UNIVERSITÀ DEGLI STUDI DI MODENA
E REGGIO EMILIA

Facoltà di Ingegneria
Corso di Laurea VOD in Ingegneria Informatica

PROGETTO E REALIZZAZIONE DELL'ALGORITMO DI ANNOTAZIONE AUTOMATICA TUCUXI

Relatore:
Chiar.mo Prof. Sonia Bergamaschi

Correlatore:
Dott.ssa Serena Sorrentino

Candidato:
Stefano Gatta

Anno Accademico 2008/2009

Parole chiave:
Word Sense Disambiguation
WordNet
Annotazione automatica
MOMIS
ALA

Indice

Indice delle Figure

Indice delle Tabelle

Introduzione

Il recente e rapido sviluppo di tutti i campi della comunicazione , e in particolare dell'*Information Technology*, ha avuto un duplice effetto sui suoi utilizzatori. Se da un lato, grazie anche all'esplorazione del *Web*, è possibile reperire una infinita *quantità* di informazioni di diversa natura, dall'altro si crea un forte deficit di *qualità* dell'informazione reperita, intesa come conformità e pertinenza alle specifiche esigenze dell'utente. Negli anni più recenti si è cercato di porre un freno a questo fenomeno di sovraccarico informativo, studiando e mettendo a punto dei sistemi, denominati per l'appunto Sistemi di Integrazione delle Informazioni, in grado di offrire una selezione ragionata dei dati provenienti da sorgenti informative diverse ed eterogenee tra loro (pagine Web, DBMS di varia natura, filmati, file xml etc) . L'agenzia nazionale **ARPA** (**A**dvanced **R**esearch **P**rojects **A**gency) (1) ha creato uno standard, denominato I^3 , che definisce un certo tipo di architettura interna cui i sistemi di integrazione dovrebbero conformarsi.

Un sistema di questo tipo è MOMIS, (**M**ediator **E**nvir**O**nment for **M**ultiple **I**nformation **S**ources), il quale consente l' integrazione di sorgenti di dati strutturate e semi-strutturate, sviluppato dal DBGROUP dell'Università di Modena e Reggio Emilia. Una fase importantissima per una corretta integrazione delle sorgenti è costituita dall'*integrazione semantica*, ossia il processo che individua le affinità esistenti tra le sorgenti diverse. Una fase saliente di tale processo è costituito dalla disambiguazione lessicale degli elementi che appartengono a tali sorgenti per estrarne i concetti principali al fine del calcolo delle affinità.

Per *disambiguazione lessicale* si intende, sinteticamente, attribuire, data una fonte di testo o di dati espressi in linguaggio naturale, il senso corretto a ciascun termine in relazione al contesto in cui è inserito. Tale processo in MOMIS è affidato ad un tool di nome **ALA** (**A**utomatic **L**exical **A**nnotator), che utilizza differenti algoritmi di disambiguazione per il fine prescritto.

Scopo principale di questa tesi è stato quello di realizzare un algoritmo di disambiguazione che si potesse inserire nel pool di algoritmi già presenti in ALA sfruttando però un approccio differente, che si basasse cioè sulle relazione semantiche presenti tra i lemmi da disambiguare.

Si è così deciso di realizzare una implementazione della parte relativa alla disambiguazione lessicale dell'algoritmo TUCUXI (In**T**elligent H**U**nter Agent for **C**oncept **U**nderstanding and Le**X**ical Cha**I**ning, pronuncia "tookooshee"), un *agente hunter* realizzato sempre dal DBGROUP dell'Università degli Studi di Modena e Reggio Emilia al fine di ottenere un *crawling* più efficace nell'esplorazione delle pagine Web.

Tale algoritmo sfrutta l'interazione con una risorsa lessicale in lingua inglese, WordNet, che contiene informazioni importanti sugli elementi lessicali e sulle relazioni che intercorrono tra essi.

Il lavoro svolto è così strutturato:

- **Capitolo 1- Il database lessicale WordNet:** questo capitolo ha lo scopo di illustrare in maniera approfondita l'organizzazione del database e le tipologie di relazioni in esso presenti, dal momento che sono state largamente utilizzate per il lavoro svolto.
- **Capitolo 2 - Il problema della disambiguazione lessicale (Word Sense Disambiguation) :** in questo capitolo si descrivono i problemi relativi alla disambiguazione lessicale, con particolare riferimento alle tecniche utilizzate da due algoritmi basati sulle

relazioni semantiche, ossia l'SSI (**S**tructural **S**emantic **I**nterconnection), sviluppato presso l'Università "La Sapienza" di Roma, e TUCUXI nella sua versione originale. Il capitolo si conclude con le possibili applicazioni di questi algoritmi, tra cui i sistemi di integrazione dei dati.

- **Capitolo 3 – L'annotazione lessicale come supporto ai Sistemi di Integrazione delle Informazioni:** dopo aver definito l'architettura di riferimento per i sistemi di integrazione delle informazioni proposta da ARPA, si passa alla descrizione dettagliata del processo di integrazione effettuato dal sistema MOMIS, e si specificheranno le modalità in cui la disambiguazione lessicale, detta *annotazione* se riguarda le sorgenti di dati, può offrire un ausilio per la sua buona riuscita.
- **Capitolo 4 – Il tool ALA di Momis: un approccio combinato alla disambiguazione:** dopo aver descritto un esempio di approccio combinato alla disambiguazione tramite il metodo CWSD (**C**ombined **W**ord **S**ense **D**isambiguation),in questo capitolo si fornirà una descrizione delle tecniche utilizzate dagli algoritmi del modulo ALA, già presente in MOMIS, per l'annotazione delle sorgenti di dati.
- **Capitolo 5 – L'algoritmo TUCUXI:** rappresenta il cuore della tesi, all'interno del quale si descriveranno dettagliatamente i passaggi logici necessari alla disambiguazione delle sorgenti di dati, seguiti da alcune note sulla implementazione effettiva e dai risultati dei test effettuati su 5 diverse sorgenti di dati.

Capitolo 1

Il database lessicale WordNet

In questo capitolo, dopo aver introdotto il concetto di *ontologia* (più volte utilizzato all'interno di questa tesi anche con accezioni differenti tra loro), verrà presentata un'ontologia lessicale in lingua inglese, chiamata WordNet, utilizzata per lo sviluppo dell'algoritmo di disambiguazione lessicale oggetto della presenta trattazione, e divenuta ormai uno strumento fondamentale nel campo della linguistica computazionale.

1.1 Il concetto di ontologia

Il termine ontologia è caratterizzato da molteplici significati, che abbracciano diverse aree della conoscenza umana. Nella sua accezione etimologica, ossia quella derivante dal greco *òntos* (genitivo singolare del participio presente ὤν di *èinai*, il verbo essere) più *lògos* (discorso) letteralmente "discorso sull'essere", assume una connotazione ampia e decisamente filosofica: si intende,cioè, un'indagine sull'Essere in senso lato, al di là degli enti e dei fenomeni attraverso cui si manifesta, di cui è invece considerato il fondamento ultimo.

In una accezione più ristretta, mutuata recentemente dal mondo della rappresentazione della conoscenza e dell'intelligenza artificiale, una ontologia rappresenta lo studio dell'essere come insieme degli enti, limitatamente a ciò che sembra esistere in concreto o risultare anche solo pensabile, dunque secondo quanto sembrerebbe attestato dai sensi o dalla psiche. Ogni domanda intorno al "soggetto", all' "oggetto" e alla loro "relazione" è una domanda ontologica.

Per estensione si è iniziato ad usare il termine, in ambito informatico, per definire generici modelli di dati. Nonostante la varietà di sensi in cui il termine viene utilizzato, nella letteratura specialistica sembra consolidata l'idea che in informatica il termine *ontologia* debba riferirsi specificamente, secondo una definizione data da Tom Gruber in (2), ad una "explicit specification of a conceptualization".

Si tratta generalmente di una struttura dati gerarchica che contiene tutte le entità rilevanti, le relazioni esistenti fra di esse, le regole, gli assiomi ed i vincoli specifici di un determinato dominio di appartenenza . In anni recenti l'utilizzo dell' ontologia come risorsa è sempre più importante a causa delle problematiche relative all'*information overload* , di cui si parlerà nel capitolo 3, e ad altre problematiche legate al rapido sviluppo dell'*information technology* e delle reti di telecomunicazioni.

Una risorsa ontologica ampiamente utilizzata all'interno di questa tesi e cui si farà continuamente riferimento è l'ontologia lessicale WordNet (3), ossia un'ontologia che descrive la concettualizzazione del linguaggio, che è caratterizzata principalmente da due tipi di conoscenza (4):

- una conoscenza lessicale derivata da un insieme di parole intese come singoli caratteri
- una conoscenza semantica, determinata dai significati espressi dai costrutti lessicali e dalle relazioni che intercorrono tra di essi.

Tale ontologia lessicale, attualmente disponibile on-line, è diventata una delle più importanti risorse nel campo della linguistica computazionale e se ne darà una descrizione dettagliata nel paragrafo successivo.

1.2 Il database lessicale WordNet

WordNet (3) è stato sviluppato al Cognitive Science Laboratory dell'Università di Princeton, sotto la direzione del Prof. George A.Miller. Si tratta di un sistema di gestione di un dizionario lessicale basato sulle attuali teorie psicolinguistiche formulate sulla memoria lessicale umana.

In realtà la sua struttura lo rende molto più completo di un semplice dizionario contenente informazioni sul significato degli elementi sintattici.

Infatti un contributo semantico importante offerto da WordNet è costituito dalla presenza di relazioni di diverso tipo tra le parole in esso contenute.

WordNet associa ad ogni termine due elementi distinti: la *word form*, dunque il lemma, ossia la forma scritta associata ad una stringa di caratteri , e il *word meaning* ovvero il significato, il concetto espresso dalla word form. Infatti una prima classificazione è basata sulle relazioni che legano un lemma ad un significato.

I significati in WordNet sono caratterizzati come insiemi di termini sinonimi che esprimono uno stesso concetto, detti *synset*. Il synset rappresenta l'unità di base dell'ontologia, in grado di descrivere un concetto univocamente attraverso una glossa, ossia un'insieme di parole, e al quale può essere associato uno o più lemmi che lo rappresentino sinteticamente.

Il discorso sulla corrispondenza tra lemma e synset, ossia tra parola e significato, diventa più chiaro se si osserva l'astrazione rappresentata dalla Matrice Lessicale (Figura 1.1): le righe di questa matrice rappresentano i significati (*word meanings*), mentre le colonne i lemmi (*word forms*).

Se è presente un elemento E_{ij} in un incrocio tra la riga i-esima e la colonna j-esima sta a significare che il concetto WM_i. Può essere espresso dal lemma WF_j . Nel caso sia presente più di un valore non nullo E_{ij} su di una stessa riga, il concetto WM_i può essere espresso da più di una word form.

La riga individuata rappresenta proprio un synset (*set of synonymous*), precedentemente descritto ed indicato in figura.

Nel caso in cui, invece, troviamo più elementi non nulli E_{ij} su una stessa colonna, siamo di fronte ad un lemma *polisemico*, che esprime,cioè, più di un significato con la stessa word form (un esempio classico può essere rappresentato dal lemma *mouse* nelle sue due accezioni di mammifero roditore e di dispositivo elettronico di puntamento)

Quando infine si ha esclusivamente un elemento E_{ij} non nullo tra una riga e una colonna, si ha a che fare con un termine *monosemico*, il cui significato è univoco e di conseguenza non ambiguo (i termini monosemici rivestono un ruolo importante in molti processi di disambiguazione).

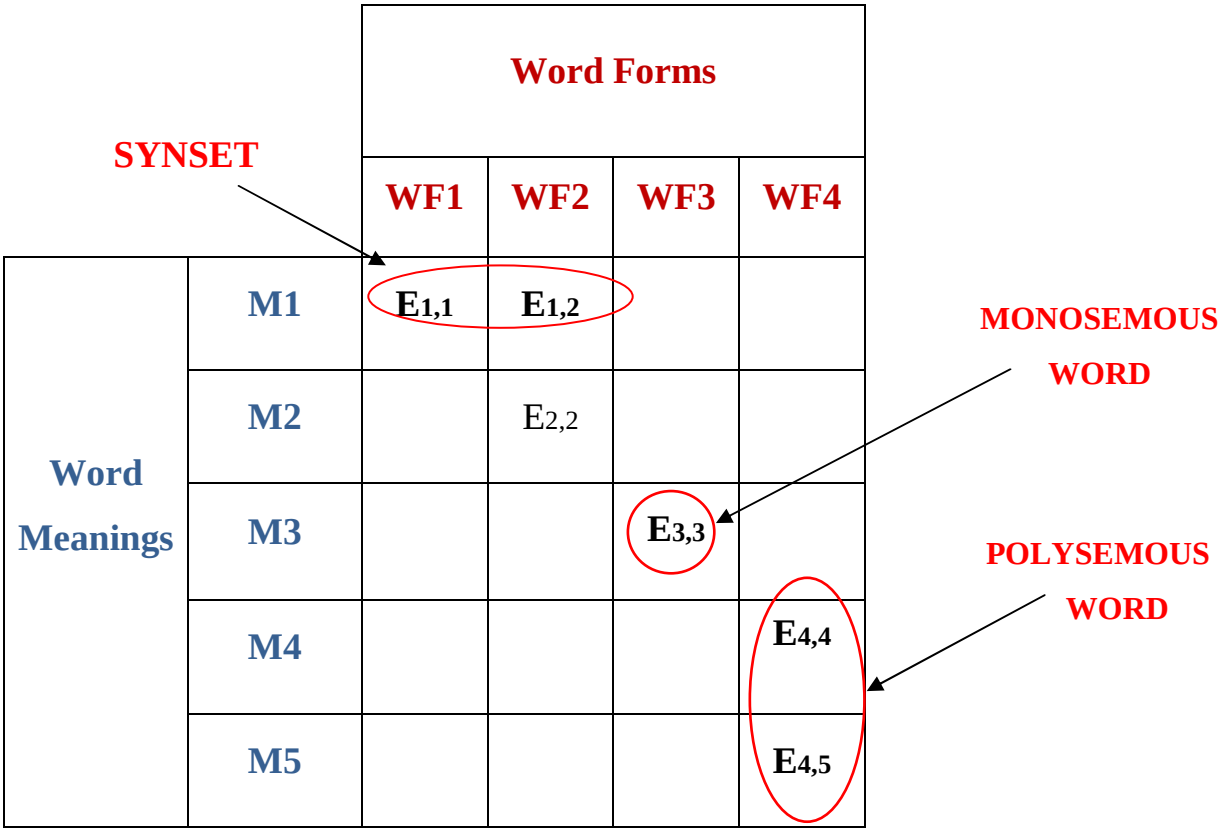

Tabella 1- La matrice lessicale di WordNet

Il database WordNet divide gli elementi lessicali in 4 categorie sintattiche principali : **sostantivi** (o più banalmente nomi), **verbi**, **aggettivi** ed **avverbi**, ognuna delle quali è costituita da elementi definiti tramite synset e posti in relazione tra loro proprio in base alla categoria sintattica di appartenenza.
Nel prossimo paragrafo si fornirà una descrizione del tipo di relazioni che interessano i synset o i lemmi di WordNet.

1.3 Le relazioni di WordNet

WordNet fa della grande informazione semantica offerta la propria ricchezza. Tale informazione si basa sulla possibilità di individuare e fornire all'utente le relazioni che si instaurano fra i concetti o fra i lemmi. Fondamentalmente WordNet contempla due tipologie di relazioni principali:

1. *relazioni lessicali fra i lemmi*
2. *relazioni semantiche fra i synset*

che adesso analizzeremo in dettaglio.

1.4.1 Le relazioni lessicali

Le relazioni lessicali sono quelle relazioni che abbracciano due word form, quindi due lemmi e non due synset, e vengono elencate di seguito:

Sinonimia

La relazione di Sinonimia gode di due definizioni. La prima attribuita a Leibniz, molto rigida e difficilmente verificabile, il quale nel suo *Scientia Generalis* afferma che due concetti sono fra loro sinonimi se la sostituzione di uno con l'altro non cambia il valore di verità della frase nella quale è effettuata la sostituzione.

L'altra, un po' più flessibile, contestualizza in qualche modo l'utilizzo dei sinonimi sostenendo che due concetti sono fra loro sinonimi in un contesto linguistico se la sostituzione di un concetto con l'altro nel contesto non ne altera il valore di verità.

La definizione che verrà utilizzata all'interno di WordNet, invece, è più debole delle altre, ed è più vicina alla definizione di similarità semantica piuttosto che di sinonimia: due lemmi sono sinonimi se presentano un synset in comune, o, da un altro punto di vista, se sono entrambi associati ad uno stesso synset. Così ad esempio i termini *albero*, *arbusto* e *pianta* secondo le definizioni date in precedenza non possono essere fra loro sempre sostituiti, in quanto esistono anche piante che non sono necessariamente alberi o arbusti, ma secondo la definizione di similarità essi appartengono allo stesso synset, in quanto ne consentono la distinzione da altri significati (5).

Antonimia

Una relazione alla quale non sempre è facile dare una definizione, ma che risulta molto familiare è l'antonimia. L'antonimia riguarda prevalentemente la categoria sintattica degli aggettivi.Un termine *y* è un antonimo di *x* se vale la proposizione *y = not x*. Ad esempio i termini *ricco* e *povero* sono fra loro antonimi anche se essere non ricchi non implica necessariamente essere poveri. L'antinomia, come detto, è una

relazione lessicale, quindi fra lemmi, e non tra synset: se infatti consideriamo i synset {*rise, ascend*} e {*fall, descend*}, essi, pur essendo concettualmente opposti, non rappresentano degli antinomi. Una relazione di antinomia, invece, è presente fra i termini *rise* e *fall*, e *descend* e *ascend* presi singolarmente in qualità di lemmi che esprimono concetti opposti (5).

Relazione di pertinenza

La relazione di *pertinenza* concerne gli aggettivi relazionali. Una caratteristica principale degli aggettivi relazionali è quella di derivare dai nomi (fraterno/fratello, musicale/musica). Nell'ambito di tali aggettivi la pertinenza svolge un ruolo che può essere riassunto nelle espressioni: *associato con*, oppure *pertinente a* o semplicemente *di* in relazione ad un sostantivo.

Relazione participiale

Questa relazione lega fra loro gli avverbi o gli aggettivi, detti participiali, rispettivamente ai nomi o ai verbi da cui derivano. Come esempio si può pensare all'aggettivo *bruciato,* derivante dal verbo *bruciare* (all'interno di WordNet esiste quindi una relazione participiale fra i lemmi *burned* e *burn*, appartenenti alle categorie sintattiche, rispettivamente di aggettivi e verbi).

Derivato da

Alcuni aggettivi derivano da antichi nomi Greci o Latini. Questa affermazione risulta essere vera, sia per la lingua italiana, che per quella inglese (idioma su cui è costruito WordNet). L'aggettivo relazionale *verbale*, deriva dal nome neutro latino *verbum* , mentre *lessicale* deriva dal corrispondente nome greco. La relazione *derivato da* lega gli aggettivi ai nomi stranieri da cui derivano.

1.4.2 Le relazioni semantiche

Le relazioni di tipo semantico, coinvolgono sempre due concetti, due significati (due synset), non semplicemente due lemmi.

Iponimia/Iperonimia

Le relazioni di iponimia e iperonimia sono relazioni fra significati e seguono la seguente definizione : un concetto rappresentato dal synset *X*= {x1, x2, x3,…} si definisce iponimo del concetto rappresentato dal synset *Y* = {y1, y2,…} se la seguente asserzione è ritenuta vera : *X è un (un tipo di) Y* (tale relazione è anche nota come relazione ISA, ossia *is a kind of*).

È evidente come questa informazione semantica implichi concettualmente una gerarchica di specializzazione secondo cui il concetto figlio (synset *X*) presenta degli aspetti che rendono più specifico il concetto padre (synset *Y*). La relazione simmetrica rispetto all'iponimia è l' iperonimia. Infatti si può affermare che un synset *X* = {x1, x2, x3,…} è iperonimo di un synset *Y* = {y1, y2,…}, se *X* ha un significato più esteso e comprensivo, ossia se rappresenta una *generalizzazione* di Y (si usa dire *X has kind Y*). Le relazioni di iponimia ed iperonimia sono le relazioni più numerose presenti all'interno del database lessicale di WordNet. Un semplice esempio, per comprendere meglio queste importanti relazioni, potrebbe essere: *salice* è in iponimo di *albero*, mentre *fiore* è iperonimo di giglio. La relazione di iponimia (assieme a quella di iperonimia), può essere utilizzata per formare una gerarchia di specializzazione fra i synset di WordNet.

Meronimia/Olonimia

La relazione di meronimia esprime il concetto di *è parte di.* Il synset *X* = {x1, x2,…} è un meronimo di un concetto rappresentato da *Y*={y1, y2,…} se si possono accettare proposizioni scritte come *X è parte di Y* (*is part of*). Una relazione di meronimia può essere rappresentata dai concetti di *maniglia* e *porta*. La relazione di meronimia generalmente è transitiva, ossia nella maggior parte dei casi se A è meronimo di B e B è meronimo di C, allora A è meronimo di C (4) . Questa proprietà però non è sempre valida : se è vero che *maniglia* è meronimo di *porta*, e *porta* è meronimo di *casa*, risulterebbe di dubbia precisione affermare che *maniglia* è meronimo di *casa*, ossia che *maniglia è parte di casa*.

Si intuisce quindi che esistono varie tipologie più specifiche di relazioni *parte di* (*is a part of*), per l'esattezza ne sono state individuate 6 :

- *componente* $\xrightarrow{is\ a\ part\ of}$ *oggetto*
- *elemento* $\xrightarrow{is\ a\ part\ of}$ *insieme*
- *porzione* $\xrightarrow{is\ a\ part\ of}$ *intero*
- *materiale* $\xrightarrow{is\ a\ part\ of}$ *oggetto*
- *azione* $\xrightarrow{is\ a\ part\ of}$ *attività*
- *località* $\xrightarrow{is\ a\ part\ of}$ *area*

e un ultimo aggiunto più tardi nel tempo:

- *fase* $\xrightarrow{is\ a\ part\ of}$ *processo*

Implicazione

La relazione di *implicazione* rappresenta l'equivalente della meronimia , riferita però a due verbi, l'implicazione non è solamente una relazione

semantica, ma è possibile avere anche implicazioni lessicali fra verbi (fra singoli termini). Questa relazione è verificata se è vera la seguente proposizione: un verbo *X* implica un verbo *Y* se *X* non può verificarsi a meno che non si sia verificato (o non si stia verificando) Y. Per comprendere meglio questo concetto si pensi ai verbi *mangiare* e *masticare*: *mangiare* implica necessariamente l'atto del *masticare.*

L'implicazione è una relazione univoca: se un verbo *X* implica un verbo *Y* non è verificato il contrario (*masticare* non implica necessariamente *mangiare*).

Relazione causale

La relazione causale è simile alla relazione di implicazione a prescindere dalla notazione temporale. Un esempio potrebbe essere *forzare* che implica *agire*.

Raggruppamento di verbi

Questa relazione viene utilizzata per produrre raggruppamenti nella categoria sintattica dei verbi. In un gruppo formato in tale maniera, i synset hanno tutti un significato semantico molto simile. Un esempio di raggruppamento di verbi è dato da: *mistake, confuse, counfound, confuse, mix_up, confuse, blur, oscure* (5).

Similarità

La relazione di *similarità* è utilizzata solamente nell'ambito della categoria sintattica degli aggettivi. Molti synset di questa categoria sono raggruppati in coppie legate da una relazione di antinomia (si pensi, ad esempio, a

synset trattanti i concetti di *alto* e *basso*, in netta contrapposizione semantica fra di loro).

Attributo

La relazione di attributo è definita per gli aggettivi e nello specifico rappresenta il legame che intercorre fra un aggettivo ed un nome di cui esprime il valore. Gli aggettivi in grado di descrivere il valore di un attributo sono gli aggettivi descrittivi. Per fare un esempio basta pensare ad una frase come: *questa persona è alta.* L'aggettivo descrittivo alta indica il valore dell'attributo *altezza* riferito a persona. Aggettivi quali *alta* o e *basso* sono, quindi, legati al nome altezza.

Coordinazione

La *coordinazione* non è un tipo di relazione base, ma si potrebbe definire derivata. Due synset sono detti coordinati se sono iponimi dello stesso iperonimo, ossia, se esprimono due concetti figli di uno stesso padre

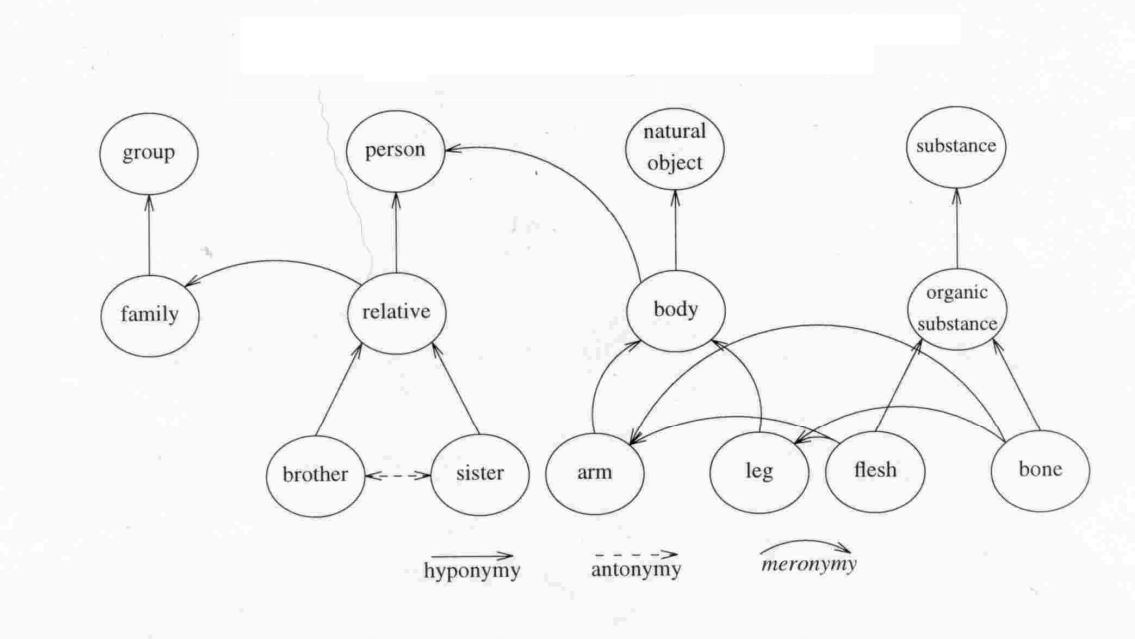

Figura 1.1 - Rappresentazione di relazioni semantiche

1.8 Le carenze di WordNet come risorsa lessicale

Nonostante WordNet abbia ottenuto un consenso molto ampio nelle applicazioni che utilizzano risorse lessicali, al punto da poter quasi essere ritenuto uno standard per molte aree della linguistica computazionale, il servizio che offre non è privo di lacune.

Anche durante la fase di sviluppo dell'algoritmo proposto in questa tesi, le carenze semantiche di WordNet sono venute a galla, confermando questo fenomeno, che si verifica soprattutto quando si utilizza WordNet come unica ontologia lessicale di riferimento.

Le cause principalmente sono due:

1. L'elevato grado di polisemia di alcuni termini porta ad un aumento a volte eccessivo di granularità nella distinzione fra i sensi dei termini. Non di rado capita di trovare in WordNet più di 40 synset associati ad uno stesso lemma, con differenze reciproche quasi impercettibili, e comunque non facilmente distinguibili quando gli elementi del contesto non forniscono informazioni utili. Nella maggior parte delle applicazioni reali, è sufficiente, se non addirittura necessario, un livello di distinzione dei sensi a granularità molto inferiore.
2. Le relazioni che WordNet afferisce sono limitate in quanto appartenenti a lemmi appartenenti alla stessa gerarchia sintattica. Ovvero WordNet non fornisce alcuna informazione,ad esempio, riguardo ai legami tra nomi e verbi

Tali lacune hanno portato al crescere di un sempre maggiore interesse nei confronti di tecniche e metodologie che consentano di estendere la risorsa

WordNet, arricchendola di nuovi contenuti informativi, di varia natura, che consentano di superare anche solo parzialmente, le mancanze di WordNet.
Sono state sviluppate, con questo proposito, delle "estensioni" di WordNet che potessero arricchire il patrimonio informativo o affiancarsi ad esso. Tra quelle effettivamente implementate la nostra analisi si è concentrata su WND (***W**ordNet **D**omains*) (6) (7), che utilizza i domini di appartenenza dei termini per offrire un contributo semantico ulteriore alle applicazioni che ne faranno uso.

1.8.1 Il concetto di Dominio

Il concetto di dominio è connesso a termini come *argomento principale, soggetto, soggetto di dominio, categoria ecc...*, termini a volte usati in maniera intercambiabile, a volte differenziati per significato.
In questa tesi, per dominio (o *domain*), intendiamo individuare un'area della conoscenza che sia, in qualche modo, riconosciuta come unitaria.
Un dominio può essere caratterizzato dal nome di una disciplina, nell'ambito della quale una certa area di conoscenza si è sviluppata (es: Chimica), oppure dal nome di uno specifico oggetto che è caratteristico di quest'area di conoscenza (es: cibo) (5).
Si può constatare però una sorta di ambiguità ,relativa ai domini, dovuta al fatto che la conoscenza si manifesta sia attraverso le singole parole, sia attraverso i testi. Così la nozione di dominio, può essere applicata sia nell'analisi semantica, dove rappresenta l'area di conoscenza alla quale un certo elemento lessicale appartiene, sia nello studio di interi testi, dove può indicare l'argomento principale che tiene "unita" la narrazione.

1.9 WordNet Domains: un'estensione di WordNet basata sui domini

Per loro natura i domini possono essere organizzati in gerarchie di specializzazione. Ad esempio possiamo certamente convenire sul fatto che

felino rappresenta una specializzazione di mammifero, o che la *chimica organica* è una branca più specializzata della *chimica* . Le gerarchie di dominio possono essere utilmente integrate con altre risorse linguistiche e proficuamente usate nell'ambito dell'elaborazione del linguaggio naturale (NLP) come Word Sense Disambiguation, Text Categorization,e Information Retrieval.

Il tentativo di inserire tali gerarchie di dominio all'interno della struttura di WordNet è stato effettuato con successo dall' ITC-irst (Istituto Trentino di Cultura- centro per la ricerca scientifica e tecnologica), che ha prodotto un'estensione di WordNet chiamata per l'appunto WordNet Domains (8) (9). All'interno di questa risorsa lessicale ogni synset di WordNet è stato etichettato con una *label* di dominio, selezionata da una gerarchia di oltre 200 domini. Grazie a questo contenuto semantico aggiuntivo, WND è stato utilizzato in svariate applicazioni e progetti, rivestendo due funzioni principali: una di classificazione dei sensi dei termini, e l'altro diclassificazione di interi testi. WND fornisce un valido supporto per la creazione di *corpora* letterari, in cui i domini vengono utilizzati come criterio discriminante per la scelta dei testi da inserire.

In fine, come verrà ampiamente descritto nel capitolo seguente, i domini rivestono un ruolo importantissimo nei processi di disambiguazione lessicale, ossia quei processi mirati ad attribuire ad un termine il significato più appropriato e coerente al contesto in cui appare. Tale importanza è dovuta al fatto che i domini contribuiscono in maniera rilevante ad individuare e definire una certa coerenza lessicale di un testo. Magnini e Strapparava sostengono, infatti, in (8) che i sensi delle parole che compaiono all'interno di una porzione di testo tendono a massimizzare l'appartenenza ad uno stesso dominio.

In secondo luogo l'informazione apportata dall'appartenenza di un synset ad un dominio è utile per ovviare alla mancanza di relazioni in WordNet tra elementi appartenenti a categorie sintattiche differenti. Nulla vieta infatti, che un sostantivo e un verbo possano appartenere ad uno stesso dominio, potenziando ulteriormente la rete di interconnessioni semantiche, e dunque la possibilità di conoscenza, tra i synset di WordNet. In figura 1.2 è presente un esempio, estratto *dall'English lexical Sample task* di annotazione dei 10 synset relativi alla parola *bank*, con le rispettive etichette di dominio.

Sense	Synset and Gloss	Domains	Semcor
#1	depository financial institution, bank, banking concern, banking company (a financial institution...)	ECONOMY	20
#2	bank (sloping land...)	GEOGRAPHY, GEOLOGY	14
#3	bank (a supply or stock held in reserve...)	ECONOMY	-
#4	bank, bank building (a building...)	ARCHITECTURE, ECONOMY	-
#5	bank (an arrangement of similar objects...)	FACTOTUM	1
#6	savings bank, coin bank, money box, bank (a container...)	ECONOMY	-
#7	bank (a long ridge or pile...)	GEOGRAPHY, GEOLOGY	2
#8	bank (the funds held by a gambling house...)	ECONOMY, PLAY	
#9	bank, cant, camber (a slope in the turn of a road...)	ARCHITECTURE	-
#10	bank (a flight maneuver...)	TRANSPORT	-

Figura 1.1 domini associati ai synset relativi a bank

Come si nota tre di questi synset possono essere raggruppati sotto il dominio di ECONOMY, mentre altri due appartengono ai domini GEOGRAPHY e GEOLOGY.

Capitolo 2

Il problema della disambiguazione lessicale (Word Sense Disambiguation)

Il processo di *disambiguazione del testo,* come è stato accennato nel capitolo precedente, consiste, nella sua accezione più generale, essenzialmente nell'identificazione dei concetti associati ad un lemma, ovvero nell'assegnare, ad ogni termine contenuto in un testo analizzato, il senso più coerente col contesto in cui è inserito.

Come descritto nel capitolo precedente, la *polisemia* che caratterizza la maggior parte dei termini rende difficoltoso questo procedimento in apparenza semplice. In particolare bisogna considerare che, oltre alle evidenti differenze presenti tra due concetti espressi da uno stesso lemma (si pensi alla parola *lion* intesa come segno zodiacale o come mammifero carnivoro), esistono una varietà di differenze a volte impercettibili tra concetti appartenenti anche allo stesso ambito (si pensi ai 10 sensi differenti associati alla parola *bank* riportati nel capitolo precedente).

Come confermato in (10) il problema della WSD (**W**ord **S**ense **D**isambiguation), ossia della disambiguazione lessicale, rappresenta pertanto il processo probabilmente più critico nell'ambito della linguistica computazionale. Nei primi tempi gli approcci alla soluzione del problema erano basati sullo sfruttamento di una conoscenza semantica che poteva essere o codificata manualmente (11) (12) , oppure estratta automaticamente da un risorsa lessicale esistente, come WordNet per l'appunto, LDOCE (13),e il Roget thesaurus (14).

Successivamente una varietà di metodologie sono state proposte, e numerosi algoritmi, i quali sfruttano diversi strumenti e differenti proprietà lessicali, sono stati implementato nel corso del tempo. Nel prossimo paragrafo si propone, a scopo esplicativo, un'insieme di criteri di classificazione degli algoritmi di disambiguazione attualmente presenti in letteratura.

