UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Facoltà di Ingegneria – Sede di Modena

Corso di Laurea in Ingegneria Informatica

Rimozione dell'ambiguità nell'interazione tra *WordNet* e il sistema MOMIS

Relatore

Chiar.mo Prof. Sonia Bergamaschi

Tesi di Laurea di

Salvatore Ricciardi

Correlatore

Dott. Ing. Domenico Beneventano

Controrelatore

Chiar.mo Prof. Flavio Bonfatti

Anno accademico 1999 – 2000

Parole chiave:

Computazione lessicale
Intelligenza Artificiale
Integrazione
Ambiguità
WordNet

Ai miei genitori

Ringraziamenti:

Ringrazio la Professoressa Sonia Bergamaschi e l'Ing. Alberto Corni per l'aiuto fornito durante la realizzazione della presente tesi
Un ringraziamento speciale va ai miei genitori, che hanno reso tutto ciò possibile.

Indice

Elenco delle figure

Introduzione

A cosa serve l'informazione se non si è in grado di sfruttarla? O meglio, a cosa serve un'elevata quantità di dati, situata su più sistemi eterogenei se non si è capaci di integrarli? A nulla: per accedere al valore aggiunto offerto dall'enorme quantità di informazione occorrono strumenti che siano in grado di integrare questi dati in modo da rendere trasparente la struttura dei sistemi su cui queste informazioni risiedono, e quindi, anche delle operazioni da compiere per interrogarli.

Il *Wordld Wide Web* costituisce l'esempio attualmente più famoso di scambio di informazione: milioni di computer connessi tra loro si scambiano, ventiquattro ore su ventiquattro e sette giorni su sette, una quantità impressionante di informazioni che rischiano di perdere il loro reale valore se non si è in grado di sfruttarle.

La possibilità di sviluppare applicazioni capaci di combinare ed utilizzare dati provenienti da una molteplicità di sorgenti è un tema di grande attualità, di interesse non solo teorico ma anche applicativo, come dimostra la sempre maggiore presenza di sistemi commerciali, quali i *Datawarehouse*, i *Dataminer*, i Sistemi di *Workflow*, ecc.

Alcune soluzioni a questo problema possono essere quelle di integrare tutto il patrimonio informativo facendolo migrare in un "unico" sistema, il che significa cambiare o potenziare la piattaforma *hardware*. Lo sforzo da sopportare in questo caso è notevole, infatti, si tratta di rivoluzionare il sistema informativo pur sapendo che tale soluzione non potrà essere definitiva poiché l'eventuale aggiunta di nuove fonti d'informazione ripresenterebbe lo stesso problema.

L'alternativa, è quella di creare un livello d'astrazione tale da nascondere completamente la natura e il formato delle fonti d'informazioni; in questo modo i dati restano dove sono, mentre, viene resa generale la tecnica di accesso ai dati.

Quest'ultima, è la soluzione adottata da MOMIS (Mediator EnvirOment for Multiple Information Sources), progetto nato all'interno del MURST 40% INTERDATA come collaborazione tra l'unità operativa dell'università di Modena e l'unità operativa dell'università di Milano. Lo strato di codice che normalizza le differenze tra le diverse fonti informative, nel sistema MOMIS, è rappresentato dalla vista integrata. In questo modo si crea un'astrazione dalle differenze sottostanti,

senza però snaturare la struttura fisica del nostro sistema permettendo un migliore sfruttamento delle risorse e, in particolare, dell'informazione.

Il primo passo verso la creazione della vista integrata è quella di generare un *thesaurus* comune di relazioni terminologiche; per ottenerlo utilizzeremo i seguenti strumenti *software*: ODB-TOOL (*ObjectDataBase-TOOL* sviluppato presso il dipartimento de Scienza dell'Ingegneria dell'università di Modena) utile per l'acquisizione e la verifica di consistenza delle basi di dati, il linguaggio dichiarativo $ODL_{\mathbf{I}^3}$ - estensione del linguaggio ODL (*Object* Data Language proposto dal gruppo di standardizzazione ODMG) per la specifica di schemi ad oggetti, ed infine il *database* lessicale *WordNet* (sviluppato dal *Cognitive Science Laboratory* dell'università di Princeton sotto la direzione del Professor George A. Miller).

In questa tesi si approfondirà il componente che, attraverso l'utilizzo di *WordNet*, estrae le relazioni lessicali, ed in particolare lo scopo di questa tesi è l'ideazione, progettazione e realizzazione di un modulo software, ARM acronimo di ***A**mbiguity **R**emoving **M**odule*. Il compito di ARM è quello di fornire degli strumenti al progettista tale da rendere il processo di rimozione dell'ambiguità delle relazioni lessicali più automatico: ambiguità dovuta alla proprietà di polisemia delle parole vale a dire dei termini (nomi di classi e attributi) presenti nelle varie sorgenti.

La struttura della tesi è la seguente:

nel primo capitolo si introducono le problematiche che un sistema di Integrazione deve affrontare. È riportata una classsificazione di questi problemi ed un'architettura di riferimento che questo tipo di sistemi dovrebbero seguire;

invece nel secondo capitolo viene descritto il sistema MOMIS, in particolare la sua architettura. Viene inoltre data una descrizione del linguaggio $ODL_{\mathbf{I}^3}$ usato per descrivere le nostre sorgenti;

nel terzo capitolo vengono descritti gli strumenti software utilizzati, come il *database* lessicale *WordNet;*

nel Capitolo 4 vengono illustrate le fasi per la creazione dello schema integrato;

la parte centrale di questa tesi è il capitolo in cui sono descritti gli strumenti realizzati per la disambiguazione dei termini;

infine, nell'ultimo, il sesto, viene descritto il *Tool* grafico ARM d'ausilio all'utente nella rimozione dell'ambiguità.

Il codice è la documentazione non sono allegati ma sono comunque reperibili sul server sparc20.dsi.unimo.it presso il Dipartimento di Scienze dell'Ingegneria, nella *directory* comune dedicata allo sviluppo del progetto MOMIS:

codice:

/export/home/progetti.comuni/Momis/prototype/modules/SIDesigner/ARM

documentazione:

http://sparc20.dsi.unimo.it/Momis/prototipo/docsOnline/sources/

La tesi è correlata delle seguenti appendici:

- Glossario di *WordNet*: utile per districarsi nella terminologia di *WordNet* mutuata dalla linguistica.
- Esempio di riferimento in $ODL_{I^{3}}$: codice dell'esempio di riferimento utilizzato in questa tesi.

Il progetto **MOMIS**, in cui si inserisce questa tesi, è stato presentato all'ultima conferenza internazionale *Very Large DataBase* {VLDB2000}, Cairo (Egitto), 10-14 settembre 2000.

Capitolo 1

L'Integrazione dell'Informazione

Da qualche tempo ci si è accorti che i documenti che un utente gestisce sono sempre più eterogenei: ai tradizionali *database,* che talvolta possono risiedere su piattaforme *hardware*/*software* diverse, si aggiungono i messaggi di posta elettronica, il *download* di file da *web*, documenti di testo e fogli elettronici. Questa diversità di fonti può, di fatto, scaturire sia per l'aggiunta di nuove strumentazioni a quelle esistenti, sia per la necessità di integrare due o più sistemi informativi diversi (come nel caso in cui due banche decidono di fondersi in un unico istituto bancario). In questi casi, ci si può trovare di fronte ad un sistema informativo in cui coesistano basi di dati di vario tipo: relazionali, gerarchiche, reticolari o ad oggetti, e spesso accade che ogni base di dati contenga informazioni d'utilità comuni che dovrebbero essere relazionate alle altre basi di dati. L'informazione si presenta quindi "frammentata" tra diversi *database.*

1.1 La necessità di un'informazione integrata

Se valutiamo l'efficienza di un sistema informativo in base alla sua capacità di minimizzare i costi per produrre ed elaborare informazioni, dobbiamo considerare che ciò dipenderà dalle seguenti variabili:

- Tempo necessario a reperire l'informazione
- Tipo d'elaborazione che l'informazione necessita
- Complessità dell'informazione

Risulta evidente che in presenza di *database* eterogenei, elaborare e produrre un'informazione "completa" sarà più costoso rispetto al caso in cui l'informazione risieda in un solo database. Da qui l'esigenza di sperimentare prodotti e tecniche per

collegare e far in qualche modo comunicare le diverse basi di dati, allo scopo di ottenere un’informazione integrata.

Al momento oggetto di studi intensivi, tale scenario che coinvolge diverse aree di ricerca e di applicazione, si arricchisce con l’inserimento dei sistemi di supporto alle decisioni (DSS, *Decision Support System*), dell’integrazione di basi di dati eterogenee, dei *datawarehouse* (magazzino), fino a comprendere i sistemi distribuiti. I *decision maker* lavorano su fonti diverse (inclusi *file system*, base di dati, librerie digitali, …) ma sono per lo più incapaci di ottenere e fondere le informazioni in modo efficiente.

L’integrazione di basi di dati, e tutto ciò che va sotto il nome di *datawarehouse*, si occupa di materializzare presso l’utente finale delle viste, ovvero delle porzioni di sorgenti replicando fisicamente i dati ed affidandosi a complicati algoritmi di “mantenimento“ di essi, per assicurare la loro consistenza a fronte di cambiamenti nelle sorgenti originali.

Con il termine *Integration Information* invece, come descritto in [5], si indicano in letteratura tutti quei sistemi che, basandosi sulle descrizioni dei dati, sono in grado di combinare tra loro dati provenienti da intere sorgenti o parti selezionate di esse, senza far uso di replicazione fisica delle informazioni. Per ottenere i risultati voluti, l’integrazione richiede conoscenza ed intelligenza volte all’individuazione delle sorgenti e dei dati nonché alla loro fusione e sintesi. Quando tale sistema di integrazione utilizza tecniche di Intelligenza Artificiale, sfruttando le conoscenze acquisite, possiamo parlare di *Intelligent Integration of Information* (I^3). Questa forma di integrazione si distingue dalle tecniche tradizionali poiché, prima di aggregare le informazioni, il sistema ne aumenta il valore analizzando i dati che le rappresentano, ottenendo nuove informazioni da sfruttare per una migliore integrazione.

1.2 L’Intelligenza artificiale

Con questi obiettivi si è quindi inserita, nell’ambito dell’integrazione, l’Intelligenza Artificiale (*IA*), che già aveva dato buoni risultati in domini applicativi più limitati. È ovvio che è pressoché impossibile pensare ad un sistema che vada bene per tutti i domini

applicativi. Per sistemi più ampi è stata proposta una suddivisione dei servizi e delle risorse che si articola in due dimensioni:

- Orizzontalmente su 3 livelli:
  - Livello utente;
  - Livello intermedio, in cui sono presenti i moduli che fanno uso di tecniche di *IA;*
  - Livello dati, dove vengono gestite le risorse di dati;

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

In questo quadro si inserisce il progetto di ricerca I^3 [6] creato e sponsorizzato, nel 1992, dall'ARPA, (*Advanced Research Projects Agency*) agenzia che fa capo al Dipartimento di Difesa degli Stati Uniti. I^3 si concentra sul livello intermedio della partizione sopra descritta, vale a dire, quello che media tra gli utenti e le sorgenti. All'interno di questo livello vi saranno diversi moduli, di cui ricordiamo i più importanti sono:

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

1.2.1 Architettura di riferimento per sistemi I^3

L'architettura di riferimento I^3, di cui si discuterà in questo paragrafo, rappresenta una sommaria categorizzazione dei principi e dei servizi che possono e

che devono essere usati nella realizzazione di un *integratore intelligente di informazioni* derivanti da fonti eterogenee. Come si vede in figura 1.1 essa è composta da cinque famiglie, di cui la più rilevante è quella individuata dai Servizi di Coordinamento. Questi ultimi giocano due ruoli:

- localizzano altri servizi I^3 e fonti di informazioni che possono essere utilizzate per costruire il sistema stesso;
- sono responsabili dell'individuazione ed invocazione a *run-tim*e degli altri servizi, necessari a dare risposta ad una specifica richiesta di dati.

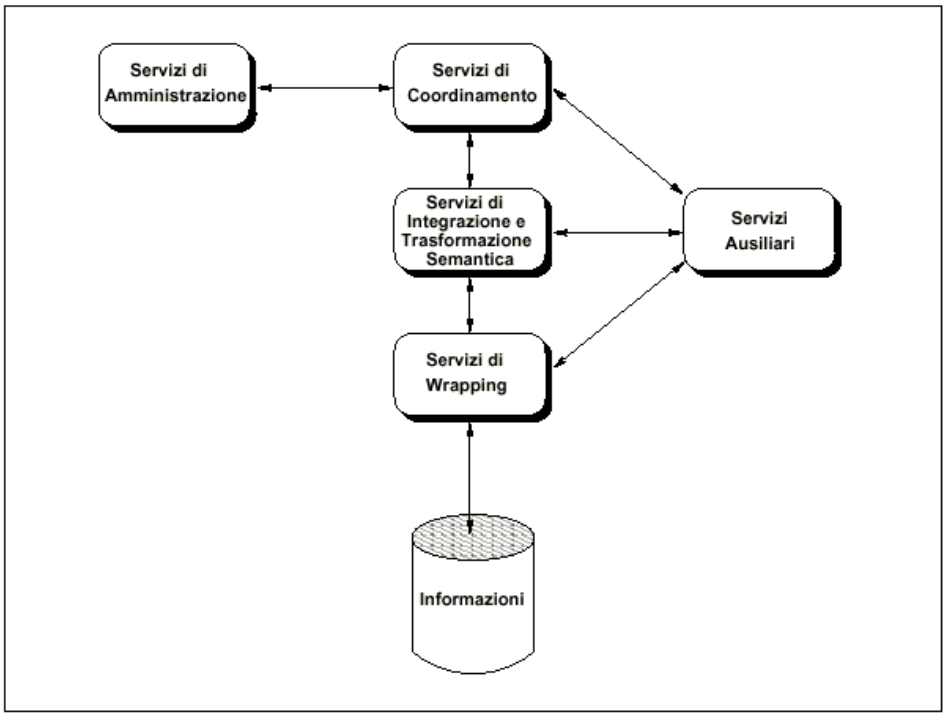


Figura 1.1: Diagramma dei servizi I^3

Importante è analizzare i due assi della figura, orizzontale e verticale, che sottolineano i differenti compiti dei servizi I^3.

Percorrendo l'asse verticale, è possibile capire come avviene lo scambio di informazioni nel sistema: i servizi di *Wrapping* provvedono ad estrarre le informazioni dalle sorgenti, queste saranno poi integrate dai servizi di Integrazione e Trasformazione Semantica per poi essere trasmesse ai servizi di Coordinamento che ne hanno fatto richiesta.

Seguendo l'asse orizzontale, si rende evidente il rapporto tra i servizi di Coordinamento ed Amministrazione, che hanno il compito di individuare le sorgenti

e i servizi. I servizi ausiliari invece, responsabili dei servizi di arricchimento semantico delle sorgenti, forniscono le funzionalità di supporto.

I **Servizi di Amministrazione** sono utilizzati dai Servizi di Coordinamento per: localizzare le sorgenti, determinare le loro capacità, creare ed interpretare i *Template* (strutture dati che descrivono i servizi e i moduli da utilizzare per trattare un insieme di sorgenti). I *Template* servono per ridurre al minimo le possibilità di decisione del sistema, consentendo di definire a priori le azioni da eseguire a fronte di una determinata richiesta. Come alternativa ad essi, sono utilizzate le *Yellow Pages*: servizi di *directory* che mantengono le informazioni sul contenuto delle varie sorgenti e sul loro stato (attiva, inattiva, occupata).

I **Servizi di Integrazione e Trasformazione Semantica** hanno come input una o più sorgenti di dati tradotte dai servizi di *Wrapping*, e, come output, la “vista” integrata o trasformata di queste informazioni. Essi vengono indicati spesso come servizi di mediazione. I principali sono:

- *Servizi di Integrazione di Schemi:* creano il vocabolario e le ontologie condivise dalle sorgenti, integrano gli schemi in una visione globale, mantengono il *mapping* tra schemi globali e sorgenti;
- *Servizi di Integrazione di Informazion*i: aggregano, riassumono ed astraggono le risposte di più sotto-*query* per fornire un’unica risposta alla *query* originale;
- *Servizi di Supporto al processo di integrazione*: sono utilizzati quando una *query* deve essere scomposta in più sotto-*query* da inviare a fonti differenti, con la necessità di integrare poi i loro risultati.

I **Servizi di *Wrapping*** fungono da interfaccia tra il sistema integratore e le singole sorgenti rendendo omogenee le informazioni. Si comportano come dei traduttori dai sistemi locali ai servizi di alto livello dell’integratore. Il loro obiettivo è, quindi, di standardizzare il processo di *wrapping* delle sorgenti, permettendo la creazione di una libreria di fonti accessibili; inoltre, il processo di realizzazione di un

wrapper dovrebbe essere standardizzato, in modo da poter essere riutilizzato per altre fonti.

I **Servizi Ausiliari** aumentano le funzionalità degli altri servizi e sono utilizzati prevalentemente dai moduli che agiscono direttamente sulle informazioni; essi vanno dai semplici servizi di monitoraggio del sistema ai servizi di propagazione degli aggiornamenti e di ottimizzazione.

1.3 Il Mediatore

Questa tesi fa parte di un progetto di ricerca più ampio che ha come obiettivo la progettazione di un ***mediatore***, ovvero del modulo intermedio dell’architettura, precedentemente descritta che si pone tra l’utente e le sorgenti di informazioni. Secondo la definizione proposta da Wiederhold in [7] “un mediatore è un modulo software che sfrutta la conoscenza su un certo insieme di dati per creare informazioni per un’applicazione di livello superiore. Dovrebbe essere piccolo e semplice, così da poter essere amministrato da uno o al più, da pochi esperti.”

Compiti di un *mediatore* sono:

- Assicurare un servizio stabile, anche quando cambiano le risorse;
- Amministrare e risolvere le eterogeneità delle diverse fonti;
- Integrare le informazioni ricavate da più risorse;
- Presentare all’utente le informazioni attraverso un modello scelto dall’utente stesso.

L’approccio architetturale adottato è quello *classico*, che consta principalmente di 3 livelli:

1. utente - attraverso un’interfaccia grafica l’utente pone delle *query* su uno schema globale e riceve un’unica risposta, come se stesse interrogando un'unica sorgente di informazioni;
2. mediatore – il mediatore gestisce l’interrogazione dell’utente, combinando, integrando ed eventualmente arricchendo i dati ricevuti dai *wrapper*, ma

usando un modello (e quindi un linguaggio interrogatore) comune a tutte le fonti;

3. *wrapper* – ogni *wrapper* gestisce una singola sorgente, ed ha una duplice funzione: da un lato converte le richieste del mediatore in una forma comprensibile dalla sorgente, dall'altro traduce informazioni estratte dalla sorgente nel modulo usato dal mediatore.

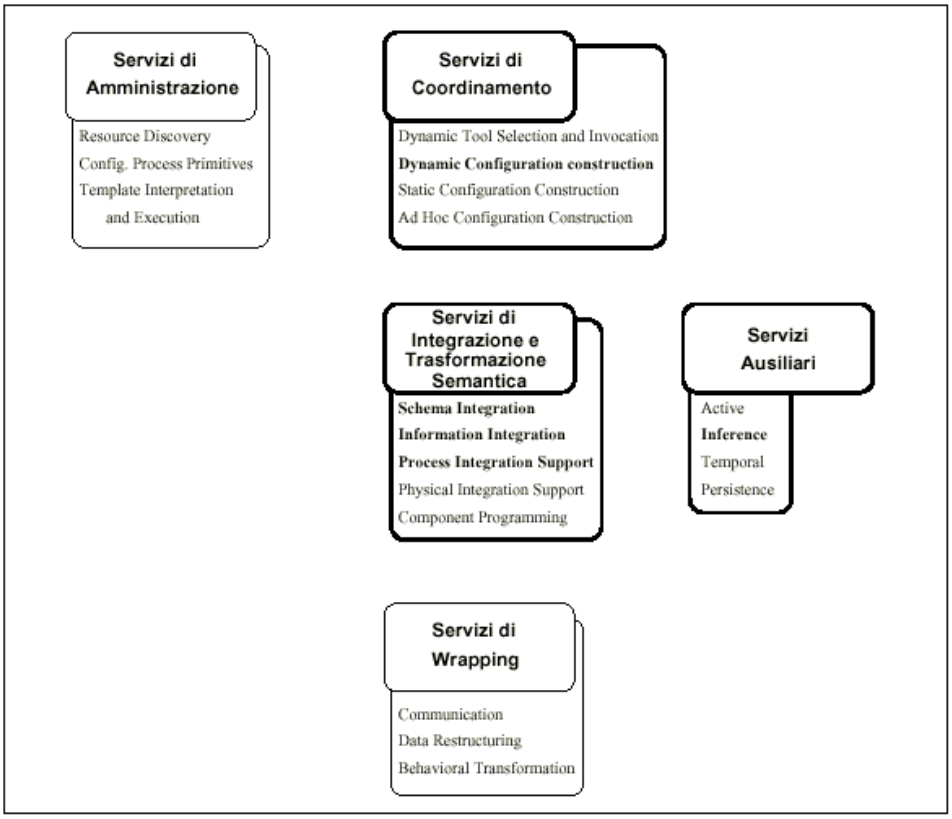


Figura 1.2: Servizi I^3 presenti nel mediatore

Facendo riferimento ai servizi descritti nelle sezioni precedenti, l'architettura del mediatore che si è progettato è riportata in figura 1.2. Parallelamente a questa impostazione architetturale inoltre, il nostro progetto vuole distaccarsi dall'approccio ***strutturale***, cioè **sintattico**, tuttora dominante tra i sistemi presenti sul mercato. L'approccio **strutturale** adottato dai sistemi quali TSIMMIS [8, 9, 10, 11], è caratterizzato dall'utilizzo di un *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 (e dunque di questa non possiede alcuno schema descrittivo) bensì e l’oggetto stesso che, attraverso delle etichette, si autodescrittive, specificando tutte le volte, per ogni suo singolo campo il significato associato.

I punti caratterizzanti di tale approccio sono quindi:

- possibilità di integrare in modo completamente trasparente al mediatore basi di dati fortemente eterogenee e magari mutevoli nel tempo, il mediatore non si basa, infatti, su una descrizione predefinita degli schemi delle sorgenti, bensì sulla descrizione che ogni singolo oggetto fa di sé. Oggetti simili provenienti dalla stessa sorgente possono quindi avere strutture differenti, ciò non avviene in un ambiente tradizionale *object-oriented;*
- per trattare in modo omogeneo dati che descrivono lo stesso concetto, o che hanno concetti in comune, ci si basa sulla definizione manuale di *rule*, che permettono di identificare i termini (e dunque i concetti) che devono essere condivisi da più oggetti.

Altri progetti, e tra questi il nostro, seguono invece un approccio definito ***semantico***, caratterizzato dai seguenti punti:

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

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

1.4 Problemi da affrontare

Pur avendo a disposizione gli schemi concettuali delle varie sorgenti, non è certamente un compito facile individuare i concetti comuni ad essi, le relazioni che possono legarli, né tanto meno realizzare una loro coerente integrazione. Tralasciando le differenze dei sistemi fisici (alle quali dovrebbero pensare i moduli *wrapper*) i problemi che si è dovuto risolvere, o con i quali occorre giungere a compromessi, sono (a livello di mediazione, ovvero di integrazione delle informazioni) essenzialmente di due tipi:

1. problemi ontologici;
2. problemi semantici.

1.4.1 problemi ontologici

Per ontologia si intende, in questo ambito, “l’insieme dei termini e delle relazioni usate in un dominio, che denotano concetti ed oggetti”. Con ontologia quindi ci si riferisce a quell’insieme di termini che, in un particolare dominio applicativo, denotano in modo univoco una particolare conoscenza e fra i quali non esiste ambiguità poiché sono condivisi dall’intera comunità di utenti del dominio applicativo stesso. Riporto una semplice classificazione delle ontologie (mutuata da [3, 4]) per inquadrare l’ambiente in cui ci si muove. I livelli di ontologia, e dunque le problematiche ad essi associate, sono essenzialmente le seguenti:

1. ***top-level ontology:*** descrive concetti molto generali (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:*** descrivono rispettivamente il vocabolario relativo a un generico dominio (come può essere un dominio medico, o automobilistico) o quello relativo a un generico obiettivo (come la diagnostica, o le vendite), dando una specializzazione dei termini introdotti nella *top-level ontology*;

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

Come ipotesi esemplificativa di questo progetto, si è assunto che tutte le fonti informative condividano i concetti fondamentali ed i termini con cui identificarli; si considera quindi di muoversi all’interno delle *domain ontology*.

