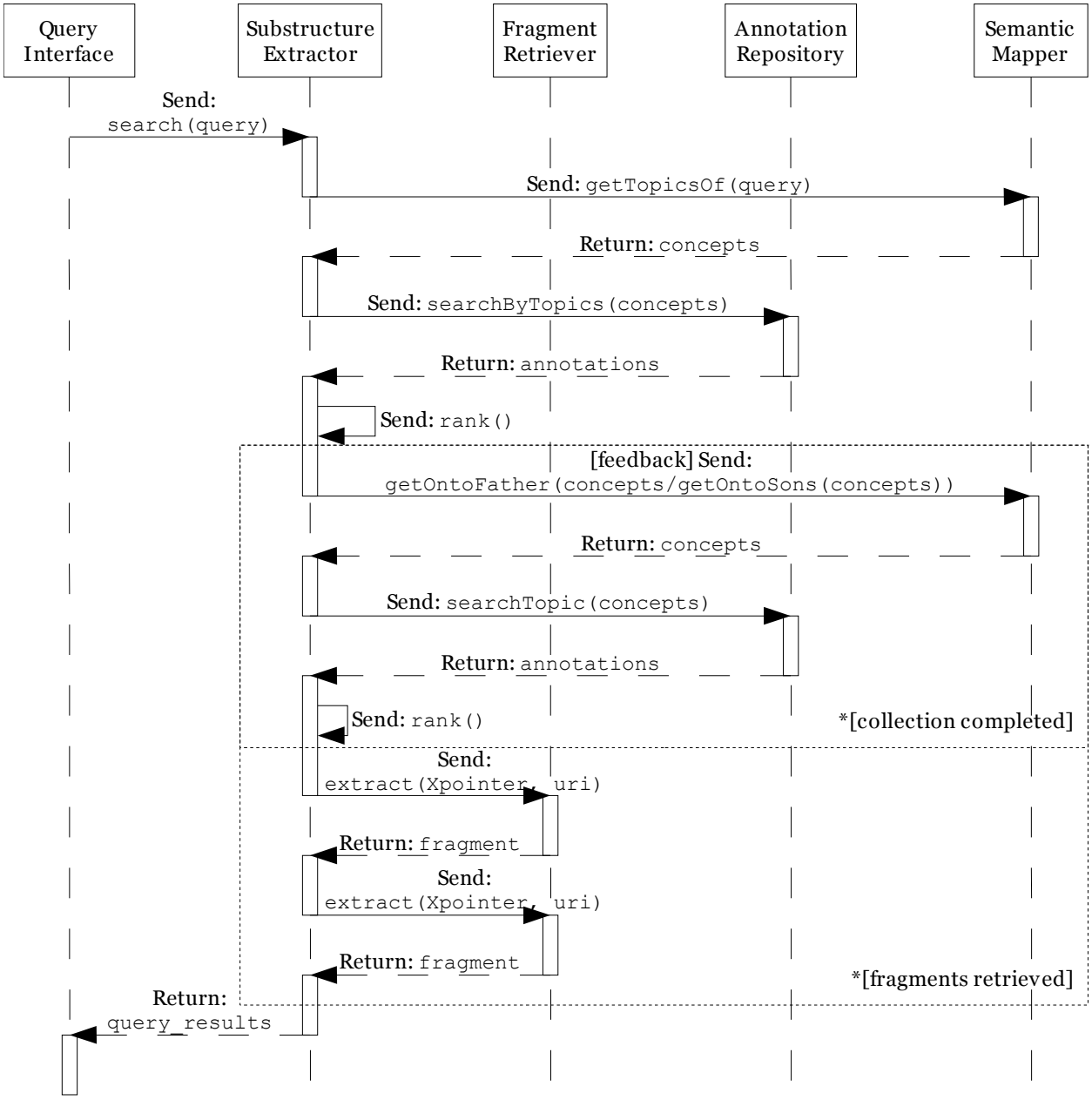

Figure 4: semantic information retrieval scenario.

The semantic information retriever module sends a search request to the Semantic Search Engine specifying the topic of interest by means of a query string or a

short text. The search module takes the received text and interacts with the Semantic Mapper for concepts extraction. The Semantic Mapper stems the query text and tries to match the text against the synsets associated to each concept in the ontology. When the text has been converted to relevant concepts the semantic search module tries to query the Annotation Repository for related annotations. The last module searches the annotation physical storage for concept matching, leveraging advanced features like taxonomic organization and Level Of Detail detection.

Once relevant annotations are retrieved, the Search Engine analyzes query results and if necessary starts a new interaction with the Semantic Mapper for query refinement, using semantic relationships occurring between ontology concepts. The Annotation Repository is then queried, and the process loops until a set of really relevant annotations is found.

When the collection of relevant annotations has been found, the Semantic Search Engine contacts the Substructure Extractor and Retriever for fragment fetching. Every fragment is subsequently returned to the querying module and the search task reaches its end.

In the second scenario (Figure 5), one possible deployment for an automated semantic indexing service based on the DOSE platform is presented. An external module like a common crawler provides a stream of URIs corresponding to a set of web resources that should be annotated.

The DOSE service access point is, in this case, the Indexer module which provides all necessary functionalities for unsupervised semantic annotation. Contacted by the external module, the Indexer starts to work interacting with the Substructure

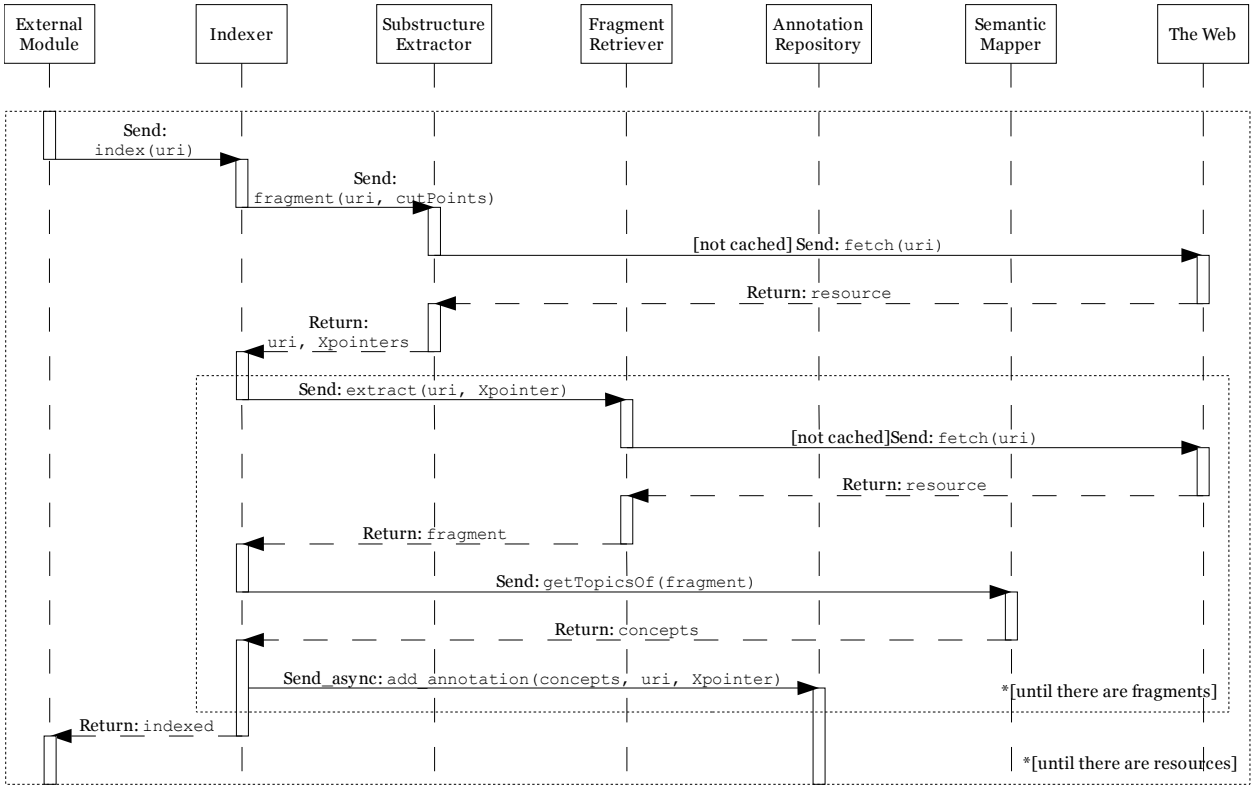

Figure 5: automatic annotation scenario.

Extractor for resource fragmentation, giving the cut point definitions for each document and obtaining a set of XPointers for identified resource substructures.

The subsequent interaction has the Fragment Retriever as target module: for each XPointer received in the first step the Indexer interacts with the Retriever in order to obtain the corresponding resource fragments.

Once a fragment has been retrieved, the Indexer sends it to the Semantic Mapper for ontology mapping via the `getTopicsOf` message. The Semantic Mapper takes the fragment and applies to that a "tdf/idf" ranking for appropriate synsets

associated to each ontology concepts: a set of concepts related to the fragment is therefore extracted and returned to the Indexer. Each concept returned has a relevance weight that establishes a ranking between the mined semantic associations.

After the last interaction the Indexer has collected the information needed for metadata creation, thus an annotation is defined for each specific topic-fragment mapping and is sent to the Annotation Repository for persistent storage. The retrieve-map-annotate cycle repeats until the end of XPointers list has been reached, taking into account semantic relationships between fragments (hierarchy, in first instance). The Indexer then alerts the higher level module for the task completion.