2.1 Criteri di classificazione degli algoritmi di disambiguazione lessicale.

Il criterio di classificazione dei processi di disambiguazione del testo, non è univoco. I vari approcci, infatti, possono differire tra loro per vari aspetti (5):

- le categorie sintattiche che consentono di disambiguare: per esempio esistono algoritmi che consentono di disambiguare solo nomi, e viceversa altri che possono essere applicati a tutte le categorie sintattiche;
- la struttura dati all'interno della quale memorizzano le informazioni necessarie al processo di disambiguazione: tale struttura può essere, un vettore, un grafo ecc…
- l'estensione dell' area in cui agisce la disambiguazione: esistono processi di *disambiguazione globale* o processi di *disambiguazione locale*. Nei primi la metodologia utilizzata consente di disambiguare contemporaneamente tutte le parole di un testo o di una porzione di testo. I secondi invece disambiguano un termine indipendentemente dai sensi attribuiti alle altre parole del contesto;
- la tipologia ed al numero di relazioni fra i sensi considerate;

- l'esigenza di essere supervisionati o meno dall'utente; si distinguono processi di disambiguazione:
  1. Supervisionati: si tratta di metodi che necessitano l'intervento umano, supportato dall'uso di collezioni di testi etichettati manualmente*;* il senso presunto corretto di un termine viene attribuito manualmente. Questa attività di generazione di corpus etichettati richiede, come si intuisce facilmente, un impegno notevole in termini di tempo.
  2. Non Supervisionati: questa tipologia di algoritmi non richiede alcuna supervisione da parte dell'utente. In genere si basa sull'utilizzo di ontologie lessicali, come WordNet o dizionari come risorsa lessicale di riferimento.
- l'elemento sul quale viene calcolata la similarità; si distinguono processi di disambiguazione:
  1. *Token-based*: hanno lo scopo di calcolare la similarità o la relazione tra il termine target che deve essere disambiguato e il suo contesto;
  2. . *Type_based:* semplicemente assegnano a tutte le istanze di un termine ambiguo, il suo senso più frequente; il senso predominante, è acquisito automaticamente da testi raw senza alcun ricorso all'annotazione manuale dei dati.
- la risorsa di conoscenza utilizzata; in questo caso si distinguono i seguenti algoritmi:
  1. Basati su collezioni di testi annotati manualmente (*hand-tagged corpora*): utilizzano come risorsa frasi all'interno di testi dove sono

2. già stati disambiguati manualmente i sensi dei termini.
3. Basati su **M**achine **R**eadable **D**ictionary (MRD): utilizzano le informazioni contenute all'interno dei dizionari.
4. Basati su Ontologie: utilizzano la conoscenza contenuta all'interno di un'ontologia.
5. Altri approcci: solitamente si ottengono a partire da due o più combinazioni degli approcci precedentemente descritti, o su altri generi di risorse meno sfruttate

.

I metodi sui quali si focalizzerà questa tesi sono quelli classificati come non supervisionati, che hanno bisogno, cioè, di una *risorsa di conoscenza lessicale* (detta propriamente **LKB, L**exical **K**nowledge **B**ase) di qualunque tipo per poter estrapolare il contenuto semantico di un testo o di una porzione di esso e dunque portare a compimento il processo di disambiguazione. Tali algoritmi vengono definiti *knowledge- based* per i motivi appena descritti.

2.2 Approcci *knowledge-based* al problema della disambiguazione

In maniera analoga alle altre applicazioni di intelligenza artificiale, la WSD *knowledge-based* ha dovuto fare i conti con il collo di bottiglia rappresentato dall'acquisizione della conoscenza. L'acquisizione manuale è un processo complicato e senza fine, e d'altro canto le informazioni semantiche reperibili nei comuni dizionari on line sono per la maggior parte dei casi non strutturate, rendendo difficile per un programma sfruttare le conoscenze lessicali codificate.

Più recentemente, tramite lo sviluppo del *machine learning*, i metodi statistici ed algebrici (15) (16) hanno avuto la meglio su i metodi knowledge-based, tracciando le linee di una nuova tendenza che emerge chiaramente nelle più importanti conferenze di Information Retrieval e nelle valutazioni comparative dei sistemi, come SIGIR, TREC e SenseVal. Tali metodi sono spesso basati sulla co-occorrenza dei termini estratti da archivi di documenti o dalle sorgenti informative del Web.

Negli anni più recenti, i risultati di più ricerche spingono verso lo sviluppo e la realizzazione di *repositories* lessicali on-line, ontologie e glossari diventano disponibili creando nuove opportunità per lo sviluppo di metodi di disambiguazione knowledge based. Il problema è che queste risorse sono spesso eterogenee, non completamente strutturate, e a volte inconsistenti. A dispetto di questi problemi, c'è però una buona probabilità che il futuro dei metodi di annotazione semantica troverà applicazione su larga scala, dipendendo in maniera critica dalla buona riuscita degli sforzi atti a integrare le esistenti risorse lessicali e a progettare algoritmi di disambiguazione che sfruttino al massimo queste conoscenze.

Si presentano nei paragrafi seguenti, due algoritmi , entrambi basati su una KLB,che utilizzano il metodo appena descritto per risolvere il complesso problema della disambiguazione , ossia sfruttano proprio le relazioni semantiche codificate all'interno di un Thesaurus comune.

Tali algoritmi sono l'SSI (**S**tructural **S**emantic **I**nterconnection) (10), ad opera di Roberto Navigli e Paola Velardi del Dipartimento d'Informatica dell'Università "La Sapienza" di Roma, e l'agente TUCUXI, realizzato da Sonia Bergamaschi , Roberta Benassi e Maurizio Vincini del Dipartimento di Ingeneria dell'Informazione dell'Università degli studi di Modena e Reggio Emilia, e verranno descritti nei paragrafi seguenti.

2.3 Algoritmo SSI

L'algoritmo Structural Semantic Interconnection sfrutta, come detto un approccio Knowledge Based, e utilizza dei grafi per descrivere gli oggetti da analizzare (i sensi da associare ai termini) e una grammatica indipendente dal contesto per individuare i percorsi semantici rilevanti tra i grafi. La classificazione dei sensi è basata sul numero e sul tipo di connessioni individuate.

La rappresentazione grafica dei sensi dei termini è costruita automaticamente a partire da diverse risorse disponibili, come ontologie lessicali, *corpora* già annotati, o dizionari, che vengono combinati in parte manualmente, in parte automaticamente.

2.3.1 Creazione delle rappresentazioni a grafo dei word senses

L'approccio al meccanismo di disambiguazione fa riferimento al modello di *structural pattern recognition* (17), il cui problema maggiore di applicazione è costituito proprio dall'apprendimento di una struttura adeguata per descrivere gli oggetti che devono essere classificati.

Nel campo della linguistica computazionale comunque, le estese basi di conoscenza lessicale e risorse già annotate offrono un punto di partenza ideale per la costruzione di rappresentazioni strutturate dei sensi associati ai termini. In queste *repositories* la conoscenza lessicale è descritta con un livello variabile di formalismi e sono state mosse molte critiche circa la consistenza e la correttezza delle informazioni codificate (con riferimento particolare agli standard dell'ingegneria informatica). Malgrado tali critiche e gli sforzi di superare alcune limitazioni, questi *depositi* di conoscenza sono diventati popolari al punto che sono state indette ogni anno conferenze dedicate all'argomento da scienziati che utilizzano queste risorse per una grande varietà di applicazioni nell'ambito dell'information technology.

La LKB utilizzata dall'algoritmo in questione per la rappresentazione a grafi dei word senses è frutto dell'integrazione di diverse risorse on line, in particolare:

- *WordNet 2.0*, ampiamente descritto nel primo capitolo
- *Etichette di Dominio descritte in* (18). Questa risorsa assegna una *label* che indica il Dominio di appartenenza (*es. turismo, sport, zoologia etc.)* della maggior parte dei synset di WordNet
- *Corpora annotati*.I testi offrono esempi sull'utilizzo delle parole all'interno di un contesto. A partire da questi testi, si estraggono automaticamente le informazioni sulla co-occorenza, ossia sulle liste di termini che compaiono insieme nello stesso contesto (di solito una frase). Una co-occorrenza semantica è invece una lista di sensi o concetti co-occorrenti.
- *Dizionari di collocation,* come ad esempio l'*Oxford Collocation* (19) o il *Longman Language Activator*. (20) Le *collocation* sono costituite da un'insieme di termini appartenenti ad un dato dominio semantico, come ad esempio (*bus,stop,station*) o (*bus,network, communication*), per i quali si registra una grande probabilità di co-occorrenza all'interno dello stesso testo.

A partire dalla LKB, possono essere facilmente generate rappresentazioni dei sensi dei termini attraverso grafi con archi etichettati. Tali grafi vengono detti *semantici* poichè rappresentano una concettualizzazione alternativa di un dato oggetto lessicale.

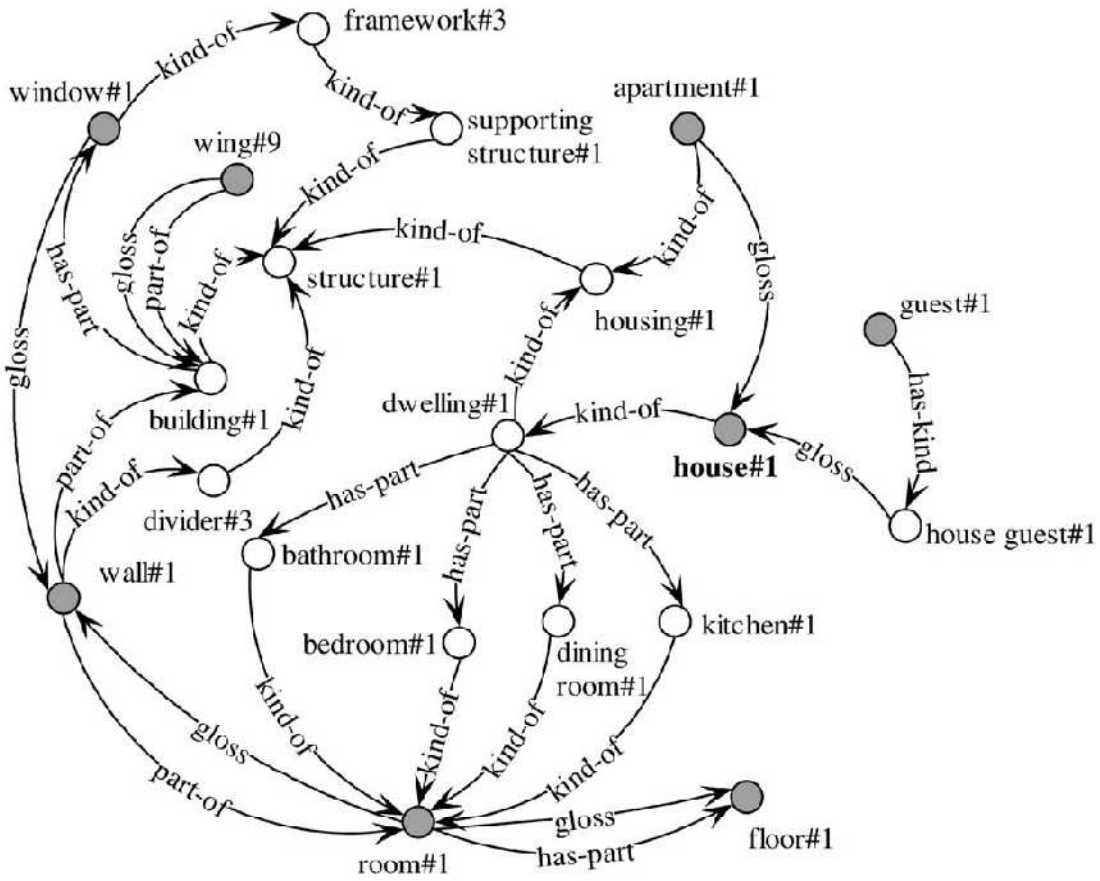

Figura 2.2 Rappresentazione a grafo del senso#1 (a) e del senso#2 (b) di *bus*

La figura 2.1 mostra i percorsi semantici che connettono il concetto *house*#1 (dove con la notazione #1 intendiamo il significato numero 1 associato al lemma *house*) con i suoi sensi correlati. I nodi in figura rappresentano concetti (ovvero synset di wordnet) e gli archi le relazioni semantiche che legano i concetti. In ogni grafo si includono al massimo synset distanti 3 archi dal nodo centrale, come suggerito dai risultati empirici al fine di ottimizzare le prestazioni di disambiguazione. Quelle che seguono sono le tipologie di relazioni prese in considerazione:

- *hyperonymy (car#1 is a kind of vehicle#1*,denotata dal simbolo $\xrightarrow{kind\text{-}of}$)
- *hyponymy* (la sua relazione inversa, $\xrightarrow{has\text{-}kind}$)
- *meronymy* (*room#1* è costituita da *wall#1*, $\xrightarrow{has\text{-}part}$)
- *holonymy* (la sua relazione inversa, $\xrightarrow{part\text{-}of}$)

- *pertainymy* (*dental#1* è relativo a *tooth#1* $\xrightarrow{pert}$)
- *attribute* (*dry#1 value-of wetness#1,* $\xrightarrow{attr}$)
- *similarity* (*beautiful#1* è simile a *pretty#1,* $\xrightarrow{sim}$)
- *gloss* ($\xrightarrow{gloss}$)
- *context* ($\xrightarrow{context}$)
- *domain* ($\xrightarrow{dl}$).

Tutte queste relazioni sono esplicitate all'interno di WordNet, fatta eccezione per le ultime tre.

Context esprime le associazioni semantiche tra concetti (*food#2 has context drink#3*) , estratte dai corpora annotati e dai dizionari di collocazioni, come spiegato precedentemente.

Gloss collega un concetto con un altro concetto presente nella sua stessa glossa cosi come definita in word net (*coach#5 ha nella sua glossa passenger#1*)

Domain infine collega un concetto alla sua *label* di Dominio (*terminal#3* ha per dominio *computer science).*

2.3.2 Il processo di disambiguazione dell'algoritmo SSI

In questo paragrafo si presenta il metodo utilizzato dall'algoritmo per disambiguare i termini in un contesto. Per mantenere una certa generalità nella trattazione, non si fa qui riferimento ai synset di WordNet ma a concetti, etichette di concetti e ontologie.

Il problema della classificazione può essere risolto come segue :

- *T* (il *contesto lessicale*) : una lista di termini correlati tra loro
- *t* : uno dei termini appartenenti a *T* da disambiguare.

- $S_1^t, S_2^t, \ldots\ldots S_n^t$: rappresentano le specifiche strutturali dei possibili concetti associati a *t* (più precisamente, i *grafi semantici*).
- *I,* il contesto semantico: è una lista di tutte le specifiche strutturali dei concetti associati ai (alcuni dei) termini di *T*. *I* contiene una oppure nessuna delle specifiche strutturali di ogni termine appartenente a *T*\{*t*}, e nessuna specifica strutturale di *t.*
- *G* : la grammatica che descrive le relazioni rilevanti tra le specifiche strutturali (più precisamente, le *connessioni semantiche* sui grafi).
- Determina, usando le regole della grammatica G, che intensità di relazione c'è tra le specifiche strutturali di *I* e ciascuna delle specifiche $S_1^t, S_2^t, \ldots\ldots S_n^t$, per ogni *t.*
- Seleziona la S_i^t con il matching più alto.

Le specifiche strutturali sono costruite a partire da dalle concettualizzazioni disponibili per l'oggetto lessicale *t* di *T*, che saranno denotate con il termine Ontologia, senza far riferimento alla specifica risorsa lessicale dalla quale provengono.

Il procedimento adottato dall'algoritmo SSI consiste di uno step di inizializzazione e uno step iterativo.

Durante una generica iterazione , l'input è costituito da una lista *T* = [$t_1, t_2, \ldots\ldots, t_n$] e una lista di sensi associati *I* = [$S^{t_1}, S^{t_2}, \ldots\ldots, S^{t_n}$] , ossia il vettore interpretazione semantica, in cui il generico termine S^{t_i} rappresenta o il senso selezionato (il risultato di una precedente iterazione) per il termine t_i o l'elemento nullo (se il termine non è stato ancora disambiguato). È inoltre mantenuto un set di termini *pendenti,* il cui senso non è ancora stato assegnato dall'algoritmo ,

ossia $P = \{ t_i | S^{t_i} = \text{null}\}$. *I* prenderà il nome di contesto semantico di *T*, ed è usato ,in ogni step, per disambiguare nuovi termini di *T*. Come detto l'algoritmo lavora in una logica iterativa, per cui, ad ogni iterazione, o un termine viene disambiguato, e dunque rimosso da *P*, oppure la procedura si arresta perché non è più possibile disambiguare alcun termine. L'output è rappresentato dalla lista *I* dei sensi associati ai termini in input *T*.

Inizialmente la lista I contiene i sensi dei termini monosemici di *T*, o un senso fissato. Ad esempio, un caso in cui un senso iniziale è disponibile si verifica quando si applica l'algoritmo al fine di disambiguare i termini delle glosse di un word sense: in questo caso *I* contiene *S* (ad esempio *bus#1*) e *P* i termini della glossa (*vehicle, carry, passenger, public transport*). Nel caso in cui non siano reperibili sensi iniziali , l'algoritmo utilizza il senso più probabile dei termini meno ambigui. A questo punto, il processo è indirizzato verso un numero di esecuzioni pari al numero dei sensi associati a *t*. Durante una generica iterazione, l'algoritmo seleziona quei termini *t* in *P* che mostrano una relazione tra almeno un senso *S* di *t* e uno o più sensi in *I*. La probabilità per un senso S di essere selezionato come corretta interpretazione della parola *t* , dato un contesto semantico *I*, è data da un funzione $f_I : C \text{ x } T \longrightarrow R$, dove *C* rappresenta il set di tutti i concetti dell'ontologia *O*

$$f_I(S,t) = \begin{cases} \rho\left(\left\{\varphi\left(S,S'\right) | S' \in I\right\}\right) & \textit{se } S \in \textit{Senses(t)} \\ 0 & \textit{altrimenti} \end{cases}$$

Dove *Senses(t)* è il sottoinsieme di *C* in *O* associati al termine *t*, mentre $\varphi(S,S')$ è la funzione che determina il peso *w* di ogni percorso congiungente il senso *S* con il senso S' . Un percorso semantico tra S e S' è

rappresentato dalla sequenza di *label* etichettate $e_1, e_2, \ldots\ldots, e_n$. Una scelta possibile per entrambe le funzioni ρ e ρ' potrebbe essere la funzione *sum* o *average sum.*

Una grammatica G = (*E*,*N*,*Sg*,*Pg*), al fine dell'attribuzione del peso ai singoli percorsi individuati, fornisce una codifica di tutti i pattern più significativi. I simboli terminali *E* sono le label etichettate, mentre i simboli non terminali *N* codificano i (sotto)percorsi tra i concetti ; *Sg* è il simbolo di partenza di *G* e *Pg* è l'insieme delle sue produzioni. Viene associato un peso a ciascuna produzione $A \rightarrow \alpha$ dove *A* appartiene ad *N*, ed α appartiene ad $(N \cup E)$, cioè α è una sequenza di simboli terminali e non. Se la sequenza delle etichette degli archi appartiene ad *L(G)*,il linguaggio generato dalla grammatica ,e determina una *G* non ambigua, allora il peso *w* è dato dalla somma dei pesi delle produzioni applicata nella derivazione $Sg \Rightarrow^{+} e_1 \cdot e_2 \cdot \ldots\ldots e_n \cdot$

Infine l'algoritmo seleziona $S^t = argmax\ f_I\ (S, t)$ come interpretazione più probabile di *t* e aggiorna la lista *I* con il senso prescelto.

Al termine di un'iterazione generica, un certo numero di termini sono stati disambiguati, e ognuno di essi è rimosso dall'insieme *P* dei termini pendenti.

L'algoritmo termina restituendo come output l'insieme *I*, quando per i rimanenti termini in *P* non può essere determinato alcun senso *S* tale che $f_I(S, t)>0$, ovvero l'insieme *P* non può essere ulteriormente ridotto. Durante ogni iterazione, le interconnessioni possono essere determinate solamente tra il senso di un termine pendente *t,* e i sensi già disambiguati durante le precedenti iterazioni.

In ogni iterazione , le connessioni sono cercate soltanto tra un termine *pendente t* e ognuno dei sensi disambiguati durante il processo di iterazione e inseriti in *I.*

Un caso speciale di applicazione dell'algoritmo di SSI si ha quando si verifica che *I = [_, _,…, _].*Questo accade quando, non è disponibile alcun contesto semantico iniziale, cioè, nell'insieme *T*, non sono presenti termini monosemici. In tal caso, si applica una politica di selezione di un termine *t,*(come abbiamo detto, ad esempio, il termine che presenta meno ambiguità), e l'esecuzione dell'algoritmo viene suddivisa in tanti processi quanti sono i sensi di *t.* Ogni processo restituisce un contesto semantico I_i . Alla fine di tutti i processi, viene selezionato il contesto più probabile I_m come

$$m = \underset{1 \le i \le n}{\operatorname{argmax}} \sum_{S^{t_j} \in I_i} f_I(S^{t_j}, t_j)$$

2.3.3 La Grammatica

La grammatica come detto ha lo scopo di descrivere il significato dei percorsi di interconnessione tra i grafi semantici rappresentanti concetti all'interno dell'ontologia *O.*

Un percorso di interconnessione è definito come una sequenza di relazioni semantiche consecutive *e1, e2, …, en,* dove e_i appartiene all'insieme dei simboli terminali *E*. Due relazioni e_i, e_{i+1} sono consecutive se gli archi etichettati con e_i, e_{i+1} sono entranti e\o uscenti dallo stesso nodo. Un percorso significativo tra due sensi *S* ed *S'* è quindi una sequenza di archi *e1, e2, …,en* appartenente all'insieme *L(G).* , dunque al linguaggio definito dalla grammatica. I simboli terminali e_i altro non sono che relazioni

concettuali estratte da WordNet e da altre risorse lessicali on-line. *G* è definita quindi dalla quadrupla (*E, N, Sg, Pg)*, dove:

$E=\{e_{kind\text{-}of}, e_{has\text{-}kind}, e_{part\text{-}of}, e_{gloss}\ldots\}$

$N=\{ S_G, S_S, Sg, S_1, S_2, S_3, S_4, S_5, E_1, E2, \ldots\}$

e l'insieme *Pg* include circa 40 produzioni. Un estratto della grammatica è mostrato in figura.

$S_G \rightarrow S_S \mid S_g$	(*all the heuristics*)
$S_S \rightarrow S_1 \mid S_2 \mid S_3$	(*simple heuristics*)
$S_1 \rightarrow E_1 S_1 \mid E_1$	(*hyperonymy/meronymy*)
$E_1 \rightarrow e_{kind\text{-}of} \mid e_{part\text{-}of}$	
$S_2 \rightarrow E_2 S_2 \mid E_2$	(*hyponymy/holonymy*)
$E_2 \rightarrow e_{has\text{-}kind} \mid e_{has\text{-}part}$	
$S_3 \rightarrow e_{kind\text{-}of}\, S_3\, e_{has\text{-}kind} \mid e_{kind\text{-}of}\, e_{has\text{-}kind}$	(*parallelism*)
$S_g \rightarrow e_{gloss}\, S_S \mid S_4 \mid S_5$	(*gloss*)
$S_4 \rightarrow e_{gloss} \mid e_{topic}$	(*gloss,context*)
$S_5 \rightarrow e_{gloss} \mid e_{is\text{-}in\text{-}gloss}$	

Figura 2.3 Un estratto della grammatica utilizzata per l'attribuzione dei pesi

Il peso *w(e1,e2, …, en)* del percorso semantico *e1*, *e2*, …, *en* è dato dalla somma dei pesi delle produzioni applicati in derivazione $Sg \Rightarrow^{+} e1, e2, \ldots, en$. I singoli pesi sono ricavabili attraverso la seguente funzione:

$$weight\ (percorso\ j) = \alpha_j + \beta_j\ (1/\ lunghezza\_percorso_j)$$

dove α_j è il peso della regola *j* in *G* e βj è il parametro di *smoothing* inversamente proporzionale alla lunghezza del percorso. Un esempio di regole con un elevato valore di α sono l'iperonimia/meronimia. In questa particolare configurazione due concetti sono legati da una sequenza di relazioni iperonimo/meronimo, ossia, ad esempio:

$$mountain\#1 \xrightarrow{has\text{-}part} mountain\_peak\#1 \xrightarrow{kind\text{-}of} top\#3.$$

2.3.4 Test sperimentali e conclusioni

Ad oggi l'algoritmo SSI trova ampia applicazione per quanto concerne le problematiche di disambiguazione del testo. In particolare il suo utilizzo è stata testato nella disambiguazione di definizioni testuali, in ontologie o glossari, e in particolare all'interno del sistema ***OntoLearn*** (21). Quest'ultimo è un sistema realizzato sempre dal dipartimento di Informatica dell'Università di Roma "La Sapienza", al cui progetto hanno fatto capo i medesimi ideatori di SSI, Roberto Navigli e Paola Velardi. O*ntoLearn* è un sistema di disambiguazione del testo, il cui scopo è quello di arricchire ed estendere in maniera automatica WordNet attraverso concetti di dominio e tramite la disambiguazione delle sue glosse. L'algoritmo SSI rappresenta il cuore di tale sistema.

Per concludere, l'algoritmo SSI è stato testato durante lo svolgimento del Senseval-3 tenutosi nel Marzo del 2004. In questa occorrenza, 17 sistemi di disambiguazione supervisionati e 9 non supervisionati sono stati testati e valutati secondo criteri stabiliti dall'organizzazione. È stato richiesto di disambiguare il più alto numero di termini, garantendo un'elevato livello di precision (che significa produrre il massimo livello di precision e di recall). Purtroppo la versione standard dell'algoritmo SSI è stata sviluppata con l'intento di ottimizzare in particolar modo la precision. Infatti, nel caso in cui non si determinasse alcun percorso all'interno del grafo, o nel caso in cui il peso di una relazione fosse inferiore ad una determinata soglia, non viene scelto alcun senso. Tuttavia , per attenersi ai criteri della valutazione, durante il Senseval – 3, è stata fornita una versione alternativa dell'SSI in cui l'algoritmo è stato è stato forzato a scegliere un senso per tutti i termini da disambiguare. È stata, pertanto, rimossa la soglia e, nel caso in cui non l'algoritmo non avesse individuato alcun percorso all'interno del grafo per un dato termine, si è scelto di selezionare il primo senso indicato da

WordNet. Infine, durante la competizione, la LKB non era ancora stata estesa con i dizionari di collocazioni (la quarta risorsa nell'elenco del paragrafo 2.3.1). Di seguito si riporteranno le tabelle contenenti i risultati dei test, valutati in termini di recall e precision, per l' algoritmo SSI così come proposto al Senseval (*with baseline*), e per la sua versione standard (10).

System	Prec.	Recall	Attempted
SSI+baseline	0.685	0.684	99.9
SSI standard	0.826	0.323	39.1

Tabella 2- Risultati nella disambiguazione delle glosse

	Nouns	Verbs	Adj.
Prec.	86.0%	69.4%	78.6%
Recall	44.7%	13.5%	26.2%

Tabella 3 - Precision e Recall in base alla categoria sintattica

Dalle tabelle si notano i buoni risultati dell'algoritmo, che si è classificato al secondo posto, molto vicino al primo e ben lontano dal terzo. Si nota anche che l'algoritmo nella sua versione standard privilegia, per la sua originale natura, la precision a discapito della recall. Un problema aggiuntivo è costituito dal fatto che le LKB, WordNet inclusa, contengono ricche informazioni per nomi e aggettivi, ma sono molto povere di verbi, come emerge analizzando i dati di recall della tabella 3. Questo aspetto è stato in parte migliorato nella versione estesa dell'LKB, ma necessita comunque di ulteriori progressi.

2.4 Algoritmo TUCUXI

TUCUXI è un In**T**elligent H**U**nter Agent for **C**oncept **U**nderstanding and Le**X**ical Cha**I**ning, realizzato dall'Università degli Studi di Modena e

Reggio Emilia, da R.Benassi ,S.Bergamaschi e M.Vincini. Esso in (22) sviluppa il meccanismo delle catene lessicali, allo scopo di realizzare una metodologia alternativa alle tradizionali tecniche di Information Retrieval che, nella ricerca di pagine nel web, utilizzano query basate su parole chiave, dove non si considera né la possibile polisemia dei termini, né le relazioni lessicali che li legano.

L'obiettivo finale dell'agente è infatti quello di effettuare un processo di ricerca dei documenti nel Web, basato non solo sull'occorrenza dei termini utilizzati dall'utente finale per esprimere una richiesta, ma anche sul contenuto semantico che essi arrecano, fornendo un *crawling* accurato che restituisca documenti più pertinenti e conformi alle esigenze dell'utente.

Come vedremo, tale algoritmo mantiene separata le fase di disambiguazione dei termini da quella di creazione delle *catene lessicali* (per maggiori informazioni sul concetto di *catena lessicale* si faccia riferimento a (23)), motivo per il quale è possibile implementare separatamente ciascuna delle due fasi, e utilizzarle all'interno di un più ampio raggio di applicazioni.

Il proposito di questa tesi è , infatti, quello di progettare e realizzare un'implementazione della parte relativa all'annotazione automatica per utilizzarla all'interno del sistema MOMIS di cui si parlerà nel prossimo capitolo. L'algoritmo di TUCUXI, pertanto, rientra nella categoria degli algoritmi di disambiguazione non supervisionati, in cui cioè l'annotazione dei lemmi provenienti da una qualsiasi sorgente dati avviene in maniera completamente automatica. TUCUXI fa di WordNet la sua risorsa lessicale , e si articola in tre steps che possono corrispondere ad altrettanti moduli indipendenti:

1. *Selezione delle candidate words.*

   Le *candidate words* rappresentano i termini utili al processo di creazione delle catene lessicali. Sotto l'assunzione che i concetti sono espressi in maniera ragionevolmente efficace dai sostantivi, soltanto questi ultimi, una volta ricondotti alla loro forma base attraverso un *Part of Speech* parser, faranno parte della lista di termini da sottoporre al processo di disambiguazione.

   Inoltre uno *Shallow* parser identificherà i termini composti, che, nella stragrande maggioranza dei casi, è conveniente mantenere inalterati (separando i lemmi che ne fanno parte con il carattere di *underscore*, es.*computer_science*), dato il maggior contenuto semantico che apportano rispetto ai singoli termini presi separatamente.

2. *Disambiguazione dei Lemmi*

   È la fase oggetto della trattazione. Dopo aver ricevuto in input l'insieme dei lemmi, ricondotti alla propria forma base, e avvalendosi di wordnet come risorsa lessicale , assocerà ad ogni lemma il synset ritenuto più probabile, tramite dei meccanismi che verranno descritti accuratamente nel paragrafo seguente. Formalmente produrrà una lista di coppie *lemma – synset* , denominate Basic Units, in cui il *lemma* rappresenta il termine da disambiguare e il *synset* rappresenta il senso attribuito dall'algoritmo tra quelli possibili recuperati dal database di WordNet

3. *Concatenazione lessicale per la costruzione di una Mappa Concettuale (Map of Meanings)*

   Il processo di creazione delle catene lessicali (23), si realizza attraverso un algoritmo di *clustering*, che riceve come input le unità di base selezionate al passo 2, e produce come output un insieme di

cluster composti da i termini disambiguati. In questo algoritmo si è deciso di selezionare solo relazioni di coesione *strong* fra i termini, ottenendo quindi delle catene lessicali che siano anch'esse *strong*. A tal fine,viene adottato il criterio di identificazione delle catene strong proposto da Barzillay e Elhadad (24), applicando tuttavia ulteriori filtri nel caso in cui si abbia a che fare con documenti ricchi d'informazione testuale. .

2.4.1 Descrizione della fase II dell'algoritmo: Word Sense Disambiguation

L'approccio adottato per la risoluzione del problema della disambiguazione si basa sulla comprensione del fatto che il linguaggio naturale preserva la semantica del testo e provvede a un recupero, fondato sui concetti, dei documenti. In particolare, per tener conto del fatto che le parole rappresentanti concetti sono naturalmente connesse tra di loro, abbiamo bisogno di identificare i significati dei termini e tener traccia delle reciproche relazioni semantiche. Halliday e Hasan, in (25), hanno fornito un valido supporto a questa intuizione: i lettori umani sono in grado di comprendere un testo scritto in virtù del fatto che ogni linguaggio ha degli strumenti che rendono possibile l'unione semantica di frasi diverse (proprietà di *coesione*) e ne favoriscono l'attribuzione di un senso logico ben definito(proprietà di *coerenza*). Mentre la proprietà di coerenza dipende dal punto di vista del lettore, la proprietà di coesione ha una natura maggiormente oggettiva, che viene supportata dalle caratteristiche lessicali del linguaggio: la coesione lessicale può essere verificata attraverso la **reiterazione (**rafforzamento dell'espressione di un concetto attraverso la ripetizione di un termine e/o dei suoi sinonimi, iponimi o ipernimi) e la **collocation** (26) (regolare combinazione di parole che spesso occorrono insieme) di un termine. La coesione lessicale pertanto rappresenta

l'elemento essenziale per recuperare le relazioni esistenti tra i termini esaminati e dunque risalire al contesto in cui tali termini appaiono. L'implementazione dell'algoritmo di disambiguazione si basa proprio su questa fondamentale proprietà del linguaggio. Il database di WordNet, e le sue estensioni create per sopperire alle non poche carenze semantiche della versione originale, fornisce gli strumenti fondamentali per la ricerca delle relazioni tra i synset associati ad ogni lemma. Di seguito lo pseudo-codice che descrive le operazioni da effettuare per il conseguimento dell'obiettivo.