1.4.2 Problemi semantici

Pur ipotizzando che anche sorgenti diverse condividano una visione similare del problema da modellare e, quindi, un insieme di concetti comuni, niente ci assicura che i diversi sistemi usino esattamente gli stessi vocaboli per rappresentare questi concetti, ne tantomeno le stesse strutture dati. Poiché le diverse strutture dati sono state progettate e modellate da persone differenti è molto improbabile che queste persone condividano la stessa “concettualizzazione” del mondo esterno, ovvero non esiste nella realtà una semantica univoca cui chiunque possa riferirsi. Ragion per cui c’è un’incertezza di interpretazione insita nell’ambiguità del linguaggio; Bates in [Bate86] scrive “*the probability of two persons using the same term in describing the same thing is less than 20%*”.

Qualche esempio ci aiuterà a capire meglio tale assunto:

Una persona *P1* disegna una fonte di informazione DB1 e un’altra persona *P2* disegna la stessa fonte DB2, sarà molto probabile che le due basi di dati presenteranno diverse semantiche: le coppie sposate potranno essere rappresentate in DB1 usando gli oggetti della classe COPPIA, con attributi MARITO e MOGLIE, mentre in DB2 potrebbe esserci una classe PERSONA con un attributo SPOSATO_A.

Come riportato in [2] la causa principale delle differenze semantiche si può identificare nelle diverse concettualizzazioni del mondo esterno che persone distinte possono avere, ma questa non è l’unica. Le differenze nei sistemi DBMS possono portare all’uso di differenti modelli per la rappresentazione della porzione di mondo in questione; partendo così dalla stessa concettualizzazione, determinate relazioni tra

concetti avranno strutture diverse a seconda che siano realizzate, ad esempio, attraverso un modello relazionale o un modello ad oggetti.

L’obiettivo dell’integratore, che, ricordiamolo, è di fornire un accesso integrato ad un insieme di sorgenti, si traduce allora nel non facile compito di identificare i concetti comuni all’interno delle sorgenti e risolvere le differenze semantiche che possono essere presenti. Possiamo classificare queste incoerenze semantiche in tre gruppi principali:

1. **eterogeneità tra le classi di oggetti:** benché due classi in due differenti sorgenti rappresentano lo stesso concetto nello stesso contesto, possono usare nomi diversi per gli stessi attributi, per i metodi, oppure avere gli stessi attributi con domini di valori diversi o ancora (dove questo è permesso) avere regole differenti su questi 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 (come può essere l’attributo SESSO in uno schema, presente invece nell’altro implicitamente attraverso la divisione della classe PERSONA in MASCHI e FEMMINE);
3. **eterogeneità nelle istanze delle classi:** ad esempio, l’uso di diverse unità di misura per i domini di un attributo, o la presenza/assenza di valori nulli.

È il caso di sottolineare l’importanza di sfruttare adeguatamente le differenze semantiche per arricchire il nostro sistema; analizzando a fondo tali differenze, e le loro motivazioni, si può arrivare al cosiddetto **arricchimento semantico** ovvero, ad aggiungere esplicitamente ai dati tutte quelle informazioni, originariamente presenti solo come *metadati* negli schemi, e come tali in un formato non interrogabile

Capitolo 2

Il sistema MOMIS

Tenendo presente le problematiche relative al mediatore, esposte nelle sezioni 1.2 e 1.3, e un insieme di progetti pre-esistenti quali TSIMMIS [8, 9, 10, 11], GARLIC [12, 13] e SIMS [14, 15], si è giunti alla progettazione di un sistema Intelligente di Integrazione delle Informazioni.

MOMIS acronimo di ***M**ediator **E**nvir**O**ment for **M**ultiple **I**nformation **S**ources* è il progetto di un sistema I^3, ideato per l'integrazione di sorgenti di dati testuali, strutturati e semistrutturati. **Momis** nasce all'interno del progetto MURST 40% INTERDATA, come collaborazione fra le unità operative dell'Università di Milano e dell'Università di Modena e Reggio Emilia.

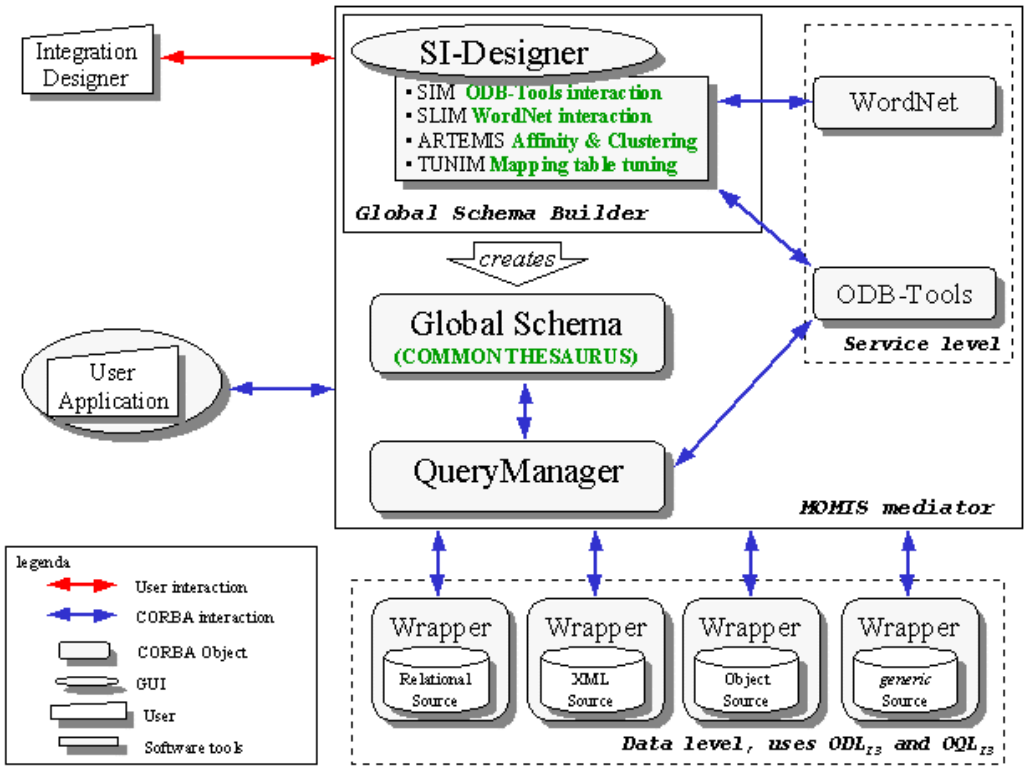


Figura 2.1: Architettura generale del sistema MOMIS

2.1 L’architettura di MOMIS

Momis è stato progettato per fornire un accesso integrato ad informazioni eterogenee, memorizzate sia in *database* di tipo tradizionale (e.g. relazionali, *object-oriented*) o *file system* sia in sorgenti di tipo semistrutturato, come quelle descritte in linguaggio XML.

Seguendo l’architettura di riferimento [1], in MOMIS si possono distinguere i componenti disposti su 3 livelli (vedi Figura 2.1):

- **Livello Dati.** Qui si trovano i *Wrapper*. Posti al di sopra di ciascuna sorgente, sono i moduli che rappresentano l’interfaccia tra il mediatore e le sorgenti locali di dati. La loro funzione è duplice:
  - in fase di integrazione, forniscono la descrizione delle informazioni in essa contenute. Questa descrizione è fornita attraverso il linguaggio descrittivo ODL_{I^3};
  - in fase di *query processing*, traducono la query ricevuta dal mediatore (espressa quindi nel linguaggio comune di interrogazione OQL_{I^3}, definito a partire dal linguaggio OQL) in un’interrogazione comprensibile 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.
- **Livello Mediatore.** Il mediatore è il cuore del sistema ed è composto da due moduli distinti:
  - *Global Schema Builder* (GSB): è il modulo che integra gli schemi locali, il quale partendo dalle descrizioni delle sorgenti espresse, attraverso il linguaggio ODL_{I^3}, genera un unico schema globale da presentare all’utente. L’interfaccia grafica di GSB, cioè il *tool* di ausilio al progettista, è *SI-Designer*.
  - *Query Manager* (QM): è il modulo di gestione delle interrogazioni. In particolare, genera le *query* in linguaggio OQL_I da inviare ai *wrapper* partendo dalla

singola *query* formulata dall'utente sullo schema globale. Servendosi di tecniche delle *Description Logics* di ODB-Tools il QM genera automaticamente la traduzione della *query* sottomessa nelle corrispondenti *sub-query* da sottoporre ai *wrapper* (*query* e sotto-*query* sono espresse in linguaggio OQL $_{I^3}$);

- **Livello Utente.** Il progettista interagisce col *Global Schema Builder* e crea la vista integrata delle sorgenti; l'utente formula le interrogazioni sullo schema globale passandole come input al *Query Manager*, che interrogherà le sorgenti e fornirà all'utente la risposta cercata.

Nella figura 1.1 compaiono altri tre *tool* che accompagnano il Mediatore nella fase di integrazione e sono:

- *ODB-Tools Engine*: un *tool* basato sulle *Description Logics* [17, 18] che compie la validazione di schemi e l'ottimizzazione di *query* [19, 20, 21].
- *ARTEMIS-Tool Enviroment*: *tool* basato sulle tecniche di *clustering affinity-based* che compie l'analisi ed il *clustering* delle classi ODL$_{I^3}$ [22].
- *WordNet:* un *database* lessicale della lingua inglese, capace di individuare relazioni lessicali e semantiche fra termini [23].

Il principale scopo che ci si è proposti con MOMIS è la realizzazione di un sistema di mediazione che, a differenza di molti altri progetti analizzati, contribuisca a realizzare, oltre alla fase di *query processing*, una reale integrazione delle sorgenti.

2.1.2 Il processo di Integrazione

L'integrazione delle sorgenti informative strutturate e semistrutturate viene compiuta in modo semiautomatico, utilizzando degli schemi locali in linguaggio ODL_{I^3} e combinando le tecniche di *Description Logics* e di *clustering*. Come mostrato in figura 2.2, le attività compiute sono le seguenti:

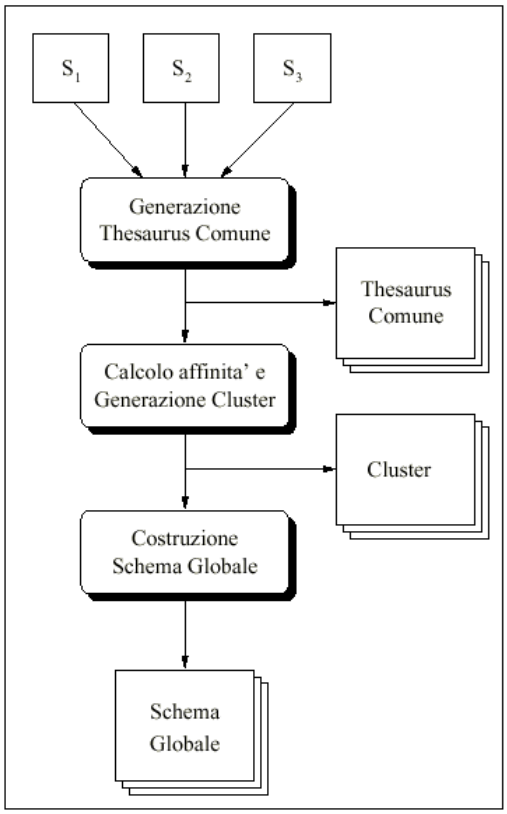


Figura 2.2: Le fasi dell'integrazione

1. Generazione de *Thesaurus* Comune, con il supporto di ODB-Tools e di *WordNet.* In questa fase, viene costruito un *Thesaurus* Comune di relazioni terminologiche. Tali relazioni esprimono la "conoscenza" inter-schema su sorgenti diverse e corrispondono alle asserzioni intensionali utilizzate in [24]. Le relazioni terminologiche sono derivate

in modo semi-automatico a partire dalle descrizioni degli schemi in ODL_{I^3}, attraverso l'analisi strutturale (utilizzando ODB-Tools e le tecniche di *Description Logics*) e di contesto (attraverso l'uso di *WordNet*) delle classi coinvolte.

2. Generazione dei *cluster* di classi ODL_{I^3} con il supporto dell'ambiente ARTEMIS-Tool. Le relazioni terminologiche contenute nel *Thesaurus* vengono utilizzate per valutare il livello di affinità tra le classi ODL_{I^3} 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 delle classi ODL_{I^3} basandosi sia sui nomi delle stesse, sia sugli attributi. Le classi ODL_{I^3} con maggiore affinità vengono raggruppate utilizzando le tecniche di *clustering* [25]. Rimandiamo per l'approfondimento alle sezioni 4.2.1 e 4.2.2.

3. Costruzione dello schema globale I *cluster* di classi ODL_{I^3} affini sono analizzati per costruire lo schema globale del Mediatore. Per ciascun *cluster* viene definita una classe globale ODL_{I^3} 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 *query* alle sorgenti locali integrate.

2.1.3 Query Processing e ottimizzazione

Quando l'utente pone una *query* sullo schema globale, MOMIS la analizza e produce un insieme di *sub-query* che saranno inviate a ciascuna sorgente informativa coinvolta. In accordo con altri approcci proposti in quest'ambito [14, 15], il processo consiste di due attività principali:

1. **Ottimizzazione Semantica e la Formulazione del Piano d'Accesso**. L'ottimizzazione semantica è basata sull'inferenza logica a partire dalla conoscenza contenuta nei vincoli di integrità dello schema globale. La

stessa procedura di ottimizzazione semantica si realizza in termini locali su ogni *sub-query* tradotta dal Mediatore nella formulazione del piano d'accesso: in tal caso ci si basa sui vincoli di integrità presenti sui singoli schemi locali.

2. **Formulazione del piano di accesso**. Il Mediator utilizza una "mappa" (*mapping table*, generata nella costruzione dello schema globale) che definisce l'associazione tra classi globali e classi locali. La *query* globale viene espressa in termini dagli schemi locali, tenendo in considerazione anche l'eventuale conoscenza di regole inter-schema definite sulle estensioni delle classi locali.

Il Mediatore agisce sulla *query* sfruttando la tecnica di ottimizzazione semantica supportata da ODB-Tools, in modo da ridurre il costo del piano d'accesso, e, dopo aver ottenuto la *query ottimizzata*, genera l'insieme di *sub-query* relative alle sorgenti coinvolte.

2.2 Il linguaggio ODL $_{I^3}$

Il linguaggio ODL **(*O**bject **D**efinition **L**anguage*) per la specifica di schemi ad oggetti, proposto dal gruppo di standardizzazione ODMG-93 [27, 28], è universalmente riconosciuto come standard. Le sue caratteristiche peculiari, al pari di altri linguaggi basati sul paradigma ad oggetti, possono essere così riassunte:

- definizione di tipi-classe e tipi-valore;
- distinzione fra intensione ed estensione di una classe di oggetti;
- definizione di attributi semplici e complessi;
- definizione di attributi atomici e collezioni (*set*, *list*, *bag*);
- definizione di relazioni binarie con relazioni inverse;
- dichiarazione della *signature* dei metodi.

Tuttavia, nonostante l'ODL sia ben progettato per rappresentare la conoscenza relativa ad un singolo schema ad oggetti, è certamente incompleto se utilizzato in un

contesto di integrazione di basi di dati eterogenee qual è quello descritto in questa tesi. Pertanto, si è reso necessario definire un'estensione di tale linguaggio, denominata ODL_{I3}, in accordo con le raccomandazioni della proposta di standardizzazione per i linguaggi di mediazione, risultato del lavoro di work-shop I^3 svoltosi nel 1995. Partendo dal presupposto che un sistema di mediazione dovrebbe poter supportare sorgenti con modelli complessi (come quelli ad oggetti) e modelli più semplici (come file di strutture), si è comunque cercato di discostarsi il meno possibile dal linguaggio ODL.

Con l'estensione di ODL al linguaggio ODL_{I3} sono stati raggiunti i seguenti obiettivi:

- per ogni classe, il wrapper può indicare nome e tipo del sorgente di appartenenza;
- per le classi appartenenti alle sorgenti relazionali è possibile definire le chiavi candidate ed eventuali foreign key;
- attraverso l'uso del costrutto "*union*" ogni classe può avere più strutture dati alternative, mentre il costrutto *optional* consente di indicare la natura opzionale di un attributo. Queste caratteristiche sono in accordo con la strategia utilizzata per la descrizione di dati semistrutturati;
- il linguaggio supporta la definizione di grandezze locali e di grandezze globali;
- il linguaggio supporta la dichiarazione di regole di mapping (o *mapping-rul*e) fra grandezze globali e grandezze locali;
- è data la possibilità di definire regole di integrità (o *if then rul*e), sia sugli schemi locali, sia sullo schema globale;
- il linguaggio supporta la definizione di relazioni terminologiche di sinonimia (SYN), iperonimia (BT), iponimia (NT) e associazione (RT);
- il linguaggio può essere automaticamente tradotto nella logica descrittiva OLCD usata da ODB-Tools, e quindi utilizzarne le capacità

nei controlli di consistenza e nell'ottimizzazione semantica delle interrogazioni.

2.3 Esempio di riferimento

Nei prossimi capitoli verranno ripresi in dettaglio tutti i passaggi che vengono effettuati in fase di integrazione e di *query* processing. Al fine di renderli più chiari introduciamo il seguente esempio di riferimento riguardante un dominio applicativo universitario (nell'Appendice B verrà riportato per esteso l'esempio di riferimento in linguaggio ODL $_{I^3}$).

Consideriamo 3 sorgenti eterogenee: una relazionale `University` (spesso abbreviata in: `UNI`), una sorgente semistrutturata `Computer_Science` (spesso abbreviata in `CS`) (originariamente un file XML) e una sorgente costituita da un semplice file `Tax_Position` (spesso abbreviata in `TP`).

La sorgente `University` contiene le informazioni riguardanti gli studenti e lo staff di un'università ed è composta da cinque relazioni: `Research_Staff`, `School Member`, `Department`, `Section` e `Room`. Ad ogni professore in `Research_Staff` è associato un dipartimento (`dept_code`) e una sezione (`section_code`). Ogni sezione (in `Section`) è associata ad un'aula (`room_code`) mentre ogni studente (in `School_Member`) è caratterizzato da un `name`, una `faculty` e dall'anno di immatricolazione `year`.

La sorgente semistrutturata `Computer_Science` contiene informazioni sulla facoltà di informatica della medesima università descritta dalla sorgente relazionale. Ci sono sei classi: `CS_Person`, `Professor`, `Student`, `Division`, `Location` e `Course`. Le informazioni rappresentate sono simili a quelle della sorgente relazionale: anche qui sono presenti professori e studenti; ogni professore è associato ad una `Division`, che è una logica specializzazione di `Department`. Di ogni studente possiamo sapere che corsi frequenta, l'anno di immatricolazione e il suo stato (`rank`), cioè se è in corso, fuori corso, laureando oppure dottorando.

L’ultima sorgente rappresenta un file, `Tax_Position`, in cui è rappresentata la posizione fiscale di ciascuno studente, in particolare, sono presenti i seguenti campi: `name`, `student_code`, `faculty_name` e `tax_fee`.

Alcune volte per evitare ambiguità tra nomi uguali in sorgenti diverse, considereremo una rappresentazione dei nomi in *dot notatio*n: per esempio, la relazione tra nomi di attributi `dept_code BT belongs_to` viene rappresentata come:

`UNI.Department.dept_code BT CS.Division.belongs_to`.

Capitolo 3

Gli Strumenti utilizzati

Da quanto visto nei capitoli precedenti, MOMIS utilizza alcuni strumenti per svolgere il suo compito:

1. **ODB-Tools,** che è utilizzato sia durante la fase d'integrazione degli schemi delle sorgenti nella costruzione del *Thesaurus* Comune, sia durante la fase di *query processing* per l'ottimizzazione delle *query;*
2. ***WordNet***, che viene impiegato per estrarre relazioni lessicali, tra i termini, che andranno ad arricchire il *Thesaurus* Comune

3.1 ODB-Tools

ODB-Tools è un sistema per l'acquisizione e la verifica di consistenza di schemi di basi di dati e per l'ottimizzazione semantica di interrogazioni su basi di dati orientate agli oggetti (OODB), ed è stato sviluppato presso il Dipartimento di Scienze dell'Ingegneria dell'Università di Modena [29, 30]. L'architettura, mostrata in figura 3.1, presenta i vari moduli integrati che definiscono un ambiente *user-friendly* basato sul linguaggio standard ODMG-93.

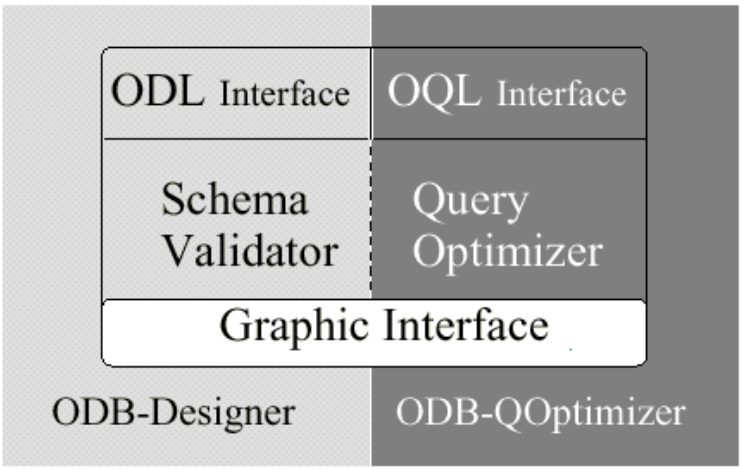


Figura 3.1: Architettura di ODB-Tools

ODB-Tools basa il suo ambiente teorico su due elementi fondamentali:

1. **OLCD (*O**bject **L**anguage with **C**omplements allowing **D**escriptive cycles*)**: linguaggio derivante dalla famiglia KL-ONE [31], proposto come formalismo comune per esprimere descrizioni di classi, vincoli di integrità ed interrogazioni e dotato di tecniche di inferenza basate sul calcolo della sussunzione, introdotte per le *Logiche Descrittive* nell'ambito dell'Intelligenza Artificiale;
2. **espansione semantica** di un tipo: realizzata attraverso l'algoritmo di sussunzione.

Il formalismo **OLCD** deriva dal precedente **OD**L, proposto in [27, 28], che già estendeva l'espressività di linguaggi di logica descrittiva al fine di rappresentare la semantica dei modelli di dati ad oggetti complessi *(CODM*s), precedentemente proposti in ambito di basi di dati deduttive e basi di dati orientate agli oggetti. In [32], ODL è stato esteso per permettere la formulazione dichiarativa di un insieme rilevante di vincoli di integrità definiti sulla base di dati. L'estensione di ODL con vincoli è stata denominata **OLCD**. Attraverso questa logica descrittiva, è possibile descrivere, oltre alle classi, anche le *regole di integrità*, permettendo la formulazione dichiarativa di un insieme rilevante di vincoli di integrità sottoforma di regole *if-then* i cui antecedenti e conseguenti sono espressioni di tipo ODL. In tale modo, è possibile descrivere correlazioni tra proprietà strutturali della stessa classe, o condizioni sufficienti per il popolamento di sottoclassi di una classe data.

In [18] è stato presentato il sistema OCDL-*Designer*, per l'acquisizione e la validazione di schemi OODB descritti attraverso **OLCD**, che preserva la consistenza della tassonomia di concetti ed effettua inferenze tassonomiche. In particolare, il sistema prevede un algoritmo di *sussunzione* che determina tutte le relazioni di specializzazione tra tipi, e un algoritmo che rileva gli eventuali tipi *inconsistent*i, cioè tipi necessariamente vuoti.

In [32] l'ambiente teorico sviluppato in [18] è stato esteso per effettuare l'ottimizzazione semantica delle interrogazioni, dando vita al sistema ODB-*QOptimizer*. Sono state inoltre sviluppate delle interfacce software per tradurre descrizioni ed interrogazioni espresse rispettivamente nei linguaggi ODL e OQL (proposti dal gruppo di standardizzazione ODMG-93 [27, 28]) in **OLCD.**

La nozione di ottimizzazione semantica di una *query* è stata introdotta, per le basi di dati relazionali, da King [33, 34] e da Hammer e Zdonik [35]. L'idea di base di queste proposte è che i vincoli di integrità, espressi per forzare la consistenza di una base di dati, possano essere utilizzati anche per ottimizzare le interrogazioni fatte dall'utente, trasformando la *query* in un *equivalent*e, in altre parole con lo stesso insieme di oggetti di risposta, ma che può essere elaborata in maniera più efficiente.