4.5 Multilinguality in DOSE

A complete overview of the DOSE platform was necessary, in order to better individuate the practical additions to be made for true multilingual support, and better define their integration with pre-existing modules. Hence, what really matters from the point of view of multilinguality is the conceptual organization of the system, and the way it deals with the two key requirements presented in Section 2.4.The key feature of DOSE in this regards can be stated as follows: *the ontology language is totally independent of the synset language*. In fact, if ontology concepts were labeled with an alphanumeric string like #NODE64, the system functionality would still remain the same. This choice would not be a good design since it would make the ontology unreadable to humans, however it clearly shows that concept labels are unrelated to synset languages. Strategies to exploit this peculiar feature of the system will be outlined in the next chapter, and a sample implementation will be given in Chapter 6.

CHAPTER 5

GOALS AND SOLUTIONS

The primary objective on which this thesis tries to focus is "multilingual support to semantic elaboration architectures" in general. More practically, the proposed approach aims at exploiting the peculiar organization underlying the DOSE platform, in order to enable full multilingual support to the typical tasks performed on a semantic architecture of this kind, namely annotation (of resource metadata) and retrieval via semantically-meaningful search queries.

This chapter is devoted to explain which strategies have been adopted to achieve multilinguality in DOSE. As first, the chapter starts by trying to translate abstract objectives into practical requirements to be satisfied in a Semantic Web architecture (DOSE in particular). Adopted solutions are then described, listing the assumptions taken into account during deployment when necessary.

5.1 Practical requirements for the current implementation

As we have seen in Chapter 4, the two main scenarios on which the DOSE platform operates are those of indexing and retrieval: the former process creates annotations (semantic links between concepts and resources) and the latter navigates them (to collect the ones whose semantic reference is requested by user queries). A proper implementation of multilingual support should then be strictly related to these two tasks, and act to extend their functionalities to the additional language parameter.

The problem of multilinguality within the indexing scenario does clearly affect the creation of annotations. An annotation is originated by the occurrence of a concept

in a resource; such concept can appear as one of many possible lexical representations, which differ consistently depending on language. The language of expression for a resource must therefore be identified before annotating, for the indexer to exploit the proper, language-dependent set of lexical entities occurring in the analyzed text. Moreover, language ID should be stored as an additional property for each annotation, to allow repository queries to distinguish between information in different languages.

Considering the search task, we can expect a double language requirement to come from the user: one for search results, and the other for query strings. If search operations can expect annotations to come with a language attribute, the goal of language-dependent information retrieval can be effectively accomplished – basically, it requires just to manage one more query parameter. In this case, the biggest issue is represented by the interpretation of queries: if a language specification for the query is not given explicitly by the user (who could select it among a list of supported ones, for example) it has to be implicitly inferred from the query text, in order to map search keywords with concepts recognized by the system – language recognition methods should then be applied again, this time to user queries. More than that, it could be sometimes very interesting to express queries in one language and obtain results in other languages: such a performance could obviously be possible only by mantaining independence between concepts and their lexical representations. The whole set of functionalities to be added to the system could be summarized as follows:

Indexing:

- discover language information about the resources to be indexed:

- when this information is explicitly stated: conform to language declaration standards, deal with implied definitions of the language property;
- when this information is missing: activate alternative strategies to attempt retrieval of language specification at the document substructure level;

- consequently setup the indexing system on the language just retrieved, in order to ensure proper coherence for contents analysis;
- store language attributes within semantic metadata, to allow convenient binding with original resources;

Search:

- accept queries (ideally) in natural language coming from the user, accompanied or not by explicit language preferences for results;
  - if a request for language is implicitly stated (i.e. inferrable from query text), activate proper strategies to attempt language detection (possibly reusing those already adopted for indexing);
- make all necessary adjustments on the search engine to allow for language-based discrimination of search results.

5.2 Main solutions adopted

End-user multilingual requirements should have been delineated, along with consequences implied on the system. The approach proposed here tries to deal with such requirements, by offering two main solutions that can be generally referred to as "extension to language" and "language recognition". Both are presented together with

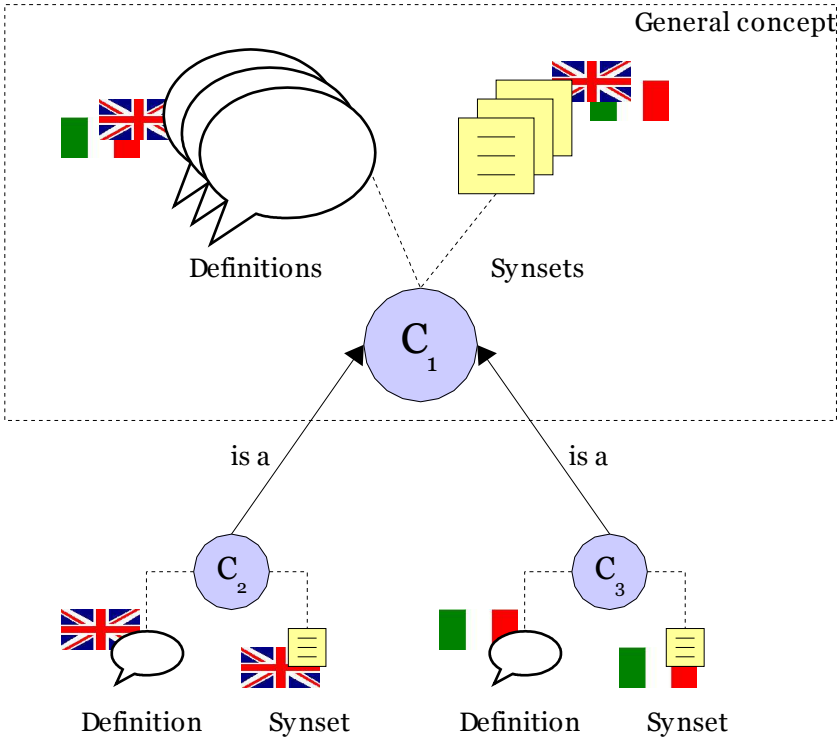

Figure 6: multilingual ontology deployment.

standard and custom assumptions, taken into account during their deployment on the architecture.

5.2.1 Extension to language

Section 4.5 pointed out an important feature of DOSE: the independence between languages used to represent ontology concepts and synset terms. The present approach exploits this language-independent ontology embodied in the system by linking each concept (defined as an high-level entity) to a set of language-specific (definition, synset) pairs (Figure 6). A concept definition is a short, human-readable text

that identifies the concept meaning as clearly as possible, and that is expressed in a specific language, while a synset – as already defined in Chapter 4 – is a set of near-synonymous words that humans usually adopt to identify that concept: now it varies according to language. Each concept is then represented as:

```
concept := concept_id, lexical_representation
lexical_representation := (lang_id, description, synset)+
synset := (word)+
```

The ontology is physically distinct from definitions and synsets, thus allowing separate management of concepts and language-specific information (the semantic and textual layer, as we could respectively call them). Language specific semantic gaps are supported by including in some concepts the definitions and synsets in the relevant languages, only. This assumption guarantees sufficient expressive power to model conceptual entities typical of each language, and at the same time reduces redundancy by collapsing all common concepts into a single multilingual entity.

Synsets and textual definitions should be created by human experts through an iterative refinement process. A multilingual team works on concept definitions by comparing ideas and intentions, aided by domain experts with linguistic skills for at least two different languages, and formalizes topics in a mutual learning cycle. At the end of the learning cycle, the team reaches a common agreement and defines a sound base of concept definitions with associated synsets. This interaction cycle eventually produces two sets of concepts: general concepts and language-specific concepts.

Formally, the two sets are modeled in the same way inside the ontology, by the definition of language independent concepts. However, concepts belonging to the first category will be linked to definitions and synsets expressed in each supported language, while those belonging to the second set will be linked to smaller subsets of languages.

The complex interaction between ontology designers, users and domain experts at design time must build upon the availability of an international network in which people cooperate to model a defined knowledge domain. Such kinds of networks have already been proposed, for example in the EU Socrates Minerva CABLE project. The CABLE project involves a group of partners with proved skills in learning and education and promotes cooperation in order to define learning material and case studies for continuous education of social workers. In CABLE, teams of experts in social sciences and education cooperate (with the support of "multilingual" domain experts) in defining case studies, teaching procedures and related semantics in a so-called "virtuous cycle". The CABLE project can effectively apply the proposed approach to multilinguality, leveraging the already defined "virtuous cycles" and defining a multilingual ontology for education in social sciences.

The outcome (i.e. ontology, definitions and synsets) constitutes an effective core for the implementation of multilingual semantic environments. Resource occupation is by a great extent comparable to that of a monolingual framework thanks to redundancy elimination, while expressive power is as effective as needed. In addition, almost no "synonymous" relationships are needed in the ontology, thus preserving the simplicity of its structure, that results more reasonable and manageable (a single modification is automatically reflected in all languages).

In this connection, one possible enhancement for the present design could be represented by the linkability to existing semantic networks like WordNet – this could spare a lot of implementation efforts for the semantic infrastructure to be used within the system. Nonetheless, the present collection of WordNet-derived networks is constituted by language-specific ontologies and synsets, which have been developed independently basing on different cultural representations of knowledge. This does not fit very well with the centralized approach proposed here; it is however possible to use WordNet synsets to expand those already provided with the system, thus enhancing the effectiveness of both the annotation and query phases.

5.2.2 Language recognition

The process of language recognition has to be mainly activated during the annotation phase, when conceptual links between available resources and concepts known by the system are created. Resources are analyzed by single substructure elements per indexing step; (almost) each element could contain a different language, which has to be recognized in order to apply the corresponding synset and collect concept references. Language detection methods should then be applied for each new document fragment that is fetched to the system.

The key idea is to retrieve standard language IDs each time they are provided with resources: this avoids the overhead caused by the use of special heuristics, taking advantage of already available information. When no clue can be found, a proper analysis strategy is applied in order to recreate the missing language specification – obviously, if even this approach fails to provide a precise language value for the current

fragment, a default one (preset to the system) is then adopted. The whole method is detailed in the following points, each one proposing an alternative solution for language detection. Solutions are listed in their application order, thus implying that the failure of a tactics in returning an acceptable language value does result in the invocation of the next one.