Algorithm 1 TUCUXI's Word Sense Disambiguation

Input: *WordNet lexical Database (WNx) and its extensions if any*
S=[s_i: s_i is one of the 1..n possible synsets contained in the text], an ordered set CW=[w_j: w_j is one of the 1..k candidate words in the text], WS_j=[ws_l: ws_l is one of the 1..t possible meanings of w_j], $j = 1..k$, scoring criteria C.
for $i = 1$ to n **do**
  ask WNx for s_i hyponyms, hypernyms, siblings,... meronyms and holonyms
  build the list of related synsets RS_i;
end for
for $i = 1$ to n **do**
  select the words in CW whose ws_l=s_i;
  update cohesion vote for the words whose ws is contained in RS_i (according to relationship strength and position of words in text, i.e. scoring criteria C);
end for
for $j = 1$ to k **do**
  select the $ws_l{}^{best}$ meaning in WS_j (with the highest score or the most frequent one in case of a tie)
  store the $ws_l{}^{best}$ meaning in the basic units list BU;
  for all ws_l in WS_j **do**
    if ws_l != $ws_l{}^{best}$ **then**
      nullify the votes expressed by the ws_l synset of the word w_j in the previous phase;
    end if
  end for
end for
Update S by deleting the s_i that are not preserved (and the related list RS_i);
Output: *a list BU of basic units, which stores the most reasonable meaning for each word in CW, a list of preserved synsets and their related ones.*

Figura 2.4 - Pseudo-codice della fase II dell'algoritmo TUCUXI

2.5 Applicabilità degli algoritmi di disambiguazione

Le possibilità di applicazione degli algoritmi di disambiguazione lessicale sono numerose. Basti pensare ai vantaggi che possono essere tratti dalla conoscenza del contenuto semantico associato ad un semplice testo scritto. Il contenuto semantico può arricchire ad esempio i criteri di ricerca basati sull'utilizzo di semplici parole chiave nei sistemi di *Information Retrieval* . Conoscere il significato corretto che ogni parola assume all'interno di un contesto migliora nettamente le prestazioni in termini di selezione e di pertinenza dei documenti recuperati. Si pensi ad esempio alla ricerca di fonti di informazione on-line, attraverso i famosi motori di ricerca. Il motore di ricerca risponde ad una query sottoposta dall'utente restituendo un elenco ordinato di pagine, basandosi solo sul contenuto del testo inserito come chiave di ricerca. In altre parole solo sulla *word form* delle keywords.Tale metodo soffre di notevoli limiti legati alla *polisemia* e alla *sinonimia* (per la quale indicare una keyword non permette di specificare automaticamente anche le parole che hanno un significato simile).
Per effetto della polisemia, il motore di ricerca tende a indicare come rilevanti documenti che contengono la parola richiesta, ma spesso essa non ha il significato desiderato. Al contrario, per effetto della sinonimia viene restituita solo una piccola parte di tutti i documenti potenzialmente rilevanti. Per ovviare a questo inconveniente, sarebbe auspicabile poter esprimere le richieste in un modo più efficace, nonchè disporre di uno strumento , come appunto un algoritmo di disambiguazione lessicale, in grado di estrarre e valutare non tanto la presenza di keywords ma il contenuto semantico di un documento.

Un altro ambito all'interno del quale può essere sfruttata la disambiguazione lessicale è quello dell'integrazione delle informazioni.

Lo scopo dei sistemi di integrazione, che verrà descritto dettagliatamente nel capitolo seguente, è , sinteticamente, quello di ottenere in maniera automatica una selezione ragionata dei dati provenienti da varie sorgenti di informazione autonome ed eterogenee tra loro, per proporne una fusione intelligente. Un sistema che tenta di concretizzare tale obiettivo è MOMIS (***M****ediator* ***E****nvir****O****ment for* ***M****ultiple* ***I****nformation* ***S****ources*), il quale, sviluppato dal gruppo DBGROUP dell'Università degli Studi di Modena e Reggio Emilia, rappresenta un progetto ideato per l'integrazione di sorgenti di dati testuali, strutturati e semi-strutturati.

Le problematiche legate alla realizzazione di tale sistema sono molteplici, ma l'attenzione di questa tesi si concentra in particolare sui problemi derivanti dalla semantica dei dati rappresentati all'interno delle sorgenti. Il problema semantico si intuisce facilmente, se si considera la possibilità che diverse persone possano fornire descrizioni, anche molto diverse tra loro, della stessa porzione di mondo.

Di conseguenza, nel processo d'integrazione, uno fra i tanti obbiettivi dovrà essere quello di risolvere le differenze semantiche fra le diverse rappresentazioni dei dati (effettuare cioè, un *mapping semantico*). Tale problematica trova un tentativo di soluzione in MOMIS attraverso l'annotazione semantica delle diverse sorgenti.

In genere, l'annotazione è definita come l'aggiunta di informazioni addizionali ad una sorgente di dati. L'annotazione semantica è un particolare tipo di annotazione che fa riferimento a una risorsa lessicale (thesaurus, rete semantica, lessico semantico). Ogni annotazione lessicale ha la proprietà di possedere una descrizione lessicale, pertanto l'annotazione di sorgenti di dati associa significati ad ogni elemento di uno schema.

Attualmente, tale annotazione è realizzata in maniera completamente manuale ,ma numerosi sono stati gli sforzi profusi affinché potesse essere

anche solo parzialmente automatizzata. L'annotazione di un termine implica ovviamente al suo interno il concetto di disambiguazione del termine stesso. Nel prossimo capitolo si analizzerà nel dettaglio l'architettura di un sistema di integrazione delle informazioni, facendo riferimento in particolare al sistema MOMIS, all'interno del quale l'algoritmo proposto da questa tesi è stato sviluppato, e si descriveranno più precisamente le modalità e i motivi per cui un algoritmo di disambiguazione può costituire un valido supporto al processo di integrazione.

Capitolo 3

L’annotazione lessicale come supporto ai sistemi di integrazione delle informazioni

Per capire pienamente il ruolo ricoperto dagli algoritmi di disambiguazione lessicale all’interno dei sistemi di integrazione delle informazioni, è necessario fornire una panoramica di tali sistemi, dei servizi offerti e della architettura cui fanno riferimento. I paragrafi seguenti si pongono esattamente questo obiettivo, ossia chiarire le modalità in cui avviene il processo di integrazione delle informazione e i motivi per cui l’annotazione automatica può offrire un valido supporto a tale processo.

Si presenterà inoltre il sistema MOMIS (27) (28), all’interno del quale il mio lavoro è inserito, che integra un modulo denominato ALA (**A**uomatic **L**exical **A**nnotator), interamente dedicato alla annotazione automatica delle sorgenti informative.

3.1 L’integrazione delle informazioni e il programma I3

Il constante sviluppo delle tecnologie legate all’informazione, cui consegue un non controllabile incremento delle sorgenti dati disponibili sulla rete, ha spianato la strada ad un fenomeno conosciuto come *information overloading*, il cosiddetto “sovraccarico cognitivo”. Questo fenomeno, considerato ovviamente negativo, consiste fondamentalmente nell’impossibilità o nella grande difficoltà, dovuta alla sconfinata mole di informazioni presenti sul web o in ambiti locali, di reperire conoscenze corrette o informazioni di reale interesse.

Strettamente correlato a questo fenomeno vi è d'altro canto, oltre alla grande quantità di informazioni, anche una grande eterogeneità dei dati disponibili, sia per quanto riguarda la loro natura (testi, immagini, audio, video, etc.), sia il modo in cui vengono descritti (Database relazionali o a oggetti, file XML, sorgenti di testo…)

Gli attuali protocolli di comunicazione (TCP/IP, ODBC, OLE, CORBA, SQL, etc.) , idonei a risolvere parzialmente i problemi relativi alle diversità hardware e software dei sistemi, non forniscono soluzioni ai problemi relativi alla modellazione delle informazioni. I modelli e gli schemi utilizzati per raccogliere e organizzare dati sono infatti eterogenei tra loro, e non è ancora stato sviluppato uno standard che ne delinei i confini.

Nasce, alla luce di queste considerazioni, la necessità di creare sistemi in grado di estrarre i contenuti informativi da fonti di conoscenza differenti tra loro, tanto per la propria architettura, quanto per la rappresentazione interna dei dati. Il termine Integrazione delle Informazioni (I_2) indica in letteratura proprio il processo adottato da quei sistemi che combinano informazioni provenienti da diverse sorgenti (o parti selezionate di esse) senza dover ricorrere alla duplicazione fisica (29).Per ottenere risultati selezionati e richiesta *conoscenza* ed *intelligenza* finalizzate alla scelta delle sorgenti e dei dati, alla loro fusione e alla conseguente sintesi. L'integrazione comporta perciò grandi difficoltà, sia a livello teorico che pratico, oltre a concrete differenze realizzative.

La dimensione e la quantità delle problematiche che insorgono qualora si desideri fondere informazioni provenienti da sorgenti autonome fanno sì che si senta l'esigenza di ideare una metodologia che garantisca:

- riusabilità, per diminuire i costi di sviluppo di tali applicazioni.
- flessibilità, per far fronte intelligentemente alle evoluzioni fisiche e logiche delle sorgenti interessate.

Le tecniche sviluppate nel campo dell'*Intelligenza Artificiale*, potendo efficacemente dedurre informazioni utili dagli schemi delle sorgenti, diventano uno strumento prezioso per la costruzione automatica di soluzioni integrate flessibili e riusabili. Quando l'Integrazione delle Informazioni fa uso di tecniche di Intelligenza Artificiale in grado di accrescere il "valore" delle informazioni recuperate e di crearne delle nuove a partire dai dati preesistenti, si parla allora di Integrazione Intelligente delle Informazioni (*Intelligent Integration of Informations , I_3*). Il programma I_3 è un ambizioso progetto di ricerca finalizzato a delineare un'architettura di riferimento che realizzi l'integrazione di sorgenti eterogenee in maniera automatica. Il programma I_3 è sviluppato dall'ARPA (1), l'agenzia che finalizzata ad indicare un architettura di riferimento che fa capo al Dipartimento di Difesa statunitense. In quel contesto si è potuto osservare che l'integrazione aumenta il valore dell'informazione ma richiede una forte adattabilità realizzativa: si devono infatti riuscire a gestire i casi di aggiornamento e sostituzione delle sorgenti, dei loro ambienti e/o piattaforme, della loro ontologia e della semantica. Costruire in modo premeditato supersistemi che interessino una gran quantità di sorgenti non correlate semanticamente, è faticoso ed il risultato è un sistema scarsamente manutenibile e poco riusabile. Secondo il programma I_3 una soluzione a questi problemi può essere rappresentata dall'introduzione di architetture modulari sviluppabili secondo i principi proposti da uno standard, il quale deve porre la basi dei servizi da realizzare contenendo allo stesso tempo i costi di sviluppo e di mantenimento.

Il paradigma suggerito per la suddivisione dei servizi e delle risorse nei diversi moduli, si articola su due dimensioni:

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

Ad esempio, si supponga di dover integrare informazioni sui trasporti mercantili, ferroviari e stradali: ciò permette all'utente di avere un'idea completa su quale mezzo di trasporto sia maggiormente vantaggioso ai propri fini. Il livello dell'architettura su cui si deve focalizzare l attenzione è quello intermedio: esso costituisce il punto nodale fra le applicazioni sviluppate per gli utenti ed i dati nelle sorgenti. Questo livello deve offrire servizi dinamici quali la selezione delle sorgenti, la gestione degli accessi e delle interrogazioni, la ricezione e la combinazione dei dati, l'analisi e la sintesi degli stessi. .Tali servizi sono forniti attraverso tre moduli principali:

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

3.2 Architettura di riferimento

Si presenterà in questo paragrafo l'architettura di riferimento così come presentata dal sito web riportato in (1), e rappresenta una classificazione generale dei principi e dei servizi da utilizzare nella realizzazione di un

integratore intelligente di informazioni. In particolare due sono le ipotesi che rappresentano la base del progetto *I3*.

La prima è, come detto, connessa con la difficoltà collegata alla ricerca delle informazioni all'interno della molteplicità delle sorgenti di informazione che in questo momento è possibile individuare. Il secondo aspetto è legato al fatto che le fonti di informazione e i sistemi informativi, pur essendo spesso semanticamente correlati tra di loro, non lo sono in una forma semplice né preordinata. Di conseguenza, il processo di integrazione di informazioni può risultare molto complesso. Va sottolineato, infine, che questa architettura non implica una soluzione implementativa, bensì vuole rappresentare alcuni dei servizi, e le interconnessioni tra essi, che devono essere forniti da un qualunque integratore di informazioni, del quale potranno essere sfruttate anche solo le funzionalità necessarie a svolgere un determinato compito.

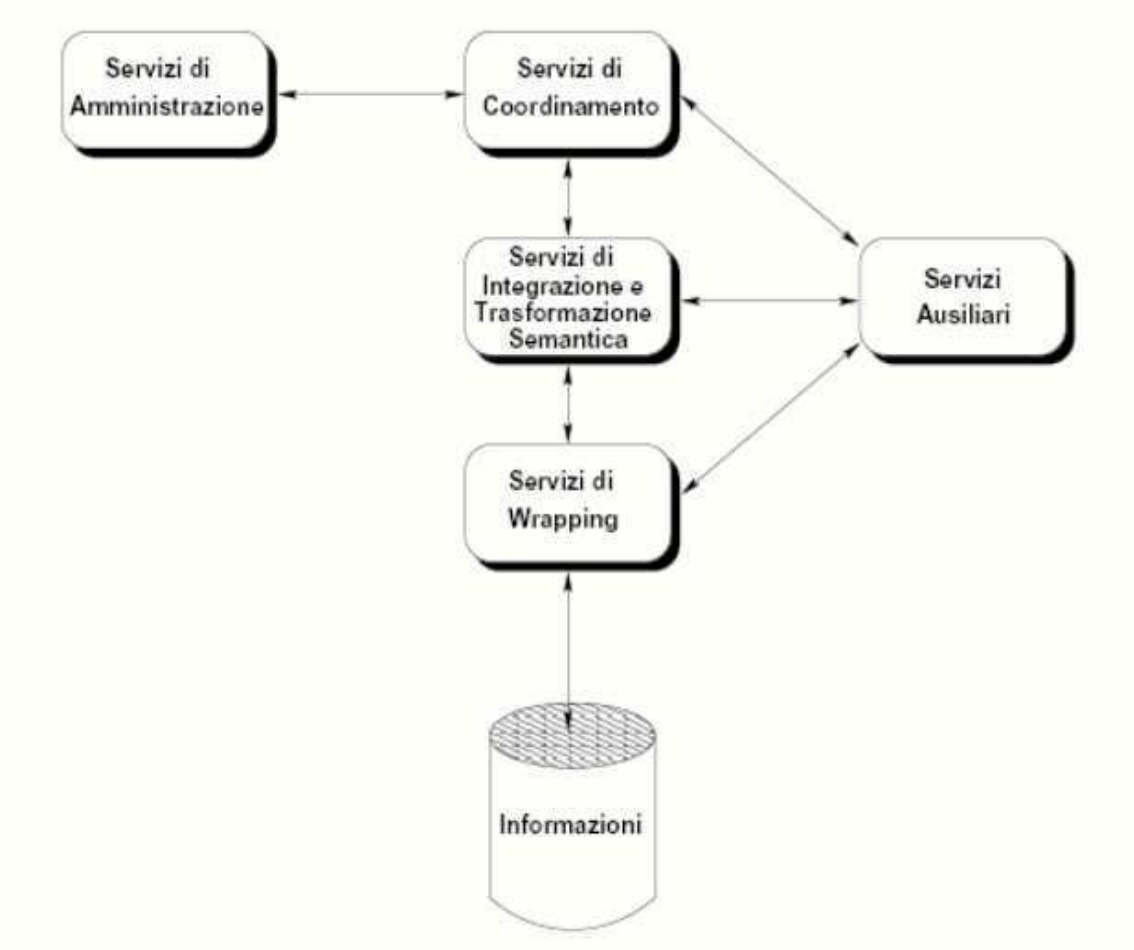

Figura 3.1 - Architettura di riferimento dei sistemi I3

3.2.1 Servizi di Coordinamento

I servizi di Coordinamento sono quei servizi di alto livello che permettono l'individuazione delle sorgenti di dati interessanti, dunque quei servizi che interagiscono direttamente con l'utente. Tali servizi possono comprendere compiti che vanno dalla selezione dinamica delle sorgenti (o brokering, per Integratori Intelligenti) fino al semplice Matchmaking, in cui il mapping tra informazioni integrate e locali è realizzato manualmente ed una volta per tutte. Quelli che seguono sono i moduli principali che possono implementare tali servizi:

- *Facilitation e Brokering Services*: l'utente manda una richiesta al sistema e questo usa un deposito di metadati per ritrovare il modulo che può trattare la richiesta direttamente. I moduli interessati da questa richiesta possono

essere o uno solo alla volta (nel qual caso si parla di Brokering) oppure più di uno(in questo caso si parla di Mediator). In questo ultimo caso una query può essere decomposta in un insieme di sottoquery, mentre si farà uso di servizi di Query Decomposition e di tecniche di Inferenza (mutuate dall'Intelligenza Artificiale) per una determinazione dinamica delle sorgenti da interrogare, a seconda delle condizioni poste nell'interrogazione. Le sottoquery vengono poi gestite da moduli differenti che interagiscono con sorgenti distinte, e forniscono risultati che vengono poi reintegrati dal mediatore per essere presentati all'utente come se fossero state ricavate da un'unica fonti. Viene così offerta una visione globale all'utente, il quale non è obbligato a conoscere i domini con i quali i vari moduli I_3 si trovano ad interagire. Risulta quindi evidente la considerevole diminuzione di complessità di interazione col sistema che deriva da una architettura di questo tipo.

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

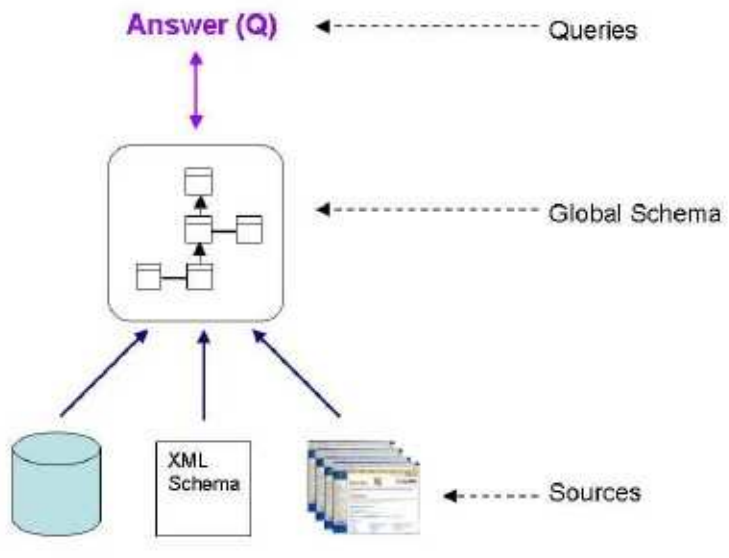

Figura 3.2 Schema di un sistema di integrazione di dati

3.2.2 Servizi di Amministrazione

Sono servizi usati dai Servizi di Coordinamento per localizzare le sorgenti *utili*, per determinare le loro capacità, e per creare ed interpretare TEMPLATE. I Template sono strutture dati che descrivono i servizi, le fonti ed i moduli da utilizzare per portare a termine un determinato task. Sono quindi utilizzati dai sistemi meno ”intelligenti”, e consentono all’operatore di predefinire le azioni da eseguire a seguito di una determinata richiesta, limitando al minimo le possibilità di decisione del sistema. In alternativa a questi metodi dei Template, sono utilizzate le **Yellow Pages**: servizi di directory che mantengono le informazioni sul contenuto delle varie sorgenti e sul loro stato (attiva, inattiva, occupata). Consultando queste Yellow Pages, il mediatore sarà in grado di spedire alla giusta sorgente la richiesta di informazioni, ed eventualmente di rimpiazzare questa sorgente con una equivalente nel caso non fosse disponibile.

3.2.3 Servizi di Integrazione e Trasformazione Semantica

Questi rappresentano i servizi più interessanti per il lavoro svolto, perché è proprio per offrire un supporto a tali servizi che la presente tesi è stata sviluppata.

Tali servizi infatti supportano le manipolazioni semantiche necessarie per l'integrazione e la trasformazione delle informazioni. Il tipico input per questi servizi è rappresentato da una o più sorgenti di dati, e l'output è la vista integrata o trasformata di queste informazioni. Essi possono essere suddivisi in :

- ***Servizi di integrazione degli schemi***. la trasformazione e l'integrazione degli schemi e delle conoscenze derivanti da fonti di dati eterogenee. Nel momento in cui devono essere scambiate informazioni da fonti eterogenee infatti, che non condividono un'unica ontologia, è necessario effettuare una trasformazione dei vocaboli e dei concetti espressi nelle singole ontologie per costruirne una condivisa. Fondamentale, a questo scopo, è la fase di individuazione dei concetti presenti in diverse fonti, e la riconciliazione delle diversità presenti sia nelle strutture, sia nei significati dei dati, tutti obiettivi il cui conseguimento può essere agevolato dai numerosi algoritmi di disambiguazione lessicale presenti nel campo della linguistica computazionale.
- ***Servizi di integrazione delle informazioni***. Provvedono alla traduzione dei termini da un contesto all'altro, ovvero dall'ontologia di partenza a quella di destinazione. Possono inoltre occuparsi di uniformare la "granularità" dei dati (come possono essere le discrepanze nelle unità di misura, o le discrepanze temporali).

- ***Servizi di supporto al processo di integrazione***. sono utilizzati quando la query deve essere scomposta in più sotto-query da inviare a fonti differenti, con la necessità di integrare, poi, i loro risultati.

3.2.4 Servizi di wrapping

Sono utilizzati per fare sì che le fonti di informazioni aderiscano ad uno standard e fungono da traduttori dai sistemi locali ai servizi di alto livello dell'integratore e viceversa quando si interroga la sorgente di dati.

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

3.2.5 Servizi Ausiliari

I servizi ausiliari aumentano le funzionalità degli altri servizi descritti precedentemente: sono prevalentemente utilizzati dai moduli che agiscono direttamente sulle informazioni. Vanno dai semplici servizi di monitoraggio del sistema, ai servizi di propagazione degli aggiornamenti e di ottimizzazione.

3.3 Il mediatore

Secondo la definizione proposta da Wiederhold in (29) *"un mediatore è un modulo software che sfrutta la conoscenza su un certo insieme di dati per creare informazioni per una applicazione di livello superiore"*. . . .

Dovrebbe essere piccolo e semplice, così da poter essere amministrato da uno, o al più pochi, esperti. Compiti di un mediatore sono allora:

1. assicurare un servizio stabile, anche nel caso di cambiamento delle risorse;
2. amministrare e risolvere le eterogeneità delle diverse fonti;
3. integrare le informazioni ricavate da più risorse;
4. presentare all'utente le informazioni attraverso un modello scelto dall'utente stesso.

L'approccio architetturale scelto per la realizzazione del mediatore MOMIS è quello classico, che si sviluppa principalmente su 3 livelli:

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

Facendo riferimento ai servizi descritti nelle sezioni precedenti, l'architettura del nuovo mediatore è riportata in Figura 3.3

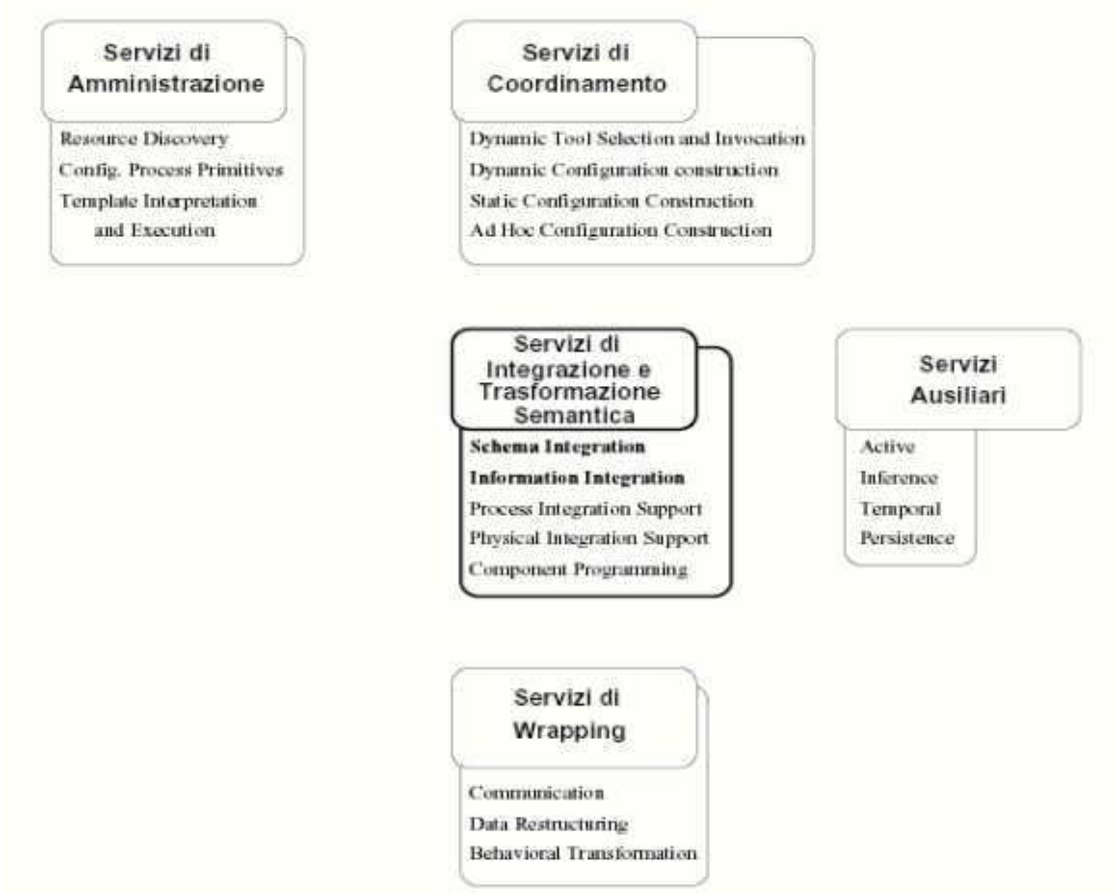

Figura 3.3 - Architettura di riferimento del mediatore

Esistono due approcci fondamentali all'architettura precedentemente descritta:

- Approccio strutturale caratterizzato dall'uso di *self-describing model* per rappresentare gli oggetti da integrare, limitando così l'uso delle informazioni semantiche a delle regole predefinite dall'operatore. In pratica, il sistema non conosce a priori la semantica di un oggetto che va a recuperare da una sorgente, bensì è l'oggetto stesso che, attraverso delle etichette, si auto-descrive, specificando tutte le volte, per ogni suo singolo campo, il significato associato.

- Approccio semantico, adottato anche dal progetto MOMIS, è caratterizzato dai seguenti punti:

1. il mediatore deve conoscere, per ogni sorgente, lo schema concettuale (metadati);
2. le informazioni semantiche sono codificate in questi schemi;
3. deve essere disponibile un modello comune per descrivere le informazioni da condividere (e dunque per descrivere anche i metadati);
4. deve essere possibile una integrazione (parziale o totale) delle sorgenti di dati.

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

3.3.1 Caratteristiche dell'approccio semantico

Il cuore dell'architettura I^3 è rappresentato dunque dai Servizi di Trasformazione ed Integrazione Semantica. Essi si occupano dell'integrazione vera e propria delle informazioni cercando di risolvere le differenze tra gli schemi, i modelli ed i tipi di dati. Un altro aspetto interessante dell'approccio semantico riguarda l'incremento del potere espressivo delle interrogazioni: molti sistemi, specialmente quelli accessibili via Web, supportano solo un ristretto numero di queries predefinite,ma con l'aggiunta di contenuti semantici è possibile ricondurre le queries più complesse in quelle predefinite o in un loro sovrainsieme.

Ma l'aspetto principale su cui si basa l'integrazione semantica sorgenti eterogenee nella struttura, riguarda l'individuazione delle sovrapposizioni delle rispettive ontologie. In altre parole individuare corrispondenze semantiche fra gli schemi delle sorgenti per scoprirne le affinità.

Tale problema non è di semplice soluzione per due diversi motivi, uno prettamente semantico, e un altro più propriamente ontologico.

Riferendoci alla prima categoria di problematiche, si può affermare che è molto improbabile che, pur condividendo concetti comuni, due persone diverse producano una modellazione di una stessa porzione di realtà utilizzando gli stessi vocaboli o tanto meno le stesse strutture dati.

Come riportato in (30) la causa principale delle differenze semantiche (anche se non l'unica) è identificabile nelle diverse concettualizzazioni del mondo esterno che persone distinte possono avere. Si possono utilizzare nomi di entità differenti, associazioni differenti, gerarchie o specializzazioni diversamente concepite. Senza considerare poi le differenze che sorgono quando si utilizzano modelli diversi di rappresentazione dei dati (un database relazionale è diverso da un database object oriented).

L'obiettivo dell'integratore di fornire un accesso integrato ad un insieme di sorgenti, si traduce quindi nel difficile compito di identificare i concetti comuni all'interno delle sorgenti e risolvere le eventuali differenze semantiche. Si possono classificare queste incoerenze semantiche in tre gruppi principali:

1. *eterogeneità tra le classi di oggetti:* benchè due classi in due differenti sorgenti rappresentino lo stesso concetto nello stesso contesto, possono usare nomi diversi per gli stessi attributi o per i metodi, oppure avere gli stessi attributi con domini di valori differenti o ancora avere regole diverse sui valori;
2. *eterogeneità tra le strutture delle classi:* comprendono le differenze nei criteri di specializzazione, nelle strutture per realizzare un'aggregazione, ed anche le discrepanze schematiche, quando cioè valori di attributi sono invece parte dei metadati in un altro schema;
3. *eterogeneità nelle istanze delle classi:* ad esempio, uso di diverse unità di misura per i domini di un attributo, o la presenza/assenza di valori nulli.

Analizzando a fondo queste differenze, e le loro motivazioni, si può ottenere un certo **arricchimento semantico**, ovvero l'aggiunta di tutte quelle informazioni strutturali e non che erano originariamente presenti solo in forma di metadati negli schemi, dunque in un formato non interrogabile.

La seconda tipologia di problemi precedentemente citata, ossia i problemi ontologici, non è prettamente inerente a questa trattazione, pertanto se ne darà una descrizione sommaria.

Nell'ambito dell'information technology Guarino in (31) individua tre livelli di ontologia:

1. *top-level ontology*: descrive concetti molto generali come spazio, tempo, evento, azione. . . , che sono quindi indipendenti da un particolare problema o dominio: si considera ragionevole, almeno in teoria, che anche comunità separate di utenti condividano la stessa top-level ontology;
2. *domain e task ontology*: descrive, rispettivamente, il vocabolario relativo a un generico dominio (come può essere un dominio medico, o automobilistico) o a un generico obiettivo (come la diagnostica, o le vendite), dando una specializzazione dei termini introdotti nelle top-level ontology;
3. *application ontology*: descrive concetti che dipendono sia da un particolare dominio che da un particolare obiettivo.

E' lecito supporre che, integrando sorgenti seppur autonome, il livello dell'ontologia sia vincolato all'interno di un certo dominio; questa restrizione è congruente con l'idea di integrare i domini comuni di sorgenti che descrivono campi almeno parzialmente sovrapposti. Non disponendo delle informazioni sui Metadati e sulle Ontologie, però risulta impossibile

realizzare una buona integrazione. In conclusione, ai fini dell’integrazione è indispensabile disporre di strumenti in grado di dedurre, direttamente dagli schemi, informazioni sui metadati e sulle ontologie.
Tali strumenti sono a presenti nell’architettura del mediatore MOMIS, che si descriverà nel prossimo paragrafo.

3.4 Il sistema MOMIS

MOMIS - acronimo di Mediator envirOnment for Multiple Information Sources - è un sistema intelligente per l'integrazione di informazioni da sorgenti di dati strutturati e semi-strutturati. Il contributo innovativo di questo progetto, rispetto ad altri similari, risiede nella fase di analisi ed integrazione degli schemi sorgenti, realizzata in modo semiautomatico (32) (28). MOMIS nasce all'interno del progetto MURST 40% INTERDATA dalla collaborazione tra i gruppi operativi dell'Università di Modena e Reggio Emilia e di quella di Milano.

3.4.1 L'architettura generale di MOMIS

Il sistema MOMIS, come descritto nel paragrafo precedente, adotta un approccio di tipo semantico e virtuale all’integrazione delle sorgenti. Con virtuale si intende che la vista integrata delle sorgenti, rappresentata dallo schema globale, non viene materializzata, ma il sistema utilizza il meccanismo descritto nel paragrafo 3.2.1 di *Query Decomposition* e di individuazione delle sorgenti da interrogare per generare le *subqueries* eseguibili localmente; lo schema globale dovrà inoltre disporre di tutte le informazioni atte alla fusione dei risultati ottenuti localmente per poter ottenere una risposta significativa.

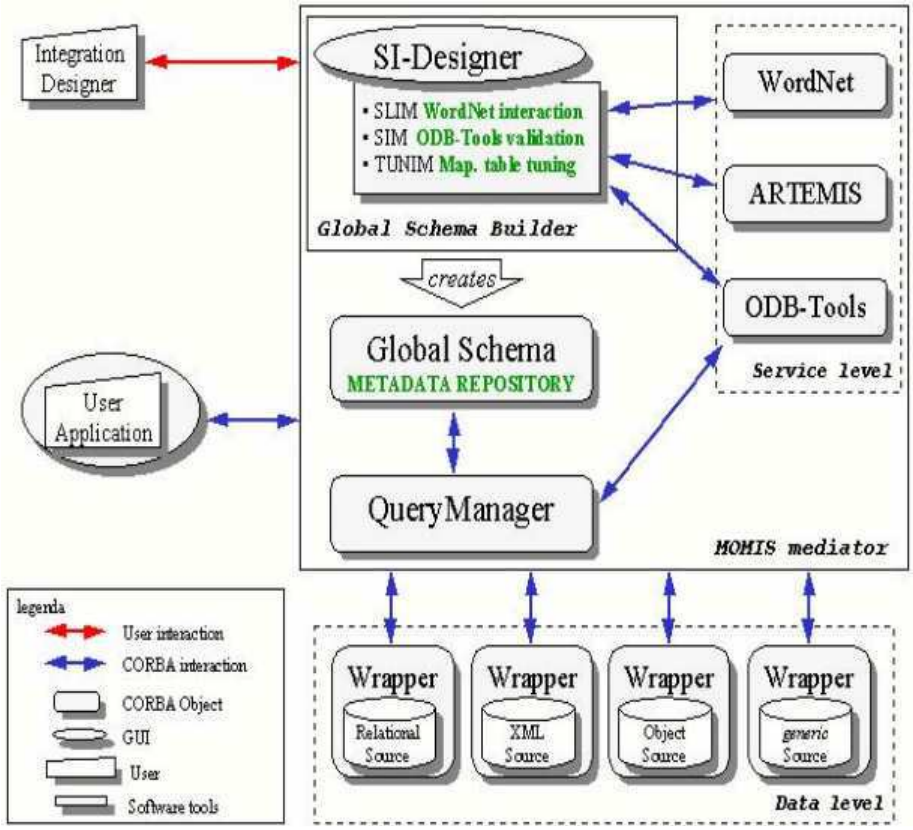

Figura 3.4 - Architettura generale del sistema MOMIS

La Figura 3.4 illustra i componenti dell'architettura generale di MOMIS. Lo schema evidenzia l'organizzazione a tre livelli utilizzata.

Livello dati. Il livello dati è rappresentato dai *Wrappers* costituiscono l'interfaccia tra il mediatore e le sorgenti; ad ogni sorgente corrisponde un determinato wrapper ed ogni wrapper deve essere disegnato esclusivamente per la sorgente (o la tipologia di sorgenti) che sovrintenderà. Ogni wrapper ha una duplice funzione:

1. in fase di integrazione deve fornire al Global Schema Builder la descrizione della sorgente da integrare in formato *ODLI*$_{3,,}$ un linguaggio orientato all'oggetti definito in MOMIS e descritto nel paragrafo 3.4.3 ;

2. in fase di query processing, traducono la query ricevuta dal mediatore (espressa quindi nel linguaggio comune di interrogazione *OQLI3*, definito in analogia al linguaggio OQL) in una interrogazione comprensibile (e realizzabile) dalla sorgente stessa. Devono inoltre esportare i dati ricevuti in risposta all' interrogazione, presentandoli al mediatore attraverso il modello comune di dati utilizzato dal sistema.