Sia il processo di consistenza e classificazione delle classi dello schema, che quello di ottimizzazione semantica di un'interrogazione, sono basati in ODB-Tools sulla nozione di *espansione* semantica di un tipo: l'espansione semantica permette di incorporare ogni possibile restrizione che non è presente nel tipo originale, ma che è logicamente implicata dallo schema (inteso come l'insieme delle classi, dei tipi, e delle regole di integrità). L'espansione dei tipi si basa sull'iterazione di questa trasformazione: se un tipo *implica* l'antecedente di una regola di integrità, allora il conseguente di quella regola può essere aggiunto alla descrizione del tipo stesso. Le *implicazioni* logiche fra i tipi (in questo caso il tipo da espandere e l'antecedente di una regola) sono determinate a loro volta utilizzando l'algoritmo di *sussunzion*e, che calcola relazioni di sussunzione, simili alle relazioni di raffinamento dei tipi definite in [28].

Il calcolo dell'espansione semantica di una classe permette di rilevare nuove relazioni *is_a*, vale a dire relazioni di specializzazione che non sono esplicitamente definite dal progettista, ma che comunque sono logicamente implicate dalla descrizione della classe e dello schema cui questa appartiene. In questo modo, una classe può essere automaticamente classificata all'interno di una gerarchia di ereditarietà. Oltre che a determinare nuove relazioni tra classi virtuali, il meccanismo, sfruttando la conoscenza fornita dalle regole di integrità, è in grado di riclassificare pure le classi base (generalmente gli schemi sono forniti in termini di classi base).

Analogamente, rappresentando a *run-time* l'interrogazione dell'utente come una classe virtuale (la *query* non è altro che una classe di oggetti di cui si definiscono le condizioni necessarie e sufficienti per l'appartenenza), questa viene classificata all'interno dello schema, in modo da ottenere l'interrogazione più specializzata tra tutte quelle semanticamente equivalenti a quella iniziale. In questo modo la *query* viene spostata verso il basso nella gerarchia e le classi cui si riferisce vengono

eventualmente sostituite con classi più specializzate: diminuendo l'insieme degli oggetti da controllare per dare risposta all'interrogazione, ne viene effettuata una vera ottimizzazione indipendente da qualsiasi modello di costo.

3.2 Il database lessicale: *WordNet*

Il *WordNet* [23] è un *DataBase* lessicale elettronico, la cui struttura è ispirata alle attuali teorie psicolinguistiche legate alla memoria lessicale umana. È sviluppato presso l'università di Princeton dal C*ognitive Science Laboratory* sotto la direzione del professor A. Gorge Miller. È anche consultabile on-line all'indirizzo http://www.cogsci.princeton.edu/~wn/.

WordNet organizza nomi, verbi, aggettivi ed avverbi in insiemi di sinonimi (*synsets*), dove ognuno di loro rappresenta un concetto. I *synset* inoltre sono legati tra loro attraverso diverse relazioni, come la sinonimia, antonimia, iponimia, iperonimia, ecc.

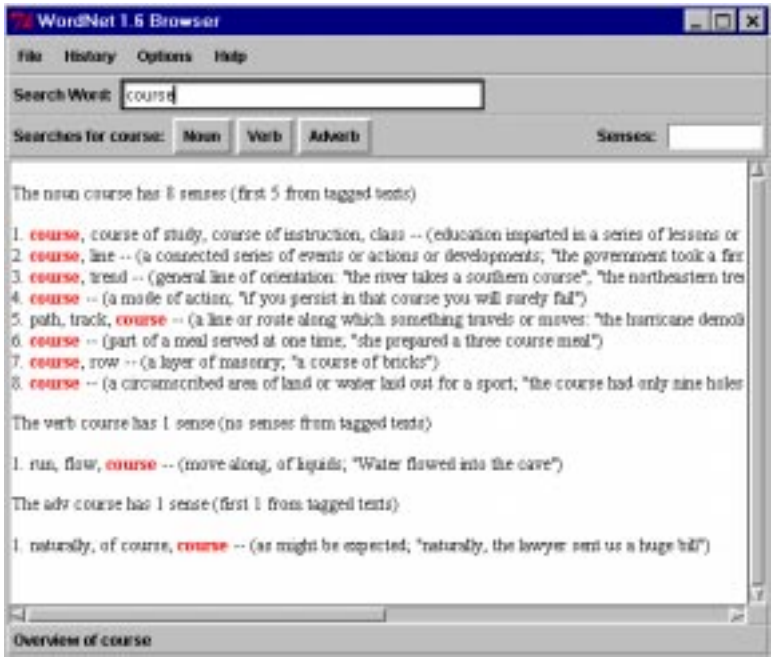


Figura 3.2: *Overview* di *course*

La differenza dalle classiche procedure per organizzare informazioni lessicali, come ad esempio i dizionari, consiste nel fatto che le parole non sono organizzate in una lunga lista ordinata, ma sono legate in modo gerarchico. In questo modo il *WordNet* conserva tutte le caratteristiche dei vecchi dizionari aggiungendo però degli strumenti in più, infatti, inserendo una parola si ottiene una *overview* (figura 3.2) della stessa, completa di tutti i significati, con i sinonimi e la categoria (nome, verbo, aggettivo, avverbio) di appartenenza, quindi si può scegliere, per ogni categoria e specificando anche l'esatto significato, il tipo di relazione da visualizzare (vedi

figura 3.), dopo di che vengono visualizzati tutti i *synset* legati in modo gerarchico alla relazione scelta.

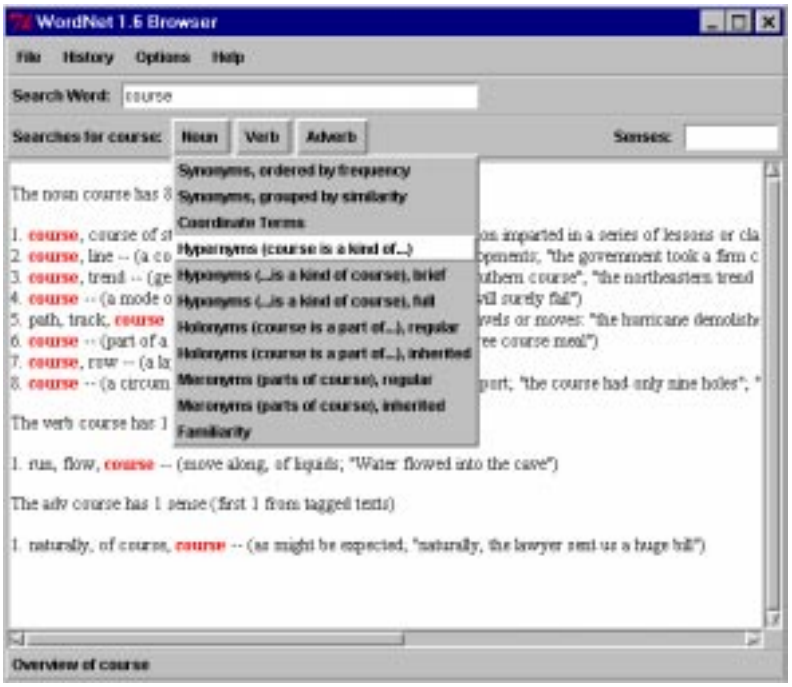


Figura 3.3: tipi di relazioni per la categoria nome

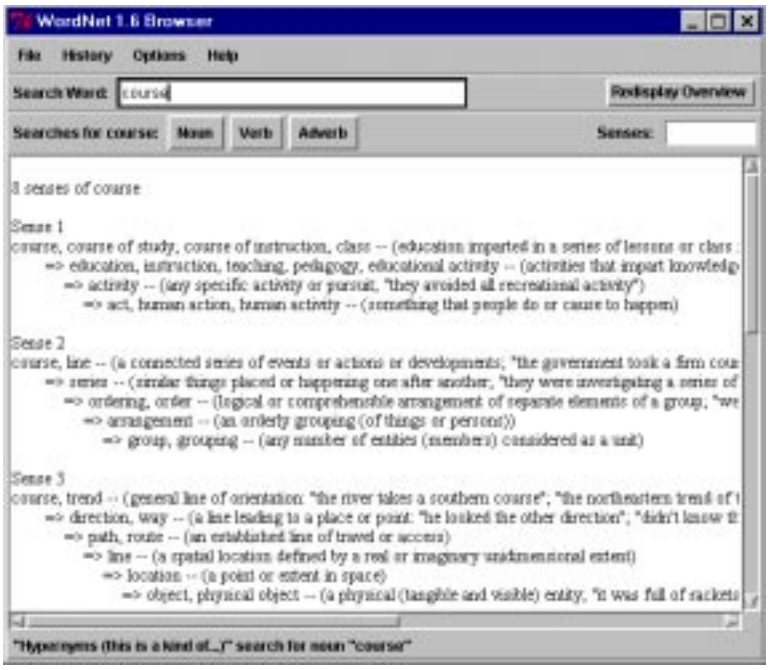


Figura 3.4: relazioni di iponimia del nome *course*

Dalla figura 3.4 si può osservare come ogni gerarchia, generata dai diversi significati di *course*, termina in un nome che viene chiamato *beginner*. Nel *WordNet* sono presenti 9 diversi *beginner* (figura 3.5), dai quali si generano tutti i nomi presenti nel *database*.

{abstraction}
{act, human action, human activity}
{entity, something}
{event}
{group, grouping}
{phenomenon}
{possession}
{psychological feature}
{state}

Figura 3.5: lista dei 9 *beginners*

WordNet è considerato la più importante risorsa disponibile per i ricercatori nei campi della linguistica computazionale, dell'analisi testuale, e di altre aree associate. Attualmente (settembre 2000), è arrivato alla versione 1.6 e contiene 129625 lemmi organizzati in 99759 *synset* divisi in 4 categorie: nomi, verbi, aggettivi e avverbi.

Questa suddivisione in categorie, pur comportando una ridondanza rispetto ai dizionari classici, (ad esempio la parola *back* che è presente in tutte le categorie), ha il vantaggio di visualizzare, in modo chiaro, le diverse categorie e di sfruttarle al meglio in base alle loro diverse strutture. Ogni categoria è organizzata in modo diverso:

- i nomi, in 9 gerarchie individuate da altrettanti *beginners*, generate da relazioni di iponimia/iperonimia.
- I verbi, in 617 gerarchie, generate da varie relazioni d'implicazione (*entailment relation*).
- Gli aggettivi e gli avverbi, in iperspazi N-dimensionali (*cluster).*

Tali diverse strutture sono necessarie, al fine di meglio rappresentare la complessità psicologica della conoscenza lessicale.

3.2.1 La Matrice Lessicale

Alla base del progetto del *WordNet* c'è la Matrice Lessicale (figura 3.7). La necessità di creare una matrice lessicale nasce dall'osservazione che una parola è un'associazione convenzionale tra il suo significato e il modo in cui viene letta o scritta. Per ridurre l'ambiguità derivata dal termine parola useremo:

- "*word meaning*" o significato, se ci riferiamo al concetto lessicale o significato
- "***word form***" o **lemma**, se ci riferiamo al modo in cui viene letta o scritta.

Quest'associazione è di molti a molti (vedi figura 3.6), e dà luogo alle seguenti proprietà:

- **Sinonimia**: proprietà di un significato di avere due o più parole in grado di esprimerlo.
- **Polisemia**: proprietà di una parola di esprimere due o più significati.

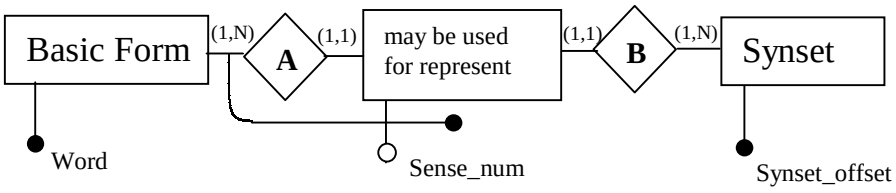


Figura 3.6: Relazione tra lemmi e significati

Questa relazione esistente tra le *word-form* e le *word-meaning* può essere sintetizzato attraverso la matrice lessicale (vedi figura 3.7), dove nelle righe sono rappresentate le *word meaning* (significati) e nelle colonne le *word form* (forma/lemma base).

Ogni cella della matrice ci indica che il lemma, in quella colonna, può essere rappresentato per esprimere il significato in quella riga; quindi, la cella $E_{1,1}$ implica

che il lemma F_1 può essere usato per esprimere il significato M_1; nel caso ci fossero due lemmi nella stessa colonna, significherebbe che il lemma è polisemo (ha più significati), mentre nel caso di due lemmi nella stessa riga significherebbe che i due lemmi sono sinonimi.

Word Meaning	Word Forms F_1 F_2 F_3 ---------- F_n
M_1	$E_{1,1}$ $E_{1,2}$
M_2	$E_{2,2}$
M_3	$E_{3,3}$
-	
-	
-	
M_m	$E_{m,n}$

Figura 3.7: Matrice lessicale

Dalla figura 3.7 possiamo affermare che F_1 è F_2 sono sinonimi e che F_2 è polisemo. Ogni cella *"e"* è una definizione (*entry*), *e=(f, m)*, dove *f* individua la *word-form* e *m* la *word-meaning*; ad esempio (*course*, 1) si riferisce ad un corso di studio; mentre (*course*, 3) si riferisce al corso di un fiume.

Word Meaning	**Word Forms** **course | course of study | course of instruction | class | line | trend**
Education imparted in a series of lessons or class meating ...	course | course of study | course of instruction | class
A connected series of events or action or developments....	course line
General line of orientation: "the river takes a southern course" ...	course trend

Figura 3.8: esempio di una vista della matrice lessicale

3.2.2 Relazioni lessicale e sintattiche

Inizialmente *WordNet* è stato ideato come un insieme di relazioni semantiche tra concetti lessicali, ma con il proseguire del progetto si è osservato che c'era un altro tipo di relazione che non poteva essere trascurato e sono le relazioni lessicali. Attualmente, anche se ad essere enfatizzate sono le sole relazioni semantiche, occorre ricordare che il *WordNet* organizza le relazioni in due principali categorie e sono:

- **Relazioni semantiche:** le quali esprimono un legame tra i significati come la specializzazione e la generalizzazione (iponimia e l'iperonimia).
- **Relazioni lessicali:** le quali esprimono un legame tra i lemmi, come la sinonimia, la polisemia e l'antonimia.

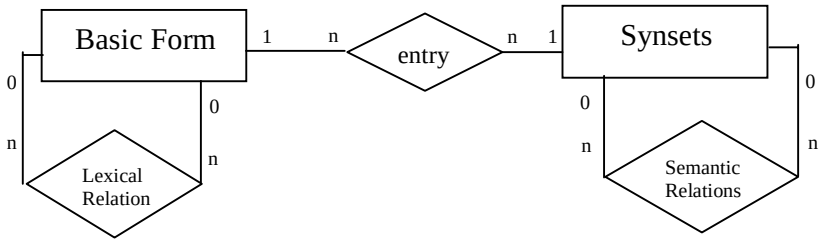


Figura 3.9: relazioni lessicali e semantiche

Abbiamo visto come *WordNet* è organizzato in relazioni semantiche, e come ognuna di esse è una relazione tra significati; abbiamo anche visto che i significati sono organizzati in *synset*, quindi è possibile immaginare una relazione semantica come un puntatore tra *synsets*. Caratteristica delle relazioni semantiche è la reciprocità: se esiste una relazione *R* tra il significato{*x,x'*,…} e il {*y,y'*,…}, allora esiste anche una relazione *R'* tra {*y,y'*,…} e {*x,x'*,…}.

3.2.2.1 Sinonimia

Da quanto argomentato, possiamo affermare che la sinonimia è la relazione più importante del *WordNet*, infatti, un buon conoscitore della lingua inglese, dalla sola relazione di sinonimia tra i vari lemmi, può risalire al significato. Consideriamo, ad esempio la parola *board*, essa può assumere i significati di "tavola di legno" e "scheda per computer", se osserviamo i due *synset* che la contengono*:* {*board, plank*} e {*board, circuit card*}, possiamo risalire all'esatto significato; quindi è bastato introdurre due sinonimi, *plank* e *circuit card*, per eliminare l'ambiguità.

Per distinguere la relazioni di sinonimia dalle altre relazioni, i termini sinonimi sono racchiusi fra parentesi graffe {}, mentre gli altri insiemi prodotti dalle altre relazioni lessicali sono racchiusi fra parentesi quadre [].

La definizione di sinonimia, generalmente attribuita a Leibinitz, secondo la quale:

> "Due espressioni sono sinonime se la sostituzione di una per l'altra non cambia il vero significato della frase nella quale è fatta la sostituzione."

Questa definizione impone una condizione estrema e rende i sinonimi molto rari o inesistenti. Una seconda definizione invece fa cadere l'ipotesi sul contesto ed è:

> "Due espressioni sono sinonime in un contesto linguistico *C* se la sostituzione di una per l'altra, nel contesto *C*, non altera il vero valore della frase."

Per meglio comprendere, prendiamo come esempio *blank*, che in un contesto di falegnameria, può quasi sempre essere sostituito con *plank* senza alterare il significato, mentre in altri contesti questa sostituzione risulterebbe inappropriata.

La definizione di sinonimia in termini di sostituibilità divide necessariamente il *WordNet* in nomi, verbi, aggettivi e avverbi, infatti, parole che appartengono a diverse categorie sintattiche non possono essere interscambiate.

3.2.2.2 Antinomia

L'antinomia è una relazione lessicale tra lemmi, la quale indica che uno è il contrario dell'altro. Solitamente l'antinomia di una parola "*x*" è "non *x*", ma ciò non è sempre vero, infatti, *ricco* e *povero* sono antinomi, ma dire *non ricco* non equivale a povero; molta gente può essere né ricca e né povera. Pur apparendo come una semplice relazione di simmetria l'antinomia è una relazione abbastanza complessa.

Occorre tener presente che l'antinomia è una relazione lessicale tra *word form*, e non una relazione semantica tra *word meaning;* i significati {*rise, ascend*} e {*fall, descend*} possono essere concettualmente opposti, ma non sono antinomi; mentre sono esempi di antinomie [*rise*/*fall*] cosi come [*ascend*/*descend*].

3.2.2.3 Iponimia

Diversamente dalle relazioni di sinonimia ed antinomia, le quali erano relazioni lessicali tra *word forms*, *iponimia/iperonimia* è una relazione semantica tra *word meanings*: per esempio, {*maple*} (acero) è un iponimo di {*tree*}, e {*tree*} è un iponimo di {*plant*}. Un concetto rappresentato dal *sinset* {*x, x', …*} è detto eponimo del concetto rappresentato da {*y, y', …*} se un madrelingua inglese convalida la frase costruita come: *x is a* (*kind of*) *y*.

La relazione di iponimia è transitiva e simmetrica ed è equivalente alla relazione di specializzazione. Ogni iponimo eredita tutte le caratteristiche dei concetti più generici e addiziona almeno uno per distinguerlo dal sovraordinato e dagli altri iponimi del sovraordinato; per esempio, *maple* eredita tutte le caratteristiche da sovraordinato tree, ma si distingue dagli altri alberi per la durezza del suo legno, la forma delle foglie, etc.

La corrispondente relazione inversa, equivalente alla generalizzazione, è l'iperonimia, la quale lega un sovraordinato, ai subordinati, ereditando dagli iponomi le caratteristiche generali.

La relazione di iponimia/iperonimia stabilisce la regola centrale per l'organizzazione dei nomi in *WordNet,* essa è definita solo per i nomi e i verbi, ed è la relazione più frequente in *WordNet*. Nel caso dei verbi la relazione di iponimia è chiamata troponimia:

> Un verbo che esprime una maniera specifica di operare di un altro verbo: *X* è un troponimo di *Y* se fare *X* è fare *Y* in qualche maniera.

3.2.2.4 Meronimia

La meronimia è una relazione semantica e corrisponde alla relazione "*has a"*. Dal punto di vista lessicale e definita come:

> Un concetto rappresentato dal *synset* {x,*x'*,…} è un meronimo di un concetto rappresentato dal *synset* {*y*,y',…} se un madrelingua inglese convalida la frase costruita come: An *y has an x* (*as a part*) *or An x is a part of y.*

Sono definiti tre tipi di aggregazione:

1. **HAS MEMBER** – esempio *school* inteso come istituzione educativa ha due aggregazioni:
   - *school HAS MEMBER: staff, faculty*
   - *school HAS MEMBER: schoolteacher, school teacher*
2. **HAS PART** – esempio *school* inteso come luogo dove si riceve l'educazione ha un asola aggregazione:
   - *school HAS PART: classroom, schoolroom*.
3. **HAS SUBSTANCE** – esempio *quartz* inteso come minerale ha due aggregazioni:
   - *quartz HAS SUBSTANCE: silicon, Si, atomic number 14*

- *quartz HAS SUBSTANCE: silica, silicon oxide, silicon dioxide.*

La meronimia è una relazione transitiva e asimmetrica, la sua relazione duale è la olonimia; anch'essa può essere usata per costruire gerarchie di concetti meronimi/olonimi, con la differenza che, in questo caso, uno stesso meronimo può avere più olonimi: in altri termini, un concetto può contemporaneamente far parte di differenti concetti composti.

I meronimi sono fattori distintivi che gli iponimi possono ereditare. Per esempio, se *beak* (becco) e *wing (*ala) sono meronimi di *bird* (uccello), e *canary* (canarino) è un iponimo di *bird*, allora, ereditando, *beak* e *wing* deve anche essere meronimo di *canary.*

Quando tutte e tre le relazioni, iponimia, meronimia, e antinomia si incrociano, il risultato è altamente interconnesso in una rete complessa (vedi figura 2.9); sapere dove una parola è situata in questa rete è un'importante informazione per la conoscenza del significato della parola.

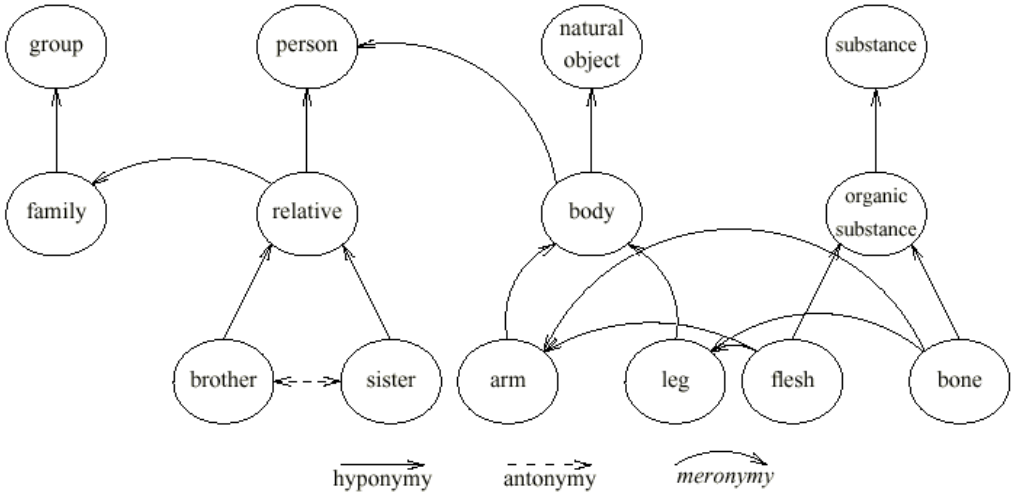


Figura 3.10: Rappresentazione delle tre relazioni

3.2.2.5 Relazioni morfologiche

Un'altra importante classe di relazioni lessicali sono le relazioni morfologiche tra *word forms*. Inizialmente questo tipo di relazione non era stato considerato ma, con l'avanzamento del progetto, cresceva la necessita di inserire questo nuovo tipo di relazioni, infatti, bastava inserire un termine al plurale, ad esempio *trees*, per far si che *WordNet* dava come risultato che il termine non era incluso nel *database*.

La morfologia delle parole nella lingua inglese è abbastanza semplice e si manifesta come:

- Declinazioni per i sostantivi, ad esempio distinzione fra singolare e plurale;
- Coniugazione per i verbi.