1. *Validate a possible user request.* Supposing to deal with an user that already knows the language of a certain set of resources to be indexed, it seemed proper to give his/her language request the maximum priority. Apart from being the most effort-relieving case for the system, this also allows for explicit user-driven resource annotation (i.e. save time by annotating half of the resources in a bilingual document). However, it is still unclear whether such possibility should be directly visible to end-users (i.e. through a "force language" option) or even be hidden by the system: a certain number of periodically re-indexed resources on which to express queries could be assumed by users, these ones having access only to the search engine;
2. *Retrieve already-present language attributes.* After considering possible user commands, this is the easiest way to find language IDs. The system should in fact be able to take advantage of standard specifications directly provided with input documents, in order to avoid the use of language recognition algorithms whenever possible. Considering to privilege conformity to one standard only among the ones listed in Section 3.4.1, the "lang" attribute was chosen. This is primarily due to the following reasons:

- it is tag-specific: in other words, it allows to specify language for single document fragments, matching a single page element with a single language (i.e. diversely from "Content-language", that sometimes could return many language values for the whole document). This exactly meets our requirements, since the system must be able to perform semantic annotation at the document substructure level;
- it is oriented to applications like the one currently provided, instead of being targeted to specific user agents, as for "Content-language" (mainly destined for browser use).

Provided that this attribute has to be individuated each time a new fragment enters the system to be indexed, the first encountered tag only is taken into account (this presumably being the one that determined the cut point). Of course, each time a resource element lacks this attribute, this strategy cannot return an acceptable language ID.

3. *Guess language via heuristics*. This approach activates a simple analysis method on fragment contents in order to collect lexical entities peculiarly related to one language or another. The lexical entities adopted for the present application are the so-called "language stopwords": they can include both functional units (such as articles, conjunctions, prepositions, etc.) and lexical forms of common use (such as conjugations of auxiliary verbs). Stopwords retain a couple of key features that make them very suitable for this use:

- they are exclusive for the language of pertinence. Apart from rare overlappings, these speech particles are the ones less likely to be shared by two or more languages, thus being the most distinctive feature of each language. Of course, uncertain situations may show up, where further disambiguation would be needed. What follows is a possible case of overlapping, where stopwords recognized by the system appear underlined:

  <H1>A day in Chicago is never enough.</H1>

  Indeed, the language of expression for the above sentence is English, but if we consider the point of view of a language detector that bases its reasoning on stopwords parsing, such an observation is not so straightforward. When scanning the tagged string left-to-right, the parser encounters "A", "in", "is", in this order; both "A" and "in" may be linked to two different languages, English and Italian ("in" is a preposition in both languages, while "A" is an article in English and a preposition in Italian). Therefore the parser cannot figure out the language of expression after two out of three stopwords, because two different languages have scored the same occurrence rate at this point – disambiguation comes with the third stopword "is", that definitely leads to the choice of English. This is just an example on a minimal sentence (far from the magnitude order of many Web documents), yet it demonstrates that possible lexical overlappings between languages of even distinct origin

(such as English and Italian) can be often overcome with a simple "stopwords count" mechanism, as explained below (when such mechanism is unresolving, document analysis in multiple languages may be taken into account);

- they have an average high occurrence. Because of their irreplaceable role in sentence structure, they usually retain high occurrence levels within short ranges: even with a minimum parsing effort, this often allows for meaningful results in language detection.

After a certain number of words within the text fragment currently processed has been evaluated, we should obtain a list of languages whose presence in it is proportional to the number of related stopwords found. The language with the highest occurrence wins the contest, i.e. it becomes the one accepted for semantic analysis. This method tries to address the average lack of language specifications, due to the typical poor standards compliance of a large majority of the documents nowadays present on the WWW. Such a method could prove ineffective only in case the structure of parsed text does not allow for a sufficient presence of stopwords within a limited text range (i.e. empty elements, keyword lists, etc.).

4. *Assume the language of the ancestor element.* Each resource element whose language is left unspecified can receive it from the closest ancestor element that declares a value for "lang". This arrangement does in fact apply the standard

specifications for the "lang" attribute: the only difference is that ancestor language information could have been originated also by heuristics. This seemed a very reasonable, additional strategy to retrieve language that builds upon the assumption that, if a certain document fragment is written using a certain language, its descendant subelements will probably be expressed with the same language too.

This approach exploits the fragmentation process carried on by the Substructure Extractor/Fragment Retriever modules (see Section 4.3) that start at the highest hierarchy level, and then continue down to lower ones recursively, using a top-down approach [37]. Higher level fragments are always extracted before lower level ones, guaranteeing that the parent language is known. At this point, an acceptable language ID is supposed to be retrieved in any case – the process can go up approaching the top hierarchy level within a document, and if even here it does not obtain acceptable results (which is very unlikely to happen), it returns the default language originally preset in the system.

CHAPTER 6

IMPLEMENTATION

This chapter describes the test implementation of language management functionalities developed under DOSE. A brief description of the current implementation of DOSE is given in Section 6.1, so as to better outline the interaction between the newly introduced language module and the rest of the system in Section 6.2. A basic test of the provided implementation is then illustrated in Section 6.3.

6.1 Technical framework

A prototype version of DOSE has been implemented, and its main functionalities are operational while refinement and optimizations are still ongoing.

The programming language chosen for its development is Java, because of its high portability and several APIs already available to suit implementation needs. Each module has been implemented as a standalone XML-RPC server, which offers specific services published to other modules by means of a common feature file (a basic form of a directory service). A single module encapsulates a number of XML-RPC clients that depends on how many external services the module interacts with.

The Substructure Extractor and Retriever was developed using the XPath Explorer API [38], in order to extract XPath/XPointer constructs from identified web resource substructures. Fragmentation is based on identifying specific tags like <H1,..,Hn> in (X)HTML and custom tags in XML. XML and XHTML documents are supported, as well as HTML documents converted into XHTML using the Tidy API [39].