Collegate ai wrapper sono quindi le Sorgenti, per questo a volte si parla anche di quattro livelli. Esse sono le fonti da integrare, possono essere DataBase (ad oggetti o relazionali) o parti di essi, file system ed anche sorgenti semistrutturate (es. XML schema).

Livello Mediatore. Il nucleo centrale del sistema è costituito dal *Mediatore* (o *Mediator*) che presiede all'esecuzione di diverse operazioni. La sua attività si può suddividere in tre funzioni principali svolte da altrettanti moduli .

- La prima funzionalità del Mediatore è quella di generazione dello Schema Globale. In questa fase il modulo preposto del Mediatore, denominato Global Schema Builder, riceve in input le descrizioni degli schemi locali delle sorgenti espressi in *ODLI3* fornite dai relativi *wrapper*. Successivamente ,utilizzando strumenti di ausilio quali ODB-Tools Engine, WordNet, ARTEMIS, il *Global Schema Builder* è in grado di costruire la vista virtuale integrata (*Global Virtual View*) utilizzando tecniche di clustering e di Intelligenza Artificiale. In questa fase è prevista anche l'interazione con il progettista il quale, grazie al tool d'ausilio chiamato SI-Designer, oltre ad inserire le regole di mapping semantico, interviene nei processi che non possono essere svolti automaticamente dal sistema

(come ad es. l'assegnamento dei nomi alle classi globali, la modifica di relazioni lessicali, . . .).

- La seconda funzionalità è quella del query processing importante modulo che compone la struttura del mediatore è il *Query Manager* In particolare,il QM genera le query in linguaggio *OQLI*$_3$ da inviare ai wrapper partendo dalla singola query formulata dall'utente sullo schema globale. Avvalendosi delle tecniche di Logica Descrittiva, il QM traduce automaticamente la query nelle corrispondenti sub-queries da sottoporre alle singole sorgenti.
- La terza funzionalità è quella di fornire un'interfaccia grafica che guidi l'utente nelle varie fasi del processo ti integrazione. Questa funzionalità è implementata dalla GUI del SI-Designer. Esso risulta a sua volta composto da quattro moduli:

  1. SIM (Source Integrator Module): estrae le relazioni inter-schema sulla base della struttura delle classi ODLI$_3$ e delle sorgenti relazionali usando ODBTools, strumento utile alla "validazione semantica" e generazione di nuove relazioni.
  2. SLIM (Source Lexical Intergrator Module): estre le relazioni inter-schema tra nomi di attributi e classi ODLI3, sfruttando il database lessicale WordNet.
  3. TUNIM (Tuning of the Mapping Table): questo modulo gestisce la fase di creazione dello schema globale.

Livello Utente: L'utilizzatore del sistema avrà la possibilità di interrogare lo schema globale. L'accesso ai dati delle diverse sorgenti risulta all'utente del tutto trasparente, in quanto è il sistema ad occuparsi di tutte le operazioni necessarie per reperire le informazioni e combinare i risultati delle query in

un'unica risposta corretta, completa e non ridondante. Tra i tools di ausilio del mediatore si possono riconoscere:

ODB-Tools: è uno strumento software sviluppato presso il dipartimento di Ingegneria dell'Università di Modena e Reggio Emilia (33) (34). Esso si occupa della validazione di schemi e dell'ottimizzazione semantica di interrogazioni rivolte a Basi di Dati orientate agli Oggetti (OODB).

WORDNET (3) il DataBase lessicale on-line in lingua inglese descritto ampiamente nel capitolo 1.

ARTEMIS-Tool Enviroment: tool basato sulle tecniche di *clustering affinity-based* che compie l'analisi ed il clustering delle classi ODLI3 (35)

3.4.2 Note sul linguaggio ODLI3

Il linguaggio ODLI$_3$ rappresenta il linguaggio intermedio, concepito e sviluppato all'interno del sistem MOMIS, di descrizione degli schemi estratti tramite i wrappers dalle sorgenti di dati.

Nella definizione del linguaggio si è cercato, discostandosi il meno possibile dalle indicazioni del progetto I$_3$, di creare un'estensione del già esistente linguaggio ODL proposto dal gruppo di standardizzazione ODMG-93 (36).

Le principali caratteristiche del linguaggio ODLI$_3$ sono:

- capacità di rappresentare sorgenti strutturate (database relazionali, ad oggetti, e file system) e semistrutturate (XML), facendo riferimento

al concetto di classe e aggregazione, e offrendo una forte indipendenza dal modello originale dei dati.

- dichiarazione di regole di integrità (*if then rule*), definite sia sugli schemi locali (e magari da questi ricevute), che riferite allo schema globale, e quindi inserite dal progettista del mediatore;
- per ogni classe, il wrapper può indicare nome e tipo del sorgente di appartenenza;
- per le classi appartenenti ai sorgenti relazionali `e possibile definire le chiavi candidate ed eventuali foreign key;
- dichiarazione di regole di mediazione, o *mapping rules*, utilizzate per specificare l'accoppiamento tra i concetti globali e i concetti locali originali;
- il linguaggio supporta la definizione di grandezze locali e di grandezze globali;
- dichiarazione di relazioni terminologiche, che permettono di specificare relazioni di sinonimia (SYN), ipernimia (BT), iponomia (NT) e relazione associativa (RT) tra due tipi.

Utilizzando questo linguaggio il *wrapper* compie la traduzione delle classi da integrare e la fornisce al mediatore. C'è da dire che non sempre lo schema locale ricevuto dal mediatore rappresenti l'intera sorgente, bensì ne descrive il sottoinsieme di informazioni visibili da un utente del mediatore, esterno quindi alla sorgente stessa.

3.4.3 Un esempio di descrizione ODLI$_3$

A scopo esplicativo si porterà un esempio di descrizione in linguaggio ODLI$_3$ di tre sorgenti diverse (37):

1. UNIVERSITY, una sorgente relazionale che contiene informazioni sugli studenti e lo staff di ricerca di un'università;

2. COMPUTER_SCIENCE, una sorgente semistrutturata che contiene informazioni sulla facoltà di informatica della università di cui al punto 1.

3. TAX_POSITION rappresentata da un file in cui è descritta la posizione fiscale di ogni studente.

La descrizione di queste sorgenti in linguaggio *ODLI3* è la seguente:

UNIVERSITY source:

```
interface Research_Staff
        ( source relational University
          extent Research_Staff
         key name
         foreign_key dept_code, section_code )
   {     attribute string name;
         attribute string relation;
         attribute string e_mail;
         attribute integer dept_code;
         attribute integer section_code;  };

interface Department
       ( source relational University
         extent Department
         key code )
   {     attribute string dept_name;
         attribute integer dept_code;
         attribute integer budget;  };
interface Room
         ( source relational University
         extent Room
         key room_code )
{        attribute integer room_code;
         attribute integer seats_number;
         attribute string notes;  };
interface School_Member
         ( source relational University
         extent School_Member
```

```
key name )
{       attribute string name;
        attribute string faculty;
        attribute integer year; };
interface Section
        ( source relational University
        extent Section
        key section_name
        foreign_key room_code )
{       attribute string section_name;
        attribute integer section_number;
        attribute integer length;
        attribute integer room_code; };
```

COMPUTER_SCIENCE source:

```
interface CS_Person
        ( source object Computer_Science
        extent CS_Persons
        keys first_name, last_name )
{       attribute string first_name;
        attribute string last_name; };
interface Student: CS_Person
        ( source object Computer_Science
        extends Students )
{       attribute integer year;
        attribute set<Course> takes;
        attribute string rank; };
interface Location
        ( source object Computer_Science
        extent Locations
        keys city, street, county, number )
{       attribute string city;
        attribute string street;
        attribute string county;
        attribute integer number; };
interface Professon: CS_Person
        ( source object Computer_Science
        extent Professors )
{       attribute string title;
        attribute Office belongs_to;
```

```
        attribute string rank; };
interface Office
(       source object Computer_Science
        extent Offices
        key description
{       attribute string description;
        attribute Location address; };
interface Course
        ( source object Computer_Science
        extent Courses
        key course_name )
{       attribute string course_name;
        attribute Professor taught_by; };

Tax_Position source;

interface Univesity_Student
        ( source file Tax_Position
        extent University_Student
        key student_code )
{       attribute string name;
        attribute integer student_code;
        attribute string faculty_name;
        attribute integer tax_fee; };
```

3.5 Il processo di integrazione

L'integrazione delle sorgenti informative strutturate e semi-strutturate, è compiuta in modo semi-automatico, utilizzando degli schemi locali descritti in linguaggio ODLI3 (figura 3.5). Le fasi salienti del processo sono tre:

1. *Generazione del Thesaurus Comune*, con il supporto di ODB-Tool e di WordNet. In questa fase è identificato un Thesaurus comune di relazioni terminologiche. Tali relazioni, esprimono la conoscenza inter-schema su

sorgenti differenti, mentre le relazioni terminologiche sono ottenute in modo semi-automatico a partire dalle descrizioni degli schemi in ODLI3, attraverso l'analisi strutturale. Il tool ALA, che verrà descritto successivamente, utilizzando delle tecniche combinate di disambiguazione lessicale, arricchisce il Common Thesaurus di relazioni ricavate dall'analisi del contesto in cui i nomi delle classi sono inseriti.

2. *Generazione dei cluster di classi ODLI3* con il supporto dell'ambiente ARTEMISTool. Le relazioni terminologiche contenute nel Thesaurus sono utilizzate per valutare il livello di affinità tra le classi ODLI3 in modo da identificare le informazioni che devono essere integrate a livello globale. A tal fine ARTEMIS calcola i coefficienti che misurano il livello di affinità tra le classi, basandosi sia sui nomi delle stesse sia sugli attributi. Le classi con maggiore affinità sono raggruppate utilizzando tecniche di clustering (38)

3. *Costruzione dello Schema Globale*. I cluster delle classi ODLI3 affini, sono analizzati per costruire lo schema globale del Mediatore. Per ciascun cluster si definisce una classe globale che rappresenta tutte le classi locali riferite al cluster ed è caratterizzata dall'unione ragionata dei loro attributi e da una *mapping-table*. L'insieme delle classi globali definite, costituisce lo schema globale del Mediatore che sarà usato per porre le queries alle sorgenti locali intergrate.

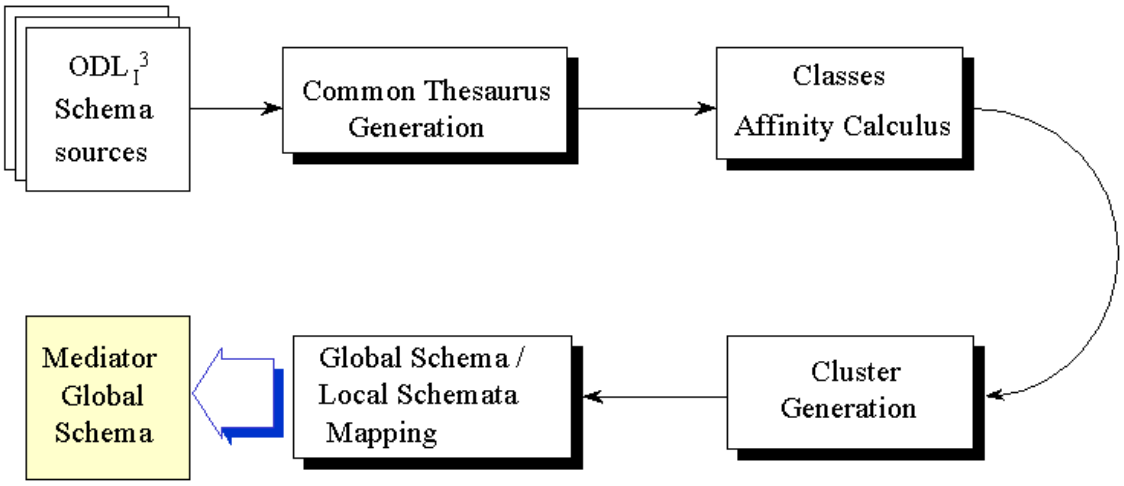

Figura 3.5 Schema generale del processo di integrazione

3.5.1 Generazione del Common Thesaurus e clustering

L'integrazione delle sorgenti realizzata dal *Global Schema Builder* si basa sull'individuazione di un'ontologia comune alle diverse fonti. In MOMIS un'ontologia di questo tipo è definita *Common Thesaurus* (Metadata Repository) che raccoglie le relazioni inter-schema ed intra-schema degli schemi delle sorgenti da integrare. Le relazioni contemplate sono:

Sinonimia SYN definita da due termini che possono essere scambiati nelle sorgenti senza modificare il concetto da rappresentare nello schema comune.

Specializzazione BT definita da due termini in cui il primo rappresenta una generalizzazione del secondo.

Aggregazione RT definita tramite due termini fra i quali esiste una relazione generica.

La costruzione del Common Thesaurus è un complesso processo incrementale in cui vengono man mano aggiunte relazioni semantiche intra-schema, relazioni lessicali, relazioni aggiunte dal progettista e relazioni intensionali inferite.

Il Common Thesaurus è man mano arricchito seguendo queste regole:

- Per le sorgenti a oggetti, possono essere agevolmente estratte le relazioni BT/NT dalla gerarchia di generalizzazione e le relazioni RT dalla gerarchia di aggregazione.
- Per le sorgenti relazionali è invece possibile ricavare relazioni RT e BT dall'analisi delle *foreign key*.
- L'estrazione di relazioni lessicali (nel senso di relazioni fra termini indipendentemente dalla classificazione fatta nel Paragrafo 2.3) può essere invece effettuata, come già accennato, grazie a preposti metodi di disambiguazione lessicale. La possibilità di estrarre importanti relazioni lessicali trova fondamento sul fatto che generalmente gli elementi che costituiscono gli schemi vengono denominati tramite espressioni significative,e a volte magari anche troppo sintetiche, del linguaggio naturale. Grazie all'interazione con la risorsa lessicale costituita da WordNet è possibile estrarre relazioni tra i nomi prescindendo dalle relazioni strutturali degli elementi cui si riferiscono. Ciò costituisce il limite di WordNet, che potrebbe essere superato grazie all'intervento di un operatore in grado di aggiungere altre relazioni importanti. In questa fase il modulo ARTEMIS può fornire importanti indicazioni circa l'affinità strutturale delle varie classi. ARTEMIS calcola l'affinità strutturale tramite un coefficiente detto *Global Affinity*, a sua volta combinazione lineare di altri due coefficienti, di cui il primo valuta l'affinità strutturale (*Structural Affinity*) delle classi, il secondo valuta l'affinità dei nomi delle classi e degli attributi (*Name affinity*). Questi tre passi devono subire un processo di validazione per garantire che vi sia compatibilità fra i tipi di dati dei termini messi in relazione, dopodiché vengono generati dei cluster di classi affini tra loro.

- L’estrazione di nuove relazioni semantiche intensionali. L'obiettivo è quello di produrre uno schema virtuale che contiene descrizioni strutturali delle sorgenti da integrare riorganizzate secondo quanto ricostruito nel Common Thesaurus. Si tratta quindi di uniformare la descrizione delle classi riconosciute simili, così come si deve fare in modo che per due classi tra le quali esista una relazione di specializzazione vi sia anche una relazione di specializzazione fra le due descrizioni e così via.

3.5.2 Generazione delle classi globali

L’ultimo passo che porta alla generazione dello schema globale,è quello che associa una classe globale ad ogni cluster generato da ARTEMIS nella fase precedente, ed è opera del SI-Designer. In ognuno di essi le classi sono rappresentate tramite struttura ad albero. Le foglie dell'albero rappresentano le classi locali. Si dice che due foglie, quindi due classi, contigue hanno un elevato grado di affinità. Lo schema globale, visibile dall’utente finale e sul quale l’utente stesso porrà le interrogazioni, è formato dall’insieme di tutte queste classi. Il processo di trasformazione di ogni cluster in una classe globale, `e articolato nelle seguenti fasi:

1. Unione degli attributi di tutte le classi appartenenti al cluster;
2. Fusione degli attributi ”simili”.

Questa cosiddetta ”Unione ragionata” degli attributi locali, `e un’operazione importante e delicata: importante perchè attraverso lo schema di attributi visibile all’utente, si deve dare a quest’ultimo la possibilità di porre queries semplici ma espressive; delicata perchè non è immediato stabilire quali attributi debbano essere collassati in uno solo.

Le uniche informazioni che il sistema ha in merito ai rapporti tra attributi appartenenti a diverse classi (o relazioni), sono quelle nemorizzate nel Thesaurus comune generato nella prima fase

L'attività di SI-Designer si conclude con la generazione di una mapping-table per ogni classe globale.

3.6 Il WordNet Editor

Abbiamo precedentemente specificato che un'importante fase della generazione di uno schema globale è costituita dai meccanismi di integrazione e informazione semantica. È stato anche ricordato che , nel sistema MOMIS, l'estrazione delle relazioni lessicali, indispensabili all'arricchimento del Common Thesaurus, è effettuata grazie all'interazione con il database lessicale WordNet. WordNet è stato perciò inserito all'interno del progetto MOMIS in un database relazionale denminato *momiswn*. Si propone ora, al fine di chiarire le modalità con cui è possibile reperire le relazioni di diverso tipo tra i synset, la struttura di *momiswn.* Lo schema relazionale è riportato in figura 3.6, ed emergono le seguenti informazioni principali:

- ***wn_synset***: contiene essenzialmente le informazioni sui synset. I campi principali sono l'identificatore *wn_synset_id,* che identifica univocamente un synset e la sua *gloss*. Il campo *syntactic_category* permette la distinzione tra nomi,verbi, aggettivi.
- ***wn_lemma:*** contiene tutti i lemmi presenti in WN e per ciascuno indica la categoria sintattica di appartenenza.
- ***wn_lemma_synset:*** associa ciascun lemma ai relativi possibili synset.
- ***wn_relationship_type****:* contiene i tipi di relazioni previsti nella versione di iniziale (1.6) di WN.

- ***wn_relationship_type_new:*** contiene i nuovi tipi di relazioni inserite all'interno della versione 2.0 di WN, comprensive di quelle della versione precedente.
- ***wn_relationship****:* contiene tutte le relazioni fra i synset di WN. Tali relazioni sono identificate principalmente dal *wn_source_synset_id*, che rappresenta il synset d'origine, e *wn_target_synset_id*, che rappresenta il synset correlato al synset d'origine da una relazione di tipo *wn_relationship_type_id*

```
WN_EXTENDER (wn_extender_id, name, description)
 AK: name

WN_SYNSET (wn_synset_id, offset,syntactic_category,
 word_cnt, gloss, wn_extender_id)
 FK: wn_extender_id references wn_extender

WN_LEMMA (wn_lemma_id, lemma, syntactic_category,
 sense_cnt, wn_extender_id)
 AK: (lemma, syntactic_category)
 FK: wn_extender_id references wn_extender

WN_LEMMA_SYNSET (wn_lemma_synset_id, wn_synset_id,
wn_lemma_id, lemma_number, sense_number, wn_extender_id)
 AK: (wn_lemma_id, sense_number ),
 (wn_synset_id, lemma_number)
  FK: wn_extender_id references wn_extender
 wn_synset_id references wn_synset
 wn_lemma_id references wn_lemma

WN_RELATIONSHIP (wn_relationship_id, wn_source_synset_id,
wn_target_synset_id, wn_source_lemma_number,
wn_target_lemma_number, wn_relationship_type_id,
wn_extender_id)
 FK: wn_extender_id references wn_extender
 FK: wn_source_lemma_number references wn_lemma
 FK: wn_target_lemma_number references wn_lemma
 FK: wn_source_synset_id references wn_synset
 FK: wn_target_synset_id references wn_synset
 FK: wn_relationship_type_id references
 wn_relationship_type

WN_RELATIONSHIP_TYPE (wn_relationship_type_id, symbol,
description, reflex)
 AK: symbol

WN_REVERSE_INDEX (wn_reverse_index_id,
term, wn_synset_id_list)
 AK: term
```

Figura 3.6 - Schema relazionale della struttura informativa di WordNet all'interno di MOMIS

L'obiettivo dell'estrazione di relazioni lessicali è, come detto, scoprire le affinità tra le classi appartenenti a differenti sorgenti di dati.

Tale obiettivo si realizza, in una fase preliminare, associando ad ogni termine il suo senso corretto corrispondente in WordNet. D'ora in poi ci si riferirà a questo processo di *mapping*, come alla *fase di annotazione delle sorgenti di dati.*

Tale fase di annotazione è attualmente eseguita da un *designer* che, attraverso l'interazione con l'interfaccia MOMIS- WordNet¸ assegna manualmente ad ogni elemento dello schema il senso ritenuto corretto in base al contesto in cui è inserito

La fase di scelta di uno o più significati da attribuire ad ogni termine, viene eseguita in due passi:

- *Scelta del termine*: durante questa prima fase il *WordNet morphologic processor* viene in aiuto al designer realizzando lo *stem* del termine. Ciò significa ricondurre un termine alla sua *base form*, quindi "depurata"dagli eventuali plurali e dai suffissi, che sarà poi cercata automaticamente in WordNet restituendo i synset associati.

  C'è anche la possibilità di aggiungere una nuova *base form* nel caso in cui non sia stata trovata un'effettiva corrispondenza in WordNet (cosa che avviene di frequente per i termini composti)
- *Scelta del significato*: il designer può scegliere di associare a ciascun elemento, zero, uno o più synset.

Una delle limitazioni principali di tale fase di annotazione, oltre ad essere completamente manuale, è quella di non consentire al *designer* di modificare il database inserendo nuovi lemmi, synset o relazioni.

Nel 2002 Veronica Guidetti in (39), integra all'interno di WordNet il componente *WordNet Editor*, il quale essenzialmente rappresenta una GUI,

che sfrutta una libreria Java e rende possibile l’estensione del database di WordNet all’interno di MOMIS.

Con WordNet Editor il designer ha l’opportunità di poter estendere WordNet allo scopo di colmare le sue lacune già evidenziate nel paragrafo 1.8 . Attraverso tale editor, è possibile, infatti, inserire nuovi lemmi, nuovi synset, nuove relazioni . Per consentire ciò, il database lessicale di MOMIS, è stato modificato aggiungendo la tabella *wn_extender* (figura 2.4), la quale tiene traccia delle informazioni riguardanti le modifiche effettuate sul database originale.

3.7 Migliorare il processo di integrazione attraverso l’annotazione

Risulta abbastanza chiaro, a questo punto, il ruolo giocato dall’annotazione all’interno del processo di integrazione delle informazioni.

Infatti, come descritto dalle sezioni precedenti, partendo dalla semantica associata agli elementi degli schemi appartenenti alle diverse sorgenti, è possibile scoprire le cosiddette *mappe concettuali* tra gli stessi elementi.

L’annotazione lessicale sembra essere un processo critico per sviluppare metodi veloci di *ontology learning e matching*, perciò, non dovrebbe sorprendere che un gran numero di sistemi includa alcune risorse lessicali (prevalentemente WordNet) come componente, e le utilizzi in qualche passo intermedio per annotare elementi di schemi e classi di ontologie grazie alle conoscenze lessicali. E non è una sorpresa il fatto che un sistema come MOMIS integri dei componenti preposti a questo oneroso compito. La nostra attenzione si focalizzerà proprio su questi tools, primo tra tutti ALA, uno strumento di disambiguazione che adotta un approccio di tipo

probabilistico alla disambiguazione, che si descriverà dettagliatamente nel prossimo capitolo, e all'interno del quale l'algoritmo sviluppato in questa tesi è stato inserito.

Capitolo 4

Il tool ALA di MOMIS: un approccio combinato alla disambiguazione

4.1 La disambiguazione lessicale applicata ai sistemi di integrazione.

Si presenteranno adesso nello specifico le problematiche che sorgono nell'applicare un processo di disambiguazione lessicale specificatamente ad una sorgente dati, piuttosto che a una sorgente testuale. Si farà riferimento, cioè, al processo precedentemente definito come *annotazione semantica* delle sorgenti di informazione.

È stato già detto che il sistema MOMIS nasce per far fronte all'esigenza di ottenere un'integrazione intelligente delle informazioni provenienti da sorgenti eterogenee e di offrirne una visione strutturata e globale che mascheri le differenze interne presenti tra le rispettive rappresentazioni logiche dei dati.

Purtroppo non sono poche le difficoltà che sorgono nel creare un sistema di integrazione e mediazione che sia affidabile, flessibile, modulare e capace di interagire con altri sistemi esistenti. Nello specifico, i sistemi di integrazione dei dati, tra cui ovviamente lo stesso MOMIS, devono far fronte a difficoltà ,classificabili come *incertezze*, che in (40) vengono individuate e inquadrate in 3 livelli:

1. Incertezze relative al *mapping* semantico tra le sorgenti di dati e lo schema globale.

2. Incertezze relative alle parole chiave da utilizzare nelle *queries.*
3. Incertezze relative alle sorgenti che possono contenere dati ambigui e non precisi.

Un potente mezzo per delineare un corretto *mapping* consiste nella comprensione dei significati esatti associati ai nomi che denotano gli elementi degli schemi che vengono analizzati (41). Nel capitolo 2 si è presentato il problema di disambiguazione lessicale testuale come uno dei problemi più complessi nel campo della linguistica computazionale. Il discorso non cambia se focalizziamo l'attenzione sulla disambiguazione di una sorgente dati strutturata o semi-strutturata.

I termini cui assegnare il senso ritenuto più corretto, in questo caso, sono rappresentati da etichette caratterizzanti una classe o un attributo, e di certo ciò non agevola il processo di disambiguazione. In particolare, come riportato in (42), oltre ai problemi classici riportati nel capitolo 2, altre difficoltà insorgono per i seguenti motivi:

- *Termini composti :* gli schemi e le ontologie spesso sono etichettate con espressioni nominali composte, come *full professor* , *football team*, etc. I termini composti però non compaiono in alcun database lessicale, a meno che non formino un composto stabile (*station wagon, computer science*). La loro annotazione risulta perciò più complicata, dal momento che è difficile associare un significato alla relazione tra i termini in un termine composto.
- *Acronimi e abbreviazioni:* gli schemi e le ontologie sono spesso etichettate con parole il cui significato non è sempre evidente e può essere mal interpretato; in tal caso è necessario raccogliere le parole

in una tabella di riferimento nella quale possano esplicitare il loro senso corretto.

- *Integrazione:* un modello/linguaggio standard per descrivere i database lessicali sappiamo che non esiste. Conseguentemente, risulta difficile integrare diverse risorse lessicali per ampliare la base di conoscenza cui fa riferimento la maggior parte degli algoritmi *knowledge-based*.

Considerando poi che l'annotazione manuale, oltre ad essere un processo noioso e *time-consuming*, incappa spesso in errori o interpretazioni scorrette quando tratta con sorgenti di dati molto estese e non esplorabili nella loro totalità , si è reso necessario lo sviluppo di tecniche di annotazione automatica.

4.2 MOMIS : approcci combinati di disambiguazione

All'interno del sistema MOMIS, pertanto, sono stati sviluppati e testati nel tempo diversi metodi che, nel tentativo di offrire una soluzione alle incertezze di livello 1, limitassero l'incidenza delle sopraccitate difficoltà.
I risultati di questi test hanno portato alla consapevolezza che un approccio che preveda la combinazione di più algoritmi supera di gran lunga le prestazioni conseguite dall'utilizzo di un singolo algoritmo WSD. Tra i metodi sviluppati in questi anni molti si muovono proprio in questa direzione. Il metodo semi-automatico MELIS (43), ad esempio, sfrutta le precedenti annotazioni manuali effettuate su un subset di elementi di una sorgente dati per produrre nuove annotazioni degli elementi rimanenti. Il metodo CWSD (44) combina insieme due differenti algoritmi che sfruttano uno la conoscenza strutturale di una sorgente e l'altro la conoscenza dei domini cui appartengono gli elementi della sorgente.

Il PWSD (42) utilizza un approccio probabilistico. Basandosi, infatti, sulla assunzione che un termine non ha necessariamente un solo ed esclusivo senso corretto, combina differenti algoritmi assegnando ad ogni annotazione pervenuta un valore di probabilità che ne riveli l'affidabilità.
Infine è stata integrata in MOMIS una GUI, col fine ultimo di offrire uno strumento utile all'utente inesperto, chiamata ALA (**A**utomatic **L**exical **A**nnotator), che risulta indipendente dagli algoritmi e dagli operatori implementati e permette al programmatore di aggiungere nuovi algoritmi senza ricompilare il codice sorgente. L'attenzione si focalizzerà nel paragrafo 4.3 sulla descrizione del sopraccitato metodo CWSD, in quanto si distingue tra gli altri per bassa dipendenza dall'intervento umano, e nel paragrafo 4.4 su ALA, che oltre a sfruttare l'approccio probabilistico ed essere il tool più "completo", è stato utilizzato in questa tesi per lo sviluppo dell'algoritmo di TUCUXI.

4.3 Metodo CWSD

Come è stato specificato nel paragrafo precedente, la combinazione di metodi differenti rappresenta una effettiva possibilità di incremento delle prestazioni nell'affrontare il problema della disambiguazione. L'idea di combinare differenti metodi risolutivi non è nuova, ma è stata sempre usata nella maggiorparte degli approcci alla disambiguazione lessicale presenti in letteratura (45), (46).
WordNet Domains, descritta ampiamente nel paragrafo precedente, ha costituito un'utile risorsa per la WSD, ed è stata utilizzata in più progetti che utilizzano algoritmi combinati tra loro. A differenza però della maggiorparte dei tool di WSD, il metodo CWSD non ha bisogno di annotazioni iniziali prima di dare il via al processo di disambiguazione.

Tale metodo, inoltre, è stato ideato è inserito all'interno del mediatore MOMIS (37) (32), per sfruttare, piuttosto che le tradizionali sorgenti testuali, la struttura delle sorgenti e la conoscenza lessicale derivata dagli elementi degli schemi (denominati d'ora in poi termini).

Effettuando questa integrazione è stato possibile alleggerire l'utente del pesante carico di annotare manualmente tutti i termini delle sorgenti da integrare.

CWSD combina un'algoritmo di tipo *structural*, che, come accennato precedentemente, utilizza le relazioni strutturali estratte dagli schemi delle sorgenti, con un algoritmo LKB basato sui *domini* di WordNet Domains, che rifinisce l'annotazione sfruttando per l'appunto una risorsa lessicale.

Al fine di disambiguare il senso di un termine passibile di diverse interpretazioni, entrambi gli algoritmi ricevono in input e lavorano in un *contesto*. In accordo con (47), molti algoritmi in letteratura rappresentano il contesto come una *bag of words*, un'insieme di parole che devono essere disambiguate, e a volte inseriscono nel contesto l'informazione relativa alla posizione relativa che ogni parola occupa all'interno del testo.

Altri approcci (48) invece considerano una *window of contest* attorno ad ogni termine target, e inviano tutte i termini appartenenti a questa finestra come input per l'algoritmo di disambiguazione.

In CSWD il contesto è composto da un set di termini (nomi di classi e di attributi) da disambiguare e un set di relazioni strutturali fra i termini incluse in un Thesaurus Comune (CT), come mostrato in figura.

Il CT è un set di relazioni $ODLI_3$ che descrivono le relazioni inter e intra-schema presenti tra gli elementi di un set di schemi sorgenti, esattamente uguale a quello utilizzato da ALA.

Il contesto di defalut per un sistema di integrazione dei dati è dato dalle sorgenti dati da integrare e da dalle relazioni locali ODLI3 delle sorgenti dati locali.

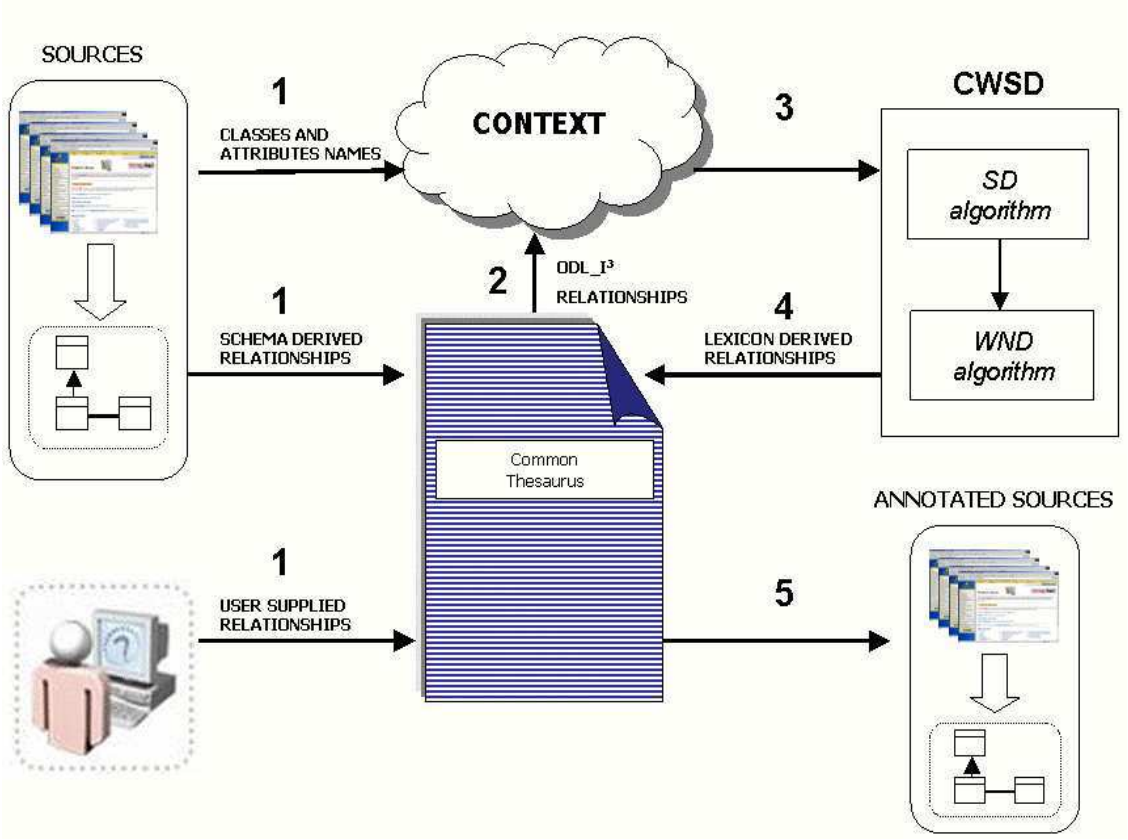

Figura 4.1 - Annotazione automatica di una sorgente dati locale tramite CWSD

4.3.1 L'algoritmo Strucural Disambiguation

Come precedentemente specificato, l'algoritmo SD sfrutta le relazioni di una sorgente dati per apportare nuove relazioni al CT grazie all'interazione con un database lessicale. Come descritto in (32) le seguenti relazioni sono estratte automaticamente:

- Per una relazione ***ISA*** tra due classi (*T1* **ISA** *T2*) si estrae una relazione di tipo

  BT : T2 ***BT*** T1 (ovvero T1 ***NT*** T2)

- Per una *foreign key* (*FK*) tra due relazioni :

  T1(A1,A2,...,AN) T2(B1,B2,....BN) FK: B1 REFERENCES T1(A1)

  si estrae la relazione di sinonimia A1 **SYN** B1

  e se B1 è una chiave della tabella T2 allora si estrae T1 **BT** T2 , altrimenti T1 **RT** T2.