Per far si che *WordNet* riconoscesse queste forme è stato inserito un software che non modifica la struttura del *database*, nel senso che, all'interno le parole non sono replicate, infatti, le parole sono memorizzate nella forma canonica. Il software introdotto ha il compito di interfacciarsi fra l'utente e il *database* lessicale, in modo tale da tradurre ogni termine in input nella forma canonica e, successivamente, inviarlo al *database.*

Nonostante la semplicità della morfologia inglese rispetto le altre lingue, la realizzazione di questo componente non lo è stata a causa della massiccia presenza di verbi irregolari.

3.2.3 Categorie sintattiche

3.2.3.1 Nomi in WordNet

Nelle moderne teorie psicolinguistiche, per definire un nome si usa un termine sovraordinato insieme a fattori che lo distinguono; questo metodo è anche alla base dell'organizzazione dei nomi in *WordNet*, infatti, la relazione di iponimia genera una gerarchia semantica. La gerarchia generata è limitata in profondità e raramente

raggiunge il dodicesimo livello. Un modo per costruire tale gerarchia è di far rientrare tutti i nomi in una singola gerarchia, in questo caso, il livello più alto dovrebbe essere semanticamente vuoto. Inizialmente si era indicato come radice {*etity*}, seguito dagli immediati iponimi: {*object*/ *thing*} e {*idea*}. Successivamente si è visto che, anche questi generici concetti astratti, portavano una piccola quantità di informazione semantica.

L'alternativa è stata di partizionare i nomi con un insieme di *unique beginner* (vedi figura 2.4) dai quali si generano altrettante gerarchie.

Approssimativamente *WordNet* contiene 57,000 nomi (*word forms*) organizzati in 48,000 *word meaning* (synset).

3.2.3.2 Aggettivi e avverbi in *WordNet*

Tutti i linguaggi assegnano agli aggettivi la proprietà di modificatori o elaboratori del significato dei nomi; quella di modificatori di nomi e la caratteristica principale associata alla categoria degli "aggettivi".

I *synset* degli aggettivi in *WordNet* contengono principalmente aggettivi, anche se sono state anche aggiunti nomi e frasi preposizionali che hanno la funzione di modificatori.

WordNet attualmente contiene approssimativamente 19,500 aggettivi in *word forms*, organizzati in 10,000 *word meanings* (*synsets*).

Gli aggettivi in *WordNet* sono suddivisi in due principali classi:

- **Descrittivi –** come *big, interesting, possibile* che attribuiscono ai nomi valori di attributi bipolari e, conseguentemente, sono organizzati in termini di opposizioni binarie (antinomia) e similitudine di significato. Un modo per definire l'aggettivo è il seguente:

  se *x* è *Adj* si presuppone che esiste un attributo *A* tale che *A*(*x*)=*Adj*.

Vale a dire: *il pacco è pesante* presuppone che esiste un attributo *PESO* tale che *PESO*(*pacco*)=*pesante*. Allo stesso modo, *basso* e *alto* sono valori per l'attributo *ALTEZZA.*

L'organizzazione semantica degli aggettivi descrittivi è completamente differente da quella dei nomi, infatti, nella loro organizzazione non esiste nessuna relazione di iponimia poiché non avrebbe nessun senso dire che un aggettivo *"is a kind of" un altro aggettivo.*

La principale relazione per gli aggettivi e l'antinomia, ciò si evince dalla funzione degli aggettivi, che è quella di attribuire valore agli attributi, e che quasi tutti gli attributi sono bipolari. Gli aggettivi che non sono bipolari sono collegati per similitudine semantica agli aggettivi bipolari.

Possiamo immaginare l'organizzazione degli aggettivi come *cluster* che contengono un *synset* principale e dei *synset satelliti*. Le coppie di antinomi sono nei *synset* principali dei *cluster* e sono dette relazioni di antinomia dirette. I *synset* principali sono collegati ai *synset* satelliti da una relazione di similarità (vedi figura 3.11).

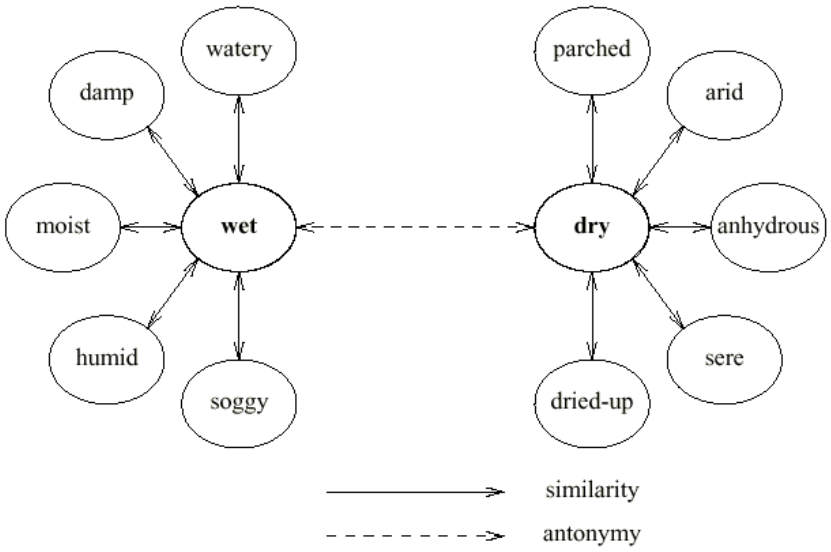


Figura 3.11: Struttura bipolare degli aggettivi

- **Relazionali –** come *presidential, nuclear, musical* che sono ritenuti varianti stilistiche dei modificatori dei nomi e sono considerati come dei riferimenti incrociati ai nomi, inoltre non hanno antinomi e non sono organizzati in *cluster.*

Gli avverbi sono simili agli aggettivi, infatti, spesso derivano da essi. La loro proprietà è di modificare il significato dei verbi.

3.2.3.3 Verbi

I verbi sono la categoria più importante, infatti, ogni frase ne deve contenere almeno uno; ma, ciò nonostante, nel linguaggio ci sono molto meno verbi che nomi. Per esempio, il "*Collins English Dictionary*" contiene 43,636 nomi e solo 14,190 verbi. Da ciò si deduce che i verbi sono più polisemici dei nomi, infatti, nel *Collins* i nomi hanno una media di 1.74 significati, mentre i verbi hanno una media di 2.11, ed inoltre sono anche più flessibili poiché cambiano il loro significato in base al contesto, mentre i nomi tendono ad essere più stabili.

Questa elevata ambiguità potrebbe essere ridotta inserendo dei riferimenti incrociati tra i *synset* dei verbi con i *synset* dei nomi, ma attualmente, in *WordNet*, non è stata ancora implementata questa soluzione.

WordNet contiene 21,000 verbi in *word forms* (dei quali circa 13,000 sono stringhe uniche) e circa 8,400 *word meaning* (*synset*). Sono anche incluse espressioni come: *look up* e *fall back.*

I verbi, come i nomi, sono organizzati in una gerarchia di specializzazione, ma anziché avere 9 *unique beginner* ne hanno 617. Inoltre queste gerarchie sono più piatte rispetto a quelle dei nomi.

La relazione di iponimia è sostituita con quella di troponimia, che è un particolare tipo d'implicazione; ogni troponimo V_1 di uno o più verbi V_2 *implica anche* V_2. Consideriamo la coppia *limp-walk* (zoppicare-camminare) se è esatta la frase: "*to limp* è anche *to walk* in una certa maniera", questo mi dice che *limp* è troponimo di *walk*; in questo caso i verbi sono anche in una relazione d'implicazione (*entailment relation*), infatti "*He is limping"* implica "*He is walking"*.

Nei verbi, le relazioni fondamentali sono quelle di implicazione o *entailment* le quali possono essere di vario tipo (vedi figura 2.11).

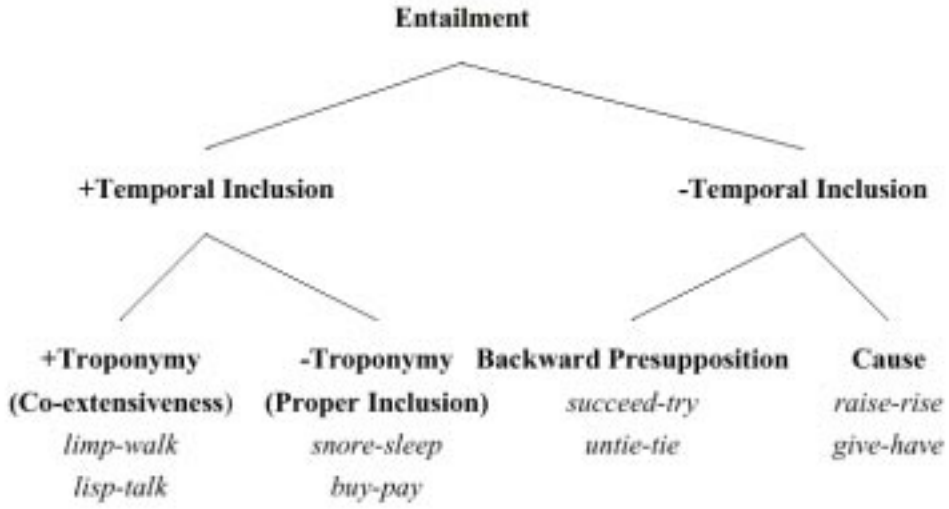


Figura 3.12 Quattro tipi di relazioni di implicazione o *entailment* tra i verbi

3.2.4 *DataBase* lessicali alternativi

Abbiamo fin qui visto le caratteristiche e le potenzialità che il *WordNet* ci offre, ma l'unico suo limite è di essere un *database* lessicale monolingue, in quanto progettato per la sola lingua inglese.

Ciò comporta un limite per MOMIS, che può sfruttare questo importante strumento solo con *database* progettati in lingua inglese, ed è proprio per questo che MOMIS si è attivato nella ricerca di altri progetti simili ma con il supporto di più lingue.

3.2.4.1 Multi-WordNet

Multi WordNet [40, 41] è sviluppato presso l'IRST (Istituto per la Ricerca Scientifica e Tecnologica), come estensione del WordNet Inglese.

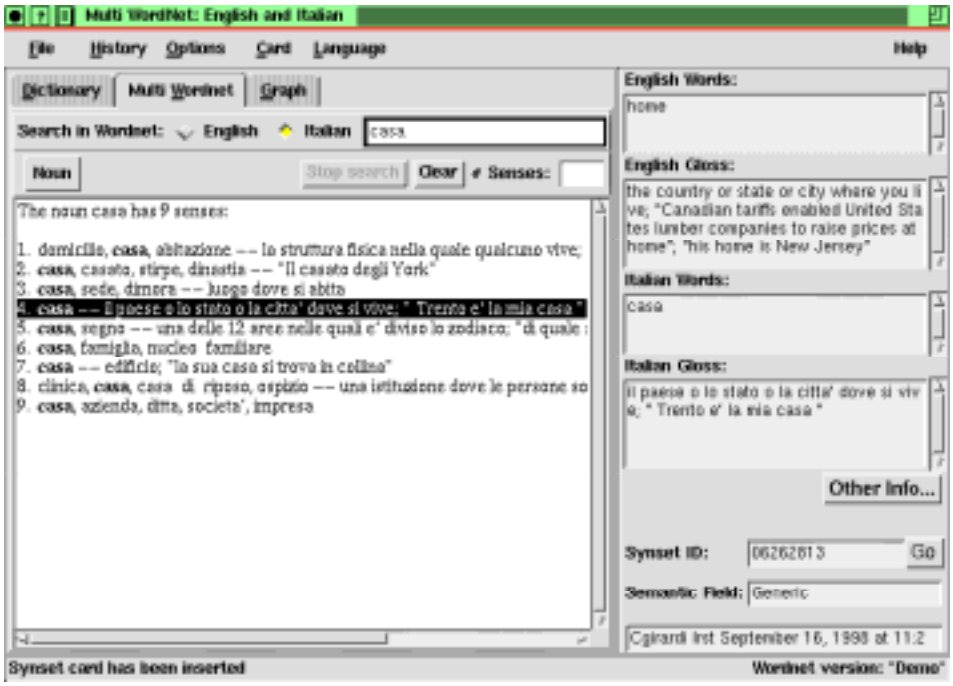


Figura 3.13 Multi-WordNet interfaccia grafica

Il progetto mira alla realizzazione di una matrice lessicale multi-lingue (MLLM) come estensione della matrice bidimensionale attualmente implementata nel WordNet, sarà quindi possibile, attraverso l'aggiunta di una terza dimensione, considerare diversi linguaggi. L'estensione della dimensione dei linguaggi verrà inizialmente considerata per l'Italiano. La figura 1 mostra le tre dimensioni della matrice: (a) parole in un linguaggio, indicate da W_j; (b) significati, indicati da M_i; (c) linguaggi, indicati da L_k. La matrice multi-lingue è stata realizzata ri-mappando le forme lessicali italiane con i significati corrispondenti (M_i), costruendo l'insieme dei synsets per l'italiano (esplicitando i valori d'intersezione E_{ij}^{I}). Il risultato è una completa ridefinizione delle relazioni lessicali, mentre per le relazioni semantiche sono state sfruttate, per quanto possibile, quelle già definite originariamente per l'inglese.

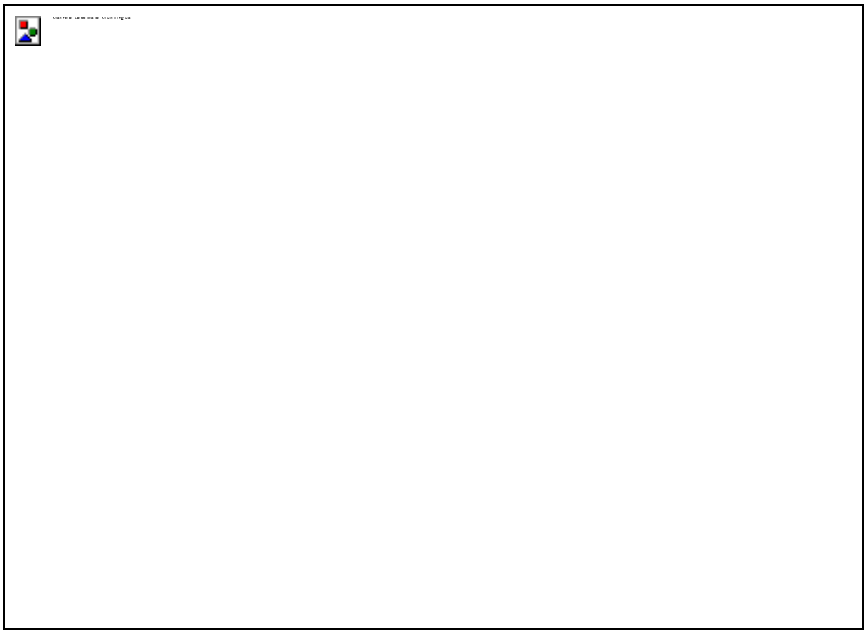


Figura 3.14: Matrice lessicale multilingua

La dimensione dei significati viene considerata costante rispetto alle lingue e alle parole di ogni lingua. Se per un certo M_k si ottiene $E_{ik}{}^{L} = \{0,...,0\}$ significa che per il linguaggio L non esiste nessuna parola che realizza lessicalmente quel significato.

Nelle gerarchie generate dalla relazione di iponimia/iperonimia in ogni lingua sono mantenuti gli stessi livelli, qualora per una lingua non esistesse il significato, il livello viene riempito con quello relativo della lingua inglese (vedi figura 3.15).

1. ditta, casa, societa`, impresa
=> azienda, impresa
=> impresa
=> organizzazione
=> gruppo_sociale
=> gruppo
2. clinica, nido, casa, ospizio
=> institution
=> establishment
=> struttura
=> manufatto
=> oggetto
=> entita`
3. casa
=> domicilio, casa, dimora, residenza, abitazione
=> housing, lodging
=> struttura
=> manufatto
=> oggetto
=> entita`
4. domicilio, casa, dimora, residenza, abitazione
=> housing, lodging
=> struttura
=> manufatto
=> oggetto
=> entita`
5. casa, famiglia
=> unit, social_unit
=> organizzazione
=> gruppo_sociale
6. casa, segno
=> parte, regione
=> ubicazione
=> oggetto
=> entita`
7. casa
=> ubicazione
=> oggetto
=> entita`
8. casa, sede, dimora
=> residenza
=> domicilio, Indirizzo
=> punto_geografico
=> punto
=> ubicazione
=> oggetto
=> entita`
9. casa
=> famiglia
=> lineage, line, line_of_descent, descent, bloodline, blood_line, blood, pedigree, ancestry, origin, parentage, stock
=> genealogia, albero_genealogico
=> cosca, clan
=> gruppo_sociale
=> gruppo

Figura 3.15 Lista degli iperonimi di casa

Il principale compito per costruire una MLKB (base di conoscenza lessicale multilingua) basata sul WordNet è stato quello di trovare la corretta corrispondenza tra le parole in Italiano e i *synset* (insieme dei sinonimi) definiti per l'inglese. Per costruire in modo automatico la matrice LKB (base di conoscenza lessicale) ci sono due principali problemi:

- L'abilità di estrarre dati relativi a parole non-Inglesi dalle sorgenti disponibili;
- Trovare l'esatta corrispondenza tra gli altri linguaggi e l'Inglese e vice versa.

La figura 2 mostra le relazioni tra queste due dimensioni per una bilingue LKB (Italiano-Inglese). La conoscenza lessicale per l'Italiano sarà acquisita da un dizionario in formato elettronico della lingua Italiana. Per la traduzione è usato un dizionario bilingue, Italiano/Inglese e Inglese/Italiano.

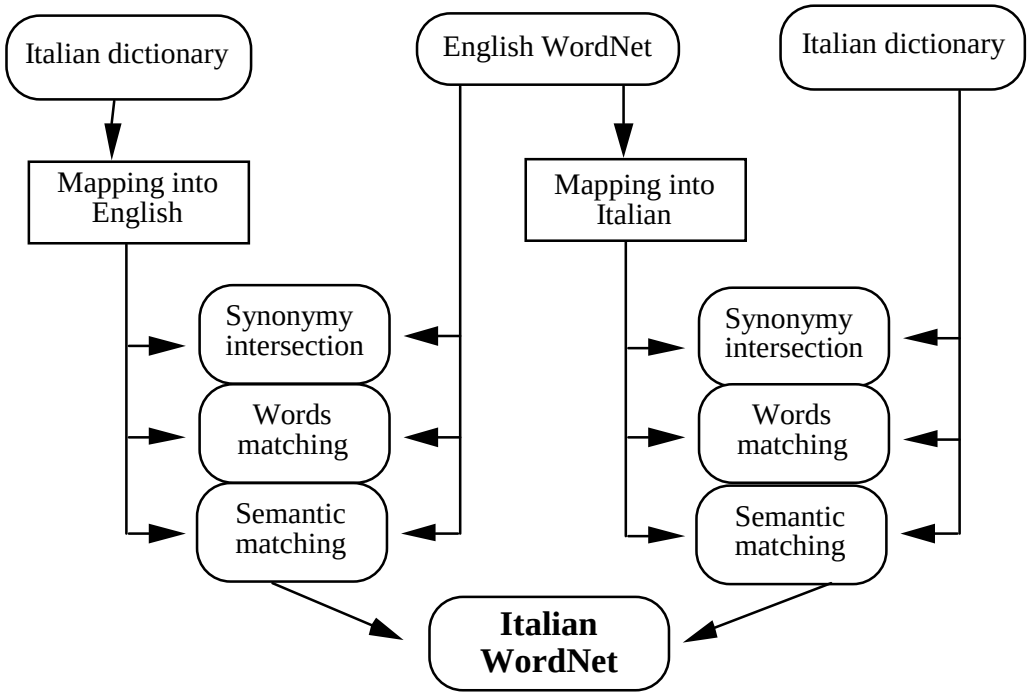


Figura 3.16: Procedura di estrazione e di confronto dei dati.

Il metodo è basato su un approccio duale che parte dalle definizioni Italiane o Inglesi; in entrambi i casi, ci saranno parole che non avranno un significato corrispondente nell'altro linguaggio. In particolare, molte parole Italiane non avranno nessun synset da riferire, e cosi saranno aggiunti nuovi nodi mantenendo la classificazione del WordNet.

Data la relativa semplicità delle definizioni Inglesi contenute nel WordNet, si avranno migliori risultati quando partiremo da esse. È importante in ogni modo l'intervento di un lessicografo per validare le scelte proposte.

Capitolo 4

La Costruzione dello Schema Integrato

Nell'ambito di un approccio semantico alla integrazione di dati (qual è quello adottato nel progetto **Momis**) è di fondamentale importanza la conoscenza delle informazioni semantiche relative al contesto e alla struttura dei vari schemi sorgenti. Tale conoscenza è contenuta nel cosiddetto *Common Thesaurus*, un dizionario all'interno del quale sono presenti un insieme di relazioni terminologiche che legano tra loro classi di attributi.

Lo sforzo compiuto a questo livello è quello di rendere il processo il più possibile automatico, minimizzando l'intervento del progettista, la cui partecipazione si rende comunque indispensabile.

4.1 Generazione de *Thesaurus*

Lo scopo di questa fase è la costruzione di un *Thesaurus* di relazioni terminologiche che rappresenta la conoscenza a disposizione sulle classi da integrare e che sarà la base per i calcoli di affinità tra le classi stesse.

Definiamo un modello di rappresentazione delle classi. Sia $S=\{S_1, S_2, \ldots, S_N\}$ un insieme di schemi di N sorgenti eterogenee, sia $c_{ij} \in S_i$ la generica classe del *source* di posto i. Ogni classe è caratterizzata da un nome e da un insieme: $c_{ij}=(n_{c_{ij}}, A(c_{ij}))$. Dove $A(c_{ij})$ rappresenta l'insieme degli attributi della classe c_{ij}, ognuno dei quali, a sua volta, possiede un nome ed un dominio di valori che può assumere: $A(c_{ij}) = \{a_1, a_2, \ldots, a_h, \ldots\}$ con $a_h = (n_h, d_h)$, dove n_h è il nome e d_h è il dominio associato ad a_h. Si parlerà inoltre generalmente di *termine* t_i indicando con esso un nome di classe e attributo.

Sia *T* l'insieme dei nomi di tutte le classi e di tutti gli altri attributi appartenenti ai vari schemi sorgenti.

Dati due diversi nomi di classi e/o attributi $t_i, t_j \in T$ tali che $t_i \neq t_j$, le relazioni che poosono legare questi due termini sono le seguenti:

- SYN (*synonym_of*): i due termini sono sinonimi, cioè possono essere interscambiati nelle sorgenti perché rappresentano lo stesso concetto della realtà. La relazione SYN gode della proprietà di simmetria, ovvero (t_i SYN t_j) - (t_j SYN t_i). Esempio: `faculty SYN faculty_name.`
- BT (*boader_term*): (t_i BT t_j) equivale ad affermare che t_i ha un significato più ampio di t_j. Esempio: `CS_Person BT Professor`
- NT (*narrower_term*): questa relazione esprime il concetto opposto della relazione BT, ovvero (t_i NT t_j) - (t_j BT t_i). Esempio: `Professor NT CS_Person.`
- RT (*related_term*) definita fra due termini che in qualche modo sono legati, perché appartenenti ad un medesimo contesto. La relazione RT gode della proprietà di simmetria, vale a dire (t_i RT t_j) - (t_j RT t_i). Esempio: `Research_Staff RT Department.`

In figura 4.1 sono evidenziate le fasi del processo che porta alla generazione del *Thesaurus* Comune.