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:j.0="http://elite.polito.it/">
<rdf:Description rdf:nodeID="A0">
    <j.0:uri rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
file:c:/testpages/test1.htm</j.0:uri>
    <j.0:xpath rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
/html/body/h1[2]</j.0:xpath>
    <j.0:topic rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
http://elite.polito.it/ontology#Cliente</j.0:topic>
    <j.0:frequency rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
1</j.0:frequency>
    <j.0:father rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
/html/body</j.0:father>
    <j.0:weight rdf:datatype="http://www.w3.org/2001/XMLSchema#double">
1.4026294550962348</j.0:weight>
    <j.0:dateTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">
26-ott-2003 17.25.35</j.0:dateTime>
  </rdf:Description>
```

Figure 7: example of automatically generated annotations.

The Annotation Repository has not yet reached its full potential, however it gives a first implementation of taxonomic organization and a preliminary realization of LOD (Level of Detail) detection.

Annotations are composed of several fields, one of which points at the "parent" of the same annotation expressed as Xpointer (Figure 7). When several annotations concerning the same document are retrieved in a single search, a generalization step is done and the parent shared by all annotations is retrieved.

The Semantic Mapper has been implemented using the Jena API [40] for ontology access and navigation and the Snowball API [41] for syntactic to semantics mapping and lexical stemming. Other than multiple, language-dependent synsets for each concept, internationalization obviously relies on the functionalities of the Snowball stemmer.

The Search Engine was implemented only to assess annotation effectiveness. It still does not implement semantic capabilities, but only acts as an interface for Annotation Repository retrieval operations: no tests have therefore been conducted on multilingual queries. Such engine accepts a set of words as input or a short fragment of text, and maps the set to ontology concepts using the service offered by the Semantic Mapper. Annotations are then retrieved from the Annotation Repository by concept matching and subsequently ranked by relevance using specified parameters (weight). When all annotations are retrieved, only a selected subset of them is forwarded to Fragment Retriever for resource fetching.

6.2 Implementation and interface

6.2.1 The language awareness problem

The first support for multilinguality in DOSE was added on a version of the architecture like the one just described, and primarily focused on the indexing scenario, i.e. the creation of language-characterized annotations from multilingual resources. One of the first encountered problems concerned the location and control of language information within the system.

Not all modules in the architecture need language awareness: in fact, language information flows, in the architecture, from upper layers to the deepest ones, being

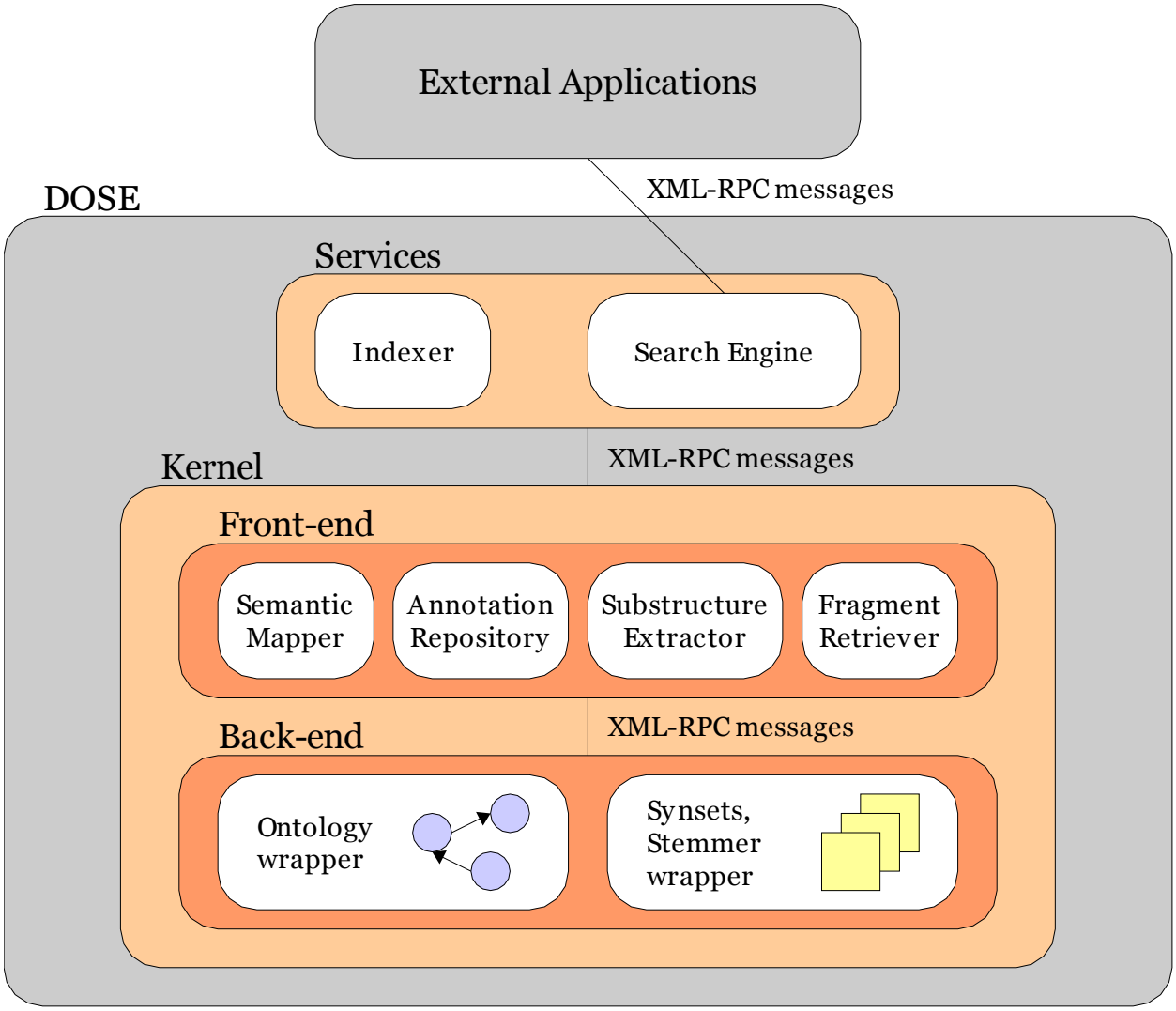

Figure 8: layered DOSE architecture.

filtered at each intermediate level. To better clarify these assertions, we can reconsider all system modules with a layer-based cathegorization: they can be arranged on three layers, starting from the interface toward external applications, down to the so-called "wrappers".

The first layer, that we call the "Service layer", includes all modules that expose interfaces to the outer world, i.e., the Indexer and the Search Engine. The "Kernel layer" lies one level below the Service one and is actually composed by two sub layers: the "Kernel front-end layer" and the "Kernel back-end layer". Core modules, i.e. the modules that offer access to semantic and syntactic information, are located at the front-end level. They are, respectively, the Semantic Mapper, the Annotation Repository, the Substructure Extractor and the Fragment Retriever. Finally, back-end modules are essentially composed by wrappers, which encapsulate ontology, synsets and definitions. A graphical depiction of DOSE layers is given in Figure 8.

Considering the need for language awareness layer by layer, we can notice that, at the service layer, all existing modules should manage language-related information; in particular, the Indexer module should know the language of the indexed resources in order to create relevant annotations, while the Search Engine can use the same information to offer complex search functionalities.

Descending one level in the architecture, we found that language identification is useful for only some modules, like the Annotation Repository and the Semantic Mapper, while the other modules could be totally language un-aware. The Annotation Repository requires language management capabilities in order to correctly store and retrieve semantic annotations – as a direct consequence, the structure of annotations should take into account language information. The Semantic Mapper is a somewhat atypical module, in which language identification is not needed for executing internal tasks; instead, it is used to correctly select underlying wrappers in order to perform significant text-to-concept mapping.