  Le relazioni estratte sono immagazzinate nel CT e sono usati nel processo di disambiguazione in accordo con un database lessicale (in questo approccio si prende in considerazione WordNet).

SD cerca di trovare una relazione lessicale corrispondente quando individua un collegamento tra due termini. In pratica, se abbiamo una relazione , diretta o indiretta che sia, tra due parole, l'algoritmo cerca di trovare i sensi correlati semanticamente e

di annotare i termini con questi significati. Una catena di relazioni è così ottenuta navigando attraverso le relazioni del database lessicale.

La figura mostra un esempio di applicazione dell'algoritmo SD , ad esempio, una parte dei primi tre livelli del sottoalbero *society* nella directory di Google. Innanzi tutto tutte le relazioni di tipo **ISA** nello schema sono estratte dalla sorgente e inserite nel CT come relazioni NT, dopodichè SD trova le corrispondenti relazioni di *iponimia* in WordNet. Le annotazioni generate utilizzando SD arricchiscono il CT di nuove relazioni ODLI 3 (tutte le relazioni lessico-derivate mostrate in figura).

Facendo uso, infine, di ODB-Tool (un componente del sistema MOMIS), il CT deduce nuove relazioni. La figura 3 mostra alcune relazioni di iponimia recuperate attraverso WordNet, e i corrispondenti synset selezionati. In particolare, per i termini *religion* e *Taoism* SD sceglie i due synset corretti , pochè due differenti relazioni di iponimia esistono tra i termini.

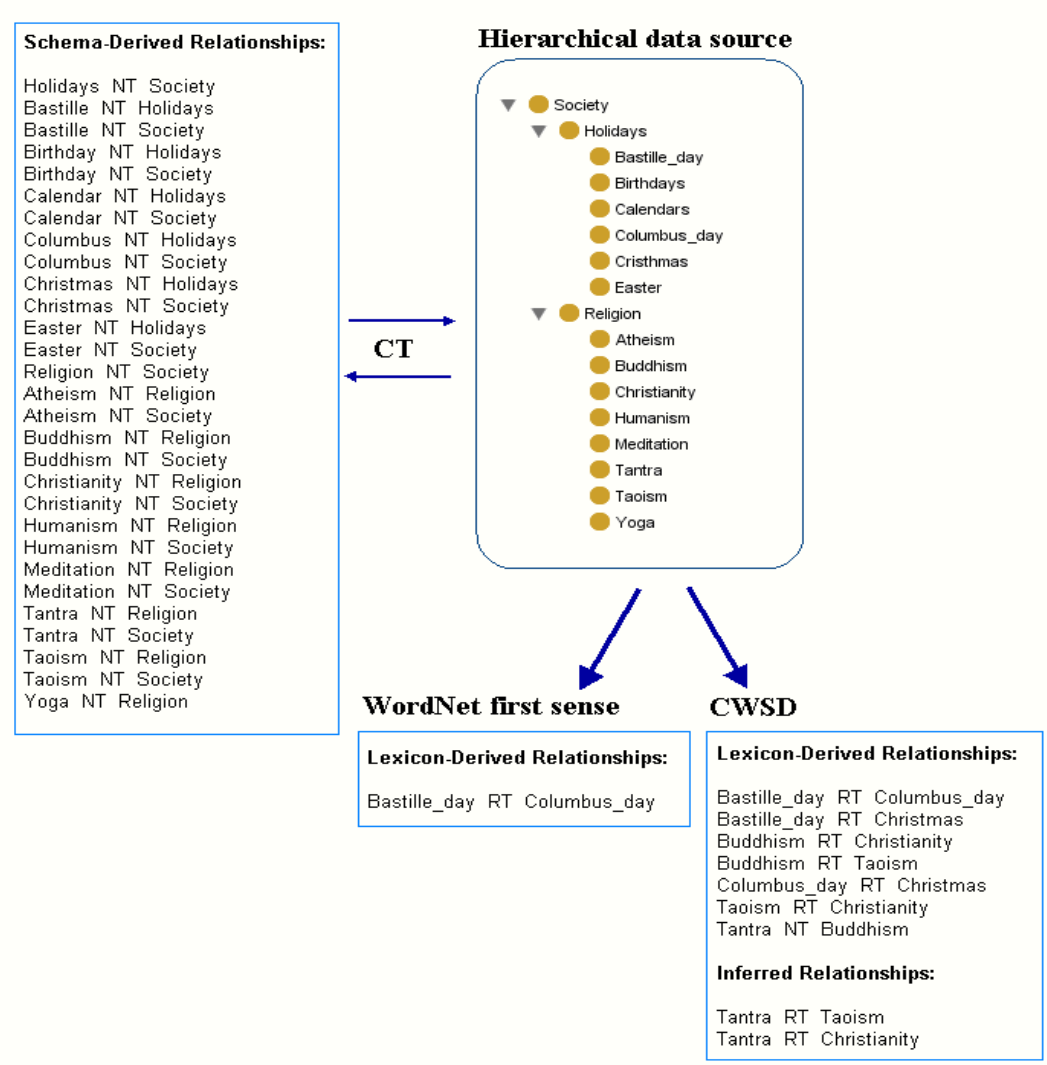

Figura 4.2 Arricchimento del CT attraverso le relazioni estratte tramite CWSD

Ricapitolando, per tuttle le relazioni di tipo NT estratte, SD trova la corrispondente relazione NT di WordNet. La relazione di equivalenza (SYN) è allo stesso modo utilizzata per estrarre la corrispondente relazione di sinonimia all'interno del database lessicale. La relazione di tipo RT è usata per individuare infine relazioni di olonimia e meronimia nel database lessicale.Infine, quando non è possibile individuare una relazione diretta tra due significati, l'algoritmo naviga attraverso WordNet alla ricerca di un percorso di relazioni lessicali che connetta i due significati. A scopo esplicativo è riportato lo pseudo codice relativo all'algoritmo cosi come comapre in (42)

Algorithm 1 Structural disambiguation algorithm

Input: *WordNet lexical Database and its extensions if any*
$T = [t_i\ i = 1\ldots Ncont]$ *the set of the terms to be disambiguated;*
$R = [r_k\ k = 1\ldots Nrel]$ *the set of the structural relationships linking two different terms* t_i *and* t_j*;*
Variables:
$S_i = [s_{iy}\ y = 1\ldots Nsyn_i]$ *the set of all possible synset related to the term* t_i*;*
$Fil = [f_{yw}\ y = 1\ldots Nsyn_i; w = 1\ldots Nf_il]$ *the set of synset linking by a chain of hypernym relationships*
of length l to one of the synset of Si;
$ANNOT_i = [syniz\ z = 1\ldots Ncsyni]$ *the set of synset chosen by the algorithm to disambiguate the term*
t_i*;*
for all r_{ij} *of* R that link two terms t_i, t_j *of* T do
  if r_{ij} is a BT relationship **then**
    determine *Si* and *Sj* ;
    initialize $l = 1$ and *annotation = false*;

    repeat
      determine F_il;
      if F_il not empty **then**
        for all s_{jz} in *Sj* do
          if exist a $f_{yw} = s_{jz}$ where s_{jz} *is of Sj* and f_{yw} *is of F* F_il then
          insert the synset s_{iy} in $ANNOT_i$;
          insert the synset s_{jz} in $ANNOT_j$;
          set *annotation*=true;
          end if
        end for
      end if
      set $l = l + 1$;
    until (*annotation = true*) or (no more hypernyms)
  end if
  if r_{ij} is a SYN relationship **then**
    if $t_i \neq t_j$ **then**
      set $ANNOT_i = ANNOT_j = S_i \setminus S_j$;
    end if
  end if
end for
Output: *different set* $ANNOT_i$ *for each term* t_i *that have a structural relation in R.*

4.3.2 L'algoritmo WordNet Domains

WordNet Domains (6) (7) può essere considerato, come già precedentemente specificato, una versione estesa di WordNet (o una risorsa lessicale) , in cui i synset sono stati annotati con una o più etichette di dominio. L'informazione offerta dai domini è complementare alle altre già presenti in WordNet. D'altronde, i domini possono raggruppare i sensi di una stessa parola in un *cluster* tematico, che possiede l'importante proprietà di ridurre il livello di ambiguità dei termini polisemici.

WordNet Domains organizza circa duecento domini in una gerarchia, in cui ogni livello è fatto di codici che indicano lo stesso livello di specificità, come descritto in (8). Alcuni synset non appartengono a nessun dominio specifico, ma possono tuttavia compareire nella maggiorparte di loro. Per questa ragione è stata creata una label generica che include questi tipi di synset: synset generici (*man# - an adult male person*), synset *stop sense* (come colori, numeri, ecc)

L'algoritmo WND (si veda la figura) trae ispirazione dall'algoritmo *domain-based* proposto in (49). In primo luogo, si esaminano tutti i possibili synset connessi ad un termine, e si estraggono i relativi domini associati. Grazie a questa informazione si calcola la lista dei *prevalent domains* nel contesto prescelto. Poi si confronta questa lista di domini con quella associata ad ogni singolo termine da disambiguare. Per ogni termine si scelgono come synset corretti tutti i synset appartenenti ai domini prevalenti cosi come sono stati individuati al passo precedente.

In WordNet Domains c'è un particolare dominio chiamato *factotum* che è il dominio associato ai synset per i quali non è stato possibile individuare un dominio specifico, e, come descritto in (8), che rappresenta nella maggiorparte dei casi il dominio più frequente in un contesto.A differenza di quanto proposto in (49), si è scelti di usare il dominio *factotum* soltanto

quando nessun altro dominio tra i *prevalent domains* è collegato con i sensi di un termine. I risultati di WND dipendono dal contesto e dalla *configurazione* prescelta. La configurazione è il massimo numero di domini selezionati per la disambiguazione, la cui scelta, insieme a quella del contesto, è demandata all'utente finale.

Algorithm 2 WordNet Domains disambiguation algorithm

Input*: WordNet lexical Database and its extensions if any*
T = [ti i = 1…Ncont] the set of the terms to be disambiguated;
NMaxDOM the maximum number of domains we want to use in the algorithm.
Variables*:*
Si = [siy y = 1…Nsyni] the set of all possible synset related to the term ti;
Dj = [dk k = 1…Ndomj] the set of the possible domains related to the synset sj ;
DOM = [domx x = 1…Ndom] an ordered set of the domains related to the set of the terms T;
FreqDOM = [fx x = 1…Ndom] the corresponding set of the frequency of the domains related to the
set of the terms T;
ANNOTi = [syniz z = 1…Ncsyni] the set of synset chosen by the algorithm to disambiguate the term ti;

```
for all ti in T do
    for all sij in Si do
       for all djk in Dj do
          if djk is in DOM then
            increase the FreqDOMk;
          else
            insert the domain djk in DOM and set FreqDOMk = 1;
          end if
       end for
     end for
end for
for all ti in T do
    if ti is a monosemic term then
      ANNOTi = syniy ;
   else
      set annotation = false;
      for x = 1 to NMaxDOM do
          for all sij in Si do
             if dx is contained in Dj then
                 insert the synset syniy in the ANNOTi;
                set annotation = true;
             end if
          end for
          if annotation = true then
            BREAK the cycle FOR;
          end if
```

```
      end for
   end if
end for
for all ti in T do
   if ANNOTi is empty then
      for all sij in Si do
         if factotum is contained in Dj then
            insert the synset syniy in the ANNOTi;
         end if
      end for
   end if
end for
```

Output: *a list DOM of the more frequent domains, and a set ANNOTi of sysnset that disambiguate each terms in T.*

La disponibilità di WorNet Domains ha consentito di tralasciare un'analisi orientata ai domini delle sorgenti di dati strutturate e semi strutturate e di implementare un effettivol algoritmo WSD basato sull'informazione di dominio.

4.3.3 Valutazione di CWSD: un'analisi comparativa

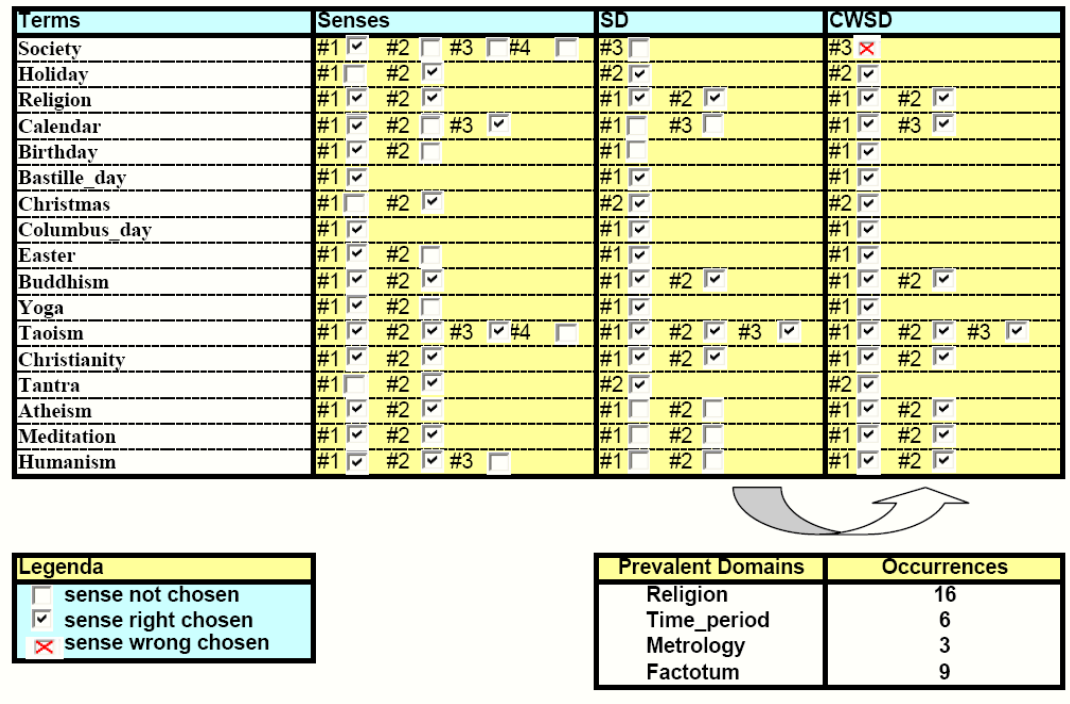

Figura 4.3 Analisi comparativa delle prestazioni di CWSD su una sorgente dati gerarchica

Nella figura precedente sono riportati i risultati finali dell'applicazione di CWSD alla sorgente dati gerarchica presentata precedentemente.

In particolare vengono confrontati i risultati ottenuti dal CWSD con i risultati ottenuti utilizzando il solo algoritmo SD. Se disambiguiamo utilizzando soltanto SD, otteniamo annotazioni corrette solamente per alcuni termini, mentre grazie all'utilizzo combinato CWSD dei due algoritmi incrementiamo le prestazioni in due direzioni:

1. La disambiguazione dei termini è più accurata; la polisemia porta ad avere più di un solo synset corretto associato ad un termine, e grazie a CWSD possiamo assegnare a questi termini più di un senso;
2. CWSD arricchisce il CT di nuove relazioni: questo fatto è particolarmente importante per il processo di integrazione (come mostrato in figura). L'unico termine annotato in maniera scorretta è *society*, poichè associato, attraverso l'algoritmo WND, al dominio *factotum*, ma il suo senso corretto è associato al dominio *anthropology* che non è presente nei *prevalent domains.*

4.3.4 Valutazione di CWSD: risultati sperimentali

Il metodo CWSD è stato sperimentato su un sorgente dati reale. In particolare sono stati selezionati i primi tre livelli di un sottoalbero delle directories di Yahoo e di Google (rispettivamente *society and culture* and *society*), che ammontano a 327 categorie per Yahoo e 408 per Google.

Nella tabella 4 sono confrontati i risultati della disambiguazione del sottoalbero delle directory appena descritte ottenuti con i diversi algoritmi: solo SD, solo WNS,solo CWSD e solo MELIS.

I risultati dell'annotazione sono stati valutati in termini di recall (numero di annotazioni corrette sul numero totale di annotazioni effettuate, ad esempio, una per ogni categoria, come definito in un golden standard) a

precision (numero di annotazioni corrette recuperate sul numero totale di annotazioni riuscite). Nella tabella i valori di recall e precision sono ottenuti considerando un elemento come propriamente annotato se l'annotazione data dall'utente è contenuta nel set di annotazioni calcolato dall'approccio WSD esaminato.

WSD approach	Recall	Precision
SD	8.00%	97.00%
WND	66.62%	69.97%
CWSD	74.18%	74.18%
MELIS	53.03%	58.85%

Tabella 4- Risultati sperimentali dei diversi algoritmi sulle sotto-directory di Google e Yahoo

L'applicazione dell'algoritmo SD sulle web directories ha sfruttato le 792 relazioni **ISA** e ha consentito di ottenere 60 annotazioni di cui 58 corrette, in modo tale da conseguire un valore molto elevato di precision ma una percentuale molto bassa di recall. Una probabile causa può essere individuata, per esperienza, dalla scarsità di relazioni presenti in WordNet.
I risultati evidenziano che un algoritmo combinato supera in performance i singoli algoritmi di cui è composto.

4.4 Il tool per l'annotazione automatica ALA

Il tool ALA, come detto, gestisce le incertezze del primo livello, quindi le incertezze relative alla giusta interpretazione del significato dei lemmi associati agli schemi costituenti la sorgente. Utilizzando una visione probabilistica dei significati associati alla *label* di uno schema, ALA esegue un'annotazione lessicale automatica dei lemmi che compaiono nella

descrizione logica dello schema. Ciò consente di trovare le probabili relazioni lessicali presenti tra sorgenti di informazione eterogenee, che vengono quindi raccolte in un **Probabilistic Common Thesaurus** (**PCT**). L'idea di un'annotazione di tipo probabilistico è una novità nel campo dell'integrazione dei dati, nonostante sia un approccio ben noto ai processi di disambiguazione del testo (41). Nel caso specifico il tool, per conseguire l'obiettivo di eliminare le ambiguità dai lemmi delle sorgenti analizzate, fa uso di una **LKB** (Lexical Knowledge Base), ossia di una risorsa lessicale , nel caso specifico WordNet, che codifica al suo interno le relazioni semantiche e lessicali esistenti tra i lemmi e indispensabili al processo di disambiguazione.

L'annotazione lessicale, dunque, consiste nell'assegnare ad ogni elemento di uno schema un significato specifico (un synset di WordNet) o più significati possibili. L'annotazione probabilistica aggiunge poi un valore di probabilità che indica l'affidabilità dell'annotazione.

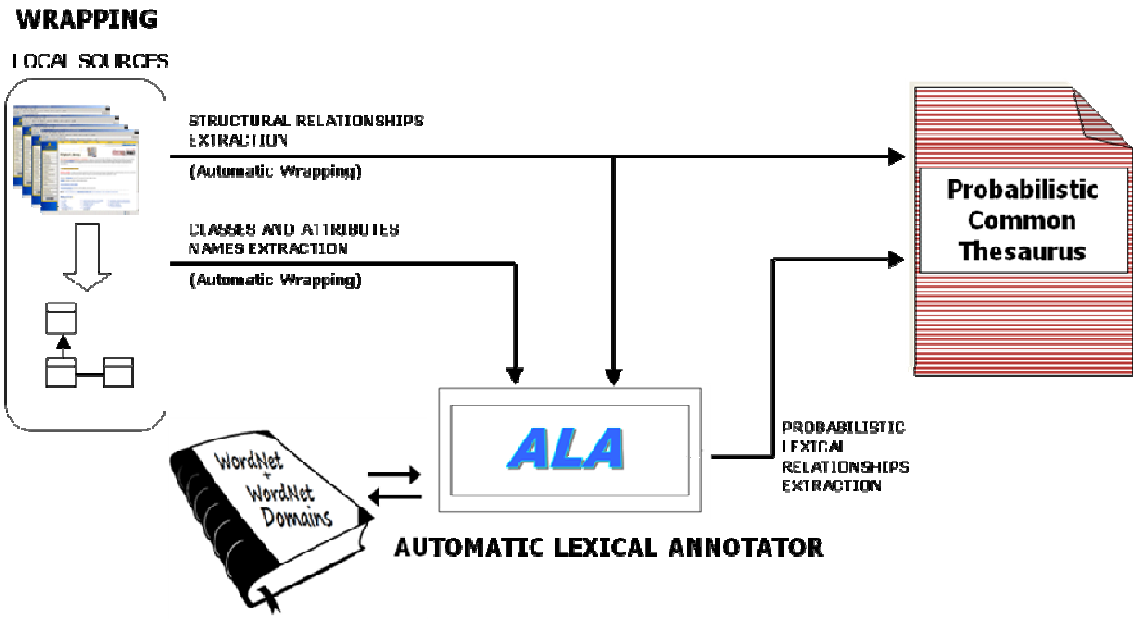

Figura 4.4 - Annotazione automatica di sorgenti locali ottenuta tramite ALA

Diverse sono le ragioni che hanno portato alla realizzazione di un'annotazione di tipo probabilistico:

- *Annotazioni multiple* : data una *label* di uno schema, un algoritmo di WSD può associare un set di significati non necessariamente ortogonali o mutuamente esclusivi.
- *Combinazioni di tecniche differenti:* un insieme di algoritmi che presentano caratteristiche e metodi diversi per la risoluzione del problema della disambiguazione possono completarsi a vicenda, coprendo le falle che ogni singolo approccio inevitabilmente crea e garantendo la massima accuratezza nell'annotazione dei termini.
- *Incertezza:* l'utilizzo di differenti algoritmi WSD conduce a una incertezza epistemica (ossia l'incertezza che risulta dalla carenza di conoscenze relative a un certo sistema) , dal momento che non sempre un algoritmo riesce a disambiguare ogni termine, e , per di più, un certo algoritmo può essere più appropriato in alcune situazioni rispetto ad un altro, causando una sorta di inconsistenza o di poca obbiettività nel valutarne il comportamento e l'attendibilità complessiva.

Il tool ALA è stato sviluppato nell'ambito del progetto del sistema MOMIS (50), ma ciò non preclude un suo eventuale impiego al servizio di qualsiasi sistema di integrazione di dati o di mapping semantico.

4.4.1 Caratteristiche di ALA

ALA utilizza i *wrappers*, i moduli precedentemente descritti , per tradurre gli schemi di ogni sorgente, usualmente rappresentati logicamente da forme eterogenee quali DB relazionali, DB ad oggetti, dati semi-strutturati XML,

in una forma intermedia definita, come detto, da un linguaggio orientato agli oggetti e concepito all'interno del sistema MOMIS, l'$ODLI_3$. I wrappers estraggono automaticamente le relazioni strutturali dagli schemi e dagli ODB-Tools (51), inferendo nuove relazioni ottenute grazie all'analisi della chiusura transitiva di quelle appena estratte. Tali relazioni strutturali sono inserite nel PCT con un valore di probabilità uguale a 1. Le relazioni strutturali sono:

- BT_{EXT} : t1 BT_{EXT} t2 se extension(t2) $\subseteq$ extension(t1) (ISA, foreign key)
- SYN_{EXT} : t1 SYN_{EXT} t2 se extension(t1) = extension(t2).

ALA fornisce un set di algoritmi e di operatori per eseguire l'annotazione lessicale. Dal punto di vista scientifico di un progettista, ALA ha una struttura modulare, che può essere facilmente estesa, dal momento che l'implementazione di nuovi algoritmi o nuovi operatori è semplice ed intuitiva.

ALA, come detto, assegna a ciascun algoritmo un valore di attendibilità (il valore di default è quello fornito da una valutazione delle prestazioni dell'algoritmo ottenuta tramite un benchmark) .L'utente può scegliere tutti o un sottoinsieme di questi algoritmi e combinarne i risultati scegliendo l'operatore più opportuno.

L'operatore *Pipe* combina le annotazioni fornite dai diversi algoritmi in base all'ordine di esecuzione che l'utente ha preventivamente stabilito. Nella fattispecie l'operatore utilizza l'output del primo algoritmo presente nell'elenco, e per quei termini non ancora disambiguati esegue il secondo algoritmo e cosi via.

L'operatore *Parallel* ogni termine è disambiguato con il contributo di tutti gli algoritmi selezionati . L'output prodotto è successivamente combinato in base alla regola di combinazione di Dempster (52).
L'operatore *Threshold* infine filtra le annotazioni che presentano un livello di probabilità inferiore ad una certa soglia stabilita dall'utente.
A partire dall'insieme di annotazioni probabilistiche ALA calcola una relazione lessicale probabilistica tra due termini se esiste, all'interno del Thesaurus , una relazione tra i rispettivi significati.
Le relazioni lessicali sono definite sempre sulla base delle relazioni di WordNet

- *SYN*: definita tra due termini che siano sinonimi;
- *BT*: (Broader Term) definita tra due termini collegati da una relazione di ipernimia (la relazione opposta della *BT* è la *NT* (Narrower Term));
- *RT*: (Related Term) definita tra due termini collegati da una relazione di meronimia o di olonimia.

Il valore di probabilità assegnato ad una relazione lessicale dipende dal valore di probabilità dei significati dei termini presi in considerazione. Per esempio, grazie alla formula dell'intersezione di probabilità, il valore associato ad una relazione lessicale che congiunga i significati $t_{\#i}$ and $s_{\#j}$ dei termini *t* e *s* rispettivamente, è definita da:

$$P(t\#i, s\#j) = P(t\#i) * P(s\#j) \qquad (\mathbf{1})$$

Al momento sono 5 gli algoritmi di cui ALA dispone:

1. *Structural Disambiguation Algorithm:*descritto dettagliatamente nel paragrafo esamina i termini che sono correlati attraverso una relazione strutturale (cioè di tipo BT_{EXT} o SYN_{EXT}) ed effettua una ricerca all'interno del Thesaurus (WordNet nello specifico) delle relazioni lessicali presenti tra i significati associati a questi termini
2. *WordNet Domains Disambiguation Algorithm:* tenta di effettuare la disambiguazione lessicale analizzando le informazioni sui domini cui i termini appartengono fornite da WordNet Domains (6).
3. *Gloss Similarity* and
4. *Iterative Gloss Similarity Algorithm:* effettuano l'annotazione sfruttando delle tecniche di similarità tra stringhe
5. *WordNet firs sense Alghorithm:* Basandosi su criteri euristici esegue la disambiguazione dei lemmi attribuendone il significato più probabile (che in WordNet è rappresentato dal primo synset cui è associato un lemma) , ossia quello statisticamente più utilizzato.

4.4.2 ALA – un esempio di utilizzo

Consideriamo , come esempio , il termine "name". In WordNet troviamo 6 diversi significati per "name" (*name*#1, *name*#2, .., *name*#6). Supponiamo ora di dover combinare tre algoritmi WSD che producono output differenti (caso mostrato in figura 2.7): WSD1 che selezione il set di significati *{name*#1, *name*#2*}*, WSD2 che fornisce *name*#1 come senso corretto e WSD3 che non restituisce alcun risultato. Ciò che vogliamo ottenere e un punteggio di probabilità assegnato ad ogni possibile senso del termine considerato.

Come si nota dalla figura 3 i termini sorgenti sono automaticamente estratti e annotati grazie all'applicazione degli algoritmi WSD.

Nel caso A, l'operatore *pipe* è applicato seguendo un ordine manuale (WSD1, WSD2, WSD3). Per il termine "name" sono scelti entrambi i sensi *name*#1 e *name*#2. La probabilità associata all'annotazione *name*#1 e *name*#2 è uguale all'affidabilità dell'algoritmo WSD1 divisa su ogni senso.

.

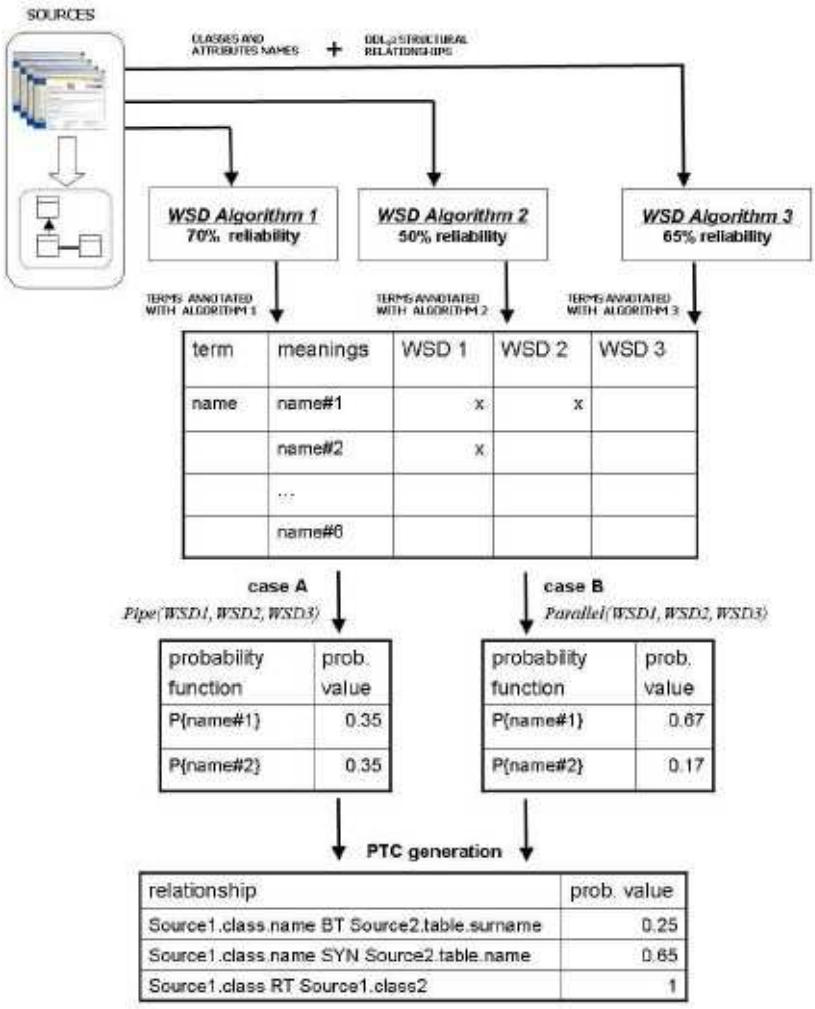

Figura 4.5 Processo di annotazione automatica di ALA: un esempio

L'operatore *parallel*, caso B, non considera gli algoritmi che non forniscono alcuna annotazione per un termine. Così ,in questo caso, la valutazione è effettuata combinando solo la funzione densità di probabilità di WSD1 e WSD2. Possiamo supporre che l'operatore *parallel* incrementa

la probabilità associata al senso *name*#1 perché entrambi gli algoritmi selezionano questo significato.

Dopo la fase di annotazione, sono estratte le relazioni lessicali, le quali vengono inserite nel PCT (la prima e la seconda relazione in figura) e sono aggiunte alle relazioni strutturali (la terza relazione in figura).

4.4.3 Gestire le annotazioni

Sviluppando il sistema ALA, ci si è posti di fronte al problema di come gestire le annotazioni riferite ad un elemento sorgente.

Si è pensato di mantenere una ricca rappresentazione semantica attraverso l'utilizzo di un *gruppo di annotazioni* (*annotation group*). Ogni *gruppo di annotazioni* è fornito di un valore di probabilità ed è composto da un set di annotazioni. Ogni annotazione indica un synset di WordNet e include un valore di probabilità.

Gli algoritmi WSD possono selezionare infatti più significati per annotare un termine; per gestire le annotazioni lessicali derivate da differenti WSD è di aiuto considerare le annotazioni fornite da ogni metodo all'interno di un gruppo di annotazioni caratterizzato dalla probabilità dell'algoritmo WSD

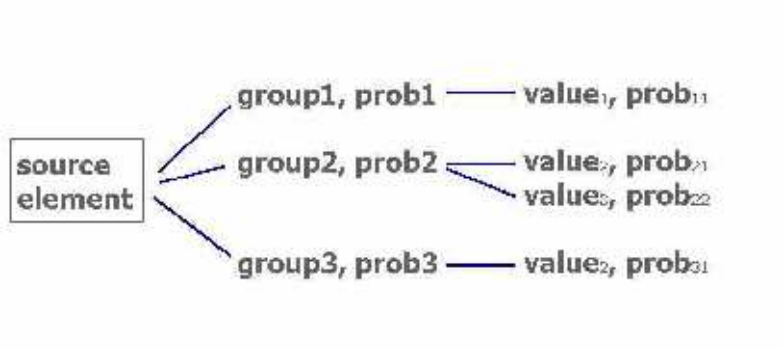

Figura 4.5 Processo di annotazione autamatica: un esempio di annotazione

I gruppi di annotazioni rappresentano il modo più semplice di rappresentare le annotazioni lessicali prodotte da ALA

Per esempio, in figura 4.6 un elemento sorgente è associato a 3 annotation groups. Ogni gruppo è caratterizzato da una probabilità assegnata al gruppo e una lista di annotazioni , in cui ogni annotazione ha un valore (un synset identifier o una tripla composta da un lemma, una syntactic category e un sense number) che può identificare un significato e una probabilità assegnata a questo valore.

Si noti che il valore può essere ripetuto in gruppi differenti, questo perché ogni gruppo definisce una annotazione specifica.

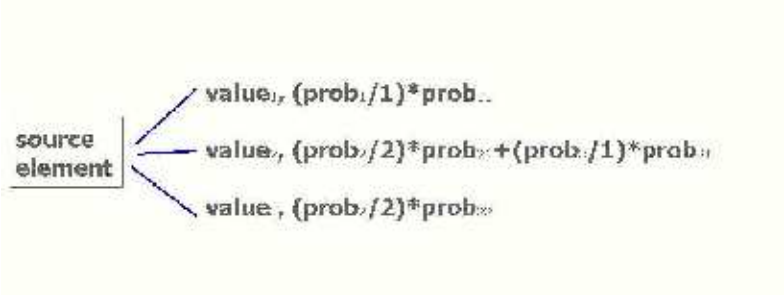

Figura 4.6 Processo di annotazione automatica : un esempio di annotation group

Mentre gli algoritmi WSD e ALA trattano con i gruppi di annotazioni, l'utente non ha modo di visualizzarli. Infatti, sebbene il numero dei synset possibili associati all'elemento sorgente è limitato (non possiamo avere più del numero totale dei possibili synset relativi ad ogni lemma di cui l'elemento sorgente è composto), non c'è un limite al numero di gruppi che un elemento sorgente può possedere. Per questa ragione è stata implementata una procedura che permetta di associare direttamente le annotazioni a un elemento sorgente (come era previsto prima dello sviluppo di ALA in MOMIS) . Ogni annotazione sarà arricchita con un punteggio di probabilità calcolato in base alla Dempster Shafer theory (52).

$$prob(value_i) = \sum_{j \in \boldsymbol{Groups}} \left(P_{\,group_j}(value_j) * \frac{P_{\,group_j}}{||group_j||} \right)$$

Importazione ed esportazione di annotazioni

Una volta che è stata effettuata l'annotazione di una sorgente (anche parziale) , il risultato può essere salvato in un file xml. Le annotazioni possono essere esportate e poi riusate in altre applicazioni o condivise da utenti diversi. Nel file le annotazioni lessicali sono salvate come gruppi di annotazioni.Importare le annotazioni da un file invece permette all'annotatore di sfruttarle per un'altra esecuzione.