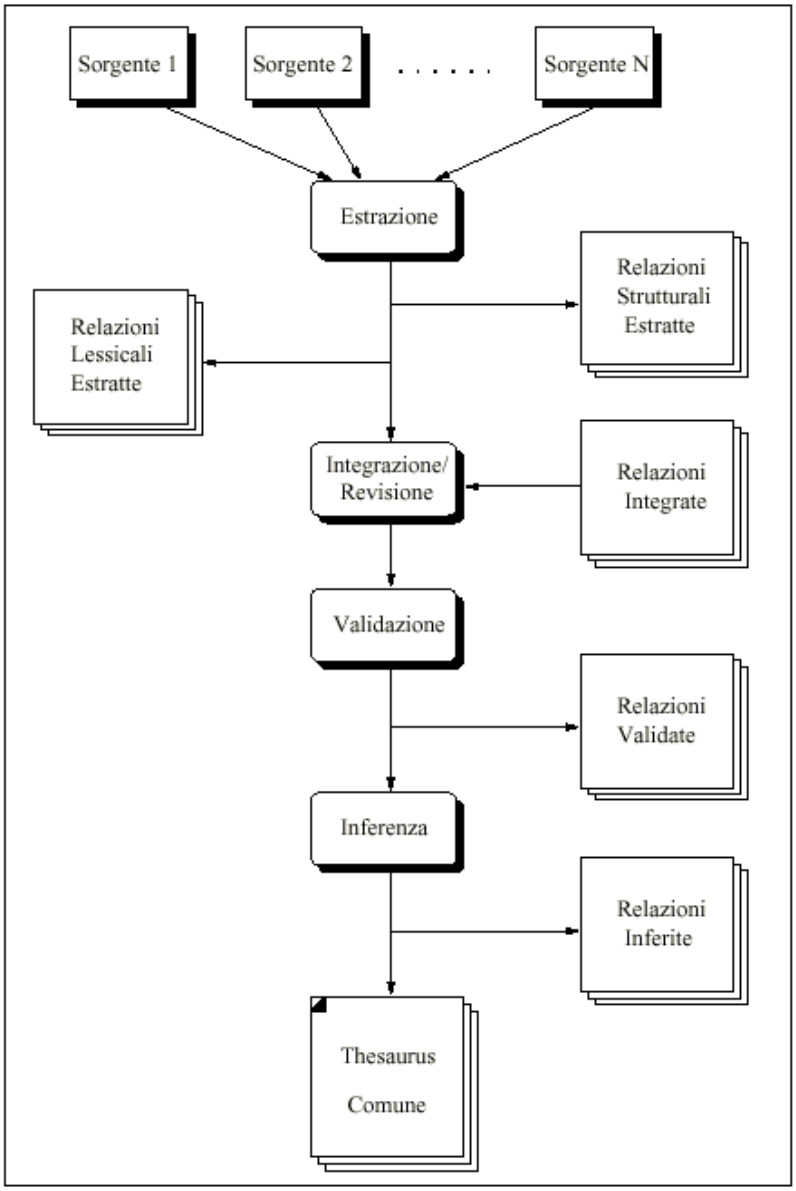


Figura 4.1: Processo di generazione del *Thesaurus* Comune

4.1.1 Estrazione delle relazioni INTRA-SCHEMA

La struttura di ogni schema locale contiene delle informazioni semantiche ricavabili dalle gerarchie di ereditarietà e aggregazione. Come primo passo verso la costruzione del *Thesaurus* c'è l'acquisizione degli schemi locali, dai quali

successivamente il sistema è in grado di estrarre in modo automatico alcune relazioni.

Il modulo che si preoccupa di estrarre le relazioni è: SIM (***S**ource **I**ntegrator **M**odule*). Le relazioni estratte in questa fase sono tutte intra-schema e non esistono relazioni di tipo SYN: infatti, all'interno di uno stesso schema, due classi strutturalmente identiche sono necessariamente la stessa classe.

1. Da un'analisi delle sorgenti relazionali sono estratte le relazioni terminologiche definite dalle *foreign key:*
    - se la *foreign key* è anche chiave primaria nella relazione di partenza e in quella referenziata allora la relazione terminologica è di tipo BT/NT;
    - altrimenti è semplicemente RT.

2. Da un'analisi delle sorgenti ad oggetti le relazioni estratte sono:
    - BT/NT ogni volta che c'è ereditarietà tra classi;
    - RT quando si è in presenza di gerarchie di aggregazione.

Esempio 1: considerando l'esempio di riferimento, si ricavano le seguenti relazioni terminologiche:

```
(CS.Professor NT CS.CS_Person)
(CS.Student NT CS.CS_Person)
(CS.Professor RT CS.Division)
(CS.Student RT CS.Corse)
(CS.Division RT CS.Location)
(CS.Course RT CS.Professor)
(UNI.Research_Staff RT UNI.Department)
(UNI.Research_Staff RT UNI.Section)
(UNI.Section RT UNI.Room)
```

Se nella relazione `UNI.Research_Staff` avessimo avuto anche la *foreign key:*

```
foreign key (first_name, last_name) references School_Member
```

allora avremmo avuto anche la relazione `(UNI.Research_Staff` BT `UNI.School_Member).`

4.1.2 Estrazione delle relazioni INTER-SCHEMA

Le relazioni estratte in questa fase sono inter-schema, vale a dire, relazioni che legano i termini (nomi di classi o attributi) di una sorgente con quelli di un'altra. Sono relazioni lessicali derivanti dai significati dei termini. Tali significati, sono stati assegnati dal progettista nella fase di progettazione della base di dati. La scarsa probabilità che due persone usino lo stesso termine, tra i vari sinonimi a disposizione in una lingua, nel descrivere la stessa cosa, sommato al grado di polisemia insita nelle parole, rendono queste relazioni ricche di ambiguità.

Poiché ogni informazione è un'opportunità che deve essere presa in considerazione, queste relazioni vengono estratte, dal modulo SLIM (*Schemata Lessical Integrator Module*) attraverso l'uso di un *database* lessicale come il *WordNet*, e dopo essere state disambiguate e validate, dal progettista, vengono inserite nel *Thesaurus* come relazioni semantiche intensionali. Compito di questa tesi è di fornire degli strumenti al progettista per la risoluzione dell'ambiguità, cercando di limitare e di facilitare i sui interventi il più possibile, quest'argomento è trattato in modo approfondito nel capitolo 5.

Nella fase di estrazione delle relazioni, può capitare che il processore morfologico di *WordNet* non riconosca alcuni termini (ad esempio `CS_Person, faculty_name),` ed è compito del progettista, attraverso un *tool* apposito inserire la **forma base** corretta (`Person, faculty`).

I tipi di relazioni lessicali estratte con l'utilizzo di *WordNet*, prima di essere inserite nel *Thesaurus* vengono tradotte in relazioni semantiche in base alla seguente corrispondenza:

- **Sinonimia:** corrisponde ad una relazione **SYN**;
- **Iperonimia:** corrisponde ad una relazione **BT**;
- **Omonimia:** corrisponde ad una relazione **RT**;
- **Correlazione:** corrisponde ad una relazione **RT**.

Esempio 2: Considerando l'esempio di riferimento, otteniamo le seguenti relazioni**:**

```
(CS.CS_Person.name SYN TP.Un_Student.name)
(CS.CS_Person.name BT UNI.Research_Staff.first_name)
```

```
(CS.CS_Person.name BT UNI.School_Member.first_name)
(CS.CS_Person.name BT UNI.Research_Staff.last_name)
(CS.CS_erson.name BT UNI.School_Member.last_name)
(TP.Un_tudent.name BT UNI.Research_Saff.first_name)
(TP.Un_Student.name BT UNI.School_ember.first_name)
(TP.Un_Student.name BT UNI.Research_Staff.last_name)
(TP.Un_Student.name BT UNI.School Member.last name)
(UNI.Research_Staff.last_name SYN UNI.School_Member.last_name)
(UNI.Research_Staff.first_name SYN UNI.School_Member.first_name)
(CS.Professor RT TP.Un_Student.faculty_name)
(CS.Professor RT UNI.School_Member.faculty)
(CS.Professor.rank SYN CS.Student.rank)
(CS.Student.year SYN UNI.School_Member.year)
(UNI.School_Member.faculty SYN TP.Un_Student.faculty_name)
(UNI.Section.room_code SYN University.Room)
```

4.1.3 Integrazione delle relazioni

Oltre alle relazioni (intra-schema, inter-schema) fin qui viste, il progettista può arricchire ulteriormente il *Thesaurus* aggiungendo ulteriori relazioni. Questa è un'operazione molto delicata, infatti, un errore ci potrebbe portare ad avere un *Thesaurus* errato. Per evitare questa situazione il progettista, nella fase di validazione, viene aiutato dal modulo ARTEMIS.

Esempio 3: Nell'esempio di riferimento il progettista aggiunge le seguenti relazioni:

```
(UNI.Research_Staff BT CS.Professor)
(UNI.School_Member BT CS.Student)
(TP.Un_Student BT CS.Student)
(UNI.Department BT CS.Division)
(UNI.Section SYN CS.Course)
(UNI.Research_Staff.dept_code BT CS.Professor.belongs_to)
(UNI.Department.dept_name SYN CS.Division.description)
(UNI.Section.section_name SYN CS.Course.course_name)
(UNI.School_Memeber.faculty SYN TP.Un_Student.faculty_name)
(CS.Division.fund SYN UNI.Department.budget)
(UNI.Department.dept area SYN CS.Division.sector)
```

4.1.4 Validazione del *Thesaurus* Comune

A questo punto il sistema analizza tutte le relazioni aggiunte nelle fasi precedenti e decide quali tra queste sono ritenute valide e quali no. La validazione

delle relazioni intensionali tra attributi e quelle estensionali tra classi sono validate utilizzando ODB-Tools.

Siano $a_t = (n_t, d_t)$ e $a_q = (n_q, d_q)$ una coppia di attributi posti in relazione tra loro. Secondo il tipo di relazione, per la validazione sono stati adottati i seguenti criteri:

- (n_t SYN n_q): la relazione è *valicata* se d_t e d_q sono equivalenti o se uno dei due è più specializzato dell'altro;
- (n_t BT n_q): la relazione è *valida* se d_t contiene d_q o è equivalente a d_q;
- (n_t BT n_q): analogamente al caso precedente la relazione è *valida* se d_t è un sottoinsieme non proprio di d_q;

Nota bene: Quando il dominio di un attributo d_q (d_t) è definito utilizzando il costruttore union, la realizzazione che coinvolge quell'attributo è *valida* se almeno uno dei suoi domini rispetta i criteri sopra elencati.

Esempio 4: Nella fase di validazione, ad ogni relazione esaminata è associato un *flag* booleano, con il valore "1" la relazione è valida mentre con "0" no.

Riferendoci all'esempio di riferimento ODB-Tools produce il seguente risultato:

```
(CS.CS_Person.name SYN TP.Un_Student.name)                          [1]
(CS.CS_Person.name BT UNI.Research_Staff.first_name)                [1]
(CS.CS_Person.name BT UNI.School_Member.first_name)                 [1]
(CS.CS_Person.name BT UNI.Research_Staff.last_name)                 [1]
(CS.CS_Person.name BT UNI.School_Member.last_name)                  [1]
(TP.Un_Student.name BT UNI.Research_Staff.first_name)               [1]
(TP.Un_Student.name BT UNI.School_Member.first_name)                [1]
(TP.Un_Student.name BT UNI.Research_Staff.last_name)                [1]
(UNI.Research_Staff.last_name SYN UNI.School_Member.last_name)      [1]
(UNI.Research_Staff.first_name SYN UNI.School_Member.first_name)    [1]
(CS.Professor.rank SYN CS.Student.rank)                             [1]
(CS.Student.year SYN UNI.School_Member.year)                        [1]
(UNI.School_Member.faculty SYN TP.Un_Student.faculty_name)          [1]
(UNI.Research_Staff.dept_code BT CS.Professor.belongs_to)           [0]
(UNI.Department.dept_name SYN CS.Division.description)              [1]
(UNI.Section.section_name SYN CS.Course.course_name)                [1]
(UNI.School_Memeber.faculty SYN TP.Un_Student.faculty_name)         [1]
(CS.Division.fund SYN UNI.Department.budget)                        [1]
(UNI.Department.dept_area SYN CS.Division.sector)                   [1]
```

L'unica relazione non validata è: `(UNI.Research_Staff.dept_code` BT `CS.Professor.belongs_to)` infatti il dominio di `dept_code` è un integre mentre quello di `belongs_to` è un oggetto `UNI.Division`.

4.1.5 Inferenza di nuove relazioni

Sulla base delle relazioni fin qui ottenute, attraverso un procedimento automatico che fa uso di tecniche di inferenza, viene generato un nuovo insieme di relazioni legali tra classi, che contribuiscono ad arricchire ulteriormente (e completamente) il *Thesaurus*.

Per agire in questo senso occorre effettuare alcune modifiche agli schemi locali, costruendo cosi uno schema virtuale; è importante precisare che tali modifiche hanno carattere provvisorio, servono solamente per il calcolo di deduzione di nuove relazioni inferite.

Le **classi** sono inserite nello schema e collegate tra loro in modo da conformarsi alle relazioni esistenti:

- ogni relazione BT e NT dà luogo a una gerarchia di ereditarietà;
- la relazione SYN produce una doppia ereditarietà. Da cui emerge come le due classi siano vicendevolmente in rapporto *is_a*;
- ogni relazione RT produce un'aggregazione.

Notare come, oltre a ricomporsi i singoli schemi locali, le relazioni lessicali aggiungono nuovi collegamenti inter-schema, in modo tale che alla fine risulti un unico schema.

Gli **attributi** coinvolti nelle relazioni validate vengono organizzati in gerarchie nelle quali:

- la relazione SYN pone due attributi allo stesso livello;
- le relazioni BT e NT pongono a livello superiore il termine più generale.

Come risultato finale avremo diversi alberi gerarchici isolati: ogni attributo è rinominato, in particolare assume il nome del termine di più alto livello dell'albero cui appartiene.

Quest'ultima operazione ha lo scopo di rendere il più possibile confrontabili le intensioni dello schema prodotto dall'analisi delle relazioni tra le classi. In questo modo l'intervento di ODB-Tools., attraverso tecniche di intelligenza artificiale basate sul calcolo di Sussunzione, è in grado di inferire tra le classi locali nuove gerarchie di ereditarietà, tali che si traducono facilmente in nuove relazioni per il *Thesaurus*.

In figura 4.2 viene mostrato lo schema relativo all'esempio di riferimento, che rappresenta anche la forma grafica del *Thesaurus*. Per quanto riguarda le relazioni tra classi locali: le linee dotate di frecce rappresentano relazioni di generalizzazione, mentre quelle in cui le frecce sono assenti denotano le relazioni di aggregazione. Un'ulteriore distinzione è fatta fra le relazioni esplicite preesistenti (linee continue) e quelle inferite (linee tratteggiate).

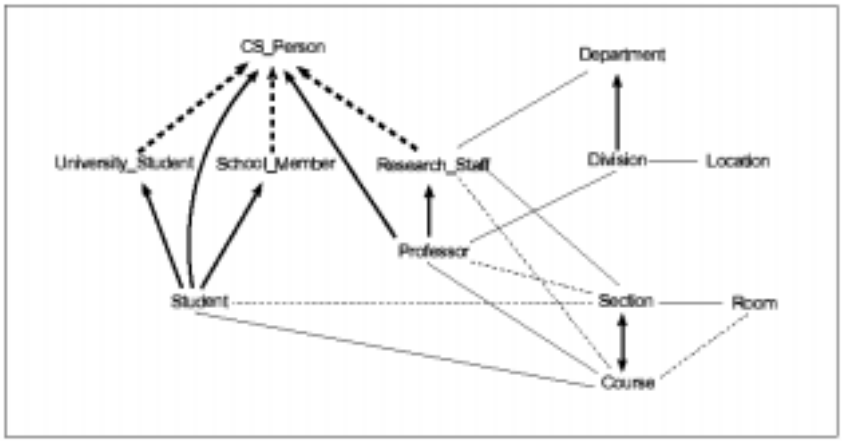


Figura 4.2: Rappresentazione grafica del *Thesaurus* comune

Esempio 5: le relazioni inferite sono le seguenti:

```
(CS.CS_Person BT UNI.Research_Staff)
(CS.CS_Person BT UNI.School_Member)
(UNI.Section RT CS.Professor)
(UNI.Research_Staff RT CS.Course)
(CS.Professor RT UNI.Department)
(CS.Professor RT UNI.Section)
(CS.Course RT UNI.Room)
(CS.Student RT UNI.Section)
```

```
(CS.CS_Person BT TP.Un_Student)
```

4.2 Generazione delle classi globali

La conoscenza semantica contenuta nel *Thesaurus* appena generato rappresenta il punto di partenza per il successivo processo che porta alla definizione dello schema globale integrato. Tale operazione viene suddivisa in tre fasi:

- calcolo delle affinità;
- generazione dei *cluster*;
- generazione degli attributi globali e delle *mapping-table*.

4.2.1 Calcolo delle affinità

Scopo di questa parte, descritta approfonditamente in [36], è quantificare il *grado di affinità* mutuo fra le classi locali, al fine di raggrupparle opportunamente in diversi *cluster*, da cui sarà possibile generare la vista globale.

L'affinità tra ogni coppia di classi è espressa attraverso un coefficiente, chiamato *Global Affinity* (*GA*) [38], calcolato come combinazione lineare di due fattori:

1. *Stuctural Affinity Coefficient*: valuta l'affinità strutturale delle classi;
2. *Name Affinity Coefficient*: valuta l'affinità che lega i nomi di classe e gli attributi

Attraverso questi coefficienti il *Thesaurus* viene riorganizzato in una struttura simile alle *Associative Network* [37], dove i nodi (ciascuno dei quali rappresenta genericamente un termine, sia esso il nome di una classe o il nome di un attributo) sono uniti attraverso relazioni terminologiche che sono presenti nel *Thesaurus*. Per dare una valutazione numerica dell'affinità tra due termini, ad ogni tipo di relazione viene associato un peso $\sigma_{R,}$ compreso nell'intervallo [0,1], che sarà tanto maggiore

quanto più questo tipo di relazione contribuisce a legare due termini: sarà quindi $\sigma_{syn} \geq \sigma_{bt/nt} \geq \sigma_{rt}$.

Il pesi adottati nelle sperimentazioni realizzate sono:

$\sigma_{syn} = 1$;
$\sigma_{bt} = \sigma_{bt} = 0.8$;
$\sigma_{rt} = 0.5$;

Un valore pari a "0" indica l'assenza di affinità trai i due termini, mentre il valore "1" indica che i termini sono affini e validi.

Durante il calcolo di questo coefficiente globale (*GA*), è comunque data implicitamente una maggiore rilevanza al *Name Affinity coefficient*, e quindi ai nomi delle classi stesse.

4.2.2 Generazione dei *cluster*

Per l'identificazione di classi affini negli schemi considerati sono utilizzate, all'interno del mediatore, tecniche di *clustering*, attraverso le quali le classi sono automaticamente classificate in gruppi caratterizzati da differenti livelli di affinità, formando un albero dove:

- Le foglie rappresentano tutte le classi locali:
  - quelle contigue sono caratterizzate da alta affinità;
  - quelle lontane tra loro rappresentano classi con bassa affinità.
- Ogni nodo rappresenta un livello di clusterizzazione ed ha associato il coefficiente di affinità tra i due sottoalberi (*cluster*) che unisce.

La procedura di *clustering* utilizza una matrice *M* di rango *r*, con *r* uguale al numero di classi ODL_{I^3} che devono essere analizzate. In ogni casella *M*[*h, k*] della matrice è rappresentato il coefficiente globale *GA*() riferito alle classi c_h e c_k.

La procedura di *clustering* è iterativa e comincia allocando per ogni classe un singolo *cluster*, successivamente, ad ogni interazione, i due *cluster* tra i quali sussiste il *GA*() di valore massimo, nella matrice *M*, vengono uniti. All'interazione successiva, viene aggiornata la matrice *M*, le righe e le colonne corrispondenti ai

precedenti *cluster* uniti vengono eliminate, e si inserisce una nuova riga ed una nuova colonna che rappresentano il nuovo *cluster* determinato. La procedura termina quando tutte le classi appartengono ad un unico *cluster*.

L'output di questa fase non è il *cluster* finale, contente tutte le classi, ma è l'albero che si è definito attraverso questa procedura di *clustering*, riportato in figura 4.3.

Il progettista può, ad ogni interazione immettere un valore di soglia in base al quale ARTEMIS costruisce i *cluster*; ogni *cluster* sarà quindi costituito da: tutte le classi appartenenti ad un sottoalbero che al nodo radice ha un coefficiente maggiore del valore di soglia. Nella figura 4.3 si è scelto come valore di soglia $\sigma = 0.5$: tutte le classi di partenza delle sorgenti dell'esempio di riferimento sono state raggruppate iterativamente (attraverso la procedura esposta sopra) in cinque *cluster* finali.

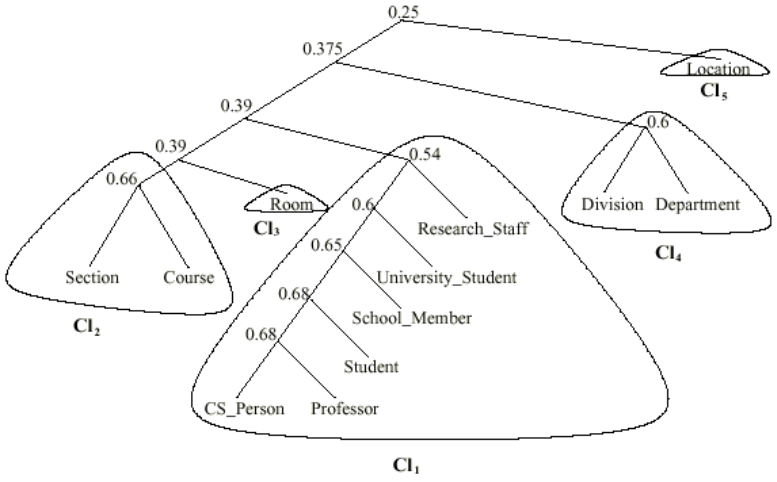


Figura 4.3: Albero delle affinità (*cluster* con $\sigma = 0.5$)

4.2.3 Costruzione delle classi globali

L'ultimo passo che porta alla generazione dello schema globale, è quello che associa una classe globale ad ogni *cluster* determinato nella fase precedente; 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* (definito come gruppo di classi ritenute affini) in classe globale, è articolato nelle seguenti fasi:

- Unione degli attributi di tutte le classi appartenenti al *cluster*;
- Fusione degli attributi simili.

Questa cosiddetta "Unione ragionata" degli attributi locali, è un'operazione importante e delicata: importante perché attraverso lo schema di attributi visibile all'utente, si deve dare a quest'ultimo la possibilità di porr *query* semplici ma espressive; delicata perché 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 memorizzate nel *Thesaurus* (vedi sezione 4.1) generato prima della fase di creazione dei *cluster*.

4.2.3.1 Unione degli attributi

Ogni *cluster* generato nella fase precedente è semplicemente una lista delle classi locali che ne fanno parte, cioè, che attraverso il calcolo di affinità illustrato nella sezione precedente, sono state ritenute simili. Questa fase consiste semplicemente nella raccolta e concatenamento, in un *insieme-unione*, di tutti gli attributi appartenenti alle suddette classi. Indicando con Cl_i il *cluster* in esame e con c_i una classe appartenente ad esso e con $A(Cl_i)$ gli attributi del *cluster* Cl_i, l'*insieme-unione* sarà dato da:

$$A(Cl_i) = \bigcup^{j} A(C_j), \quad \forall\, c_J \in Cl_j$$

Esempio 5: Riferendoco al *cluster* Cl_1 di Figura 4.3 l'unione degli attributi delle classi locali che li raggruppa fornisce il seguente *insieme-unione*:

$A(Cl_1)$ = {`first_name, last_name, relation, e_mail, Home_email, phd_email, dept_code, section_code, name, student_code, faculty_name, tax_fee, year, takes, title, belongs_to, faculty`}

Si noti come vi sia una certa ridondanza tra gli attributi: `faculty_name` e `faculty` in realtà hanno lo stesso significato di “nome della facoltà”, così come gli attributi `first_name, last_name,` e `name` rappresentano nomi di persona.

4.2.3.2 Fusione degli attributi

In questa fase il sistema è in grado di riconoscere le affinità degli attributi in base alle relazioni del *Thesaurus* Comune che li legano: tutti gli attributi affini vengono *fusi*. in uno solo, riducendo così le ridondanze presenti nell’*insieme-unione*.

In ogni sottoinsieme *S* di *U* viene scelto un attributo a_g e da *U* viene eliminato l’insieme di attributi $\{a: a \in S, a \neq a_g\}$, ovvero l’insieme di attributi simili ad a_g che per la classe globale sarebbero ridondanti: questa operazione prende il nome di *fusione* poiché è come se gli attributi *a* eliminati da *U* si siano *fusi* con a_g.

Al termine di questa fase, *U* è l’insieme di attributi non ridondanti della classe globale che si sta costruendo.

4.2.3.2.1 Attributi legati da relazioni validate

Una relazione validata che lega due attributi indica che c’è compatibilità tra i domini: il dominio di un attributo coincide o contiene il dominio dell’altro oppure, se gli attributi possono avere più domini (è il caso delle sorgenti semistrutturate), almeno un dominio dell’uno coincide o contiene un dominio dell’altro**.**

Sono eseguiti due tipi di fusione: la prima considera gli attributi legati da relazioni di sinonimia (SYN) mentre la seconda considera quelle validate di specializzazione (BT o NT). Le relazioni (RT), anche se validate non vengono considerate, perché rappresentano legami molto *deboli*.