```
<rdf:Description rdf:nodeID="A7">
    <j.0:uri rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
file:c:/testpages/testIt.htm</j.0:uri>
    <j.0:xpathrdf:datatype="http://www.w3.org/2001/XMLSchema#string">
/html/body/p[4]</j.0:xpath>
    <j.0:topic rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
http://elite.polito.it/ontology#Corsia preferenziale</j.0:topic>
    <j.0:language rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
it</j.0:language>
    <j.0:frequency rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
1</j.0:frequency>
    <j.0:father rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
/html/body</j.0:father>
    <j.0:weight rdf:datatype="http://www.w3.org/2001/XMLSchema#double">
1.1020799941005643</j.0:weight>
    <j.0:dateTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">
21-ott-2003 14.01.20</j.0:dateTime>
</rdf:Description>
```

Figure 9: extra language property for annotations.

The last layer of the DOSE platform does not need language awareness, but synsets and definition wrappers should both retain a language identifier that enables selection by the Mapper. Almost all service layer modules will then require language awareness, while kernel back-end modules will be language-unaware.

6.2.2 Extension to language

To extend DOSE for multilinguality, it was necessary to apply the conceptual solution detailed in Section 5.2.1 to the current platform prototype. This resulted in the use of a single ontology (described in a .rdfs file) and two distinct synsets for Italian and

English languages (two .rdf files), matched with the ontology: the Semantic Mapper can thus have the correct synset when a resource in one of these two languages is processed. This structural change affects several modules.

The synset wrapper now retains a `language` field, which keeps track of the synset stemmer currently used, and allows the correct language-dependent synset file to be loaded any time it has to be changed. The Semantic Mapper must now handle an additional language argument for its primitive `getTopicsOf`, charged to perform language-dependent semantic mapping; synset switching can be controlled via the new `setupSyn(language)` private method. The Annotation Repository adds an extra `language` property to each annotation (see Figure 9), and provides an additional `getLanguage` method to discriminate annotations based on language IDs. Finally, also the Indexer manages an additional language specification in its methods, to coordinate sublayer modules and pass them the language ID for each indexing step.

6.2.3 Language recognition

All discussed modifications have been implemented into the original architecture and are based on the existence of language IDs. To retrieve such IDs and ensure continuous operation in multilingual semantic environments, a new module "Language Detector" has been introduced. The module offers language recognition facilities and is called whenever a language parameter is missing; it has been located at the front-end level, as provided services are usually required by service layer modules. The Detector offers the main primitive `findLanguage(text)`, available via XML-RPC call, that is usually invoked by the Indexer module to retrieve the language ID for each

analyzed resource. Such primitive exploits the two private methods `lang(text)` and `detect(text)`, which apply the strategies for "lang" value retrieval and heuristics on stopwords outlined in Section 5.2.2, points 2 and 3 respectively.

The "ancestor" language solution is instead implemented in the Indexer: since this solution relies on the storage of language IDs previously retrieved for parent fragments, and the whole system is implemented with scalability in mind, no state information can be held in the Detector. The Indexer is the only module (together with the Search Engine) able to manage user sessions; it is thus provided an `ancestorLanguage(xpath)` method that retrieves the language of ancestor elements with respect to the current Xpath. Two parallel strategies have been actually implemented for this task: one that uses the Annotation Repository, and another one that relies on a stack within the Indexer.

The first strategy starts by rebuilding the correct parent Xpath for the fragment currently analyzed, and then queries the Repository to recollect its language; if the operation is unsuccessful, the process continues recursively until the top level is reached. The second strategy uses a stack, which keeps track of languages along the root-to-leaf path in the document hierarchy, the leaf being the current fragment.

The whole number of changes and additions in system modules is summarized in Table III.

Module	*Change/addition*	*Description*
Synset wrapper	Field `language`	Allows for the use of multiple synsets/stemmers
Semantic Mapper	Method `setupSyn(language)`	Switches to the synset corresponding to `language`
	Argument `language` in `getTopicsOf`	Extends mapping to multiple languages
Annotation Repository	Property `language` in annotations	Stores the language ID of each annotated resource
	Method `getLanguage`	Helps to discriminate annotated resources basing on language
Indexer	Parameter `language` in method `index`	Extends indexing to multiple languages
	Method `ancestorLanguage (xpath)`	Retrieves language ID of `xpath` ancestor elements
Language Detector	Method `findLanguage (text)`	Finds language ID of `text` by exploiting `lang` and `detect` methods

Table III: system changes and additions for multilingual support.

6.2.4 Multilingual indexing

To better explain how system modules interact during a multilingual indexing phase, a proper UML sequence diagram is given in Figure 10. The only difference between a monolingual scenario stands between the retrieval of each fragment and its submission to the Semantic Mapper: if a fixed language has not been specified by the external module, the Indexer queries the Language detector on such fragment, by sending the message `findLanguage(fragment)`. If the Detector does not return an

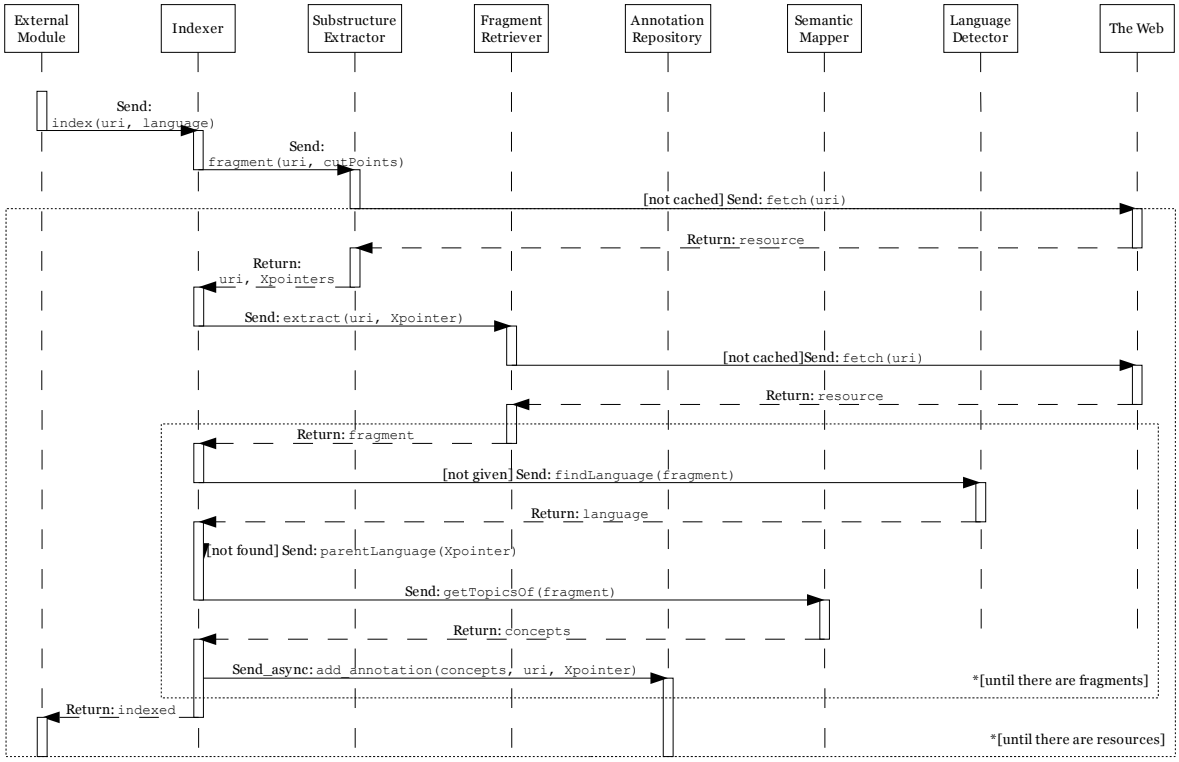

Figure 10: multilingual indexing scenario.

acceptable value, the Indexer itself retrieves a parent language specification via the `ancestorLanguage` method. The obtained language attribute is then sent to the Mapper, which will set up the correct synset file. Annotations will be stored with a language attribute directly specified by the Indexer.

6.3 Experimental setup

A simple experimental setup has been defined for this first multilingual version of DOSE, in order to assess the feasibility of the approach and to show how the open

architecture of the platform can easily support multilinguality by exploiting its peculiar organization. The system uses an ontology on disability, which has been developed in collaboration with the Passepartout service of the City of Turin, a public service for disabled people integration and assistance in Turin, Italy.

Since the Passepartout service does not offer a bilingual version of its site for the moment, the first question to solve was about finding a sufficient collection of documents to evaluate the newly introduced multilingual capabilities of DOSE. More than that, as the semantic domain of disability chosen for the ontology (core of the whole semantic processing) is quite peculiar – both in terms of lexicon and contents – only another dedicated site like the Passepartout one would have provided a sound testing base. The requirements for a suitable set of test documents could then be summarized in the following points:

- *directly or indirectly related to disability.* As already stated, in order to exploit an ontology centered of the issue of disability, documents to be analyzed should concern disability (in whatever form), so to maximize both the number of possible matches with synset terms, and deriving annotations;
- *multilingual (bilingual at least).* Obvious for the present evaluation purposes; for the moment, synsets to back the available ontology concepts have been provided for English and Italian only. Therefore, the set of resources to be chosen should incorporate these two languages;
- *with parallel structure.* "Parallel structure" is required in order to test one of the most peculiar features of DOSE, i.e. the ability to produce annotations on

resources whose actual localization is refined "at the document substructure level". We could define two "parallel documents" as documents retaining the same structure, showing the same contents expressed with a different language in each parallel version of the same element. Provided that a multilingual website is found, where multiple versions of the same document share the same markup and content layout, differing only on the language of expression, such a document set could be accepted because of its "parallel structure";

- *substantial enough*. A good number of documents is needed for the tests to prove sufficiently reliable – the more documents are available, the larger possibly becomes the portion of ontology that can be matched and tested;
- *containing highly descriptive, organically structured documents*. Pages should be hierarchically structured with respect to paragraph contents; such paragraphs should be as descriptive as possible, so to effectively simulate the real kind of information a user is tipically looking for. Lists of links and keywords should be avoided, in that they are even lacking important text elements such as stopwords (refer to Section 5.2.2, point 3).

To find a collection of resources that retains all of the above features is far from easy, especially with respect to multilinguality requirements. As a matter of fact, information on disability often concerns special legal dispositions or local initiatives, and for the most part is therefore country-dependent: even on the Internet, this causes the diffusion of many monolingual related websites, but very few multilingual (or even bilingual) ones.

All experiments have been conducted on twenty documents coming from www.asphi.it [42], the website of an italian association committed in promoting computer science technologies addressed to the support of disabled people; the site is available both in English and Italian. Testing has been divided in two phases: the first one takes into account the strategy proposed to extend DOSE for multilinguality (using "parallel annotations"), while the second one simply consists of performance evaluation for the new language detection module.

6.3.1 Parallel annotation

This experimental scenario evaluates the recent extension of DOSE to multilinguality, trying to demonstrate that this new version of the system is able to perform language-transparent contents mapping: this can be obtained by showing that two sets of parallel documents are (almost) simmetrically mapped to the same concepts. Such documents are the ones mentioned before [42], and multilingual indexing of these pages produced approximately 5100 annotations, both for the English and Italian versions, totalling about 24 Mb in .RDF files. This annotation phase has been carried out by explicitly specifying the language for resources (without usage of any language detection method – Section 5.3.2, point 1 only), in order to improve the quality of the tests by avoiding the tampering of results due to possible defects of the language detector (whose exclusive testing is discussed in Section 6.3.2).

At the semantic level, the platform should (ideally) annotate all pairs of translated fragments with an equivalent set of concepts. Such achievement has been verified by computing the correlation factor between parallel document fragments at

different hierarchy levels, upon previous analysis of the annotations stored in the repository for both Italian and English documents. In this case, the <BODY> and <Hx> fragmentation levels have been considered.

The adopted correlation measure is directly inherited from the "Vector Space Model" methodology [36] – traditionally used to quantify the degree of correlation between a query string and the corresponding result pages, this technique expresses the correlation between two parallel page fragments as the cosine of the angle between the vectors representing such fragments. Correlation is defined according to the classical vector space model and considering each concept as an independent dimension: the higher a concept is weighted within a fragment, the stronger the corresponding component will be in its vectorial representation. The resulting table with correlation factors for all possible fragment pairs at <BODY> level is shown in Figure 11, where rows and columns are labeled according to the source of fragments, using a "language/page/fragment" syntax.

Sub-table A	EN/Page1/BODY	EN/Page2/BODY	EN/Page3/BODY	EN/Page4/BODY	EN/Page5/BODY	EN/Page6/BODY	EN/Page7/BODY	EN/Page8/BODY	EN/Page9/BODY	EN/Page10/BODY
IT/Page 1/BODY	0,56	0,09	0,14	0,14	0,28	0,38	0,3	0,28	0,26	0,35
IT/Page 2/BODY	0,17	0,49	0,33	0,43	0,23	0,18	0,22	0,17	0,19	0,15
IT/Page 3/BODY	0,15	0,28	0,57	0,34	0,17	0,14	0,14	0,14	0,15	0,19
IT/Page 4/BODY	0,2	0,32	0,4	0,63	0,22	0,21	0,23	0,24	0,28	0,2
IT/Page 5/BODY	0,26	0,12	0,12	0,15	0,47	0,17	0,27	0,22	0,27	0,19
IT/Page 6/BODY	0,37	0,16	0,18	0,25	0,24	0,5	0,34	0,36	0,32	0,36
IT/Page 7/BODY	0,33	0,09	0,1	0,2	0,24	0,32	0,59	0,28	0,4	0,26
IT/Page 8/BODY	0,27	0,08	0,08	0,14	0,18	0,41	0,29	0,4	0,26	0,26
IT/Page 9/BODY	0,3	0,06	0,08	0,18	0,21	0,39	0,38	0,28	0,45	0,22
IT/Page 10/BODY	0,25	0,14	0,22	0,17	0,18	0,39	0,31	0,28	0,22	0,43
IT/Page 11/BODY	0,34	0,08	0,08	0,14	0,26	0,36	0,37	0,32	0,38	0,32
IT/Page 12/BODY	0,31	0,09	0,08	0,12	0,24	0,42	0,31	0,29	0,26	0,37
IT/Page 13/BODY	0,39	0,11	0,12	0,17	0,22	0,4	0,37	0,36	0,31	0,35
IT/Page 14/BODY	0,28	0,08	0,2	0,13	0,18	0,33	0,25	0,3	0,19	0,28
IT/Page 15/BODY	0,44	0,25	0,29	0,36	0,34	0,42	0,39	0,4	0,33	0,34
IT/Page 16/BODY	0,35	0,13	0,16	0,16	0,24	0,35	0,37	0,31	0,27	0,33
IT/Page 17/BODY	0,29	0,26	0,39	0,29	0,23	0,35	0,26	0,22	0,26	0,3
IT/Page 18/BODY	0,28	0,32	0,22	0,35	0,24	0,26	0,33	0,28	0,27	0,24
IT/Page 19/BODY	0,24	0,33	0,34	0,4	0,22	0,18	0,24	0,21	0,18	0,17
IT/Page 20/BODY	0,29	0,22	0,38	0,27	0,18	0,33	0,26	0,2	0,24	0,28

Table IV: correlation factors at the <BODY> level (subtable A).

Sub-table B	EN/Page11/BODY	EN/Page12/BODY	EN/Page13/BODY	EN/Page14/BODY	EN/Page15/BODY	EN/Page16/BODY	EN/Page17/BODY	EN/Page18/BODY	EN/Page19/BODY	EN/Page20/BODY
IT/Page 1/BODY	0,34	0,41	0,42	0,29	0,39	0,33	0,17	0,24	0,15	0,26
IT/Page 2/BODY	0,2	0,21	0,19	0,15	0,27	0,19	0,31	0,36	0,37	0,28
IT/Page 3/BODY	0,15	0,14	0,13	0,16	0,21	0,17	0,24	0,26	0,26	0,4
IT/Page 4/BODY	0,2	0,14	0,16	0,16	0,27	0,16	0,24	0,35	0,37	0,35
IT/Page 5/BODY	0,26	0,22	0,14	0,17	0,26	0,21	0,12	0,25	0,14	0,2
IT/Page 6/BODY	0,31	0,38	0,41	0,27	0,42	0,34	0,18	0,25	0,2	0,26
IT/Page 7/BODY	0,25	0,28	0,28	0,14	0,4	0,26	0,13	0,24	0,17	0,22
IT/Page 8/BODY	0,24	0,22	0,33	0,24	0,42	0,22	0,12	0,2	0,13	0,23
IT/Page 9/BODY	0,26	0,26	0,26	0,13	0,37	0,19	0,13	0,2	0,12	0,21
IT/Page 10/BODY	0,26	0,34	0,39	0,2	0,33	0,28	0,15	0,26	0,09	0,31