4.4.4 Il pannello di controllo di ALA

Come mostrato in figura ALA consente di combinare diversi algoritmi di disambiguazione (mostrati sul lato sinistro), che l'utente potrà scegliere e per i quali potrà selezionare una delle modalità d'esecuzione viste nel paragrafo precedente (*pipe*, *parallel* e *formula*).

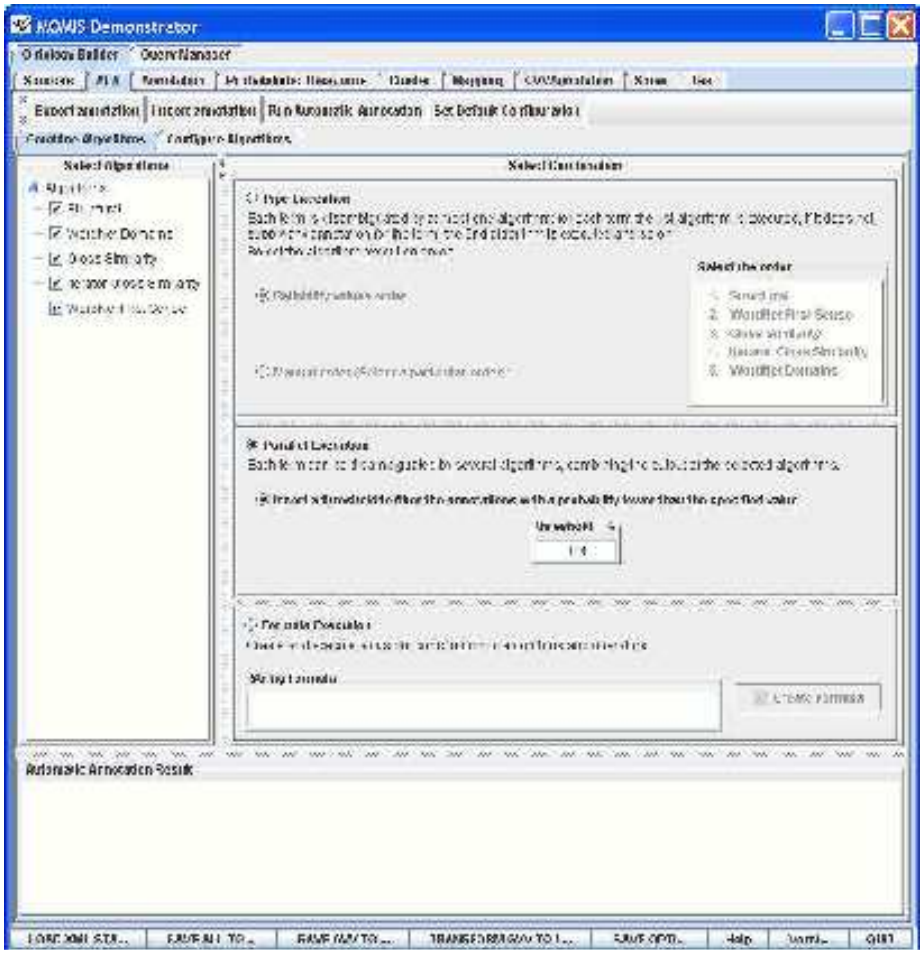

Figura 4.8 - Screenshot del pannello di controllo di ALA

4.4.5 Inserimento di un nuovo algoritmo in ALA

Alla luce delle considerazioni effettuate sui vantaggi apportati dall'esecuzione combinata di più algoritmi al processo di disambiguazione lessicale, e dal momento che la flessibilità di ALA consente una semplice estensione o integrazione di altri elementi, si è pensato di arricchire il set dei sopraccitati algoritmi con un ulteriore algoritmo di annotazione automatica , basato anch'esso su una LKB, che sfrutti direttamente le relazioni semantiche esistenti tra i termini da disambiguare. L'algoritmo presentato rappresenta una re-implementazione della seconda fase dell'algoritmo TUCUXI presentato nel paragrafo 2.5.1, e se ne fornirà una descrizione dettagliata nel capitolo seguente.

Capitolo 5

L’algoritmo TUCUXI

5.1 Introduzione

Il tool ALA di MOMIS, descritto nel capitolo precedente, allo stato attuale non ha ancora a disposizione, nel pool di algoritmi implementati, un algoritmo che utilizzi, come metodologia da adottare per il processo di disambiguazione, le relazioni presenti tra i termini da disambiguare. I due approcci KLB alla disambiguazione descritti nel capitolo 2, ossia l’algoritmo SSI e TUCUXI, sfruttano entrambi la proprietà lessicale di coesione, e dunque le relazioni esistenti tra lemmi appartenenti ad uno stesso contesto, per effettuare l’annotazione.

L’obiettivo di questa tesi è, per l’appunto, quello arricchire il set di algoritmi di ALA con uno che abbia queste caratteristiche. Tra i due algoritmi sopraccitati la scelta è ricaduta su quello di TUCUXI.

Come è stato ampiamente descritto nel capitolo 2, TUCUXI può essere diviso in tre moduli ben definiti e autonomi , il secondo dei quali ha il compito di disambiguare una lista di termini che viene passata come input. L’algoritmo che si propone rappresenta effettivamente una reimplementazione di questo modulo.

Di seguito si riporta il nuovo pseudo-codice che descrive le operazioni da effettuare per l’annotazione automatica.

Algorithm TUCUXI- Modificato

Input:
WordNet lexical Database and its Domains extensions
CW = [cw_i i = 1…k] *the set of the terms (candidate words) to be disambiguated;*

Variables:
S = [s_i i = 1…n] *the set of all synset belonging to all the terms to be disambiguated*
RS$_i$ = [rs_{ik} k = 1…h] *the list of all related synset of the generic synset* s_i
WS$_i$ = [ws_{ij} j = 1…t] *the set of all possible synset related to the term* cw_i*;*
bu$_{ij}$ = [cw_i ,ws_{ij}, *score*] *a triple formed by the association of one candidate word and one of its own possible meaning, reporting a cohesion score that indicates the strength of the couple* cw_i ,ws_{ij} *into disambiguation process*
BUlist=[bu_z z = 1…n] *a list including all the* bu_{ij}

for all cw_i in *CW* **do**
determine WS_i
for j=1 to t **do**
insert distinct synset $ws_{ij} \in WS_i$ in *S;*
initialize score = 0.0;
build bu_{ij} = [cw_i ,ws_{ij}, *score*];
insert bu_{ij} in the list *BUlist;*
end for
end for
for all bu_z in the *BUlist* **do**
determine RS_z of synset $ws_z \in bu_z$
for all rs_{zk} in RS_z **do**
if exists $s_i \in S$ *where* $rs_{zk} = s_i$ **then**
update cohesion score for bu_z according to strength of relationship with rs_{zk} and to the relative relationship path;
end if
end for
end for
for all cw_i in *CW* **do**
retain from *BUlist* those *bu* whose *cw* = cw_i having the MAX cohesion score;
delete the other ones
end for
Output: *the list of BU including only the best meaning for a candidate word*

5.2 Definizione delle variabili utilizzate

Come si nota dallo pseudo-codice l'algoritmo dovrebbe ricevere in input, oltre al database WordNet e alla sua estensione WordNet Domains, una struttura dati principale:

- l'insieme **CW** =[$cw_{1,}$ $cw_{2,}$ ……., cw_{k}] delle *candidate words* $\mathbf{cw_i}$, ossia i lemmi selezionati dal *POS parser* (nel nostro caso saranno soltanto elementi appartenennti alla categoria sintattica dei sostantivi) che potranno essere disambiguati. Mutuando una espressione utilizzata da R.Navigli e P.Velardi in (10), potremmo chiamare questo insieme il *contesto lessicale* in cui è inserito ogni lemma.

A partire da tale insieme l'algoritmo provvede a generare altre variabili:

- *k* insiemi $\mathbf{WS}_j$, ($j = 1……k$): tanti quante sono le *candidate words,* ognuno di essi rappresenta l'insieme dei synset associati ad un determinato lemma. In particolare il j-esimo insieme è riferito al lemma $\mathbf{cw_j}$ e avrà una struttura del tipo $\mathbf{WS}_j = [ws_{j1,}\ ws_{j2,}\ ……ws_{jt}]$ dove ogni synset $\mathbf{ws_{jk}}$ è un possibile significato da attribuire a $\mathbf{cw_j}$.

- un insieme **S** =[$s_{1,}$ $s_{2,}$ ……., s_n] costituito da tutti i synset $\mathbf{s_i}$ associati a tutti i lemmi da esaminare.

- *n* insiemi $\mathbf{RS}_i$ =[$rs_{i1,}$ $rs_{i2,}$ ……., rs_{ih}] . Ognuno di essi è calcolato a partire dal synset $\mathbf{ws_i}$ preso in esame dall'algoritmo, ed è costituito da tutti i synset $\mathbf{rs_{ij}}$ con cui il synset $\mathbf{ws_i}$ in esame presenti una relazione semantica di qualsiasi natura (iponimia, iperonimia, meronimia , olonimia , domini in comune etc.)

- *n* strutture dati denominate *basic unit* **bu**. La *basic unit* è una variabile che consente di associare ogni synset al lemma corrispondente, più un valore che indica la *forza* di tale associazione.

In particolare ogni basic unit **bu** è costituita dalla tripla (**cw**, **ws**$_j$, **score**), ossia il lemma che l'algoritmo sta esaminando, un suo j-esimo senso possibile **ws**$_j$, e un punteggio di coesione, calcolato durante l'esecuzione dell'algoritmo, che indica quanto è significativa l'associazione tra il lemma **cw** e il suo significato **ws**$_j$. Tale valore verrà utilizzato per selezionare quale, tra i synset **ws**$_j$, sarà il synset da attribuire al lemma **cw**

- Una lista **BUlist** contenente tutte le **bu** determinate, che serviranno successivamente per il processo di disambiguazione.

5.3 Procedimento logico dell'algoritmo

L' obiettivo specifico dell'algoritmo è quello di trovare il maggior numero di relazioni utili tra i synset associati ai lemmi da disambiguare, per poi selezionare, come verrà chiarito fra poco, quei synset maggiormente "inerenti" al contesto, ossia maggiormente "connessi" tra loro.

È possibile articolare il processo di disambiguazione in tre fasi:

1. Costruzione delle strutture dati necessarie all'algoritmo per individuare le relazioni presenti tra i synset appartenenti ai lemmi da disambiguare.
2. Analisi delle strutture dati appena create finalizzata al recupero delle relazioni e all'attribuzione del punteggio di coesione per ogni basic unit
3. Selezione, per ogni termine da disambiguare, delle basic unit aventi il punteggio di coesione più elevato. Tale punteggio decreterà la scelta del synset da assegnare al termine.

5.3.1 Step 1 - Costruire le strutture dati

Nella prima fase dell'algoritmo, per la costruzione delle variabili su cui si agirà nelle fase sucessive, si riceve in input la lista delle *candidate word* ***CW*****.** Per ogni $\mathbf{cw}_i \in \boldsymbol{CW}$ si determina, grazie all'interazione con la risorsa lessicale WordNet, che in MOMIS è implementata in un database chiamato *momiswn* , l'insieme $\mathbf{WS}_i = [ws_{i1}, ws_{i2}, \ldots\ldots, ws_{it}]$, contenente i synset ad essa associati. Ogni synset $\mathbf{ws_{ij}}$ pervenuto sarà aggiunto alla lista **S**, inizialmente vuota, che terrà traccia di tutti i synset del *contesto lessicale*. Si nota subito che l'insieme **S** altro non è che l'unione di tutti gli insiemi $\mathbf{WS}_i$, ossia $\boldsymbol{S} = \cup_{\boldsymbol{i=1}}^{\boldsymbol{k}} \boldsymbol{WS_i}$.

Per ognuno dei synset $\mathbf{ws_j}$ si costruisce la relativa basic unit $\mathbf{bu_{ij}} = [\mathbf{cw_i}, \mathbf{ws_j}, \mathbf{score}]$, ossia una variabile contenente il synset $\mathbf{ws_j}$, il lemma $\mathbf{cw_i}$ cui il synset fa riferimento, e un punteggio di coesione, inizializzato a zero, determina la "bontà" della scelta del synset $\mathbf{ws_j}$ come significato del lemma $\mathbf{cw_i}$ **.** Tutte le basic unit così determinate vengono infine inserite in una lista, la **BUlist** per l'appunto, che rappresenterà ,per così dire, l'input della seconda fase dell'algoritmo.

5.3.2 Step 2 - Individuare le relazioni tra i synset e attribuire il punteggio di coesione

Nella seconda fase, il cuore dell'algoritmo, si procede con la ricerca delle relazioni tra i synset associati ai termini per poi provvederne la successiva disambiguazione, grazie all'attribuzione di un punteggio di coesione per ogni relazione individuata.

Il meccanismo di recupero delle relazioni

In questa fase si analizzano tutte le $\mathbf{bu_z} \in$ **BUlist ,** così come create nella fase precedente. L'algoritmo si posiziona sulla prima **bu** , e in particolare sul lemma $\boldsymbol{cw_i}$ e sul synset $\mathbf{ws_z}$, che sarà il synset di partenza della nostra analisi. Per ogni synset $\mathbf{ws_z}$ viene quindi recuperato l'insieme $\mathbf{RS}_z = [rs_{z1}, rs_{z2}, \ldots\ldots, rs_{zh}]$ contenente tutti i synset $\mathbf{rs_{zj}}$ che presentino una relazione semantica di qualsiasi tipo con $\mathbf{ws_z}$. Nel paragrafo successivo e nella sezione relativa ai test effettuati, si capirà come la scelta dei tipi di relazioni da considerare e il *livello di parentela* che i synset correlati hanno con il synset d'origine influenzi non poco le prestazioni dell'algoritmo.

A questo punto si controlla se $\mathbf{RS}_z$ contiene uno o più elementi in comune con l'insieme $\mathbf{S}/\{\mathbf{WS}_i\}$ (ricordiamo che **S** contiene tutti i synset del *contesto lessicale* e $\mathbf{WS}_i$ rappresenta l'insieme dei possibili synset cui appartiene il lemma $\boldsymbol{cw_i}$ in esame). Ciò equivale a cercare le eventuali relazioni che il synset $\mathbf{ws_z}$ presenta con tutti gli altri synset di **S** ,ossia del contesto, escludendo però dal processo di ricerca i synset appartenenti al lemma corrente, che non fornirebbero informazioni utili ai fini della disambiguazione (non avrebbe senso cercare le relazioni esistenti tra i synset contenuti nell'insieme $\mathbf{WS}_i$ di una stessa *candidate word* $\boldsymbol{cw_i}$).

Per comprendere meglio il procedimento logico adottato per trovare relazioni tra i synset e le strutture dati appena descritte si può far riferimento ad una rappresentazione grafica come quella in figura 5.1.

Supponiamo di voler trovare le relazioni che il lemma $\mathbf{cw_1}$ in figura presenta con uno o più lemmi del *contesto lessicale*. Per trovare queste relazioni, bisogna valutare se uno o più sensi di $\mathbf{cw_1}$ sono in qualche modo collegati ad uno o più sensi di ciascun altro termine.

Si analizzi, per cominciare, il synset $\mathbf{ws_1}$ (per semplicità omettiamo il primo pedice che identifica la **cw** di appartenenza), e si valuti l'insieme $\mathbf{RS}_1$ di

tutti i synset correlati a **ws*1*** per mezzo di una qualsiasi relazione semantica (supponiamo per il momento che si tratti solo di relazioni di primo livello). Si nota che il synset **rs*1*** di **RS**$_1$, che ad esempio presenta una relazione di *iperonimia (is a kind of)* con **ws*1*** , è uguale al synset **s*9*** presente nell'insieme **S,** e in particolare nell'insieme **WS**$_3$ dei synset relativi al lemma **cw*3***. Si crea in tal modo una relazione di *iperonimia* che abbraccia i due termini **cw*1*** e **cw*3*** nella loro accezione descritta rispettivamente dai sensi **ws*1*** e **s*13***.(che come detto rappresenta un possibile synset di **cw*3*** incluso in **WS**$_3$.).

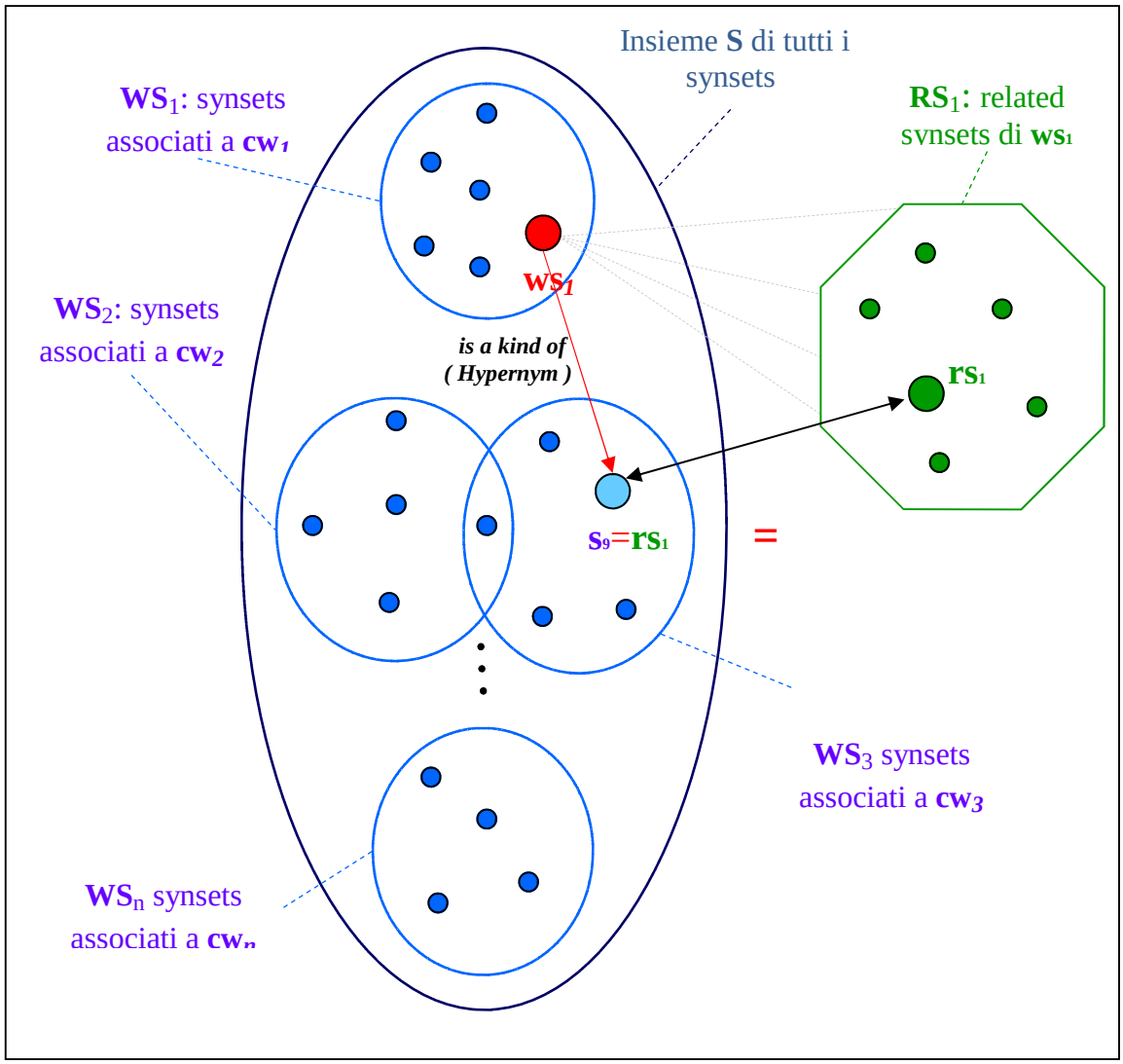

Figura 5.1 Il processo di individuazione delle relazioni tra synset

Ciò vuol dire che questi due synset avranno una probabilità maggiore di essere scelti come possibili synset corretti con cui annotare i due termini considerati, rispetto, ad esempio, a due synset che invece non presentano alcun tipo di relazione con il contesto. Si noti che non è insolito che due insiemi $\mathbf{WS}_j$ e $\mathbf{WS}_k$ si intersechino reciprocamente mostrando uno o più synset in comune (è il caso di $\mathbf{WS}_2$ e $\mathbf{WS}_3$ in figura) . In questo caso i due termini relativi $\mathbf{cw}_j$ e $\mathbf{cw}_k$ sono considerati sinonimi, e dunque il synset interessato avrà una possibilità ancora più alta di essere eletto come synset corretto per entrambi i termini.

Meccanismo di attribuzione del punteggio di coesione alle relazioni individuate

Nel momento in cui una relazione tra due synset viene individuata, è necessario attribuire a tale relazione un punteggio che andrà inserito nel campo score della **bu** considerata nell'iterazione generica. Bisogna però fare attenzione al fatto che l'insieme $\mathbf{RS}_1$ non necessariamente contiene soltanto i synset che presentano una relazione diretta con $\mathbf{ws}_1$. Ciò sta a significare che è possibile, per aumentare la probabilità di trovare una relazione, considerare un percorso di relazioni che possa congiungere un synset d'origine ad un synset di destinazione (*target synset*). Il punteggio, in tal caso, può essere attribuito in maniera euristica non solo in base alla "forza" del tipo di relazione individuata (una relazione di ipernimia è considerata genericamente più "forte" dal punto di vista semantico rispetto ad una relazione di meronimia), ma anche in base alla lunghezza del percorso di relazioni (*path*) che congiunge il synset d'origine con il *target synset*.

Per rendere più chiaro cosa si intenda per *path* di una relazione, e per comprendere come sia possibile rappresentare synset e relazioni attraverso un grafo, può essere conveniente fare un esempio.

Supponiamo di aver iterato il procedimento di determinazione delle relazioni appena descritto per due dei synset appartenenti all'insieme $\mathbf{WS}_1$ della $\mathbf{cw}_1$. Supponiamo che il termine in questione sia il lemma *bus* e che i due synset siano i due concetti *bus#1* e *bus#2:* il concetto (synset) *bus#1* è inteso come *veicolo di trasporto*, e il concetto *bus#2* inteso come *connettore di dispositivi* nei sistemi elettronici .

Al termine delle iterazioni saranno recuperate tutte le relazioni , dirette e indirette fino al livello desiderato, che si diramano dai due synset. Per ognuno di essi si ottiene così una mappa di relazioni semantiche che possono essere rappresentate graficamente, analogamente all'algoritmo SSI (3) come nella figura seguente:

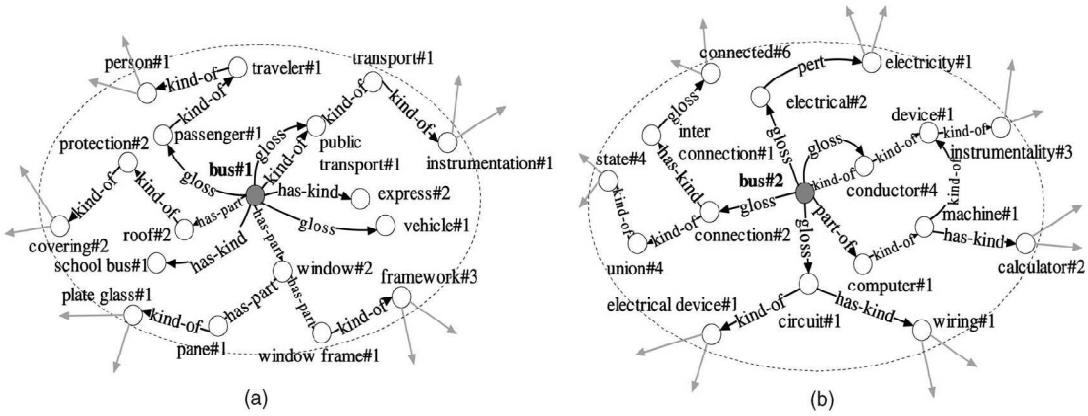

Figura 5.2 Rappresentazione a grafo del senso#1 (a) e del senso#2 (b) di bus

Nel caso mostrato nelle due figure si nota che non sono stati considerati soltanto i synset collegati direttamente a *bus#1*e *bus#2* , ma anche synset che presentano fino a tre gradi di parentela con il nodo d'origine.

È evidente a questo punto come si sia generata una struttura a grafo G = (V, E) , in cui l'insieme V dei nodi rappresenta l'unione dell'insieme $\boldsymbol{S}$ e di tutti gli insiemi $\boldsymbol{RS}_i$ costruiti a partire da $\boldsymbol{S}$, e l'insieme E degli archi congiungenti i nodi rappresenta le relazioni che intercorrono tra i synset.

In tal modo si può definire un percorso come sequenza dei nodi che gli archi collegano tra loro: in un percorso (a_0 $a_{1,}$ a_1 a_2 ,…, a_{k-1} a_k) ogni elemento a_i è un synset appartenente a V , ed ogni coppia a_{i-1} a_i rappresenta un arco , appartenente a E, congiungente i due synset a_{i-1} $a_{i.}$

Si definisce *cammino* ("*path*") un percorso i cui nodi sono tutti distinti tra loro. In questo modo, se *n* è il livello di parentela prescelto, si ottengono dei cammini di relazioni lunghi al più *n* tra il synset di partenza ed un suo target synset.

Per chiarire ora il metodo di attribuzione dello score ad un cammino, e dunque alla basic unit che contiene il suo synset d'origine, consideriamo il percorso in figura 5.2 (b) :

$$bus\#2 \xrightarrow{is\ a\ part\ of} computer \xrightarrow{is\ a\ kind\ of} machine\#1 \xrightarrow{has\ kind} calculator\#2$$

In questo caso il synset *bus*#2 è correlato al synset *calculator*#2 dalla sequenza di tre relazioni: *meronimia/iperonimia/iponimia* . Il peso da attribuire a tale percorso sarà direttamente proporzionale alla "forza" delle singole relazioni interessate, e inversamente proporzionale alla lunghezza del percorso. Per valutare la "forza" di un certo tipo di relazione e attribuirne uno score adatto si sono presi come valori di riferimento i risultati empirici pubblicati in (3), ma, nel corso dei test effettuati, tali valori sono stati continuamente variati per ottenere una configurazione che più delle altre ottimizzasse l'accuratezza dell'algoritmo.

Definito con la notazione $RW(\mathbf{s}_j,\mathbf{s}_{j+1})$ il peso associato al tipo di relazione esistente (*relationship weight*) tra due synset intermedi consecutivi $\mathbf{s}_j$ e $\mathbf{s}_{j+1}$, la formula utilizzata per calcolare il peso del cammino i-esimo congiungente il generico synset d'origine **s** con il synset di destinazione $\mathbf{s}_k$ è dato da:

$$weight(\boldsymbol{s}, path_i) = \frac{\sum_{j=0}^{k} RW(s_j, s_{j+1})}{lenght(path_i)} \qquad (f1)$$

Dove $lenght(path_i)$ è la lunghezza del percorso i-esimo congiungente i due synset e l'elemento $\mathbf{s_0}$ rappresenta il synset $\boldsymbol{s}$ all'interno della sommatoria . Tornando all'esempio del synset *bus*#2, e ipotizzando un peso per la relazione di iperonimia pari a 0.8 (equivalente al peso dell'iponimia, sua simmetrica) e pari a 0.5 per la meronimia, il punteggio da associare al synset sarà pari a :

$$weight(bus\#2 \xrightarrow{is\ a\ part\ of} computer \xrightarrow{is\ a\ kind\ of} machine\#1 \xrightarrow{has\ kind} calculator\#2) = \frac{0.5+0.8+0.8}{3} = 0.7$$

Tale punteggio viene poi attribuito alla **bu** corrispondente al synset *bus*#2, e viene sommato ai punteggi di tutti i *relationship paths* trovati che abbiano *bus#2* come synset d'origine. Un synset di un determinato lemma che presenta più di una relazione con elementi del contesto, infatti, deve ottenere un punteggio maggiore rispetto ad un altro synset , appartenente allo stesso lemma, ma scollegato dal contesto. Pertanto, supponendo che esistano *n* cammini di lunghezza variabile che si diramano dal generico synset d'origine $\mathbf{s_s}$, definiamo il punteggio totale $W(\mathbf{s_s})$ da attribuire al synset:

$$W(s_s) = \begin{cases} \sum_{i=1}^{n} weight(path_i) & \textit{se } n > 0 \\ 0 & \textit{altrimenti} \end{cases} \qquad (f2)$$

dove *i = 1.....n* è l'indice del singolo percorso individuato.

In figura 5.2 è riportato un esempio del meccanismo di valutazione del punteggio: il synset generico $\mathbf{s_0}$ presenta una relazione diretta con il target synset $\mathbf{ts_2}$, una relazione di secondo livello con il target synset $\mathbf{ts_1}$, e una relazione di terzo livello con il target synset $\mathbf{ts_3}$. Per ognuna delle relazioni che costituiscono ogni percorso sono riportati i relativi punteggi stabiliti euristicamente, ossia gli $RW(\mathbf{s_j},\mathbf{s_{j+1}})$.

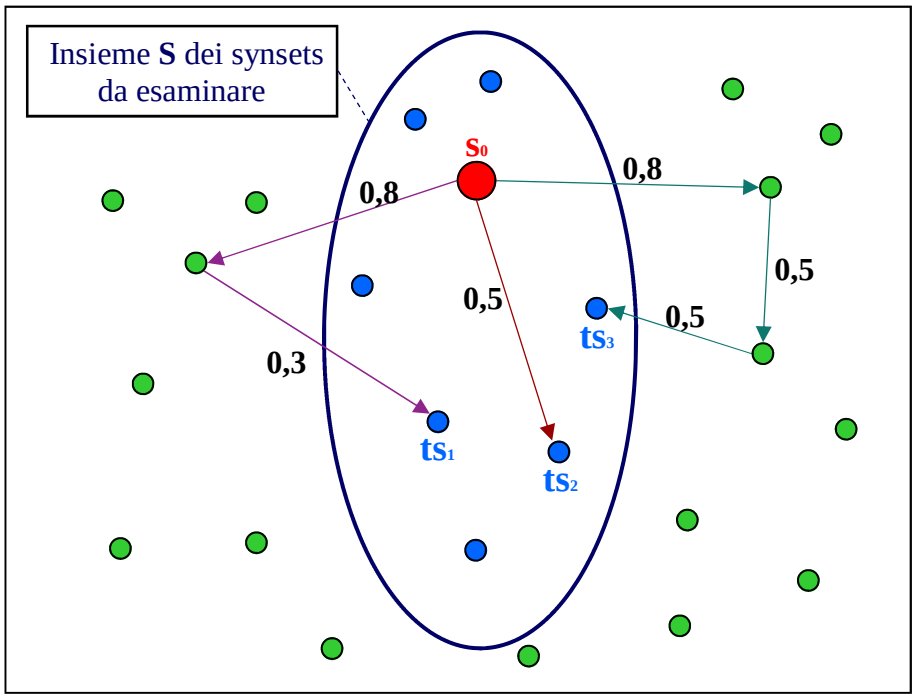

Figura 5.2 – Percorsi indiretti di connessione tra synset

Ciascuno dei tre percorsi recherà poi il proprio punteggio $weight(\boldsymbol{s_0}, path_i)$,calcolato in base alla formula (*f1*). Si può notare che calcolare il punteggio dei due cammini indiretti equivale a costruire due nuove relazioni, come quelle tratteggiate in figura 5.3, il cui punteggio di coesione rappresenta la media pesata dei pesi delle relazioni coinvolte . Il peso W(s_s), infine, è dato dalla somma dei $weight(\boldsymbol{s_0}, path_i)$ appena ricavati, e rappresenta il punteggio complessivo da attribuire, al termine dell'analisi del generico synset $\mathbf{s_0}$, alla basic unit **bu** che lo contiene.

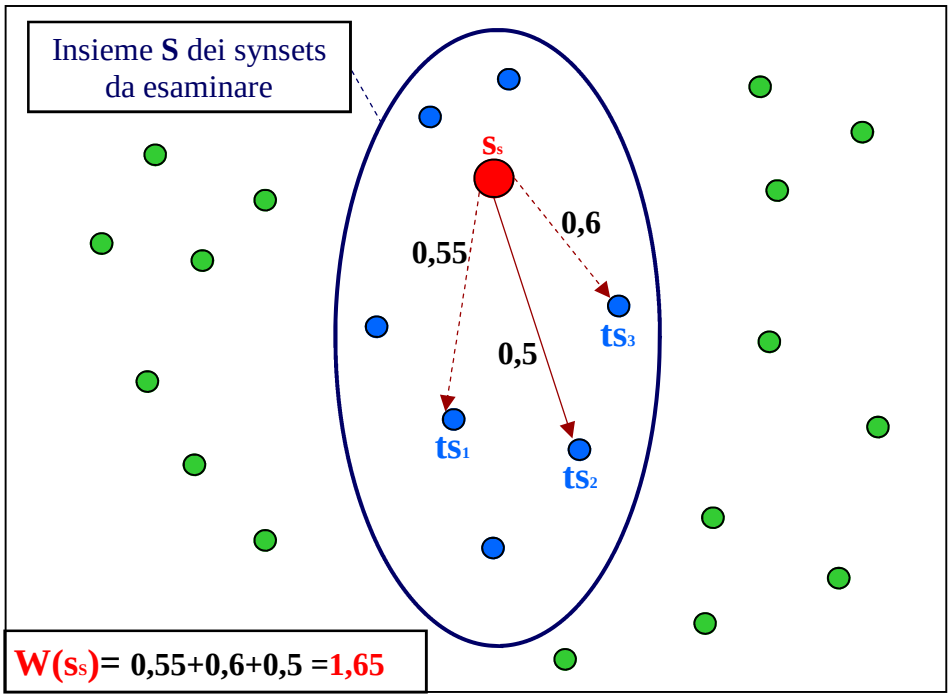

Figura 5.3 – Percorsi equivalenti di connessione tra synset

5.3.3 Step 3 - Estrarre la lista dei termini annotati

Dopo aver iterato il meccanismo di valutazione dei punteggi per tutte le **bu** della **BUlist,** ogni synset recherà il proprio punteggio di coesione con il lemma cui è associato. A questo punto l'ultima operazione da effettuare consiste nel selezionare, per ogni lemma, il synset con il punteggio più elevato. Nel caso in cui due synset presentassero lo stesso score, verranno mantenuti entrambi come possibili sensi corretti.

In altre parole, il synset $\mathbf{ws_m} \in \mathbf{W_i}$ selezionato relativo al lemma generico $\mathbf{cw_i}$ sarà tale per cui

$$m = \underset{ws_j \in \mathrm{W_i}}{\operatorname{argmax}}\{W(ws_j)\}$$

dove il generico elemento $W(ws_j)$ rappresenta,come detto, lo score associato ad uno dei possibili sensi $\mathbf{ws_j} \in \mathbf{W_i}$ del lemma che stiamo esaminando.