La fusione avviene nel modo seguente:

- Tutti gli attributi sinonimi (o omonimi) vengono collassati in uno solo il cui nome potrebbe essere quello di uno solo dei partecipanti o, meglio,

potrebbe essere dato dal concatenamento dei singoli nomi, lasciando poi al progettista il compito di trovare un nome conciso ed appropriato.

- Ogni gerarchia di termini legati da relazioni di specializzazione (BT e NT) viene sostituita da un unico attributo, il cui nome coincide col termine più generale della gerarchia stessa.

4.2.3.2.2 Attributi legati da relazioni non validate

La fusione di attributi coinvolti in relazioni validate è un'operazione automatica; lo stesso non si può dire nel caso di relazioni non validate, per le quali il problema è più complesso.

Le relazioni non validate del *Thesaurus* indicano che nonostante i concetti espressi dai nomi degli attributi siano correlati, l'analisi sui domini mostra che gli attributi in questione sono incompatibili.

Per capire meglio il problema, prendiamo in esame l'esempio di riferimento. Una delle relazione non validate è la seguente:

```
UNI.Research_Staff.dept_code BT CS.Professor.belongs_to
```

Questa relazione non è valida perchè l'attributo `dept_code` appartiene all'insieme degli interi, mentre `belongs_to` è una collezione OID che realizza un'associazione tra le classi `CS.Professor` e `CS.Division`. Tuttavia un'analisi attenta dei sorgenti, mostra che l'attributo `dept_code` è definito come *foreign key* nella relazione `UNI.Research_Staff` (`UNI` è una sorgente relazionale) e realizza un'aggregazione fra le relazioni `UNI.Research_Staff` e `UNI.Department`.

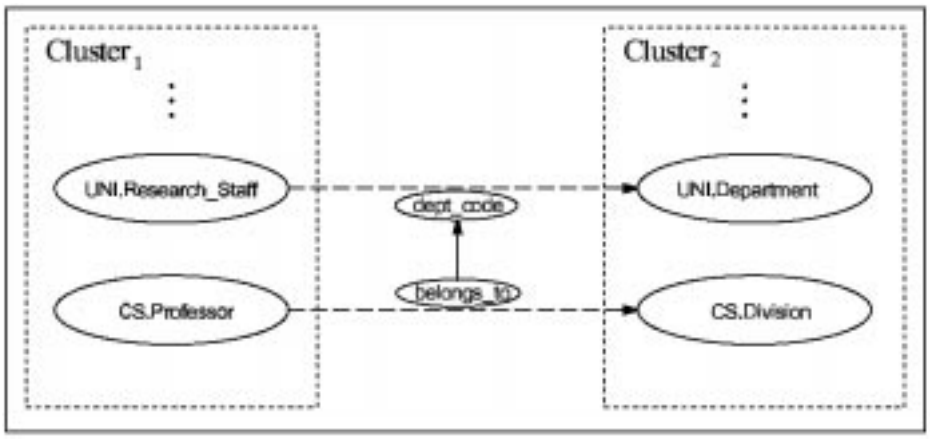


Figura 4.4: Fusione degli attributi contenuti in relazioni non valide

Dalla figura 4.4, che mostra graficamente il significato complesso dei due attributi, si nota che le classi *puntate* dai due attributi appartengono entrambe allo stesso *cluster*. In tal caso è opportuno che i due attributi sono fusi in uno solo:

`dept_code, belongs_to › works`

In questo caso, il dominio assunto dall'attributo globale sarà un dominio particolare che il gestore delle interrogazioni riconoscerà e si comporterà di conseguenza.

Possiamo quindi definire le condizioni affinché sia possibile raggruppare i due attributi appartenenti a relazioni non valide:

1. gli attributi devono rappresentare entrambi una gerarchia di aggregazione (collezione di attributi complessi o *foreign key*)
2. le rispettive classi locali mappate devono appartenere allo stesso *cluster* e la stessa condizione deve valere anche per le classi di appartenenza, ma questo è sempre vero perché, per la fusione, vengono considerate solamente le relazioni che legano attributi appartenenti ad un medesimo *cluster*;
3. gli attributi devono avere lo stesso significato semantico: condizione garantita se esiste una relazione di sinonimia o di specializzazione che li lega.

4.2.3.3 Generazione della *mapping table*

La *mapping table* (*MT*) rappresenta la conoscenza, che il sistema ha, sulla corrispondenza che esiste tra gli attributi globali con quelli locali; infatti, quando viene fatta una *query*, essa viene rivolta allo schema globale, e successivamente il sistema la rigira ai singoli schemi locali. Per fare ciò il sistema mantiene in una tabella i legami tra gli attributi globali (derivati da una manipolazione di quelli locali) e gli attributi locali che gli hanno originati, a questa tabella è dato il nome di *mapping table*.

Ogni *mapping table* è una tabella *MT*[*C*][*A*] dove *C* è l'insieme delle classi locali che appartengono al *cluster* a cui la *mapping table* si riferisce e *A* è l'insieme degli attributi globali ottenuto dopo la fusione degli attributi simili. Indicando con $c_i \in C$ il nome di una classe locale e con $a_g \in A$ il nome di un attributo globale, ogni elemento $MT[c_i][a_g]$ della tabella contiene una *mapping rule* ODL_{I^3}: una regola che indica il tipo di legame tra l'attributo globale a_g e le informazioni della classe c_i ovvero quali valori assume a_g ogni volta che il *Ouery Manager* accede *a* c_i. Non è detto tuttavia che venga definita una *mapping rule* per ogni coppia attributo globale-classe locale: può anche essere che un attributo globale non assuma alcun valore in corrispondenza ad una certa classe locale. In figura 4.5 sono riportati alcuni esempi di *mapping table*.

Dato un attributo globale, i possibili casi di *mapping* sono:

- **corrispondenza semplice**, che si verifica quando:
  - a_g rappresenta solamente le informazioni dell'attributo a_i per la classe c_i e non assume alcun valore per tutte le altre classi. Esempio l'attributo locale `employee_nr` della classe `CS.Division` non è coinvolto in nessuna relazione del *Thesaurus* quindi deve essere mappato in un attributo globale, che sarà caratterizzato dallo stesso dominio e assumerà lo stesso nome;
  - l'analisi di relazioni di sinonimia porta alla fusione di più attributi, appartenenti a classi locali diverse, in un unico attributo. È il caso degli attributi `faculty_name` e `faculty`, delle classi

`TP.University_Student` e `UNI.School_Member`, che sono fusi nell'attributo locale `faculty`.

- l'analisi delle relazioni di specializzazione porta alla fusione di più attributi, appartenenti a classi locali diverse, in un unico attributo. Ad esempio, gli attributi `dept_code` e `belongs_to`, rispettivamente delle classi `UNI.Research_Stafff` e `CS.Division`, sono fusi nell'attributo globale `works`.

- **Corrispondenza in *and*:** è usata in concomitanza alla fusione di più attributi appartenenti ad un'unica classe locale c_i e i valori che deve assumere l'attributo globale a_g sono il concatenamento dei valori contenuti negli attributi fusi. Ad esempio, gli attributi `first_name` e `last_name`, appartenenti alla medesima classe locale `UNI.Research_Staff`, sono fusi nell'attributo `name`.
- **Corrispondenza in *union*:** è analoga a quella in *and.* La differenza è che in questo caso, l'attributo globale a_g può assumere il valore di uno solo tra gli attributi fusi appartenenti alla classe c_i. La scelta del valore che l'attributo globale assume avviene quando il *Query Manager* deve eseguire l'integrazione delle risposte ottenute dalle sorgenti, ed è determinata dal valore di un attributo locale, denominato *tag*, appartenente sempre alla classe c_i.

  Nell'esempio di riferimento, per avere un'integrazione corretta, l'attributo globale `email` deve assumere il valore `home_email` o `phd_email` a seconda del valore assunto dall'attributo locale `rank` (attributo *tag*), appartenente anch'esso alla classe locale `CS.Student`.

Nelle classi locali dove un attributo globale non fa *mapping*, a volte può essere necessario aggiungere un valore di *default* per quell'attributo. In alcuni casi il progettista è a conoscenza di tale informazione, in altri casi l'informazione stessa è già presente negli schemi locali come *metadato* e, se non si opera in questo senso, si rischia di perderla nello schema globale, riducendo la capacità si ottenere risposte corrette dalle interrogazioni.

A volte l'unione e la successiva fusione di attributi non sono sufficienti per rappresentare, nello schema globale, alcune informazioni ricavabili dalla struttura dei singoli schemi locali. L'aggiunta di uno o più attributi globali da parte del progettista risolve il problema e arricchisce ulteriormente il contenuto informativo dello schema integrato.

University_Person	name	rank	works	faculty	email	...
Research_Staff	first_name and last_name	'Professor'	dept_code	null	email	...
School_Member	first_name and last_name	'Student'	null	faculty	null	...
CS_Person	name	null	null	'Computer_Science'	null	...
Professor	name	rank	belongs_to	'Computer_Science'	null	...
Student	name	rank	null	'Computer_Science'	case rank of 'course':home_mail 'phd':phd_mail	...
University_Student	name	'Student'	null	faculty_name	null	...

Workplace	name	area	employer	budget	...
Department	dept_name	dept_area	null	budget	...
Division	description	sector	employer	fund	...

Figura 4.5: alcune *mapping table* create con l'esempio di riferimento: le classi globali `University _Person` e `Workplace`

Dopo la generazione di una *mapping table* per ogni classe globale, il sistema conclude l'integrazione con la creazione della descrizione ODL_I dello schema globale [38]. Vedere la figura 4.6 per una parte della descrizione ODL_{I^3} di una classe globale.

```
interface University_Person
(extent Research_Staffers, School_Members, CS_Person
        Professors, Students, University_Students
 key    name)
{attribute string name
    mapping_rule (University.Research_Staff.first_name and
                  University.Research_Staff.last_name),
                  (University.School_Member.first_name and
                  University.School_Member.last_name),
                  Computer_Science.CS_Person.name,
                  Computer_Science.Professor.name,
                  Computer_Science.Student.name,
                  Tax_Position.University_Student.name;
 attribute string rank
    mapping_rule University.Research_Staff = 'Professor',
                 University.School_Member = 'Student',
     ... }
```

Figura 4.6: Esempio di classe globale in ODL_{I^3}

Capitolo 5

Rimozione dell’ambiguità

Lo scopo di questo lavoro è di automatizzare la fase di integrazione delle relazioni lessicali, cercando di limitare l’intervento del progettista nella scelta dei significati dei termini che compongono le relazioni. Il problema nasce dall’ambiguità data dal grado di polisemia dei termini, vale a dire la proprietà di una parola di assumere più di un significato (vedi Figura 5.1). Risolvere l’ambiguità non è un problema di facile soluzione poiché la lingua, anche se formata da regole ben definite, presenta sempre delle eccezioni.

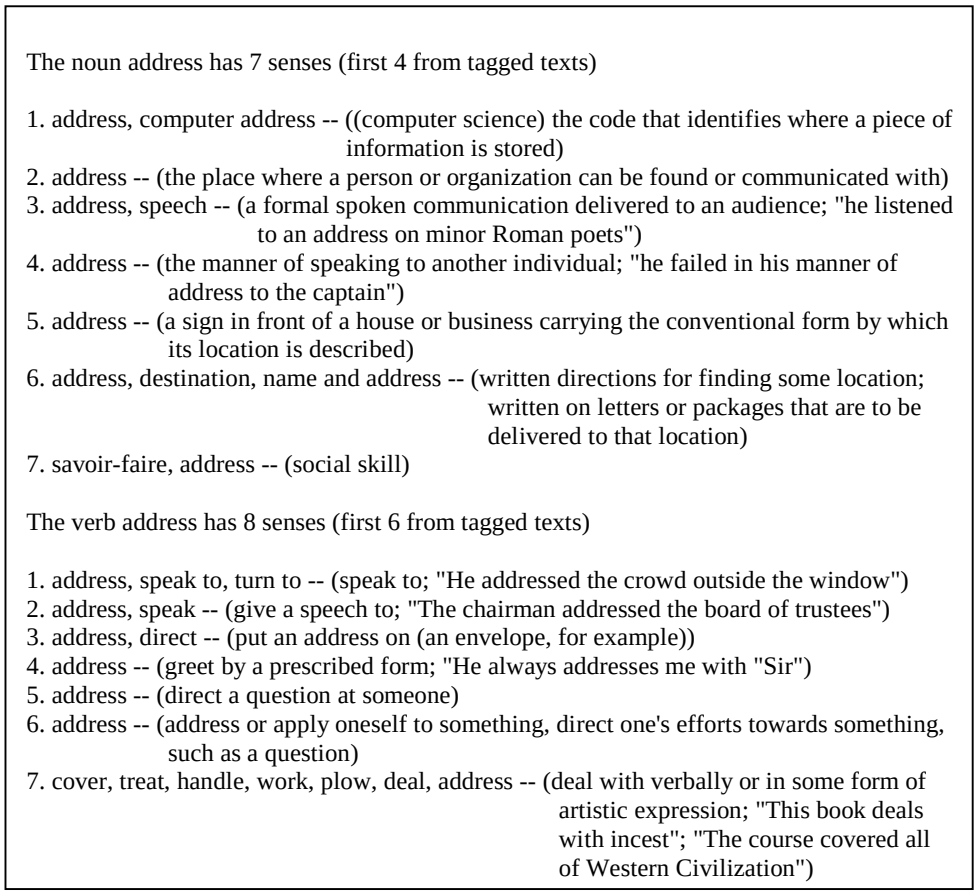


Figura 5.1 significati di address dati dal *WordNet*

Tenendo presente i problemi che derivano dall'ambiguità di una lingua, sono state sviluppate delle tecniche che più si adattano al nostro specifico caso, secondo una filosofia "*think big act* small".

Precedentemente, l'unico metodo per attribuire i significati ai termini era quello manuale; il progettista per ogni termine era costretto a scegliere uno fra significati presenti nel *WordNet,* mentre ora sono stati introdotti strumenti in grado di sfruttare sia le informazioni degli schemi delle sorgenti, sia la struttura del *WordNet* attraverso l'uso di *KeyWord* e *SemanticField,* rendendo in tal modo il processo più automatico.

5.1 Ipotesi semplificativa.

Prima di procedere con l'analisi degli strumenti di disambiguazione occorre formulare un'ipotesi semplificativa che nasce dalle seguenti osservazioni:

I[a] osservazione – Il generico termine di cui dobbiamo trovare le relazioni è o un nome di una generica classe o uno dei suoi attributi e, sempre o quasi sempre, questi sono dei nomi (punto di vista grammaticale).

II[a] osservazione – Nella lingua inglese, diversamente da quell'italiana, per ogni parola che identifica un verbo (un'azione) ne esiste quasi sempre, un'uguale che identifica un nome (il nome dell'azione). Consideriamo il seguente esempio:

the verb to "run" -- (move fast by using one's feet, with one foot off the ground at any given time)

the noun "run" -- (a race run on foot; "she broke the record for the half-mile run");

nella lingua italiana avremmo avuto due parole completamente differenti (correre e corsa)

III[a] osservazione – "*... the language has far fewer verbs than nouns. For example, the Collins English Dictionary lists 43,636*

> *different nouns and 14,190 different verbs. Verbs are more polysemous than nouns: the nouns in Collins have on the average 1.74 senses, whereas verbs average 2.11 senses*
> *The higher polysemy of verbs suggests that verb meanings are more flexible than noun meanings*." (estratto da: "*Introduction to WordNet*" file "5papers.pdf " scaricabile dal sito della *Princeton University*)
> È evidente, quindi, che l'elevata polisemia che caratterizza i verbi tende ad accrescere l'ambiguità.

In base alle suddette osservazioni possiamo formulare l'ipotesi di considerare i nostri termini (nomi di classi ed attributi) solo come nomi (in realtà rientrano in questi nomi anche 8400 verbi che soddisfano la seconda osservazione).

5.2 Utilizzo della struttura degli schemi per acquisire conoscenza lessicale e diminuire l'ambiguità

Un metodo idoneo ad assegnare il giusto significato è quello di sfruttare la struttura degli schemi, in modo da trovare delle relazioni strutturali che siano allo stesso tempo anche lessicali.

5.2.1 Utilizzo delle relazioni interschema

Uno dei metodi utilizzati è quello che sfrutta le relazioni intra-schema. Il processo di estrazione dei significati è completamente automatico, occorre però l'intervento del progettista, il quale può accettare o scartare i significati trovati.

Analizziamo i passi del processo:

1. Acquisisco le relazioni intraschema estratte da SIM, utilizzando ODB-Tools, descritte nel linguaggio descrittivo odl_{i_3}.
2. Arricchisco la "conoscenza" lessicale dei termini sfruttando le relazioni intraschema:

- Utilizzando il *WordNet*, verifico se esistono delle relazioni intraschema che siano in relazione anche dal punto di vista lessicale.

  Consideriamo come esempio la relazione intraschema <`CS_person BT professor`> dalla struttura del WordNet (Figura 5.2) riesco a dedurre in modo automatico il *sense* di *person* infatti il *sense* 1 (a human being; “there was too much for one person to do”) di person è in relazione con professor anche dal punto di vista lessicale.

  Sfruttando le relazioni che soddisfano il p.to precedente, sarà quindi possibile ricavare in modo **automatico** il *sense* di alcuni dei termini.

3. Infine il progettista decide se accettare o scartare i significati trovati.

Come si può osservare dalla figura 5.2 il nome `professor` è una specializzazione di `person`, ciò indica che i due nomi sono in relazione, oltre che dal punto di vista strutturale (relazione intraschema `CS_person BT professor`) anche da quello lessicale, quindi, dal p.to di vista statistico è più probabile che l’esatto significato da assegnare, per il nostro contesto, sia il primo (*a human being; "there was too much for one person to do"*) e di conseguenza si scartano i rimanenti due.

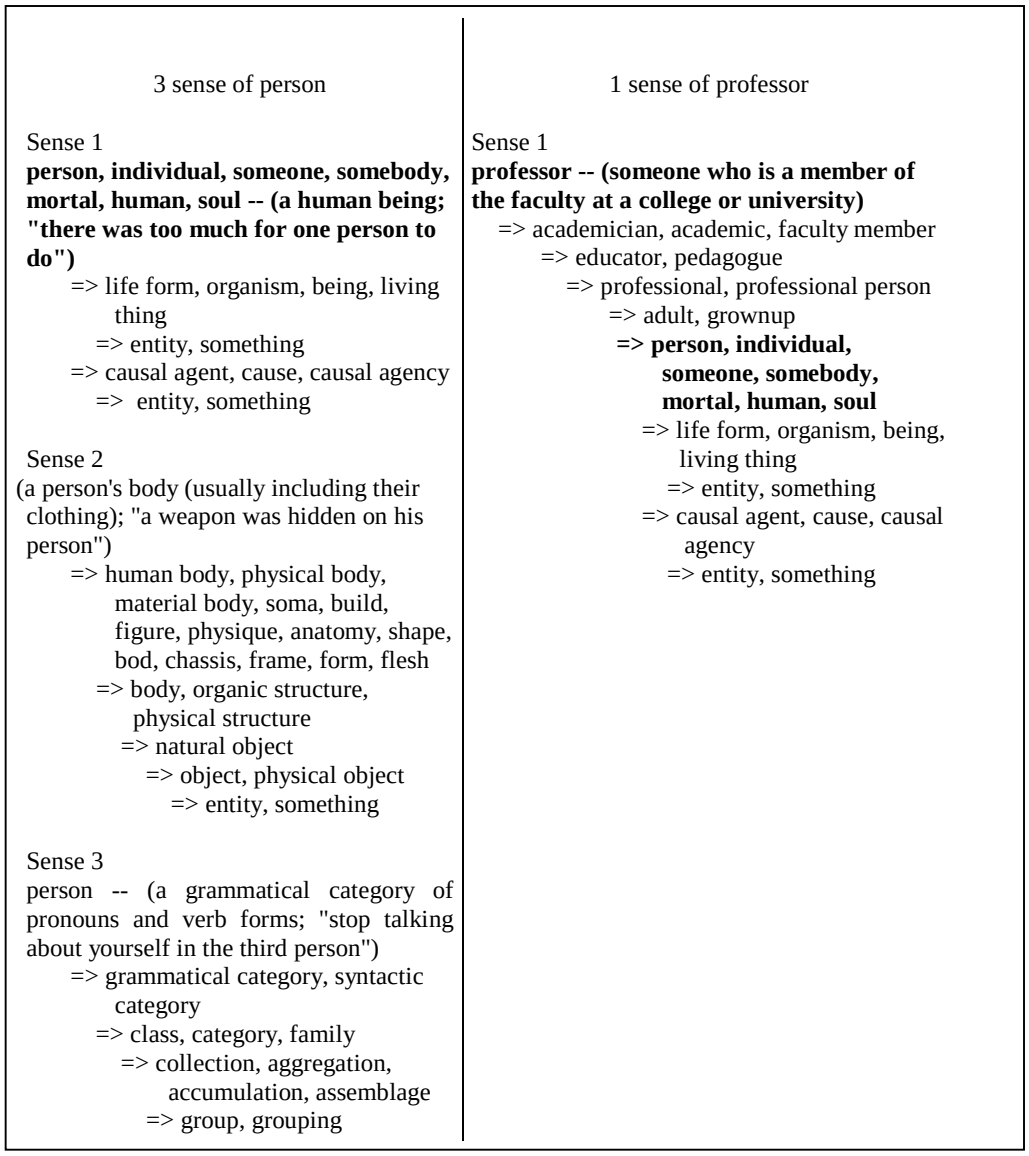


Figura 5.2: Struttura di WordNet dei nomi `person` e `professor`.

5.2.2 Utilizzo del legame tra nome di classe e attributi.

Un altro metodo implementato per estrarre i significati è quello di controllare se esiste per un nome di classe un legame lessicale con i suoi attributi, infatti, se esso esiste è molto probabile che i significati dei due termini in relazione siano quelli relativi al nostro contesto (es.: considero l'attributo `city` appartenente alla classe `location` `CS.location.city` vedi figura 5.3).

3 senses of location

Sense 1
location -- (a point or extent in space)
=> object, physical object
=> entity, something

Sense 2
location, locating, placement, position, positioning, emplacement, situating -- (the act of putting something in a certain place or location)
=> activity
=> act, human action, human activity

Sense 3
localization, localisation, location, locating, fix -- (a determination of the location of something; "he got a good fix on the target")
=> determination, finding
=> discovery, find, uncovering
=> deed, feat, effort, exploit
=> accomplishment, achievement
=> action
=> act, human action, human activity

3 senses of city

Sense 1
city, metropolis, urban center -- (a large and densely populated urban area; may include several independent administrative districts; "Ancient Troy was a great city")
=> municipality
=> urban area
=> geographical area, geographic area, geographical region, geographic region
=> region
=> location -- (a point or extent in space)
=> object, physical object
=> entity, something
=> administrative district, administrative division, territorial division
=> district, territory
=> region
=> **location**
=> object, physical object
=> entity, something

Sense 2
city -- (an incorporated administrative district established by state charter; "the city raised the tax rate")
=> administrative district, administrative division, territorial division
=> district, territory
=> region
=> **location -- (a point or extent in space)**
=> object, physical object
=> entity, something

Sense 3
city, metropolis -- (people living in a large densely populated municipality; "the city voted for Republicans in 1994")
=> municipality
=> gathering, assemblage
=> social group
=> group, grouping

Figura 5.3 Struttura di *WordNet* dei nomi `location` e `city`

I passi da fare sono i seguenti:

1. Estrarre le informazione delle nostre sorgenti descritte in odl_{i^3}.
2. Utilizzando il *WordNet* si verifica se esistono delle relazioni lessicali tra nome di classe e i suoi attributi.
3. Visualizzare i risultati al progettista il quale decide di accettare o scartare i significati assegnati.

Il terzo passo, anche se abbassa il livello d’automazione, è fondamentale perché, alcune volte, si può verificare che ci siano dei significati non inerenti, infatti, se si osserva la figura 5.3, si vede che `city` e `location` sono legati lessicalmente da due relazioni che individuano un unico significato per il nome `location` (*sense* 1):

1. location -- (a point or extent in space);

e due significati distinti per il nome *city* (*senses* 1 e 2):

1. city, metropolis, urban center -- (a large and densely populated urban area; may include several independent administrative districts; "Ancient Troy was a great city");
2. city -- (an incorporated administrative district established by state charter; "the city raised the tax rate").