IT/Page 11/BODY	0,41	0,38	0,39	0,2	0,34	0,29	0,16	0,18	0,1	0,22
IT/Page 12/BODY	0,38	0,53	0,42	0,26	0,41	0,31	0,13	0,2	0,11	0,24
IT/Page 13/BODY	0,32	0,34	0,52	0,26	0,39	0,31	0,14	0,19	0,13	0,26
IT/Page 14/BODY	0,23	0,27	0,35	0,37	0,36	0,25	0,14	0,18	0,15	0,33
IT/Page 15/BODY	0,34	0,35	0,44	0,31	0,63	0,39	0,22	0,42	0,28	0,35
IT/Page 16/BODY	0,29	0,38	0,41	0,24	0,43	0,45	0,18	0,25	0,17	0,28
IT/Page 17/BODY	0,28	0,28	0,28	0,22	0,37	0,25	0,42	0,4	0,31	0,48
IT/Page 18/BODY	0,27	0,23	0,25	0,25	0,31	0,21	0,34	0,56	0,34	0,4
IT/Page 19/BODY	0,18	0,21	0,24	0,16	0,31	0,2	0,29	0,32	0,56	0,33
IT/Page 20/BODY	0,23	0,21	0,23	0,21	0,31	0,22	0,28	0,38	0,28	0,48

Table V: correlation factors at the <BODY> level (subtable B).

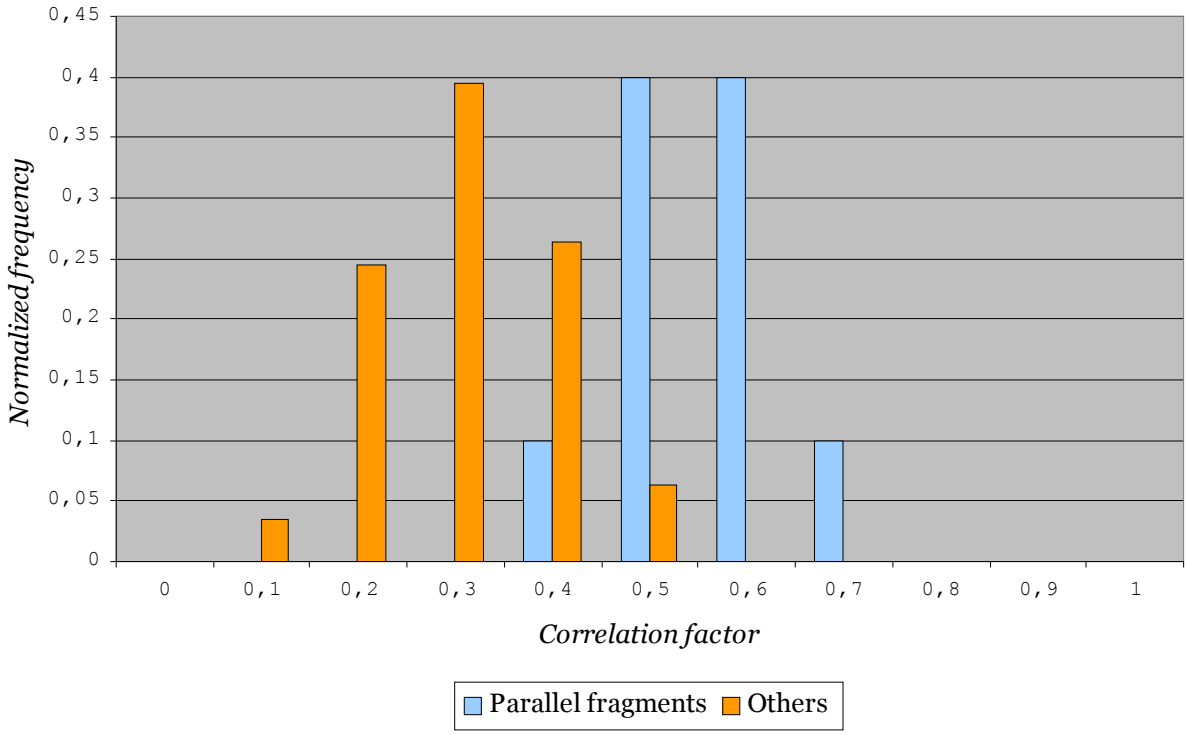

Figure 11: correlation distributions at <BODY> level.

Correlation data at each fragmentation level has been grouped into two sets: the first is composed by elements lying on the table diagonal, while the second includes all remaining elements. Elements lying on the table diagonal are the correlation factors between parallel fragments in the two languages: these are properly expected to hold higher values with respect to all others. Figures 11 and 12 depict the correlation distributions for the parallel sets at the <BODY> and <Hx> fragmentation levels.

Correlation values for "parallel" and "nonparallel" fragment pairs could be grouped in two different distributions, which give a visual representation of hit and miss

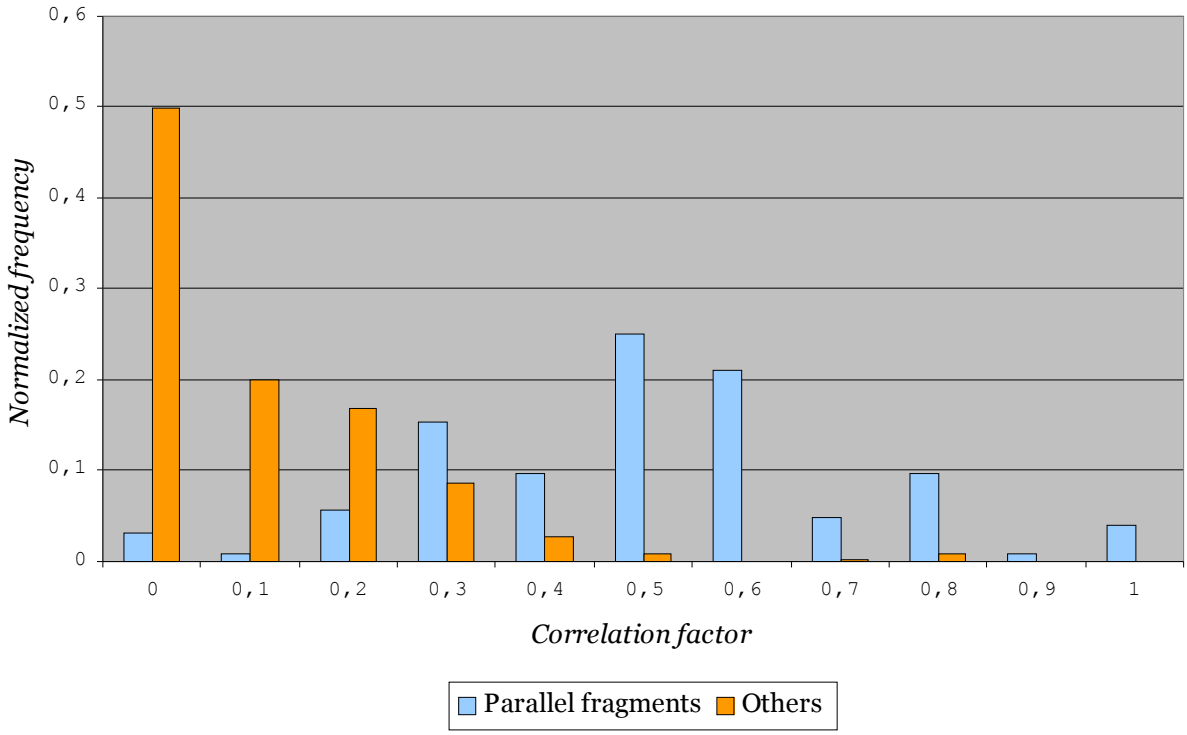

Figure 12: correlation distributions at <Hx> level.

ratios scored by the indexing system, respectively. For the sake of simplicity, such distributions could be seen as (approximate) Gaussian ones; considering for instance the <BODY> case, average values are obviously represented by the peaks centered around approximate values of 0.3 (nonparallel) and 0.55 (parallel).

Here one of the first positive responses from the system is represented by the average correlation assigned to parallel pairs. As said before, parallel fragments are expected to hold an higher correlation value, because of the sharing of most of their concepts – DOSE actually seems able to meet the expectations (0.55 > 0.3).

However, absolute correlation peaks for parallel fragments are not substantially showing high performance rates yet. This becomes quite evident when we consider the numerical meaning of such results in practice: an average correlation of 0.55 between documents with (more or less) the same contents means to have missed nearly half of the common concepts. But besides this consideration on (relatively poor) absolute values scored by the system on semantic correlation, what really matters as an encouraging result on the adopted methodology is the overall behavior of the two (approximate) distributions.

As a matter of fact, diagrams show that the distribution of the first and the second set are nearly separated. This means that corresponding fragments in different languages are annotated as belonging to a common set of concepts, while non-corresponding fragments are kept distinct by annotating them with reasonably different concepts. In spite of absolute rating, the platform is then able to distinguish between situations where correlation matters and other ones where it is not expected. The phenomenon becomes even more evident when considering a three-dimensional representation of all possible correlation samples, as shown in Figure 13 and 14, where the diagonal peak series stands apart from the rest of the graph.

This discussion on achieved results has mainly focused on the <BODY> level, but it can be easily re-applied to the <Hx> case: the main difference is that the range of concepts spanned by each fragment is usually smaller than before and focused on a more precise topic. As shown in Figure 12, here results are better for nonparallel fragments and worse for parallel ones: this is probably a natural consequence of the deeper semantic specialization of each fragment. As a matter of fact, random overlaps on

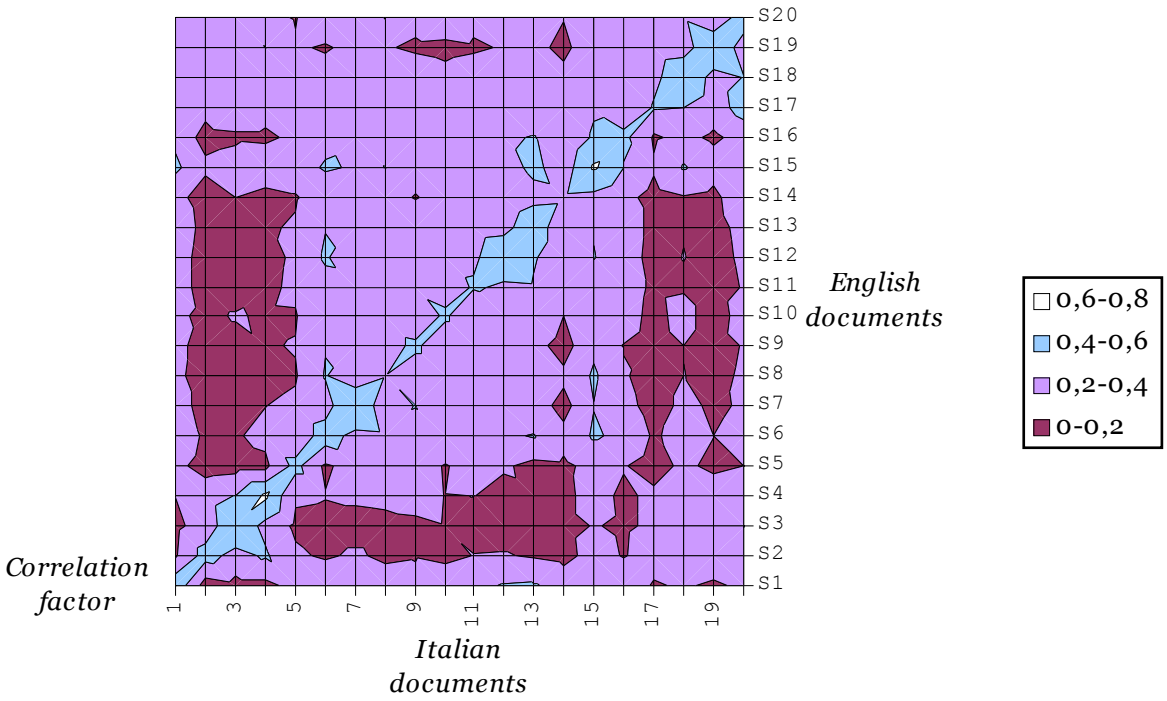

Figure 13: correlation factors at <BODY> level.

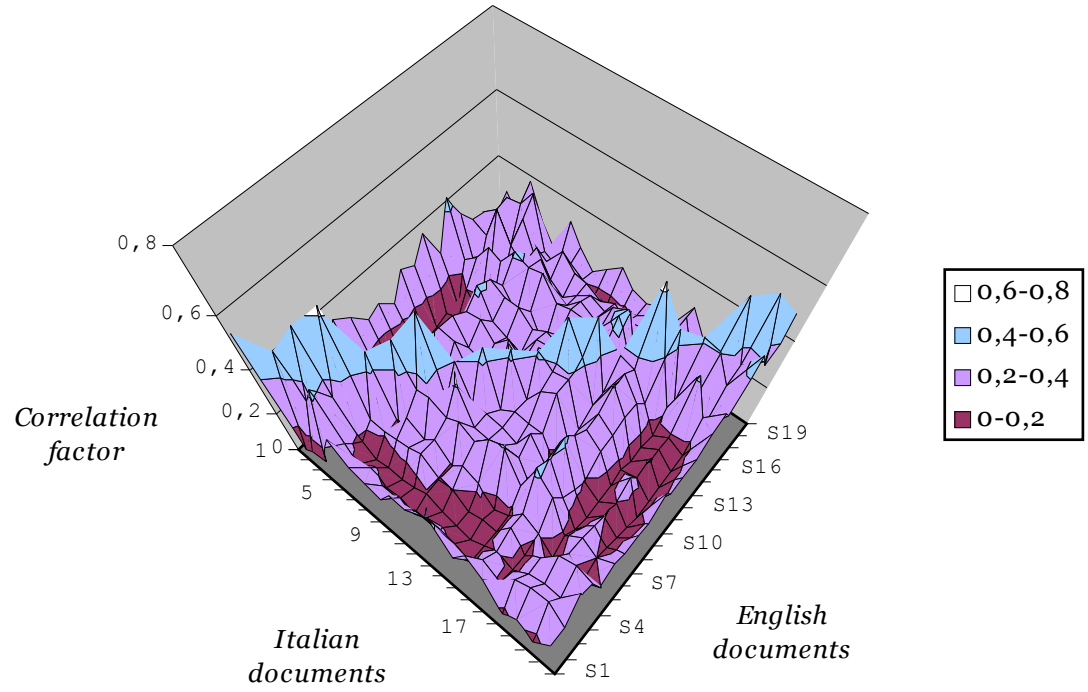

Figure 14: correlation factors at <BODY> level (alternate view).

some concepts are now more unlikely to happen between unrelated fragments; on the other hand, correspondencies between parallel fragments are based on a narrower set of concepts, thus being harder to spot.

6.3.2 Language recognition

A second experimental setup has been carried out, in order to test the effectiveness of the provided language detection module (described in Section 6.2.3). All documents analyzed in the previous section have been fetched again to the system, for batch language recognition at the document substructure level. Italian and English pages are alternated within the above sequence: for each page, this avoids fake language recognition due to the accidental restoring of default language settings directly inherited from previously processed documents. Such arrangement is motivated by explaining that a default language value is reassigned to each page, as the one chosen for the previous page in batch processing; with subsequences of documents written in the same language, this design feature could invalidate the test.

Two distinct evaluation phases have been conducted within this scenario: they are distinguished only by the strategy chosen to recollect language IDs of ancestor fragments. Considered implementations have already been introduced in Section 5.2.3 – however, only results obtained via the "Annotation Repository" solution are presented here, as the "Language Stack" alternative has not shown acceptable performance yet. The diagram presented as Figure 15 shows the percentage of correctly interpreted language IDs (fragments) per page.

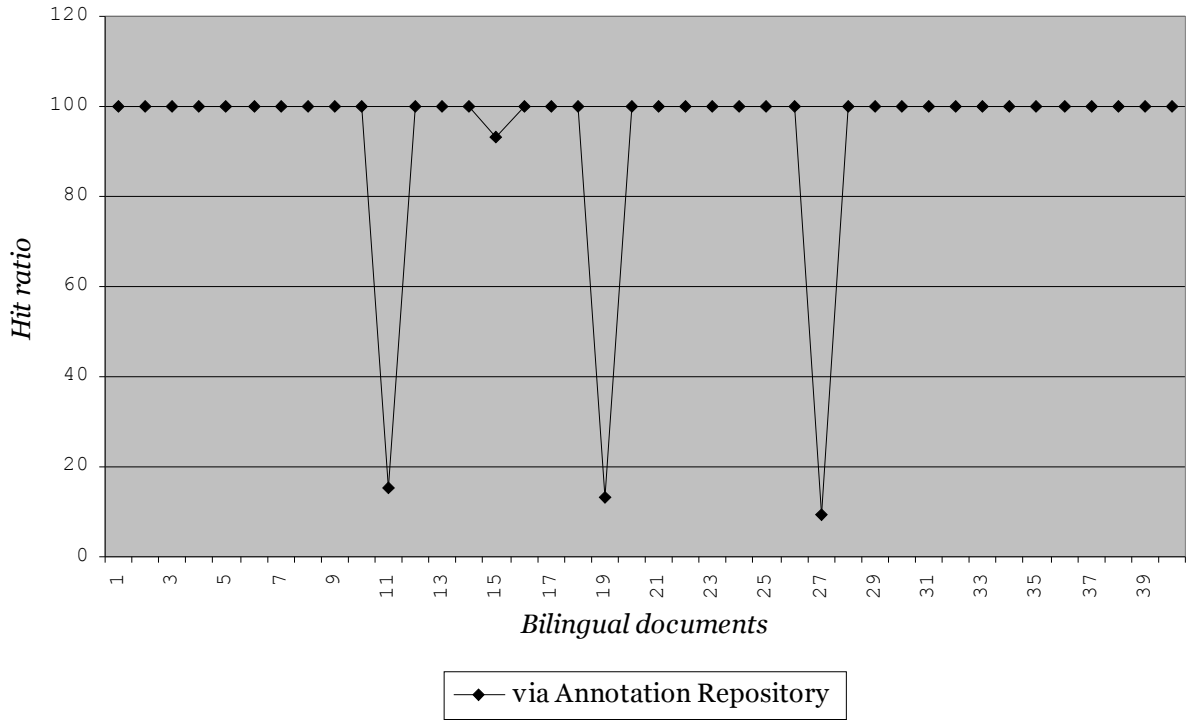

Figure 15: language recognition results ("via Annotation Repository").

For most of the documents in the collection, language recognition was nearly perfect. Of course, occasional lacks of accuracy did occur, but these results show that the strategies for language recognition proposed in this implementation are to be improved and/or integrated with additional techniques, rather than to be replaced with alternative ones.

CHAPTER 7

CONCLUSIONS

This thesis represents the first specific attempt at implementing multilinguality on a Semantic Web platform like DOSE. The proposed approach should have highlighted the favourable support offered by the conceptual organization of the system, by exploiting it with a simple, preliminary implementation.

From the practical point of view, DOSE has been extended in its pre-existing functionalities with the addition of language support both for annotation and search. The indexing process is now able to recognize the language of resources and create annotations accordingly; language IDs stored with annotations can be utilized to perform language-specific search queries. An additional module with language management capabilities has been provided in its core functionalities, in order to assess the global validity of the strategies adopted with respect to language recognition and switching.

Extensive further work is however needed, both to assure higher reliability for the system architecture, and possibly to deploy it for larger scale utilization. The primary need is for consistent testing on an heterogeneous base of resources – experimentation is still at the beginning, and more has to be done to definitely prove that the proposed strategy is effective from all points of view.