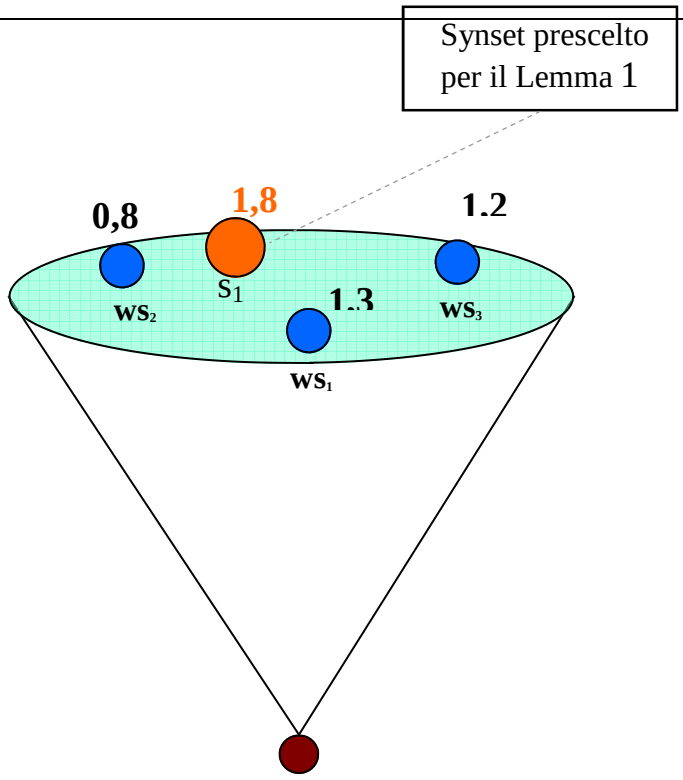

Figura 5.4 – Selezione del synset punteggio di coesione maggiore

La figura 5.4 chiarifica il processo di selezione del synset con lo score più elevato. A questo punto, dunque, il processo di annotazione lessicale automatica è terminato. L'output prodotto dall'algoritmo è rappresentato da una lista i contenente l'insieme dei lemmi associati al senso ritenuto corretto, così come mostrato nello schema seguente

Sentences extracted from
http://www.cs.stanford.edu/Courses/Index.html

Class information & Courses.
The Computer Science Education Center
has information on undergraduate CS courses.

Candidate words	Selected Meaning
class(1)	3591
information(2)	27555
course(3)	3591
computer science(4)	28610
education(5)	3589
center(6)	27928
information(7)	27555
undergraduate(8)	47915
cs(9)	28610
course(10)	3591

(a) *Word Sense disambiguation*

Candidate Words	Possible Meanings (Synsets and WordNet Glosses)
class(1)	37377 - a collection of things sharing a common attribute; 38085 - a body of students who are taught together; 37296 - people having the same social or economic status… 3591 - education imparted in a series of lessons or class meetings…
information (2)(7)	33347 - formal accusation of a crime 38929 - a collection of facts from which conclusion may be drawn 27555 - knowledge acquired through study or experience…
course(3)(10)	3591 - education imparted in a series of lessons… … 15044 - a circumscribed area of land or water…
computer science(4)	28610 the branch of engineering science…
education(5)	3589 - activities that impart knowledge; 28190 - knowledge acquired by learning and instruction… …
center(6)	39134 - an area that is approximately central…
undergraduate(8)	47915 - a university student who has not yet received a first degree
cs(9)	62950 - a soft silver-white ductile metallic element 28610 the branch of engineering science…

(b) *Candidate words and Their Possible Meanings.*

Figura 5.5- Esempio di disambiguazione tramite TUCUXI

In figura è presente il risultato della disambiguazione di una frase estratta dal sito http://www.cs.stanford.eu/Courses/index.html e usata come esempio in (22). La sezione (b) mostra tutti i possibili synset relativi ad ogni sostantivo da disambiguare nella frase, mentre la figura (a) mostra il risultato finale dell’annotazione, ossia l’attribuzione del synset ritenuto più corretto ad ogni lemma esaminato.

5.4 Dettagli implementativi

5.4.1 L’ambiente di sviluppo

L’algoritmo di TUCUXI è stato interamente realizzato in Java, utilizzando la release 1.6 della JDK (Java Development Kit) di Sun Microsystem. L’ ambiente di sviluppo utilizzato per l’implementazione è stata la piattaforma integrata Eclipse Java EE IDE, framework contenente tool e utilità di alto livello utili alla programmazione ed un potente editor per Java. Eclipse è, peraltro, l’ ambiente utilizzato per lo sviluppo del sistema MOMIS. Infatti all’interno del progetto erano già presenti diverse classi, come ad esempio quelle che gestiscono la comunicazione con le ontologie lessicali (WordNet in questo caso) o con gli schemi ODLI$_3$ memorizzati in file xml, che rappresentano le sorgenti di dati utilizzate ai fini dell’annotazione automatica.

L’inserimento, poi, della classe principale che descrive l’algoritmo TUCUXI all’interno del pool di algoritmi di ALA è stato agevolato dalla flessibilità e dalla modularità dell’organizzazione del codice: è disponibile infatti un’interfaccia cui ogni algoritmo di ALA deve conformarsi.

Le scelte implementative sono state adottate cercando il giusto compromesso tra la precisione dell’algoritmo e le sue prestazioni: abbiamo visto nella sezione precedente come il nucleo centrale attorno a cui ruota

l'esecuzione è rappresentato dalla ricerca delle relazioni esistenti tra termini da disambiguare. Ebbene, in WordNet sono presenti alcuni synset che presentano migliaia di relazioni,e si intuisce come non siano da sottovalutare i costi , in termini di tempi di analisi e di utilizzo di risorse, che bisogna sostenere al crescere del numero di termini da disambiguare.

5.4.2 Le tipologie di relazioni considerate

Tra le relazioni di WordNet, ampiamente descritte nel paragrafo 1.2.1, e della sua estensione WordNet Domains, l'algoritmo prende in considerazione prevalentemente relazioni semantiche, e in particolare:

1. *Iponimia/iperonimia*
2. *Meronimia/Olonimia*, suddivisa in tre categorie (delle 6 elencate nel paragrafo 1.2.1) :
   - *componente* $\xrightarrow{is\ a\ part\ of}$ *oggetto* (*Part Meronymy/Holonymy*)
   - *elemento* $\xrightarrow{is\ a\ part\ of}$ *insieme* (*Member Meronymy/Holonymy*)
   - *materiale* $\xrightarrow{is\ a\ part\ of}$ *oggetto* (*Substance Meronymy/Holonymy*)
3. *Domini in comune*. Questo tipo di relazione non è definita esplicitamente in WordNet ma derivata da altre relazioni di dominio, come vedremo.
4. *Sinonimia*. L'unica relazione lessicale presa in considerazione, si individua quando due lemmi hanno un synset in comune.

Tali relazioni sono state esaminate, durante il lavoro svolto, soltanto per i termini appartenenti alla categoria sintattica dei nomi. Questa scelta è sottesa da due motivazioni:

- In un qualsiasi testo scritto in linguaggio naturale, una parte ragionevole del contenuto suo informativo può essere ricavato dall’analisi dei soli sostantivi.
- l’algoritmo in questione è stato realizzato principalmente per l’annotazione automatica di sorgenti di dati strutturate e semistrutturate, e dunque di nomi che principalmente descrivono classi e attributi.

Ad ognuna di queste relazioni, come detto, è stato attribuito un peso che dovrebbe essere indice di “vicinanza semantica” tra due synset, ossia dovrebbe rappresentare la “forza” di una relazione diretta tra, adottando la terminologia di WordNet, un *source synset* e un *target synset.*

La scelta del peso da assegnare a ciascuna relazione influenza significativamente i risultati del processo di disambiguazione, motivo per cui si è pensato di renderne il valore *parametrizzabile*. Ciò sta a significare che l’utente potrà attribuire ad ogni relazione, fatta eccezione per la sinonimia, che rappresenta la massima misura di vicinanza semantica, un peso arbitrario all’interno di un intervallo che va, per livelli decrescenti di rilevanza, da 1 a 0.01. Va precisato, infine, che le relazioni simmetriche sono valutate allo stesso modo (ad esempio alle relazioni di iperonimia e di iponimia sarà attribuito lo stesso punteggio) .

5.4.3 Valutazioni sulla complessità della ricerca delle relazioni

Nei capitoli precedenti si è spiegato che è possibile ricavare una rappresentazione “a grafo” della rete di relazioni che connette i synset appartenenti ad uno stesso *contesto lessicale:* i nodi del grafo individuano i synset, e gli archi congiungenti i nodi rappresentano le relazioni che collegano i synset.

Dal punto di vista informatico, una implementazione possibile di un grafo richiede la creazione una matrice *n x n* in cui gli indici delle righe e delle colonne individuano univocamente i nodi del grafo e i cui elementi rappresentano le relazioni che intercorrono (ovviamente ogni elemento non deve necessariamente indicare la presenza o l'assenza di un di una relazione , ma può disporre di più proprietà, come il tipo di relazione interessata e il peso attribuito) tra i nodi identificati dagli indici.

Risulta però evidente come nel caso in esame una matrice sparsa,ossia una matrice il cui numero di elementi non sia predeterminato, sarebbe preferibile rispetto alla matrice *n x n,* dal momento che con una matrice *n x n* si dovrebbero gestire non pochi elementi nulli nel caso in cui due synset non fossero collegati.

Tuttavia, in una fase preliminare allo sviluppo dell'algoritmo, è stato valutato che anche la scelta della matrice sparsa, seppur corretta dal punto di vista implementativo, avrebbe appesantito eccessivamente l'esecuzione, poiché la creazione e il mantenimento di una struttura dati complessa richiede comunque un impiego notevole di risorse, specialmente quando si ha a che fare con synset che presentano migliaia di relazioni. Va poi considerato che anche la *profondità* del grafo semantico di un synset influisce sulle prestazioni: costruire, a partire da un synset d'origine, una mappa dei synset ad esso collegati è un procedimento che presenta una complessità proporzionale al numero di archi implicati nel collegamento. In altre parole, è proporzionale alla distanza, intesa come numero di archi, che separa il synset d'origine dai synset ad esso correlati.

Alla luce di queste considerazioni sono state prese due decisioni:

- Pur offrendo all'utente la possibilità di inserire il livello di *profondità* desiderato per la costruzione di un grafo semantico, si è

posto un limite massimo quantificato in 3 archi di distanza dal synset d'origine.

- È stato implementato un metodo che, invece di costruire una matrice e mantenerne gli elementi in memoria, possa dinamicamente creare oggetti e distruggerli quando non è più necessaria la loro allocazione, liberando cosi le risorse impegnate. Tale metodo è sinteticamente descritto nel prossimo paragrafo.

Infine, dal momento che non tutte le tipologie di relazioni sono sempre significative ai fini della disambiguazione, è stata offerta all'utente la possibilità di scegliere quali relazioni cercare tra i termini da disambiguare.

5.4.5 Il metodo *setScoreForSynset*

Il metodo *setScoreForSynset* è la soluzione proposta per calcolare dinamicamente il punteggio di coesione da attribuire al synset contenuto nella basic unit $\mathbf{bu}_i$ che l'algoritmo sta analizzando (si veda in proposito lo pseudo codice in figura) . Il metodo accetta come parametri in input un oggetto *sourceNode*, che rappresenta il synset di partenza (ossia il nodo d'origine), un insieme di synset *synsetToAnalize*, che rappresenta l'insieme $\mathbf{S}/\{\mathbf{WS}_i\}$ dei synset del contesto lessicale con i quali si devono cercare le relazioni, e una lista di simboli *symbols* che rappresentano le tipologie di relazioni che si desidera individuare. Il risultato prodotto dall'algoritmo è direttamente punteggio di coesione da attribuire al sourceNode.

Come prima cosa il metodo valuterà se sono presenti altri lemmi da disambiguare che presentino un synset in comune con il *sourceNode*, e per ogni sinonimo trovato memorizzerà il punteggio stabilito per la relazione di sinonimia.

A seconda del livello di profondità scelto dall'utente il metodo si posiziona sul *sourceNode* ed individua tutti i synset che presentino relazioni (*target*

synset), delle tipologie indicate dai *symbols*, esattamente fino a quel livello. Nel caso in cui un target synset recuperato sia un synset appartenente al contesto, vuol dire che due lemmi sono legati da una relazione, per cui si calcola il punteggio parziale secondo la formula (1) descritta nel paragrafo 5.3.2 , che poi verrà sommato a tutti i punteggi parziali ottenuti per ogni altra relazione individuata. Per quanto riguarda la relazione che si instaura tra due synset che appartengono allo stesso dominio (individuato anch'esso da un synset di WND), è stato implementato un altro metodo, all'interno di *setScoreForSynset*, che individua le intersezioni tra l'insieme dei domini cui il *sourceNode* appartiene e l'insieme dei domini cui appartengono tutti gli altri synset rappresentati dai *synsetToAnalyze*. Per ogni intersezione recuperata, il metodo aggiorna, con il punteggio selezionato dall'utente per la relazione di dominio, lo score totale del *sourceNode* che sarà successivamente restituito come output del metodo principale *setScoreForSynset*.

Quando il metodo *setScoreForSynset* sarà stato chiamato per tutte le basic units relative ad uno stesso lemma e avrà inserito in ognuna di esse il punteggio di coesione calcolato, un altro metodo, *findMaxScoringBasicUnits* provvederà immediatamente ad individuare la/le basic unit con lo score massimo e a rimuovere dalla lista *BUlist* le basic unit con il punteggio più basso. Quando il procedimento avrà "toccato" tutti i termini da disambiguare, la lista *BUlist* rappresenterà l'output prodotto dall'algoritmo, ossia una lista di termini cui è stato attribuito uno o più significati.

5.4.6 I termini composti

Uno dei problemi che caratterizzano l'annotazione automatica di sorgenti di dati è rappresentato, come specificato nel paragrafo 4.1, dalla

disambiguazione dei *compound terms*, i termini composti, il cui utilizzo è ampiamente diffuso per denominare gli elementi degli schemi di sorgenti di dati, a prescindere dalla loro natura. Per termini composti qui intendiamo termini separati da un carattere "blank", da un carattere di "underscore" oppure scritti in CamelCase (ad esempio *UniversityStudent* o *departmentCode*). Purtroppo le ontologie lessicali per il momento non offrono un valido aiuto, dal momento che pochi sono i termini composti che siano di così vasto utilizzo da permetterne l'inserimento e la catalogazione. La maggior parte degli elementi degli schemi, invece, riportano dei nomi composti creati ad hoc per esprimere caratteristiche conformi alle esigenze del progettista della sorgente di dati. Alla luce di queste considerazioni, mantenendo l'approccio attualmente utilizzato nel tool ALA, si attribuirà un significato a ciascuno dei lemmi che formano il *compund term.*

5.4.7 Note sull'esecuzione

Nella fase preliminare all'esecuzione, viene effettuato il *parsing* di ogni sorgente dati al fine di individuarne i nomi delle classi e degli attributi, che costituiranno la lista dei termini da disambiguare (le *candidate words* descritte precedentemente). Il *parsing* viene effettuato su file xml che contengono la descrizione ODLI$_3$ delle sorgenti considerate.

Per cercare di minimizzare i tempi di esecuzione del codice, sono stati adottati i seguenti accorgimenti:

1. Se, ad una generica iterazione, l'algoritmo si posiziona su un lemma che occorre più di una volta all'interno delle *candidate words*, e che è stato già disambiguato in un passo precedente, allora copia il risultato della disambiguazione precedente al lemma corrente.

Questo accorgimento evita di ripetere il costoso processo di ricerca di relazioni per un lemma le cui relazioni sono già state individuate.

2. Ogni volta che viene individuata una relazione tra due synset appartenenti all'insieme **S**, del tipo:

   $Synset1 \xrightarrow{has\ part} intermediateSynset \xrightarrow{is\ kind\ of} Synset2$

   il punteggio parziale attribuito al *Synset1* per tale relazione sarà attribuito anche al *Synset2*. Questa soluzione è giustificata dal fatto che la relazione simmetrica

   $Synset2 \xrightarrow{has\ kind} intermediateSynset \xrightarrow{is\ part\ of} Synset1$

   recherebbe lo stesso score, dal momento che attribuiamo lo stesso peso alle relazioni simmetriche, ed eviterebbe la ricerca relazioni già trovate tra due synset.

Va precisato un ultimo aspetto relativo al punto 1: anche se non si ripete la ricerca di relazioni quando l'algoritmo si posiziona su un termine che è già stato disambiguato, c'è da dire che il numero delle sue occorrenze all'interno delle *candidate words* è comunque importante. Infatti supponiamo che uno tra i synset del termine X sia connesso,attraverso un cammino qualsiasi, ad un synset del termine Y, e che il termine Y compaia 3 volte nella lista delle *candidate words*. Il synset di X, in questo caso, riceverà un punteggio pari a 3 volte il punteggio del cammino che lo connette al synset di Y.

Ad esempio, ipotizziamo che il lemma *country* compaia 3 volte all'interno dell'elenco dei termini da disambiguare e che il lemma *area,* presente anch'esso nella lista, sia collegato ad esso da una relazione diretta di iperonimia tra *area#1* e *country#2*, il cui peso è di 0.8. Al synset *area#2* in questione sarà attribuito un peso pari a 2.4, ossia al peso della singola relazione per il numero di volte che tale relazione viene trovata. Questo approccio trova la sua *ratio* nel fatto che anche la ***reiterazione*** dei termini**,**

come già spiegato precedentemente, è un elemento che caratterizza la ***coesione*** del testo. Per questo motivo se un synset associato ad un termine ha una relazione con un altro synset associato ad un termine che si ripete più volte,è giusto attribuirgli un peso maggiore.

Un'ultima premessa riguarda l'estrazione dei termini disambiguati: se per un lemma appartenente al *contesto lessicale* non viene individuata alcuna relazione, il lemma non viene disambiguato, anche se si tratta di un lemma *monosemico*, dunque non ambiguo. Questa scelta è dovuta al fatto che si desidera valutare il ruolo effettivo che le relazioni rivestono nella disambiguazione del testo.

5.5 Analisi delle prestazioni dell'algoritmo

L'algoritmo realizzato è stato testato su 5 sorgenti di dati. Due di queste sono di tipo relazionale, e tre sono sorgenti semi strutturate definite in file xml. La prima, *Mondial*, contiene informazioni riguardanti la geografia di un territorio, le altre contengono informazioni di diverso tipo relative al campo dell'editoria.

L'analisi delle prestazioni dell'algoritmo è stata effettuata tramite l'utilizzo dei classici fattori che misurano l'efficacia di un sistema di *Information Retrieval* ,vale a dire i valori di ***Precision*** e ***Recall,*** definiti come:

$$\textbf{\textit{Precision}} = \frac{numero\ di\ annotazioni\ corrette}{numero\ di\ annotazioni\ effettuate}$$

$$\textbf{\textit{Recall}} = \frac{numero\ di\ annotazioni\ corrette}{numero\ totale\ di\ annotazioni\ attese}$$

dove per annotazioni corrette si intendono i significati da me attribuiti manualmente agli elementi degli schemi sorgenti. Per annotazioni attese si intendono invece tutti gli elementi delle sorgenti presi indistintamente, e dunque anche quei termini (pochi in realtà) che non appartengono alla categoria sintattica dei nomi, gli acronimi, e tutti quei termini che non sono stati catalogati in WordNet.
Come abbiamo accennato nei paragrafi precedenti, i valori di precision e recall sono stati ricavati, per ogni sorgente, in funzione :

1. delle tipologie di relazioni considerate e del peso da attribuire alle relazioni scelte. È infatti possibile passare all'algoritmo una lista delle tipologie di relazioni che si vogliono prendere in considerazione per il processo di disambiguazione
2. della lunghezza massima selezionata per i cammini che connettono due synset correlati. In altre parole se si seleziona una lunghezza massima pari a 2, l'algoritmo cercherà i synset correlati al synset di partenza tramite un cammino costituito al più da 2 archi. Come abbiamo detto è possibile selezionare un valore che va da 1 a 3.

La possibilità di effettuare tali scelte è resa da un semplice pannello di configurazione che permette di selezionare la lunghezza massima dei cammini e il peso desiderato per ogni tipologia di relazione che si desidera includere nella lista da passare al programma.
Ciò ha consentito di avere un quadro più approfondito dei fattori che influenzano la precision e la recall dell'algoritmo, testando differenti configurazioni di pesi per diversi livelli di profondità selezionati.

Gli obiettivi fondamentale dei test effettuati sono :

- *Valutare le prestazioni dell’algoritmo su ogni sorgente*, utilizzando diverse configurazioni di pesi, diversi livelli di profondità dei cammini e scegliendo differenti tipologie di relazioni da includere nella lista.
- *Ottenere una configurazione di pesi da attribuire alle relazioni che massimizzi le prestazioni*, in termini di precision e recall, dell’algortimo valutate su tutte le sorgenti considerate.

Tralasciando i risultati ottenuti per livello di profondità unitario, che come si può intuire, nella maggior parte dei casi (ma non in tutti) non porta a prestazioni soddisfacenti, si riportano i risultati di due test:

1. **Test 1 :** è stato selezionato una lunghezza massima per i cammini individuati pari a **2 archi** e sono stati calcolati i valori di precision e recall per ognuna delle configurazioni descritte nel paragrafo 5.6.
2. **Test 2:** è stata selezionata una lunghezza massima pari a **3 archi** e sono state utilizzate le stesse configurazioni del punto 1.

5.6 Le configurazioni dei pesi utilizzati per i test

Si riportano ora le tabelle di configurazione, ossia le tabelle contenenti i diversi pesi scelti per le tipologie di relazioni che si è desiderato prendere in considerazione. Per non appesantire troppo la trattazione, si è scelto di riportare le 4 configurazioni più rilevanti:

1. **Configurazione A:**

RELATIONSHIP TYPE	WEIGHT
SYNONYMY	**1**
HYPERONIMY/HYPONIMY	**0,8**
SUBST MERONYMY/HOLONYMY	**0,5**
MEMB MERONYMY/HOLONYMY	**0,5**
PART MERONYMY/HOLONYMY	**0,5**
DOMAIN IN COMMON	**0.01**

Tabella 5 – Configurazione A dei pesi selezionati per le relazioni

Questa rappresenta la configurazione più completa, tutte le relazioni contemplate sono state prese in considerazione, ed è stato attribuito loro un peso che va da 0.01 a 1. Il punteggio di appena 0.01 attribuito a due synset che presentano un dominio in comune è dovuto al fatto che generalmente queste relazioni sono numerosissime, poichè non è infrequente due synset che appartengono ad uno stesso dominio, soprattutto se parliamo di domini molto vasti (del tipo *person*, *animal* etc.). Per cui si è pensato di tener basso lo score per non inficiare i risultati "coprendo" gli score di relazioni importanti come l'iperonimia/iponimia (ricordiamo che un synset viene scelto in base al punteggio di coesione ottenuto, e, assegnando un punteggio elevato alle relazioni di dominio, queste finirebbero per essere l'unica discriminante nella scelta dei significati).

2. **Configurazione B:**

RELATIONSHIP TYPE	WEIGHT
SYNONYMY	**NC**
HYPERONIMY/HYPONIMY	**0,8**
SUBST MERONYMY/HOLONYMY	**0,5**
MEMB MERONYMY/HOLONYMY	**0,5**
PART MERONYMY/HOLONYMY	**0,5**
DOMAIN IN COMMON	**NC**

Tabella 6 – Configurazione B dei pesi selezionati per le relazioni

Questa configurazione, invece, lascia spazio alle relazioni prettamente semantiche, tralasciando i sinonimi e le relazioni di dominio. NC infatti sta per *Non Considerata*. Il peso di 0,5 per le relazioni di meronimia è stato scelto in base ad una semplice assunzione: un percorso diretto tra due synset legati da una relazione di meronimia deve avere un peso maggiore rispetto ad una relazione indiretta (quindi formata da due archi) di iperonimia/iponimia (che avrebbe in questo caso un peso di 0.8/2 = 0.4).

3. **Configurazione C:**

RELATIONSHIP TYPE	WEIGHT
SYNONYMY	NC
HYPERONIMY/HYPONIMY	0,8
SUBST MERONYMY/HOLONYMY	NC
MEMB MERONYMY/HOLONYMY	NC
PART MERONYMY/HOLONYMY	NC
DOMAIN IN COMMON	NC

Tabella 7 – Configurazione C dei pesi selezionati per le relazioni

Come si nota l’unica tipologia di relazione presa in considerazione è quella di iponimia/iperonimia, ossia la relazione semantica statisticamente più diffusa in WordNet. La scelta di questa semplice configurazione è mirata a capire quanto siano influenti ai fini della disambiguazione tali relazioni.

4. **Configurazione D:**

RELATIONSHIP TYPE	SCORE
SYNONYMY	1
HYPERONIMY/HYPONIMY	NC
SUBST MERONYMY/HOLONYMY	0,6
MEMB MERONYMY/HOLONYMY	0,6
PART MERONYMY/HOLONYMY	0,6
DOMAIN IN COMMON	0,05

Tabella 8 – Configurazione D dei pesi selezionati per le relazioni

Questa configurazione, infine, è servita per avere una controprova della scelta effettuata per la configurazione C. Si vuole valutare, in altre parole, la capacità di disambiguazione dell'algoritmo prescindendo completamente dalla relazione considerata fondamentale, ossia quella di iponimia/iperonimia, e incrementando leggermente il peso dei domini nel processo.

5.7 Test 1: lunghezza massima cammini pari a 2 archi

Si riportano in questa sezione i risultati dei test per ogni configurazioni di pesi precedentemente descritta, considerando sempre una lunghezza massima dei cammini di connessione tra synset pari a 2 archi

5.7.1 Configurazione A: analisi dei risultati

SCHEMA SORGENTE	LEMMI ESAMINATI	LEMMI DISAMBIGUATI	LEMMI DISAMBIGUATI CORRETTAMENTE	PRECISION	RECALL
Mondial	138	114	76	0,67	0,55
Amalgam1	103	42	11	0,26	0,11
Ontology1	185	164	83	0,51	0,45
Ontology2	205	150	72	0,48	0,35
Ontology3	207	155	76	0,49	0,37

Tabella 9 - Precision e recall per profondità dei cammini pari a 2 e configurazione di pesi A

Come si nota dall'analisi dei dati in tabella, i valori di precision sono discreti. Ricordiamo che la configurazione A prevede l'utilizzo di tutte le relazioni, sinonimi compresi, per disambiguare i termini degli schemi. Il

basso livello di recall è da attribuire sia alle carenze lessicali di WordNet, che al grande uso di abbreviazioni e acronimi per la definizione degli elementi annotabili. Un’altra osservazione va fatta in merito alla sorgente *Amalgam1*. I valori di precision per questa sorgente si discostano dalla media dal momento che i termini di cui è costituito questo database sono pochi ma ripetuti più di una volta. Per questo motivo, date le osservazioni fatte al punto 1 del paragrafo 5.5.4 , l’ annotazione errata di un termine si ripete per il numero di occorrenze di quel termine abbassando notevolmente i valori di precision. I valori medi di precision e recall, calcolati su tutte le sorgenti, relativi a questa configurazione sono dunque

AVG PRECISION	AVG RECALL
0,48	0,36

Tabella 10 - Precision e recall medi per profondità dei cammini pari a 2 e configurazione di pesi A

5.7.2 Configurazione B: analisi dei risultati

SCHEMA SORGENTE	LEMMI ESAMINATI	LEMMI DISAMBIGUATI	LEMMI DISAMBIGUATI CORRETTAMENTE	PRECISION	RECALL
Mondial	138	78	53	0,68	0,38
Amalgam1	103	41	14	0,34	0,14
Ontology1	185	150	80	0,53	0,43
Ontology2	205	136	77	0,57	0,38
Ontology3	207	133	76	0,57	0,37

Tabella 11 - Precision e recall per profondità dei cammini pari a 2 e configurazione di pesi B

Contrariamente a quanto ci si poteva aspettare, non considerare i sinonimi e le relazioni di dominio ha avuto un effetto positivo generale sulla precision dell'algoritmo. Dobbiamo questo risultato al fatto che le relazioni di dominio individuate, anche se consentono di aumentare la dimensione della rete di relazioni che partono da ogni synset (a volte anche di due ordini di grandezza), portano spesso a valutazioni non corrette. Per fare un esempio, se il lemma *number* dello schema *Ontology1* è stato annotato, non utilizzando i domini, con il suo senso corretto *number#7*, ossia col significato di "*numero di un periodico*", utilizzando le relazioni di dominio sarà annotato con il suo senso *number#1*, ossia col significato più generale del termine numero, inteso come *insieme di simboli che identificano una cifra numerica*. Va però detto che questo discorso non ha valenza universale, e che comunque non considerare relazioni numerose come quelle di dominio porta ad un decremento della percentuale dei termini annotati rispetto al numero degli elementi di uno schema, che si fissa in questo caso su una media del 62%. Di seguito i valori medi di precision e recall su tutte le sorgenti:

AVG PRECISION	AVG RECALL
0,54	0,34

Tabella 12 - Precision e recall medi per profondità dei cammini pari a 2 e configurazione di pesi B

5.7.3 Configurazione C: analisi dei risultati

SCHEMA SORGENTE	LEMMI ESAMINATI	LEMMI DISAMBIGUATI	LEMMI DISAMBIGUATI CORRETTAMENTE	PRECISION	RECALL
Mondial	138	78	53	0,68	0,38
Amalgam1	103	41	10	0,24	0,10
Ontology1	185	142	83	0,58	0,45
Ontology2	205	136	72	0,53	0,35
Ontology3	207	133	70	0,53	0,34

Tabella 13 - Precision e recall per profondità dei cammini pari a 2 e configurazione di pesi C

Questa configurazione, che ricordiamo determina i risultati della disambiguazione utilizzando il contributo della sola relazione di iperonimia/iponimia, ha confermato un interessante risultato: la relazione di iperonimia/iponimia è di gran lunga la più influente per l'attribuzione del significato corretto ad ogni termine. Per avere validità, però, questa affermazione deve essere comprovata da un decremento delle prestazioni nel caso in cui la relazione di iponimia/iperonimia non venisse presa in considerazione. Di seguito i valori medi di recall e precision per la configurazione C:

AVG PRECISION	AVG RECALL
0,51	0,32

Tabella 14 - Precision e recall medi per profondità dei cammini pari a 2 e configurazione di pesi C

5.7.4 Configurazione D: analisi dei risultati

SCHEMA SORGENTE	LEMMI ESAMINATI	LEMMI DISAMBIGUATI	LEMMI DISAMBIGUATI CORRETTAMENTE	PRECISION	RECALL
Mondial	138	114	69	0,61	0,50
Amalgam1	103	42	12	0,29	0,12
Ontology1	185	142	71	0,50	0,38
Ontology2	205	150	58	0,39	0,28
Ontology3	207	155	60	0,39	0,29

Tabella 15 - Precision e recall per profondità dei cammini pari a 2 e configurazione di pesi D

Le ipotesi precedentemente vagliate sono verificate pienamente da questa configurazione, che ricordiamo, tralascia il contributo delle relazioni di iperonimia/iponimia nell'attribuzione del senso corretto. Si nota che la percentuale di termini disambiguati è rimasta pressoché la stessa della configurazione A (70% contro il 72%). Ciò sta a significare che tutte le altre tipologie di relazioni sono sufficienti a creare un numero adeguato di percorsi di interconnessioni tra i synset, ma tali percorsi non sembrano essere quelli corretti. Ecco , in conclusione, la media dei valori di precision e recall, che rappresentano i valori più bassi di tutti i test effettuati sulle 5 sorgenti

AVG PRECISION	AVG RECALL
0,43	0,31

Tabella 16 - Precision e recall medi per profondità dei cammini pari a 2 e configurazione di pesi D

5.7.5 Conclusioni sui risultati del Test 1

Come è già stato descritto nei paragrafi precedenti, gli aspetti fondamentali emersi dall’analisi dei risultati proposti dall’algoritmo in funzione delle diverse tipologie di relazioni considerate, e dei diversi pesi ad esse associate, possono essere così riassunti:

- La configurazione che massimizza il valore di precision è la configurazione B, come si nota in figura 5.7, ossia quella che prende in considerazione esclusivamente le relazioni semantiche.
- La tipologia di relazioni più influente ai fini di una buona disambiguazione dei termini è la relazione di iperonimia/iponimia
- Le relazioni di dominio consentono un’incremento del valore di recall, ma un decremento della precision per le motivazioni precedentemente elencate

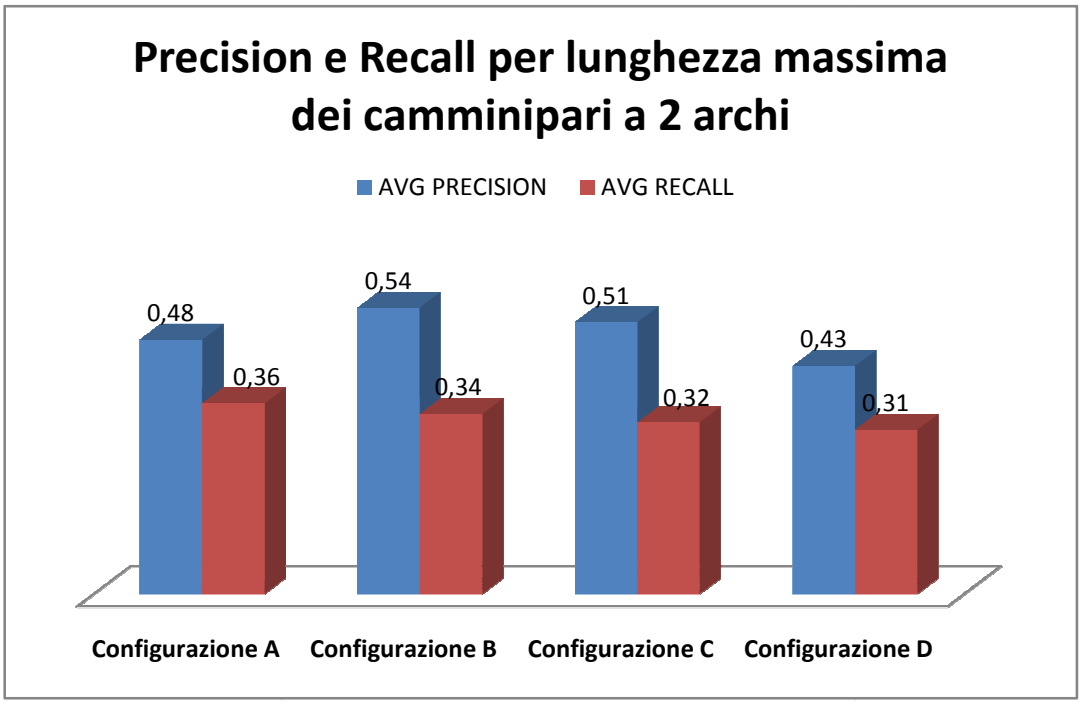

Figura 5.7 – Precision e Recall in funzione delle quattro configurazioni

C'è da specificare, in ultima istanza, che questi valori di precision e recall seppur discreti, non sono del tutto soddisfacenti. Le motivazioni alla base di tali risultati sono state individuate in:

- *carenze di WordNet, sia lessicali che relazionali*. Spesso, infatti, WordNet non contempla relazioni anche evidenti tra due synset, oltre a non poter dare informazioni semantiche su termini composti.
- *uso eccessivo di acronimi e abbreviazioni* nelle denominazioni degli elementi degli schemi, che non consentono di recuperare alcun tipo di relazione significativa.
- *Basso livello di profondità per le relazioni individuate*. Considerare, infatti relazioni soltanto relazioni di secondo livello non consente di individuare percorsi sì più lunghi, ma comunque significativi per una disambiguazione corretta.