Sono stati aggiunti altri due metodi che come i precedenti qui descritti sfruttano la struttura degli schemi e sono:

- sfruttamento del legame tra attributi di una stessa classe.
- sfruttamento del legame tra sorgenti di una stessa classe.

Questi due metodi verranno descritti nel paragrafo 5.5.

5.3 Ridurre l'ambiguità raggruppando i significati.

Le due tecniche che seguono, *keyword* e *semanticfield*, forniscono uno strumento per la risoluzione dell'ambiguità diverse dalle precedenti, vale a dire si astraggono dalla struttura delle sorgenti, col tentativo di raggruppare i significati in gruppi, effettuando così una disambiguazione non su un singolo termine ma su un gruppo. In particolare le *keyword* ragrupperanno i significati secondo le gerarchie del *WordNet* e i *semanticfield* in base al contesto.

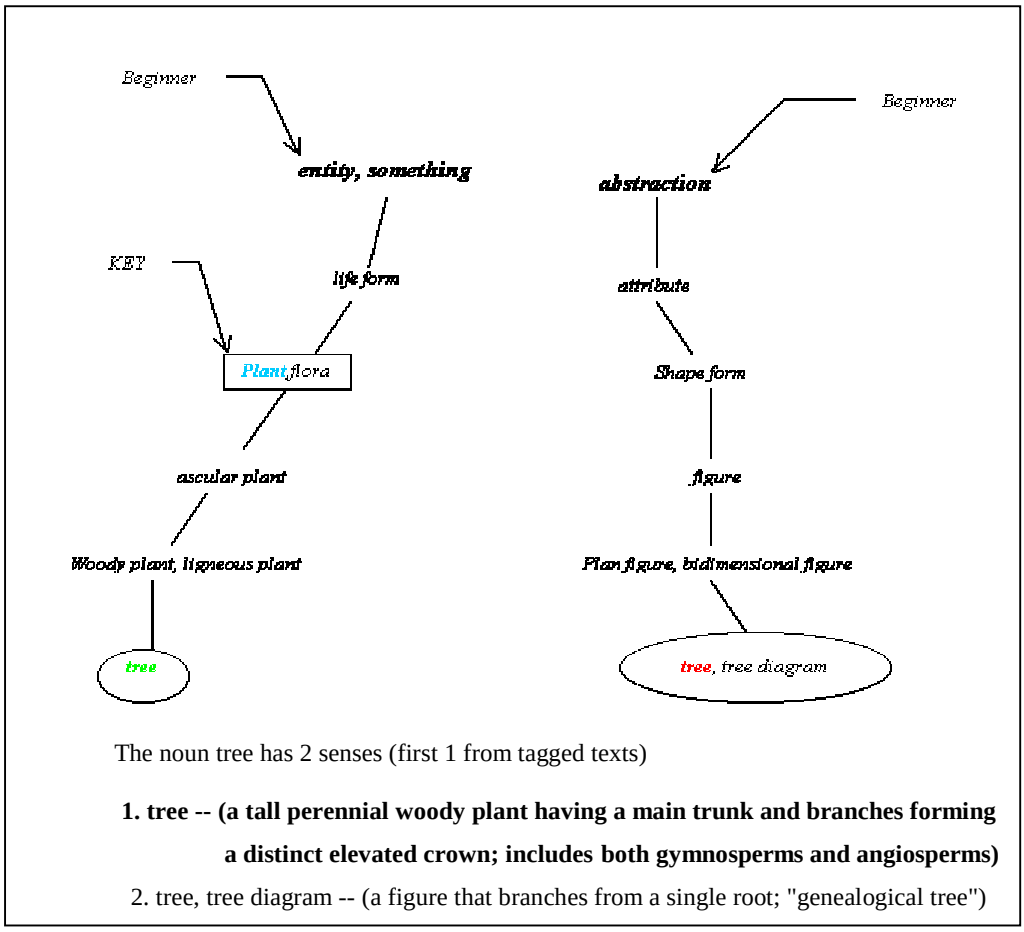
5.3.1 Acquisizione della conoscenza lessicale e diminuzione dell'ambiguità attraverso l'inserimento di *KeyWord*

Un altro strumento creato per automatizzare il processo di disambiguazione consiste nell'inserire delle *KeyWord*, vale a dire dei nomi con le seguenti proprietà:

- **Attinenza** – Tali nomi devono essere attinenti al contesto della sorgente es.: se la sorgente contiene dati relativi alla botanica (figura 5.4), una possibile chiave per descriverla potrà essere *plant* così, nel caso dovesse venir fuori una relazione di *synonymy* tra due nomi come *tree*, le relazioni riferite al *sense* 2 di *tree* (*a figure that branches from a single root*; "*genealogical tree*") saranno messe da parte.
- **Appartenenza** – Le parole chiavi devono essere contenute nel database del *WordNet* altrimenti non avrebbero influenza.
- **Assenza** – le parole chiavi devono essere prive d'ambiguità, cioè devono essere complete anche del *sense* per non creare un ulteriore livello d'ambiguità, es.: nel caso di prima *plant* ha quattro significati, ma ciò che ci interessa è quello relativo al *sense* 2: *a living organism lacking the power of locomotion*.

Le *KeyWord* agiscono nel processo di disambiguazione assegnando un peso ai termini sottostanti nella gerarchia di *WordNet.* Il peso assegnato può avere effetti su

di essi sia positivi che negativi. Una trattazione più approfondita su come i pesi vengono assegnati verrà elaborata nel paragrafo 5.5.



The noun tree has 2 senses (first 1 from tagged texts)

1. tree -- (a tall perennial woody plant having a main trunk and branches forming a distinct elevated crown; includes both gymnosperms and angiosperms)

2. tree, tree diagram -- (a figure that branches from a single root; "genealogical tree")

Figura 5.4: Struttura di WordNet dei 2 significati di *tree*.

Per facilitare il progettista nell'uso di questo strumento si è creato un pannello guida, il quale visualizza solo una parte del *database* di *WordNet*, cioè viene creato un *database* locale che contiene solo i nomi dei termini che appartengono alle nostre sorgenti, limitando di molto la scelta delle possibili *KeyWord*. Inoltre in questo modo saranno sempre verificate le proprietà di:

- **Appartenenza** al *WordNet*, poiché tutti i termini visualizzati nel pannello guida sono ricavati direttamente dal *WordNet.*
- **Assenza** di ambiguità, poiché ogni termine visualizzato è completo di significato.

Mentre queste due proprietà sono sempre rispettate, poiché imposte dalla struttura del pannello guida, lo stesso non si può dire per la proprietà di **attinenza,** infatti**,** questa dipende molto dalla bravura del progettista nello scegliere una *KeyWord* attinente allo specifico contesto. La limitazione del *database* di *WordNet*, ai soli termini delle nostre sorgenti, facilita il compito al progettista, in quanto è più semplice focalizzare i nomi, infatti, senza questa limitazione il progettista si sarebbe trovato a scegliere tra tutti i nomi di *WordNet* che sono all’in circa 57000, mentre, considerando l’esempio di riferimento, il *database* locale creato ne contiene all’incirca 200 (dipende dai nomi che si assegnano alle forme basi).

Osservazione: Un’ulteriore restrizione potrebbe essere fatta dopo l’estrazione delle relazioni lessicali; infatti, una volta eseguite tale restrizione, i nomi che rimarranno saranno solo quelli presenti nelle relazioni. Ciò comporta una riduzione dei nomi di circa l’85%, infatti, considerando l’esempio di riferimento, da circa 200 termini iniziali, ne rimangono all’in circa 30, facilitando di molto il compito del progettista.

Quest’ultima restrizione richiede però un elevato tempo di calcolo delle relazioni, ché deve essere effettuato prendendo in considerazione tutti i nomi, ma si è preferito velocizzare il calcolo delle relazioni effettuandolo solo sui nomi privi di ambiguità.

5.3.2 Semantic Field

I *Semantic field*, nella struttura di *WordNet*, sono delle *label* presenti nel campo del significato del nome e sono situate all'inizio del campo significato racchiuse da parentesi (vedi Figura 3.5).

In neretto sono evidenziati i *SemanticField*

address, computer address -- (**(computer science)** the code that identifies where a piece of information is stored)
=> code, computer code -- (**(computer science)** the symbolic arrangement of data or instructions in a computer program or the set of such instructions)
=> coding system -- (a system of signals used to represent letters or numbers in transmitting messages)
=> writing, symbolic representation -- (letters or symbols written or imprinted on a surface; "he turned the paper over so the writing wouldn't show"; "the doctor's writing was illegible")
=> written communication, written language -- (communication by means of written symbols)
=> communication -- (something that is communicated between people or groups)
=> social relation -- (a relation between living organisms; esp between people)
=> relation -- (an abstraction belonging to or characteristic of two entities or parts together)
=> abstraction -- (a general concept formed by extracting common features from specific examples)

Figura 5.5: *SemanticField* nella struttura di *WordNet*

Nel *database* di *WordNet s*ono presenti circa 200 *labels* (*Semantic Field*), ma solo 3500 *synsets* le contengono; quindi per ora solo il 3.5% dei *synsets* sono coperti da *labels.* Riporto alcuni dei 200 *SemanticField* presenti:

Computer Science, Informal, Geometry, Linguistic, Usually plural, Botany, Biology, Astronomy, Baseball, Law, Christianity, Sports, Games, Music, Pathology, Electricity, Medicine, Engineering, Genetics, Pharmacology, Astrology.

I *semanticfield* sono usati in *WordNet* per specificare l'uso di un nome e quindi rendere più facile l'individuazione del significato, infatti, se data una parola ad esempio "*sheet*" ed un'altra parola indicante il contesto "*sleeping*", molto facilmente il significato che verrà attribuito a questa parola sarà: "*bed linen*" invece di "*piece* of

paper". Questo quindi può essere un facile strumento per l'individuazione del significato e in più ci permette di raggruppare i nomi in base al contesto.

L'uso di tali *label* sarebbe utilissimo nell'effettuare un'analisi contestuale e, quindi, abbassare il livello d'ambiguità in modo intuitivo è semplice ma, per ora, i benefici che ci da questo strumento è limitato dalla scarsa presenza di *SemanticField* nel *database* di *WordNet* (solo il 3.5%).

Anche qui per facilitare il progettista nell'uso di *questo* strumento si è creato un pannello guida, il quale visualizza i soli *SemanticField* presenti nel campo *gloss* dei nostri termini e, per ognuno di essi, vengono visualizzati i termini che li contengono con il relativo significato.

5.4 Differenza tra *KeyWord* "Positive", "Negative" e "Restrittive".

L'esigenza di differenziare le *KeyWord* introducendo termini "positivi", "negativi" e "restrittivi" nasce dall'osservazione che, le sole *KeyWord* positive non sono sufficienti ad effettuare un'analisi di contesto in quanto, ci potremmo trovare in casi (vedi Figura 5.6 considerando *name* come *KeyWord* positiva) dove dei rami sottostanti a *KeyWord* "positive" contengano nomi (*title*) con significato diverso da quello inteso nel contesto della sorgente, per cui risulta necessario introdurre *KeyWord* negative o restrittive per associare ad essi un peso negativo o rimuoverli del tutto.

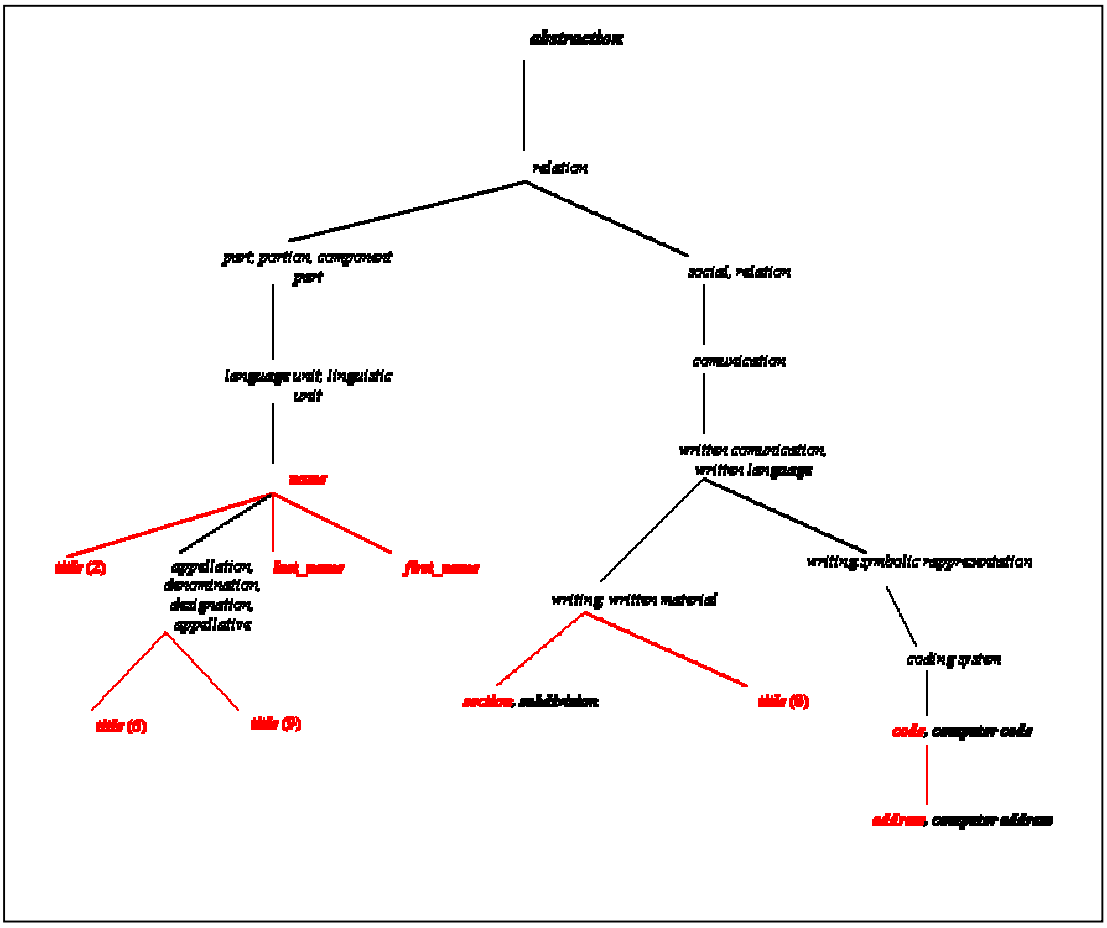


Figura 5.6 Struttura gerarchica di *abstraction*

Differenze tra *KeyWord* positive e *KeyWord* negative e restrittive:

1. Le *KeyWord* positive devono essere nomi attinenti al contesto della nostra sorgente, mentre quelle negative e restrittive no.

2. Per le *KeyWord* positive, come abbiamo visto, è possibile trovare, al di sotto di esse, nomi con significato non attinente al contesto inteso nella sorgente (rendendo inutile la *KeyWord* che introduce in questo caso un informazione errata), mentre per le *KeyWord* negative e restrittive qualsiasi termine (esempio: vedi Figura 3.7 se considerassimo *military unit* come *KeyWord* negativa o restrittiva) al di sotto di queste non sarà attinente al contesto inteso nella sorgente

(rispettando la funzione della *KeyWord*). Inoltre, in una sorgente ci sarà più di un contesto; se consideriamo una sorgente universitaria avremo diversi contesti, ad esempio relativi al personale, alla gestione economica, alla gestione dei corsi etc. e, ognuno di questi, sarà individuato da una o più *KeyWord* positiva. E' probabile che vi siano nomi con più di un significato dove ognuno può appartenere ad un diverso contesto che non necessariamente è coerente con quello specifico della sorgente. In tal caso, nonostante la presenza di *KeyWord* positive che individuano i diversi contesti, ci sarà ancora ambiguità che richiederà un ulteriore intervento del progettista.

Con *KeyWords* restrittive o negative, invece, l'intervento del progettista non è necessario poiché, anche se sono presenti per uno stesso nome più significati, tutti quelli che hanno come *broader term KeyWords* restrittive o negative, saranno direttamente eliminati, oppure avranno associato un peso negativo.

Vantaggi:

- *KeyWord* positive – Danno un'informazione più "utile" in quanto individuano i termini da accettare.
- *KeyWord* negative e restrittive – Non creano ulteriore ambiguità ai termini sottostanti, infatti, tutti i *narrower terms* di queste, possono essere associati a pesi negativi o eliminati direttamente.

Svantaggi

- *KeyWord* positive – Possono creare un'ulteriore ambiguità (caso, già esaminato, in cui una *KeyWords* individua più significati di uno stesso nome).
- *KeyWord* negative e restrittive – E' possibile eliminare o diminuire l'ambiguità solo per esclusione.

Differenze tra *KeyWord* positive e negative e *KeyWord* restrittive:

1. *KeyWord* positive e negative danno un peso alle relazioni o ai nomi sotto di esse, che può essere "positivo" o "negativo" provocando solo una modifica dei pesi associati ai nomi e, di conseguenza, l'inserimento in DUT (*Depository of Unambiguous Term*) o l'eliminazione dei nomi che supereranno i valori di soglia, mentre quelle restrittive effettuano, a partire dalla *KeyWord*, una restrizione del *database* di *WordNet* eliminando i rami sottostanti ad esse e, di conseguenza, anche i nomi appartenenti a quei rami.

Uso di *KeyWord* e *SemanticField*.

Entrambe possono essere inserite sia prima dell'estrazione di relazioni sia dopo.

- ✓ Prima dell'estrazione delle relazioni.

  Può essere utile fare un'analisi del contesto in modo tale da inserire delle *KeyWord* e dei *SemanticField.* Quest'operazione limiterà la dimensione del *database* di *WordNet* e il numero di *synset* da relazionare, diminuendo, allo stesso tempo, l'ambiguità e velocizzando il processo d'estrazione.
- ✓ Dopo l'estrazione delle relazioni.

  L'uso di *KeyWord* e di *SemanticField* permette di rendere il processo di rimozione dell'ambiguità più automatico.

Full Hyponyms list of military unit

military unit, military force, force -- (a unit that is part of some military service; "he sent Caesar a force of six thousand men")
=> command -- (a military unit or region under the control of a single officer)
=> enemy -- (an opposing military force; "the enemy attacked at dawn")
=> task force -- (a temporary military unit formed to accomplish a particular objective)
=> army unit -- (a military unit that is part of an army)
=> corps -- (an army unit usually consisting of two or more divisions)
=> Reserve Officers Training Corps, ROTC -- (a training program to prepare college students to be commissioned officers)
=> division -- (an army unit large enough to sustain combat; "two infantry divisions were held in reserve")
=> Special Forces, U. S. Army Special Forces -- (a division of the United States Army that is specially trained for guerrilla fighting)
=> battle group -- (an army unit usually consisting of five companies)
=> regiment -- (army unit smaller than a division)
=> brigade -- (army unit smaller than a division)
=> battalion -- (an army unit usually consisting of a headquarters and three or more companies)
=> company -- (small military unit; usually two or three platoons)
=> platoon -- (a military unit that is a subdivision of a company; usually has a headquarters and two or more squads; usually commanded by a lieutenant)
=> detachment -- (a small unit of troops of special composition)
=> guard, bodyguard -- (a group of men who escort and protect some important person)
=> yeomanry -- (a British volunteer cavalry force organized in 1761 for home defense later incorporated into the Territorial Army)
=> Praetorian Guard -- (the elite bodyguard of a Roman Emperor)
=> patrol -- (a detachment used for security or reconnaissance)
=> picket -- (a detachment of troops guarding an army from surprise attack)
=> press gang -- (a detachment empowered to force civilians to serve in the army or navy)
=> provost guard -- (a detachment under the command of a provost marshall)
=> rearguard -- (a detachment assigned to protect the rear of a (retreating) military body)
=> vanguard, van -- (the leading units moving at the head of an army)
=> section -- (a small army unit usually having a special function)
=> squad -- (a smallest army unit)
=> firing squad, firing party -- (a squad formed to fire volleys at a military funeral or to carry out a military execution)
=> troop -- (a group of soldiers)
=> shock troops -- (soldiers who are specially trained and armed to lead an assault)
=> troop -- (a cavalry unit corresponding to an infantry company)
=> artillery, artillery unit -- (an army unit that uses big guns)
=> battery -- (group of guns or missile launchers operated together at one place)
=> musketry -- (musketeers and their muskets collectively)
=> cavalry -- (a highly mobile army unit)
=> squadron -- (a cavalry unit consisting of two or more troops and headquarters and supporting arms)
=> horse cavalry -- (an army unit mounted on horseback)
=> mechanized cavalry -- (an armored unit of a modern army equipped with motor vehicles)
=> infantry, foot -- (an army unit consisting of soldiers who fight on foot; "there came ten thousand horsemen and as many fully-armed foot")
=> paratroops -- (infantry trained and equipped to parachute)
=> naval unit -- (a military unit that is part of a navy)
=> Marine Corps, US Marine Corps -- (an amphibious division of the US Navy)
=> division -- (a group of ships of similar type)
=> squadron -- (a naval unit that is detached from the fleet for a particular task)
=> air unit -- (a military unit that is part of the airforce)
=> division -- (a unit of the US air force usually comprising two or more wings)
=> wing -- (a unit of military aircraft)
=> air group -- (a unit of the US air force larger than a squadron and smaller than a wing)
=> squadron -- (a unit of the US air force larger than a flight and smaller than a group)
=> flight -- (a unit of the US air force smaller than a squadron)
=> legion -- (a large military unit; "the French Foreign Legion")
=> echelon -- (a body of troops arranged in a line)
=> phalanx -- (a body of troops in close array)
=> militia, reserves -- (civilians trained as soldiers but not part of the regular army)
=> territorial, territorial reserve -- (a territorial military unit)
=> National Guard, home reserve -- (US military reserves recruited by the states and equipped by the federal government; subject to call by either)
=> Territorial Army -- (British unit of nonprofessional soldiers organized for defense of GB)
=> commando -- (an amphibious military unit trained for raids into enemy territory)
=> contingent, detail -- (a temporary military unit; "the peace-keeping force includes one British contingent")
=> headquarters -- (a military unit consisting of a commander and the headquarters staff)

Figura 5.7 lista completa degli hiponomi di *military unit*

5.5 Proposte su come attribuire il peso:

Per rendere automatico il processo di disambiguazione è necessario attribuire dei pesi ai nomi e, di conseguenza, alle relazioni. La disambiguazione è un processo necessario quando esistono termini ai quali sono associati più di un significato (grado di polisemia maggiore di uno). Tale processo permette, attraverso l'attribuzione di un peso ai vari significati presenti, di individuare quello relativo al nostro contesto.

Appreso che l'ambiguità presente nei nostri nomi è data dal grado di polisemia, propongo un possibile metodo di assegnazione dei pesi.

1. Ogni nome (n_i) inizialmente ha un peso (w_i) dato dal grado di polisemia (p_i), il peso minimo per un nome sarà quindi uno, e ciò significa che il nome avrà un solo significato e sarà privo di ambiguità. L'obiettivo è quello di far tendere, in modo automatico, i pesi dei nomi al valore minimo o ad una soglia poco superiore ad esso.

2. Il peso assegnato inizialmente subisce, durante il processo di rimozione dell'ambiguità, variazioni provocate da:
   2.1 la **struttura degli schemi** delle sorgenti sfruttando:
      2.1.1 le **relazioni intra-schema,** ciò avviene quando due nomi oltre ad essere in relazione dal punto di vista strutturale lo sono anche dal punto di vista lessicale. A questi nomi attribuiamo un peso $\boldsymbol{w_i=r_j}$ ($\boldsymbol{r_j}$ è il peso associato alle relazioni è j indica la distanza dei due termini della relazione nella gerarchia di *WordNet*).
      2.1.2 Il **legame tra nome di classe e attributi**, questo avviene quando esiste un legame lessicale tra il nome di una classe e uno dei suoi attributi. A questi nomi viene attribuito un peso $\boldsymbol{w_i=r_j}$, dove $\boldsymbol{r_j}$ è lo stesso del punto precedente.
      2.1.3 Il **legame tra attributi di una stessa classe,** accade quando esiste una relazione lessicale tra gli attributi di una stessa classe, e il peso dei nomi individuati diventa $\boldsymbol{w_i= r_j}$
      2.1.4 Il **legame tra nomi di classe di una stessa sorgente**, avviene quando esiste una relazione lessicale tra due nomi di classe appartenti alla stessa sorgente, il peso assegnato ai termini è: $\boldsymbol{w_i=r_j}$

2.2. le ***KeyWord***, che hanno un peso k_j che si ricava dalla distanza della stessa dal *beginner*, e a seconda che siano:

2.2.1.**positive**, il peso di ogni nome che si trova al disotto di esse nella gerarchia di *WordNet* diventerà $w=(p_i/k_j+1)$.