In addition to that, the base version of the language module has to be improved, by adding refinements both for language ID retrieval and detection – such an intent could be summarized in the following points:

- offer facilities for semantic search queries;

- support compliance to all Web standards for language ID specification;
- implement concurrent management of multiple languages at the same time, in order to improve overall performance;
- optimize heuristics by supporting alternative methodologies that can effectively apply to all languages.

Work is currently going on, as part of a joint effort that aims at improving the overall performance of DOSE, both by adding new modules that should provide new functionalities to the system, and by updating the existing ones to support forthcoming Semantic Web technologies.

BIBLIOGRAPHY

[1] T. Berners-Lee, J. Hendler, and O. Lassila: "The Semantic Web". Scientific American, 5/01, May 2001.

[2] T. Bray, J. Paoli et al.: "Extensible Markup Language (XML) 1.0", 2000. http://www.w3.org/TR/REC-xml

[3] D. C. Fallside (ed.): "XML Schema Part 0: Primer", 2000. http://www.w3.org/TR/xmlschema-0

[4] A. Swartz: "The Semantic Web in breadth". http://logicerror.com/semanticWeb-long

[5] T. Bray, D. Hollander, A. Layman (eds.): "Namespaces in XML", 1999. http://www.w3.org/TR/1999/REC-xml-names-19990114

[6] D. Connolly (ed.): "Naming and addressing: URIs, URLs, ...", 1998. http://www.w3.org/Addressing/

[7] O. Lassila, R.R. Swick (eds.): "Resource Description Framework (RDF), Model and Syntax Specification", 1999.

[8] Tom Gruber: "What Is An Ontology?". http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

[9] D. Brickley, R.V. Guha (eds.): "Resource Description Framework (RDF) Schema Specification", 2000. http://www.w3.org/TR/2000/CR-rdf-schema

[10] F. Van Harmelen, I. Horrocks (eds.): "Reference description of the DAML+OIL ontology markup language", 2000. http://www.daml.org/2000/12/reference.html

[11] D. L. McGuinness, Frank Van Harmelen: "OWL Web Ontology Language overview", 2003. http://www.w3.org/TR/owl-features/

[12] R. Cole et al.: "Survey of the State of the Art in Human Language Technology (Web Edition)". Cambridge University Press and Giardini, 1997.

[13] M. Agnesund: "Supporting multilinguality in ontologies for lexical semantics: an object-oriented approach (1997)". M.S. Thesis in Computational Linguistics, Göteborg University.

[14] Luisa Bentivogli, Emanuele Pianta, Fabio Pianesi: "Coping With Lexical Gaps When Building Aligned Multilingual WordNets (2000)".

[15] Douglas W. Oard: "Alternative approaches for Cross-Language Text Retrieval". In AAAI Symposium on Cross-Language Text and Speech Retrieval, American Association for Artificial Intelligence, March 1997.

[16] Carbonell, J., Yang, Y., Frederking, R., Brown, R. D., Geng, Y., and Lee, D.: "Translingual information retrieval: a comparative evaluation (1997)". In Proceedings of the Fifteenth international Joint Conference on Artificial Intelligence.

[17] H. Alvestrand: "Content-Language Headers" (RFC 3282), The Internet Society, 2002. http://rfc.sunsite.dk/rfc/rfc3282.html

[18] W3C recommendation: "HTML language information and text direction". http://www.w3.org/TR/REC-html40/struct/dirlang.html

[19] "WordNet: a Lexical Database for English Language". http://www.cogsci.princeton.edu/~wn/

[20] L. Denoue, L. Vignollet: "An annotation tool for web browsers and its applications to information retrieval". In Proceedings of RIAO 2000, Paris, France, April 12-14, 2000.

[21] J. Kahan, M. Koivunen, E. Prud'Hommeaux, R. Swick: "Annotea: An Open RDF Infrastructure for Shared Web Annotations". In Proceedings of the WWW10 International Conference, Hong Kong, May 1-5, 2001.

[22] M. Koivunen, R. Swick: "Metadata-based annotation infrastructure offers flexibility and extensibility for collaborative applications and beyond". In K-CAP 2001 workshop on Knowledge Markup and Semantic Annotation, Victoria, BC, Canada, 21 October, 2001.

[23] S. Handschuh, S. Staab, A. Maedche: "CREAM - Creating relational metadata with a component-based, ontology-driven annotation framework". In Proceedings of ACM K-CAP 2001 – First International Conference on Knowledge Capture, Victoria, BC, Canada, October 21-23, 2001.

[24] S. Staab, A. Maedche, S. Handschuh: "An annotation framework for the Semantic Web". In Proceedings of the First Workshop on Multimedia Annotation, Tokyo, Japan, January 30-31, 2001, 2001.

[25] L. Gangmin, V. Uren, E. Motta: "ClaiMaker: weaving a semantic web of research papers". In Proceedings of ISWC 2002 – First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002.

[26] Carr, L., Bechhofer, S., Goble, C., & Hall, W.: "Conceptual linking: ontology-based open hypermedia (2001)". In WWW10 – Proceedings of the 10th International World Wide Web Conference, Hong Kong, China, pages 334-342.

[27] S. Handschuh, S. Staab, F. Ciravegna: "S-CREAM - Semi-automatic CREAtion of Metadata". In Proceedings of EKAW02 – 13th International Conference on Knowledge Engineering and Knowledge Management, Siguenza, Spain, October 1-4, 2002.

[28] M. Vargas-Vera, E. Motta, J. Domingue, S. Buckingham Shum, M. Lanzoni: "Knowledge extraction by using an ontology-based annotation tool". In K-CAP 2001 workshop on Knowledge Markup and Semantic Annotation, Victoria, BC, Canada, 21 October, 2001.

[29] P. Kogut, W. Holmes: "AeroDAML: Applying information extraction to generate DAML annotations from web pages". In K-CAP 2001 workshop on Knowledge Markup and Semantic Annotation, Victoria, BC, Canada, 21 October, 2001.

[30] N. Collier, K. Takeuchi, K. Tsuji: "The PIA project: learning to semantically annotate texts from an ontology and XML-instance data". In Proceedings of SWWS 2001 - International Semantic Web Working Symposium, Stanford University, CA, USA, July 30 - August 1, 2001.

[31] J. Li, L. Zhang, Y. Yu: "Learning to generate semantic annotation for domain-specific sentences". In K-CAP 2001 workshop on Knowledge Markup and Semantic Annotation, Victoria, BC, Canada, 21 October, 2001.

[32] D. Bonino, F. Corno, L. Farinetti: "DOSE: a Distributed Open Semantic Elaboration Platform". In proceedings of ICTAI'03, Sacramento, California, November 2003.

[33] R. Baeza-Yates, B. Ribeiro-Neto: "Modern Information Retrieval". Addison-Wesley, 1999.

[34] P. Grosso, E. Maler, J. Marsh, N. Walsh: "XPointer Framework". World Wide Web Consortium, 2002.

[35] The Apache Software Foundation: "Apache XML-RPC". http://ws.apache.org/xmlrpc/

[36] G. Salton: "Developments in automatic text retrieval". Science, 253:974--980, 1991.

[37] W3C recommendation: "Document Object Model (DOM) Level 1 Specification". http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/

[38] XPath Explorer. http://www.purpletech.com/xpe/index.jsp

[39] D. Ragget et al.: "HTML Tidy project". http://tidy.sourceforge.net/

[40] McBride, B.: "Jena: a semantic Web toolkit", Internet Computing, IEEE , Volume: 6 Issue: 6, Nov/Dec 2002, pages 55–59.

[41] Porter MF: "An algorithm for suffix stripping (1980)". Program, 14: 130-137.

[42] Avviamento e Sviluppo di Progetti per ridurre l'Handicap mediante l'Informatica. http://www.asphi.it

[43] Bonino D., Corno F., Farinetti L., Ferrato A.: "Multilingual Semantic Elaboration in the DOSE platform". In SAC'2004 Symposium on Applied Computing, Nicosia, Cyprus, March 14-17, 2004.

[44] DOSE: a Distributed Open Semantic Elaboration platform. http://dose.sourceforge.net