Verificheremo la veridicità dell'ultimo punto nel paragrafo seguente, in cui si effettuerà l'analisi delle prestazioni dell'algoritmo utilizzando le stesse configurazioni di pesi indicate nel paragrafo precedente ma un livello massimo di profondità per i cammini individuati tra synset pari a 3.

5.8 Test 2: lunghezza massima cammini pari a 3 archi

La nostra analisi si sposta ora sui risultati ottenuti considerando un livello di profondità massima di relazioni tra i synset pari a 3. Saranno utilizzate le stesse configurazioni descritte nel paragrafo 5.5.

5.8.1 Configurazione A: analisi dei risultati

SCHEMA SORGENTE	LEMMI ESAMINATI	LEMMI DISAMBIGUATI	LEMMI DISAMBIGUATI CORRETTAMENTE	PRECISION	RECALL
Mondial	138	114	91	0,80	0,66
Amalgam1	103	42	11	0,26	0,11
Ontology1	184	164	98	0,60	0,53
Ontology2	205	152	86	0,57	0,42
Ontology3	207	157	89	0,57	0,43

Tabella 17 - Precision e recall per profondità dei cammini pari a 3 e configurazione di pesi A

La prima cosa emerge dai dati in tabella è un incremento sostanziale e generale, sia per quanto riguarda la precision che la recall, delle prestazioni dell’algoritmo. Considerare tre livelli di relazioni infatti, se da un lato rallenta un po’ i tempi di esecuzione, dall’altro offre risultati soddisfacenti, che toccano addirittura l’80% riportato sullo schema mondial. I motivi sono ovvi, in quanto, considerando synset distanti 3 archi da un synset d’origine, è possibile coprire quasi tutte le relazioni significative ai fini della disambiguazione. La recall, per gli stessi motivi, va incontro ad un prevedibile aumento. Se non si considerasse nella media lo schema *Amalgam1* , per il quale valgono le stesse considerazioni fatte nel paragrafo 5.6.1, la precision media salirebbe al 64%. Tuttavia i dati che rappresentano la media effettiva di precision e recall per la configurazione A calcoltata sulle 5 sorgenti sono:

AVG PRECISION	AVG RECALL
0,56	0,43

Tabella 18 - Precision e recall medi per profondità dei cammini pari a 3 e configurazione di pesi A

5.8.2 Configurazione B: analisi dei risultati

SCHEMA SORGENTE	LEMMI ESAMINATI	LEMMI DISAMBIGUATI	LEMMI DISAMBIGUATI CORRETTAMENTE	PRECISION	RECALL
Mondial	138	110	87	0,79	0,63
Amalgam1	103	41	17	0,41	0,17
Ontology1	184	163	95	0,58	0,52
Ontology2	205	151	90	0,60	0,44
Ontology3	207	148	89	0,60	0,43

Tabella 19 - Precision e recall per profondità dei cammini pari a 3 e configurazione di pesi B

Anche in questo caso si confermano le osservazioni fatte nel paragrafo 5.7.2. Sembra cioè definitiva l'ipotesi per cui le relazioni semantiche sono sufficienti ad ottenere delle soddisfacenti prestazioni, considerando sempre le difficoltà maggiori presenti nell'annotazione di una sorgente di dati rispetto ad una annotazione testuale. Questa configurazione che offre, in maniera assoluta, i massimi valori di precision e recall , ed è la configurazione da consigliare ad un eventuale utente che volesse utilizzare come metodo di disambiguazione l'algoritmo di TUCUXI. I valori medi ottenuti sulle sorgenti sono:

AVG PRECISION	AVG RECALL
0,60	0,44

Tabella 20 - Precision e recall medi per profondità dei cammini pari a 3 e configurazione di pesi B

5.8.3 Configurazione C: analisi dei risultati

SCHEMA SORGENTE	LEMMI ESAMINATI	LEMMI DISAMBIGUATI	LEMMI DISAMBIGUATI CORRETTAMENTE	PRECISION	RECALL
Mondial	138	109	86	0,79	0,62
Amalgam1	103	41	12	0,29	0,12
Ontology1	184	163	93	0,57	0,51
Ontology2	205	150	100	0,67	0,49
Ontology3	207	147	97	0,66	0,47

Tabella 21 - Precision e recall per profondità dei cammini pari a 3 e configurazione di pesi C

Anche per quanto riguarda le relazioni di terzo livello l’ipernimia/iponimia si conferma relazione fondamentale per ottenere positive prestazioni dall’algoritmo. Tuttavia questa volta la precisione media è di un punto percentuale inferiore alla configurazione A, nella quale sono comprese tutte le tipologie di relazioni. È probabile che le relazioni di meronimia/olonimia di terzo livello abbiano determinato, questa volta, tale leggero incremento di prestazioni. I livelli di precision e recall medi sono:

AVG PRECISION	AVG RECALL
0,55	0,44

Tabella 22 - Precision e recall medi per profondità dei cammini pari a 3 e configurazione di pesi C

5.8.4 Configurazione D: analisi dei risultati

SCHEMA SORGENTE	LEMMI ESAMINATI	LEMMI DISAMBIGUATI	LEMMI DISAMBIGUATI CORRETTAMENTE	PRECISION	RECALL
Mondial	138	113	69	0,61	0,50
Amalgam1	103	42	14	0,33	0,14
Ontology1	184	167	63	0,38	0,34
Ontology2	205	152	61	0,40	0,30
Ontology3	207	149	58	0,39	0,28

Tabella 23 - Precision e recall per profondità dei cammini pari a 3 e configurazione di pesi D

I dati riportati in tabella evidenziano che pur considerando tre livelli di profondità per le relazioni cercate, non è possibile escludere dalla lista quelle di iponimia/iperonimia se si desiderano ottenere prestazioni soddisfacenti. I valori medi di precision e recall anche in questo test sono i più bassi rinvenuti a parità di livello:

AVG PRECISION	AVG RECALL
0,42	0,31

Tabella 24 - Precision e recall medi per profondità dei cammini pari a 3 e configurazione di pesi

5.8.5 Conclusioni sui risultati del Test 2

L’andamento dell’istogramma che confronta i valori di precision e recall dell’algoritmo in funzione della configurazione di pesi selezionata non si discosta molto dal grafico prodotto per le relazioni di secondo livello. Tranne la configurazione D, che ha ottenuto risultati migliori nel Test 1, le altre tre configurazioni hanno visto un incremento di precision e recall che trova il suo massimo nel + 8% della configurazione A. Le osservazioni da fare sono identiche a quelle del paragrafo 5.6.5, con poche differenze:

- Le relazioni di dominio considerate hanno un effetto meno “fuorviante” nell’attribuzione dei sensi corretti agli elementi da annotare se si selezionano 3 livelli di profondità per le relazioni
- Considerare un livello di profondità delle relazioni tra i synset pari a 3 rappresenta un compromesso accettabile tra la velocità di esecuzione e l’accuratezza dell’algoritmo nel processo di disambiguazione.

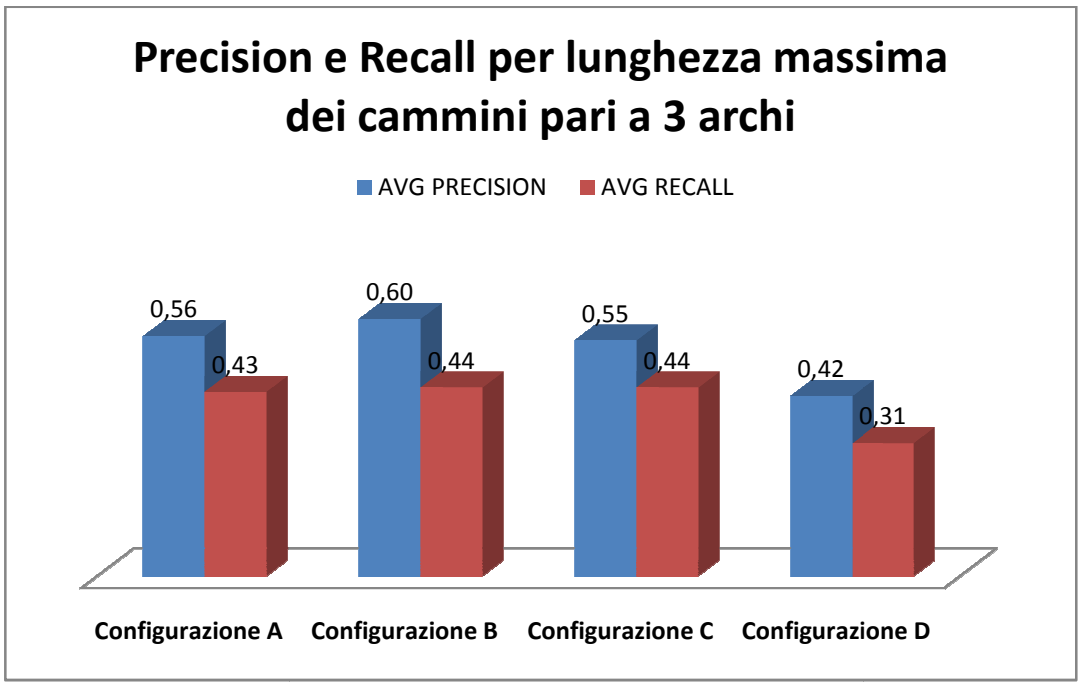

Figura 5.8 – Valori medi di precision e recall per relazioni di III livello in base alle configurazione

Per il resto, altri aspetti come il rilevante ruolo ricoperto dalle relazioni di iperonimia/iponimia (configurazione C) e la poca affidabilità delle altre tipologie di relazioni come unica discriminante (configurazione D), sono rimasti invariati, pur considerando una profondità maggiore di relazioni.

5.9 Valutazione globale delle prestazioni dell'algoritmo

Dal momento che uno degli obiettivi dei test è quello di trovare una configurazione che massimizzi le prestazioni dell'algoritmo, necessario riportare il confronto tra i due test effettuati. Il primo grafico mostra il confronto tra i valori di precision ottenuti nel test 1 con quelli ottenuti nel test 2

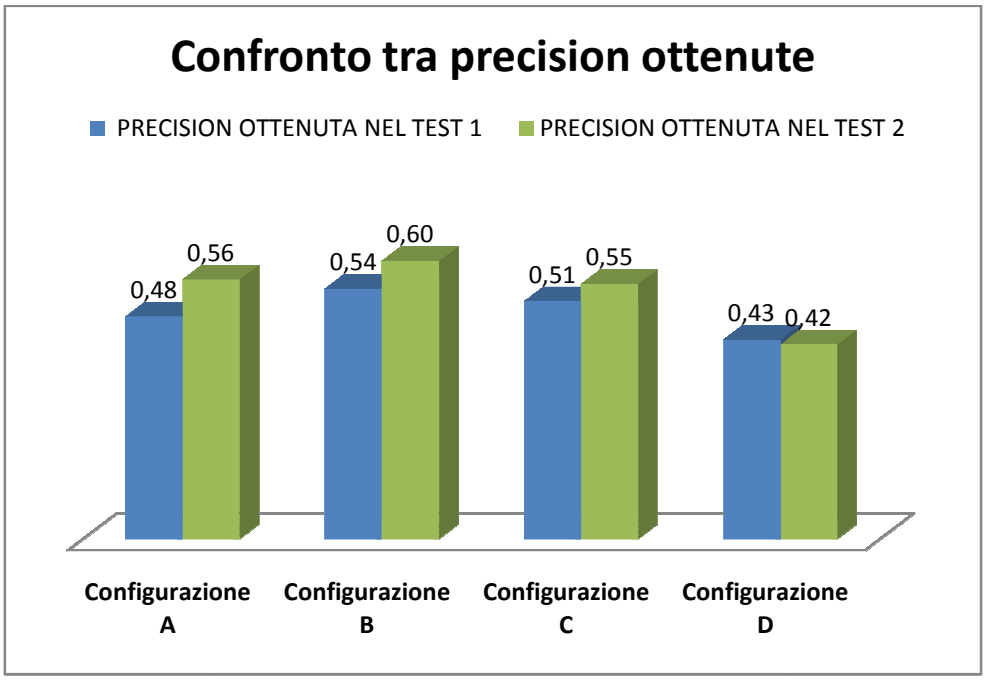

Figura 5.9 – Confronto tra precision ottenute nel test1 e nel test2

Il secondo invece riporta il confronto tra i valori di recall del test 1 e del test2. Si noti che la precision e la recall sono riferite in questi due casi ai valori medi ottenuti, per ogni configurazione di pesi, sulle 5 sorgenti.

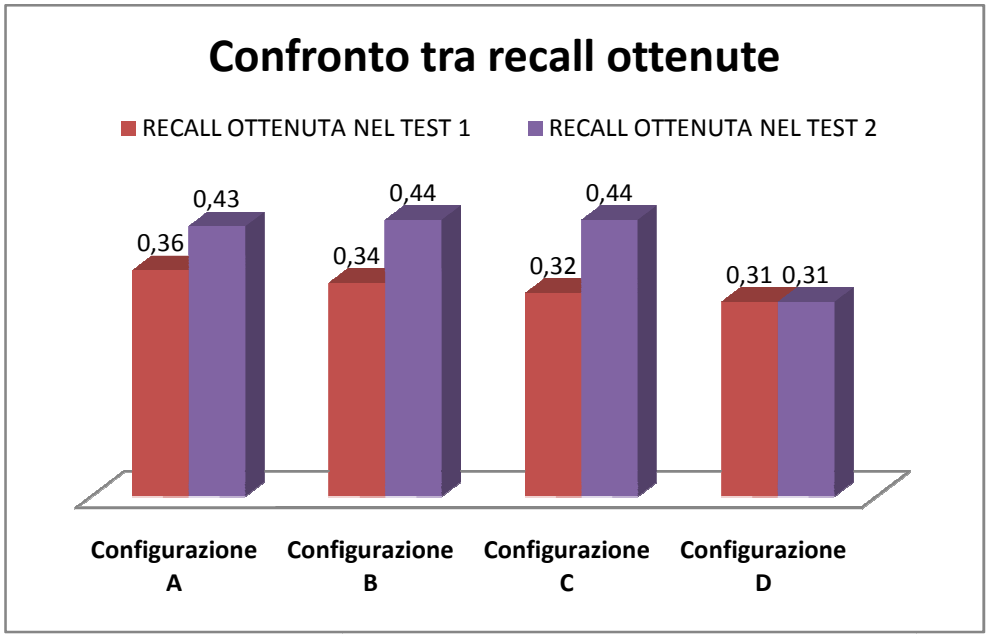

Figura 5.10 – Confronto tra recall ottenute nel test 1 e nel test 2

In entrambi i grafici appare evidente che il test 2 ha fornito risultati più incoraggianti. Ciò significa che cercare percorsi di interconnessione più lunghi tra i synset del *contesto lessicale* porta ad una più efficace disambiguazione. Questo aspetto, che potrebbe essere intuitivo, tuttavia non è sempre verificato. Sono stati riscontrati ad esempio, per alcune sorgenti, valori di precision più elevati considerando solo relazioni di iperonimia/iponimia e addirittura soltanto un livello di relazioni tra i synset.

5.9.1 I limiti interni

Pur avendo conseguito delle buone prestazioni medie, l'algoritmo presenta delle limitazioni, che rappresentano dei punti di partenza per eventuali sviluppi futuri, che possono essere così riassunte:

- Il punteggio che si attribuisce ad un percorso qualsiasi che collega due synset non tiene conto della distanza che intercorre tra i lemmi inseriti nella lista dei termini candidati. Ciò causa disambiguazioni errate nel caso in cui uno stesso lemma abbia connotazioni differenti all'interno di una stessa sorgente dati. Per di più, un errore di disambiguazione commesso su un lemma che occorre più di una volta, si ripete per tutte le occorrenze di tale lemma inficiando le prestazioni dell'algoritmo.
- L'algoritmo, pur considerando le relazioni semantiche principali e la sinonimia, non contempla altri tipi di relazioni derivate, come la similarità o la presenza di un termine da disambiguare all'interno della glossa di un altro termine. Tali relazioni potrebbero essere utili all'annotazione.
- Il punteggio che viene attribuito ad un cammino che connette due synset risulta essere una media dei pesi delle singole relazioni che delineano tale cammino. Definire un meccanismo di *scoring* che consideri le sequenze possibili di relazioni coinvolte in un percorso e ne attribuisca un punteggio ad hoc (strategia adottata dall'algoritmo SSI descritto nel capitolo 2) potrebbe contribuire ad un incremento di prestazioni.

Quelli appena descritti rappresentano i limiti interni, relativi all'implementazione e alla metodologia di ricerca delle relazioni. A questi vanno aggiunti i limiti esterni, che sono definiti nel prossimo paragrafo.

5.9.2 I limiti esterni

I limiti esterni rappresentano quegli aspetti che portano ad un livello non soddisfacente di prestazioni indipendentemente dall’implementazione dell’algoritmo, e possono essere individuati nei seguenti punti:

- L’algoritmo non sfrutta le relazioni strutturali, quelle cioè relative alla realizzazione delle sorgenti di dati. Tale informazione sarebbe utile ai fini dell’annotazione automatica, ma farebbe perdere all’algoritmo il carattere di generalità che lo caratterizza (attualmente può essere applicato indifferentemente a sorgenti di dati o a sorgenti di testo). Tuttavia questa mancanza può essere facilmente ovviata dall’inserimento dell’algoritmo nel tool ALA, che come abbiamo visto, combina metodi differenti, tra cui l’algoritmo SD, per l’annotazione automatica.
- Si verifica una certa carenza di contenuti lessicali e semantici all’interno delle ontologie lessicali di riferimento, che porta a non sfruttare a massimo le potenzialità degli algoritmi di disambiguazione
- Manca, ad oggi, uno standard che permetta di integrare tra loro tali ontologie lessicali per offrirne una visione strutturata e più ricca di informazioni dal punto di vista semantico.

5.9.3 Osservazioni finali

Nonostante sia noto che ogni sorgente, di dati o di testo che sia, presenta delle caratteristiche che si prestano ad un metodo di disambiguazione piuttosto che ad un altro, i risultati ottenuti sono comunque soddisfacenti e hanno portato a galla tre aspetti fondamentali:

- Non è il numero complessivo di relazioni che si riescono ad estrarre tra i synset del *contesto lessicale* che influenza positivamente le prestazioni, ma la tipologia di relazioni. In altre parole conta più la *qualità* delle relazioni estratte che la *quantità*.
- La configurazione B, associata ad una lunghezza pari a 3 archi nella costruzione dei cammini tra i synset, si è rivelata essere la configurazione ottimale, e pertanto sarà proposta come configurazione di base per un utente che desideri effettuare l'annotazione automatica tramite l'algoritmo TUCUXI.
- Le relazioni di domino non sempre sono utili, e pertanto dovrebbero esser estratte soltanto nel caso in cui si vogliano annotare più termini possibili o nel caso in cui si gestiscano sorgenti con un numero di elementi limitato . Tale scelta è supportata anche dall'eccessivo aumento dei tempi di esecuzione che la ricerca di queste relazioni comporta.

La tabella seguente riporta i valori di precision e recall ottenuti con il Test 2, che complessivamente ha fornito i risultati migliori.

La configurazione B, ossia quella che sfrutta esclusivamente le relazioni semantiche, è la configurazione che massimizza i valori medi di precision e recall, fissati rispettivamente al 60% e al 44%.

	Configurazione A	**Configurazione B**	**Configurazione C**	**Configurazione D**
Precision	**56%**	**60%**	**55%**	**43%**
Recall	**43%**	**44%**	**43%**	**32&**

Tabella 26 – Precision e Recall ottenute dal test 2 in base alle diverse configurazioni di peso

In conclusione, è stato realizzato un algoritmo in grado di disambiguare, in media, il 70% dei termini con una precisione del 60%.
Si ritiene che, in cooperazione con gli altri algoritmi inseriti all’interno del tool ALA, l’algoritmo di TUCUXI possa offrire un valido supporto per l’annotazione automatica delle sorgenti di dati di medie dimensioni.

Capitolo 6

Conclusioni e sviluppi futuri

L'obiettivo di questa tesi è stato quello di realizzare una implementazione dell'algoritmo di TUCUXI presentato in (22). TUCUXI rientra nella categoria degli algoritmi non supervisionati di disambiguazione lessicale, ossia quegli algoritmi che, data una sorgente, dati o testuale, si pongono l'obiettivo di attribuire ad ogni termine, espresso in linguaggio naturale, il significato più corretto a seconda del contesto in cui è inserito.

Il processo di disambiguazione lessicale in questa trattazione è stato riferito esclusivamente a sorgenti di dati strutturate e semi-strutturate , ed avviene in maniera del tutto automatica, prendendo il nome di annotazione lessicale automatica.

L'algoritmo è infatti stato concepito per offrire un supporto ai sistemi di integrazione intelligente delle informazioni (I^3), ed è stato realizzato nell'ambito del progetto MOMIS, un sistema che consente di integrare sorgenti informative eterogenee tra loro nella struttura e nella rappresentazione interna dei dati per offrirne una visione globale e strutturata. Il processo di integrazione vede una delle sue fasi salienti nella *Estrazione della conoscenza lessicale*, ossia nella fase che consente di individuare le affinità lessicali tra sorgenti eterogenee. Risolvere le ambiguità presenti tra i termini utilizzati dai progettisti per denominare gli elementi di ogni schema è un passo fondamentale per la creazione dei *mapping semantici*, e in MOMIS tale compito è demandato ad un modulo denominato ALA, che sfrutta un approccio probabilistico alla disambiguazione. Al momento il tool è composto da 5 algoritmi, ognuno dei quali sfrutta un approccio diverso alla disambiguazione, ma manca tra

questi un metodo che si basi sulle relazioni semantiche presenti tra i termini da disambiguare.

Tale metodo è utilizzato dall'algoritmo TUCUXI, che, sfruttando la proprietà lessicale di *coesione* caratteristica del linguaggio naturale, consente, grazie alle conoscenze semantiche estratte da un'ontologia lessicale di nome WordNet, di individuare tutti i cammini che collegano i diversi significati dei termini presenti all'interno di un contesto per costruirne una *mappa semantica*. Tramite l'uso di un meccanismo di scoring, viene attribuito un peso ad ognuno di questi cammini. Tale peso rappresenta il *valore di coesione* che un significato ha con il contesto in cui è inserito. L'output prodotto dall'algoritmo sarà una selezione dei sensi considerati maggiormente connessi a tale contesto.

L'algoritmo è stato testato su diverse sorgenti di dati, e sono state valutate diverse configurazioni di utilizzo, che hanno portato alla selezione della modalità che consente un maggior livello di prestazioni in termini di *precision* e *recall.* Sono infine stati analizzati i limiti, interni ed esterni di tale algoritmo e valutati i possibili sviluppi tesi a migliorane l'accuratezza e l'efficienza. Si ritiene, comunque, che l'inserimento di TUCUXI all'interno del pool di algoritmi di annotazione automatica già disponibili in ALA, possa fornire uno strumento in più per garantire una corretta integrazione semantica delle sorgenti utilizzate da MOMIS.

Attualmente, a causa della complessità del linguaggio naturale, non è ancora stato possibile sviluppare un algoritmo di WSD in grado di annotare i termini di una sorgente con precision e recall pari al 100%. Tuttavia gli incoraggianti risultati ottenuti dagli approcci LKB e CWSD, potrebbero determinare in futuro un incremento delle prestazioni in questo senso.

Bibliografia

1. **R. Hull, R. King et altri.** Arpa I3 reference architecture. *Available at http://www.isse.gmu.edu/I3 Arch/ index.html.* 1995.

2. **Gruber, T. R.** A translation approach to portable ontologies. *Knowledge Acquisition.* 1993.

3. **Miller, Gorge A.** WordNet: a lexical database for english. *Communications for the ACM.* 1995.

4. **F.Niero.** WordNet e sue applicazioni.Revisione e implementazione di un database di termini matematici. *Tesi di Laurea Ingegneria Informatica.* 2005 2006.

5. **S., Sorrentino.** Metodi di disambiguazione del testo ed estensioni di WordNet nel sistema Momis. *Tesi di laurea specialistica Ingegneria Informatica.* Modena disponibile sul sito www.dbgroup.unimo.it : s.n., 2006.

6. **A. M. Gliozzo, C. Strapparava, and I. Dagan.** Unsupervised and supervised exploitation of semantic domains in lexical disambiguation. *Computer Speech & Language.* 2004.

7. **Gliozzo, A.M., Giuliano, C., Strapparava, C.** Domain kernels for word sense disambiguation. *ACL, The Association for Computer Linguistics.* 2005.

8. **Magnini, B., Strapparava, C., Pezzulo, G., Gliozzo, A.** The role of domain information in word sense disambiguation. *Natural Language Engineering, special issue on Word Sense Disambiguation.* 2002.

9. **A.Gliozzo, C.Strapparava, I. Dagan.** Unsupervised e Supervised Exploitation of Semantic Domains in Lexical Disambiguation. 2002.

10. **Navigli R., Velardi P.** Structural Semantic Interconnection: A knowledge-based approach to Word Sense Disambiguation. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE.* JULY 2005. Vol. 27, 7.

11. **Wilks, Y.** A Preferential Pattern-Seeking Semantics for Natural Language Inference. *Artificial Intelligence.* 1978. Vol. 6.

12. **R. Schank, R. Abelson.** Scripts, Plans, Goals, and Understanding. Hillsdale, N.J. : Lawrence Erlbaum, 1977.

13. **R. Krovetz, W.B. Croft.** Word Sense Disambiguation Using Machine Readable Dictionaries. *Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval.* 1989.

14. **Yarowski, D.** Word-Sense Disambiguation Using Statistical Model of the Roget's Categories Trained on Large Corpora. *Int'l Conf. Computational Linguistics (COLING-92).* 1992.

15. **W. Gale, K. Church, D. Yarowsky.** One Sense per Discourse. *Proc. DARPA Speech and Natural Language Workshop.* 1992.

16. **W. Gale, K. Church, and D. Yarowsky.** A Method for Disambiguating Word Senses in a Corpus. *Computer and the Humanities.* 1992.

17. **Fu, K.S.** Syntactic Pattern Recognition and Applications. *Englewood Cliffs, N.J.* s.l. : Prentice Hall, 1982.

18. **B. Magnini, G. Cavaglia.** Integrating Subject Field Codes into WordNet. *Proc. Second Int'l Conf. Language Resources and Evaluation.* 2000.

19. **Lea, D.** Oxford Collocations. s.l. : Oxford Univ. Press, 2002.

20. **Longman, K.** Longman Language Activator. s.l. : Pearson Education.

21. **R. Navigli, P. Velardi.** Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites. *Computational Linguistics* . s.l. : MIT Press, June 2004.

22. **R.Benassi, S.Bergamaschi, M.Vincini.** TUCUXI: the Intelligent Hunter Agent for Concept Understanding and LeXical ChaIning.

23. **R.Hasan, K.Halliday e.** An Introduction to Functional Grammar. Londra : Edward Arnold. Vol. 1985.

24. **R. Barzilay, M.Elhadad, in.** Using Lexical Chains for Text Summarization. *ACL/Eacl-97 summarization workshop, pages 10-18, Madrid 1997.* Madrid : s.n., 1997.

25. **M.A.K Halliday, R.Hasan.** Cohesion in English. London : Longman, 1976.

26. **J.Morris, G.Hirst.** Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text. *Computational Linguistic.* 1991.

27. **S. Bergamaschi, S. Castano, S. De Capitani di Vimercati, S. Montanari, M.Vincini.** An intelligent approach to information integration.

Proceedings of the International Conference on Formal Ontology in Information Systems. Trento : s.n., 1998.

28. **S. Bergamaschi, S. Castano, S. De Capitani di Vimercati, S. Montanari, M.Vincini**.Exploiting schema knowledge for the integration of heterogeneous sources. *Sesto Convegno Nazionale su Sistemi Evoluti per Basi di Dati SEBD1998.* Ancona : s.n., 1998.

29.**G. Wiederhold et altri.** Integrating Artificial Intelligence and DataBase Technology . *Journal of Intelligent Information Systems.* 1996. Vol. 2/3.

30. **F. Saltor, E. Rodriguez.** On intelligent access to heterogeneous information. *Proceedings of the 4th KRDB Workshop.* Athens, Greece : s.n., Agosto 1997.

31. **Guarino, N.** Semantic matching: Formal ontological distinctions for information organization, extraction and integration. *Technical Report, Summer School on Information Extraction.* Frascati : s.n., 1997.

32. **Bergamaschi, S., Castano, S., Beneventano, D., Vincini, M.** Semantic integration of heterogeneous information sources. *Journal of Data and Knowledge Engineering.* 2001. 36.

33. **Domenico Beneventano, Sonia Bergamaschi, Claudio Sartori, Maurizio Vincini.** ODB-QOPTIMIZER: A tool for semantic query optimization in oodb . *Int. Conference on Data Engineering - ICDE97.* 1997.

34. **D. Beneventano, S. Bergamaschi, C. Sartori, and M. Vincini.** Odbtools: a description logics based tool for schema validation and semantic query optimization in object oriented databases. *In Sesto Convegno AIIA.* Roma : s.n., 1997.

35. **Antonellis, S. Castano and V. De.** Deriving global conceptual views from multiple information sources . *preProc. of ER'97 Preconference Symposium on Conceptual Modeling,Historical Perspectives and Future Directions.* 1997.

36. **Cattel, R.G.G.** The object Database Standard: ODMG93. San Francisco, CA : Morgan Kaufman, 1997.

37. **Bergamaschi, S., Castano, S., Vincini, M.** Semantic integration of semistructured and structured data sources. s.l. : SIGMOD Record, 1999. Vol. 28, 1.

38. **Everitt., B.** Computer-Aided Database Design: the DATAID Project. *Heinemann Educational Books Ltd.* s.l. : Social Science Research Council, 1974.

39. **V.Guidetti.** Intelligent information integration system: Extending a lexicon ontology. *Master thesis in Computer Science, Università di Modena e Reggio Emilia.* 2002.

40. **Dong, X. L.** Data integration with uncertainty. *VLDB.* 2007.

41. **F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-match.** an algorithm and an implementation of semantic matching. *Semantic Interoperability and Integration.* 2005.

42. **Po, L.** Automatic Lexical Annotation: an effective technique for dynamic data integration. *PhD Thesis.* s.l. : Available at http://www.dbgroup.unimo.it/ po/., 2009.

43. **S. Bergamaschi, P. Bouquet, D. Giacomuzzi, F. Guerra, L. Po, M. Vincini.** An incremental method for the lexical annotation of domain ontologies. *J. Semantic Web Inf. Syst.* 2007.

44. **S. Bergamaschi, L. Po, and S. Sorrentino.** Automatic annotation for mapping discovery in data integration systems. [aut. libro] I. Infantino, and D. Saccµ S. Gaglio. *SEBD.* 2008.

45. **Rigau G., Atserias J., Agirre E.** Combining unsupervised lexical knowledgemethods for word sense disambiguation . *CoRR cmp-lg/9704007.* 1997.

46. **Mihalcea, R., Moldovan, D.I.:.** An iterative approach to word sense disambiguation. [aut. libro] J.N., Manaris, B.Z. Etheredge. *AAAI Press.* s.l. : FLAIRS Conference, 2000.

47. **Pahikkala, T., Pyysalo, S., Ginter, F., Boberg, J., JÄarvinen, J., Salakoski, T.** Kernels incorporating word positional information in natural language disambiguation. [aut. libro] I., Markov, Z. In Russell. *AAAI Press* . s.l. : FLAIRS Conference, 2005.

48. **Banerjee, S., Pedersen, T.** An adapted lesk algorithm for word sense disambiguation using wordnet. [aut. libro] A.F Gelbukh. *Lecture Notes in Computer Science.* s.l. : CICLing, 2002.

49. **Magnini, B.** *Experiments in word domain disambiguation for parallel texts.* 2000.

50. **D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini.** Synthesizing an integrated ontology. *IEEE Internet Computing.* Sep-Oct 2003.

51. **D. Beneventano, S. Bergamaschi, and C. Sartori.** Description logics for semantic query optimization in object-oriented database systems. *ACM Trans. Database Syst.* 2003.

52. **Shafer, G.** A Mathematical Theory of Evidence. *Princeton University Press.* 1976.

53. **Fellbaum, C., Miller, G.** WordNet: An electronic lexical database. *The MIT Press* . 1998.

54. **Yarowsky, P. Resnik and D.** Distinguishing systems and distinguishing senses: new evaluation methods for word sense disambiguation. *Natural Language Engineering.* 2000.

55. **B.Magnini, C.Strapparava, G.Pezzullo e A.Ghiozzo.** Comparing Ontology-based and Corpus-Based Domain Annotation in WordNet. 2002.

56. **R.Mihalcea, Dan I Moldovan.** EXtended WordNet: progress report. *Proceeding NAACL Workshop on WordNet and other Lexical Reources.* Pittsburg, PA : s.n., 2001.

Grazie alla professoressa Sonia Bergamaschi e alla dottoressa Serena Sorrentino per l'aiuto fornito durante la stesura della presente tesi.
I ringraziamenti più sentiti vanno, però, ai miei genitori, che mi hanno permesso di iniziare, proseguire e, con il dovuto ritardo, portare a termine il percorso di studi intrapreso.
Un ringraziamento speciale va a mia nonna Grazia, che mi è sempre stata vicina: grazie ai ceri accesi in voto ad un numero imprecisato di Santi mi ha fornito, in alcune circostanze, un aiuto quasi miracoloso.
Grazie a Giulia, che, ormai alla fine della carriera universitaria, mi ha dato quella serenità indispensabile per gli ultimi sforzi prima della meta.
Adesso passiamo agli amici. Con ognuno di loro ho condiviso un periodo della mia vita, nel bene, nel male e specialmente nelle immancabili e salutari perdite di tempo. Li ringrazio tutti indistintamente, ma per elencarli non basterebbe una pagina intera. Mi limito, pertanto, a citare soltanto due amici fraterni, che conosco ormai da una vita, Lorenzo e Magoo, la cui presenza e il cui costante supporto non è mai stato messo neanche in discussione, nonostante le mie solite mancanze.
In ultimo volevo ringraziare tutto il tempo, non perso, ma dedicato ad altre inutili attività extrauniversitarie: d'altronde, quando si giunge alla fine di un percorso, tutte le sue tappe acquistano quasi un senso…

"…non insegnate ai bambini
non insegnate la vostra morale…
è così stanca e malata
potrebbe far male...
forse una grave imprudenza
è lasciarli in balia di una falsa coscienza…"

Giorgio Gaber