2.2.1.1.Per i nomi uguali a quelli del punto 2.2.1 ma aventi un *sense* diverso, il peso diventa $w_i=(p_i*k_j+1)$

2.2.2. **negative,** il peso di ogni nome che si trova al disotto di esse nella gerarchia di *WordNet* diventerà $w_i=(p_i*k_j+1)$.

2.2.3. **restrittive**, i nomi che si trovano al di sotto di esse nella gerarchia di *WordNet* verranno eliminati.

2.3. I ***SemanticField***, ai quali associamo un peso **s**, varieranno a seconda che siano:

2.3.1. **positivi**, il peso dei nomi che li contengono che diventerà $w_i=(p_i/s+1)$, mentre il peso dei nomi con significato diverso da quello individuato dal *semanticfield* diventa $w_i=(p_i*s+1)$

2.3.2. **negativi**, il peso dei nomi che li contengono che diventerà $w_i=(p_i*s+1)$.

2.3.3. **restrittivi**, i nomi verranno eliminati.

3. Ogni volta che un nome con relativo *sense* viene eliminato - o perché supera il valore di soglia massimo, o perché si trova al di sotto di una *KeyWord* restrittiva, o perché contiene un *SemanticField* restrittivo - il grado di polisemia dei nomi uguali ma con *sense* diverso diventerà: $p_i=p_i-1$, e il peso verrà ricalcolato considerando il nuovo valore di p_i.

Verranno qui di seguito proposti, a titolo esemplificativo dei possibili valori di soglia che dovranno essere affinati attraverso ulteriori sperimentazioni. Invece, per per quanto già detto sopra, un ritocco potrebbe essere fatto su k_i che indica l'importanza delle *KeyWords*.

I valori di soglia per i termini sono:

Min –> 1,6 Il peso minimo che i nomi possono assumere è uno (sono quelli aventi grado di polisemia 1); invece quelli ricavati attraverso il passo 2 avranno per costruzione un valore maggiore di uno, di

conseguenza, la soglia dovrebbe avere un valore compreso tra 1 e 2 (quest'ultimo valore sta ad indicare che un nome ha due significati).

- Max -> 19 Il grado massimo di polisemia che, sulla base delle ricerche eseguite, è stato riscontrato all'in circa 13. Si è però preferito supporre che esista un margine più ampio.

Valori di soglia proposti	
Min.	**Max.**
1.6	19

Tabella 5.1: Tabella riassuntive dei valori di soglia

Tabella riassuntiva per il calcolo dei pesi:

Eventi che influenzano il peso		**peso**	**Significato**
Condizione iniziale		$w_i=p_i$	Ad ogni nome è associato inizialmente un peso dato dal relativo grado di polisemia
Struttura degli schem	Relazioni intraschema	$w_i = r_j$	Ad ogni nome appartenente ad una relazione strutturale che sia anche lessicale gli associamo un peso r_j
	Legame tra classi e attributi	$w_i = r_j$	Ad ogni attributo e classe tra cui esiste una relazione lessicale viene associato un peso r_j
	Legame tra attributi di una stessa classe	$w_i = r_j$	Ad ogni coppia di attributi in relazione dal punto di vista strutturale, gli associamo un peso r_j
	Legame tra classi di una stessa sorgente	$w_i = r_j$	Il peso di ogni coppia di classi in relazione dal punto di vista lessicale divinata uguale ad $w_i = r_j$
KeyWord	**Positiva**	$\boldsymbol{w_i=(p_i/k_j+1)}$	**Il peso dei nomi che si trovano al disotto di una KeyWord positiva, nella gerarchia di WordNet, diventerà $w_i=(p_i/k_j+1)$, mentre gli altri con significato diverso $w_i=(p_i*k_j+1)$.**
	Negativa	$\boldsymbol{w_i=(p_i*k_j+1)}$	**Il peso dei nomi che si trovano al disotto di KeyWord negativa, nella gerarchia di WordNet, diventerà $w_i=(p_i*k_j+1)$.**
	Restrittiva	-	**I nomi che si trovano al di sotto di KeyWord restrittive nella gerarchia di WordNet verranno eliminati.**
SemanticField	**Positiva**	$\boldsymbol{w_i=(p_i/s+1)}$	**Il peso dei nomi che contengono i SemanticField positivi diventerà $w_i=(p_i/s+1)$ mentre gli altri con significato diverso $w_i=(p_i*s+1)$.**
	Negativa	$\boldsymbol{w_i=(p_i*s+1)}$	**Il peso dei nomi che contengono i SemanticField negativi diventerà $w_i=(p_i/s+1)$**
	Restrittiva	-	I nomi che conterranno i SemanticField restrittivi verranno eliminati.
Ogni volta che un nome con relativo sense viene eliminato, - o perché supera il valore di soglia massimo, o perché si trova al di sotto di una KeyWord restrittiva, o perché contiene un SemanticField restrittivo, - il grado di polisemia dei nomi uguali ma con sense diverso diventerà: $p_i=p_i-1$, e il nuovo peso di essi verrà ricalcolato considerando il nuovo valore di p_i.			

Legenda:	
n_i	nome i_esimo
w_i	peso i_esimo del nome n_i
p_i	grado di polisemia del nome n_i
r_j	peso da associare a nomi che sono in relazione sia dal punto di vista lessicale che strutturale (dipende dalla distanza dei determini nella gerarchia di *WordNet*).
k_j	peso associato alla KeyWord j_esima (si ricava dalla distanza della stessa dal beginner)
s	peso associato ai SemanticField

5.5.1 Calcolo del peso

Al significato di ogni termine è associato un grado di polisemia *p*, un numeratore *num*, e un denominatore *den*. Inizialmente il grado di polisemia per ogni nome coincide con quello assegnatogli dal *WordNet*, mentre numeratore e denominatore sono posti uguale ad 1.

Definiamo *K* il contributo che $\boldsymbol{r_j}$, $\boldsymbol{k_j}$, $\boldsymbol{s}$ danno al peso (a $\boldsymbol{k_j}$ che indica la distanza della *keyword* dal *beginner* si somma 0.5, per fare in modo che anche il *beginner* abbia un peso qualora venisse considerato come *keyword*). Ogni volta che viene modificato il peso di un nome (completo di significato), vengono seguiti i seguenti passi:

1 A secondo di come si vuole influenzare il peso

1.1 in modo positivo viene posto: $K = r_j = k_j = s$,
a seconda se viene influenzato da una *keyword*, da un *semanticfield* o da una relazione ricavata dalla nostra struttura.

1.2 in modo negativo viene posto: $K = \frac{1}{rj} = \frac{1}{kj} = \frac{1}{s}$

2 il denominatore dei nomi individuati diventa: $den = den * K$

3 il peso dei nomi individuati diventa: $w = \frac{p * num}{den} + 1$

4 Se i pesi vengono influenzati in modo positivo, il numeratore, dei significati diversi da quelli individuati, viene posto: $num = num * k$ e si calcola il peso come al punto 3.

5 Vengono controllati i pesi, se sono all'interno dei valori di soglia (min 1.6, max 19).

5.1 I nomi che oltrepassano la soglia minima vengono considerati non ambigui, e posti in un deposito di termini disambiguati (il processo di disambiguazione non avrà più influenza per questi nomi). Tutti gli altri significati del nome disambiguato vengono eliminati.

5.2 Se il peso oltrepassa la soglia massima viene eliminato. Per i significati restanti è decrementato di uno il grado di polisemia e ricalcolato il peso.

Capitolo 6

ARM: il *tool* che automatizza il processo di rimozione dell'ambiguità

L'obiettivo del modulo ARM (***A**mbiuty **R**emoving **M**odule*) è quello di aiutare il progettista nella fase di rimozione dell'ambiguità. Nel processo di integrazione questo si colloca tra la fase di acquisizione e convalida delle relazioni intra-schema, che effettuata dal modulo SIM (*Source Integrator Module***)** con l'ausilio di ODB-Tools, e la fase di estrazione delle relazioni lessicali compiuta dal modulo SLIM (*Schemata Lessical Integrator Module*) [39] con l'ausilio di *WordNet.*

6.1 ARM all'interno di MOMIS

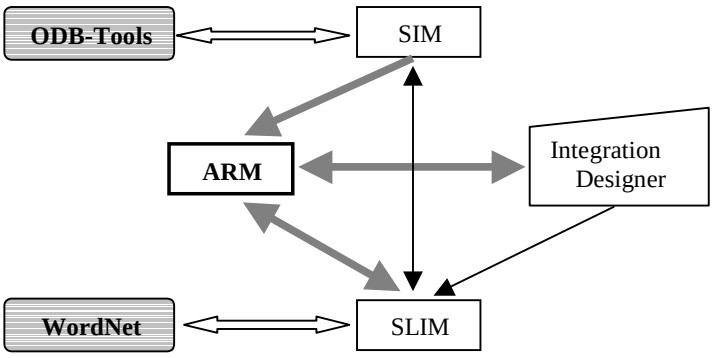


Figura 6.1: Interazioni di ARM in MOMIS

La figura 6.1 mostra le interazioni di ARM nel sistema MOMIS: questa figura serve solo ad individuare, a livello logico, i moduli con cui ARM interagisce, poiché nella realtà della struttura MOMIS, ARM è inserito in un componente SI-Designer che mette a disposizione un oggetto *proxy* con il quale il modulo ARM e tutti gli altri comunicano. Il *proxy* è utilizzato anche dal SI-Designer per tener traccia delle operazioni di integrazione.

Quindi è attraverso il metodo *getGlobalSchema()*, che ARM acquisisce un oggetto di tipo *Schema* con il quale ottiene sia le informazioni relative alle sorgenti, nomi di classi e attributi (*getGlobalSchema().getSource()*), che le relazioni intra-schema presenti nel *thesaurus* (*getGlobalSchema().getThesRelation()*).

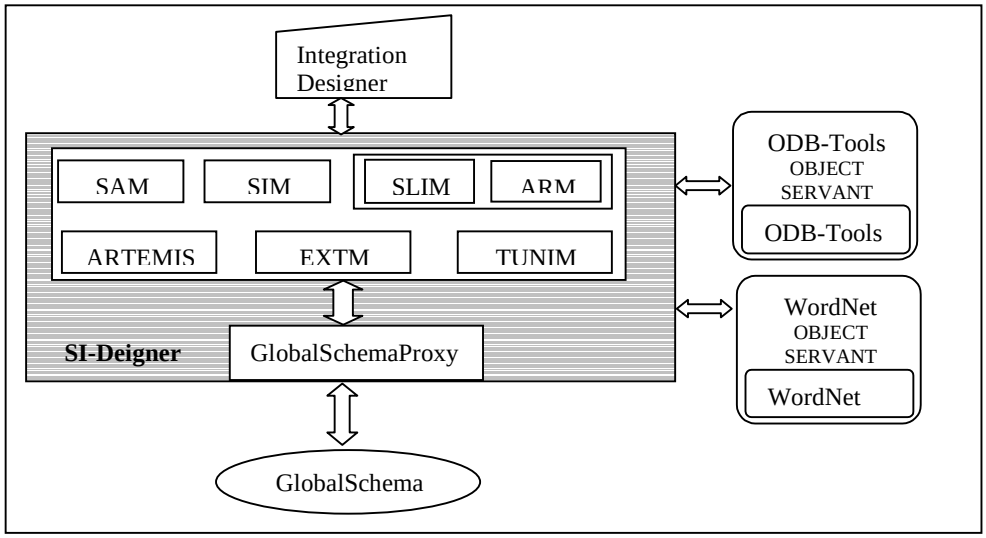


Figura 6.2: Architettura del SI-Designer

Osservando la figura 6.2, si nota che all'interno dell'architettura del SI-Designer, i due, moduli ARM e SLIM, sono integrati in un unico modulo, che serve sia per l'assegnazione e quindi la disambiguazione dei significati sia per l'estrazione delle relazioni lessicali.

6.2 ARM e SLIM due moduli per assegnare i significati

In fase di progettazione i moduli ARM e SLIM sono stati concepiti come moduli indipendenti, vale a dire con strutture dati completamente diverse, e come unica via di scambio dei dati il *GlobalSchemaProxy*. Visto però la similarità dei dati con cui lavoravano e le loro frequenti interazioni, si è preferito unirli creando un unico modulo caratterizzato da un'unica struttura dati.

Il modulo ARM rappresenta così un'evoluzione del modulo SLIM, basato sull'aggiunta di nuove funzioni e strumenti che aiutano il progettista nella fase di rimozione dell'ambiguità.

6.2.1 Come viene risolta l'ambiguità da SLIM

Prima dell'integrazione di ARM, SLIM, era solo formato da due parti, di cui una caratterizzata dalla presenza di un *JTree* che memorizzava su ogni foglia tutte le informazioni relative ai termini (ambiguo, ignorato, significato assegnato), e l'altra da una tabella che mostrava le relazioni lessicali estratte (vedi Figura 6.3).

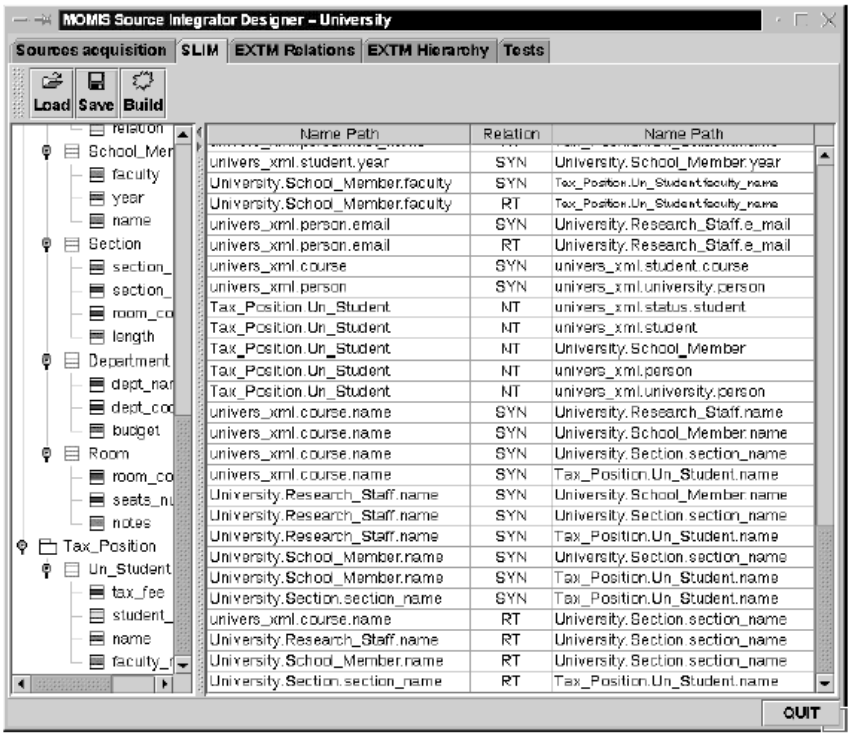


Figura 6.3: SLIM prima dell'integrazione di ARM.

L'unico modo con cui il progettista interagiva per assegnare i significati era di cliccare su una foglia, e, attraverso un menu contestuale (vedi figura 6.4), poteva scegliere se ignorare il termine, assegnarli un significato o inserire una formabase.

Quest'operazione doveva essere effettuata per ogni termine rendendo così il processo di disambiguazione molto dispendioso in termini di tempo. Questo è ancora più rilevante se si pensa che prima di assegnare un significato ad un termine l'utente doveva leggere tutti i significati (vedi figura 6.5), per poi scegliere quello esatto rispetto al contesto in esame.

Considerando il nostro esempio di riferimento, composto di circa 50 termini (nomi di classi e attributi), e considerando inoltre che in media ogni termine ha

Figura 6.3: SLIM prima dell'integrazione di ARM.

quattro significati, il progettista sarebbe stato costretto a leggere all'incirca 200 significati prima di eliminarne l'ambiguità, (è facile immaginare un caso reale in cui questo numero risulterebbe di gran lunga più elevato!).

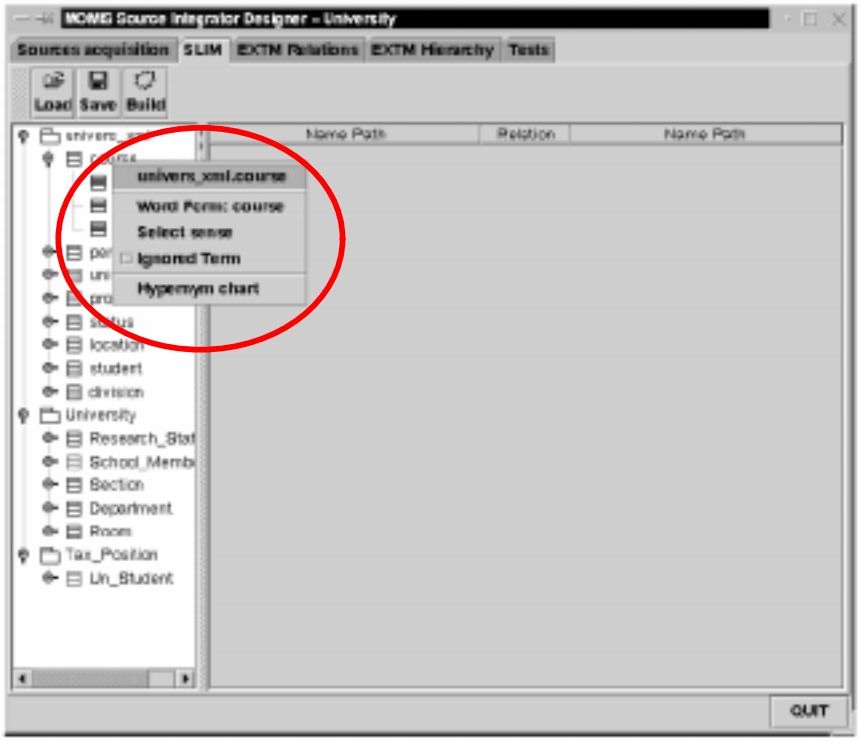


Figura 6.4: Menù contestuale

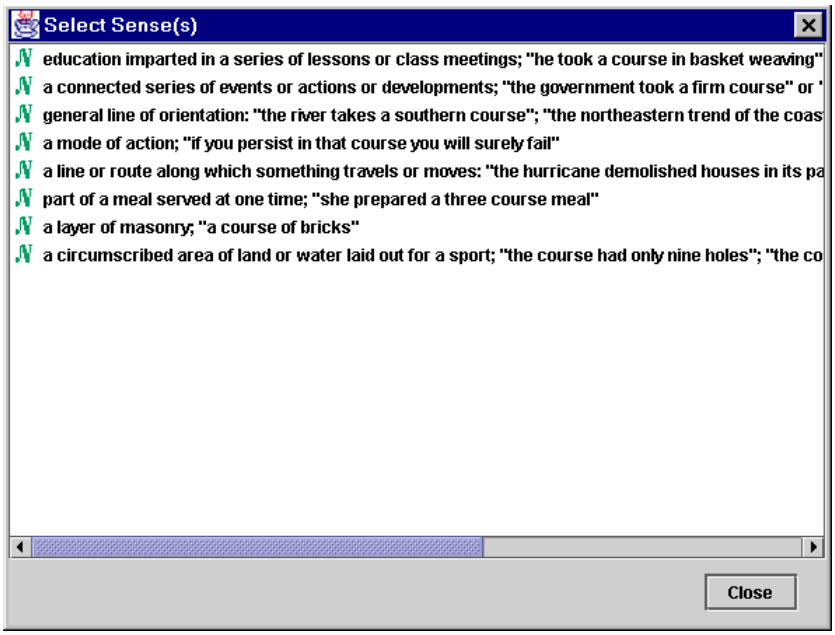


Figura 6.5: Finestra di dialogo relativa a *corse* per la scelta del significato

Figura 6.3: SLIM prima dell'integrazione di ARM.

6.2.2 ARM l'evoluzione di SLIM

Con l'inserimento di ARM tutte queste funzionalità sono state conservate. La struttura dati risultante dalla fusione dei due moduli è rimasta quella del *JTree*, che memorizza su ogni foglia i termini presenti nelle sorgenti, in più si è aggiunto ad ogni foglia, un vettore che memorizza le informazioni (peso, polisemia, *num*, *den*, etc.) sul significato del termine. Quest'aggiunta è stata necessaria in quanto SLIM manteneva solo lo stato delle foglie (termini), ma per effettuare algoritmi di *ranking* è stato necessario aggiungere informazioni relative ad ognuno dei significati, e in più la storia delle operazioni effettuate in modo da poter permettere all'utente di annullare, in caso di errore, le operazioni fatte.

Dal punto di vista estetico il modulo finale risultante è molto simile a SLIM, l'unico cambiamento è nella parte destra dove invece di un semplice *JPanel*, usato da SLIM per visualizzare le relazioni lessicali estratte, c'è un *JTabbedPane* dove ogni linguetta rappresenta uno strumento utile nella fase di rimozione dell'ambiguità (vedi figura 6.6).

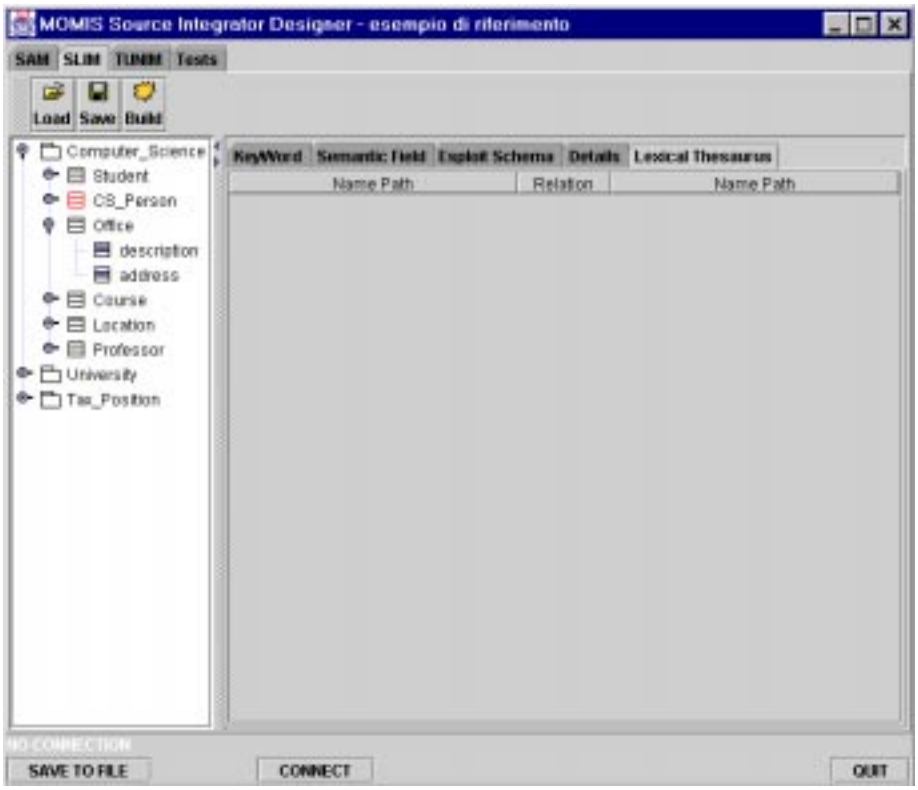


Figura 6.6 ARM e SLIM integrati

Figura 6.3: SLIM prima dell'integrazione di ARM.

6.3 Esempio di funzionamento di ogni singolo pannello

I nuovi strumenti che ARM mette a disposizione sono quelli descritti nelle sezioni 5.2, 5.3 e 54, vale a dire *KeyWord*, *SemanticField* e altri strumenti ancora che danno al progettista la possibilità di sfruttare delle informazioni ricavabili dalla struttura degli schemi sorgenti, in modo da ricavarne delle informazioni lessicali da sfruttare poi per la rimozione dell'ambiguità. L'altra novità è l'introduzione di un panello "*Details*" che visualizza lo stato di ognuno dei significati.
Nelle sezioni che seguono ne analizzeremo le caratteristiche.

6.3.1 KeyWord

Ciccando sull'etichetta "*KeyWord*" viene visualizzato un pannello guida che permette l'inserimento delle *keyword* (vedi figura 6.7).

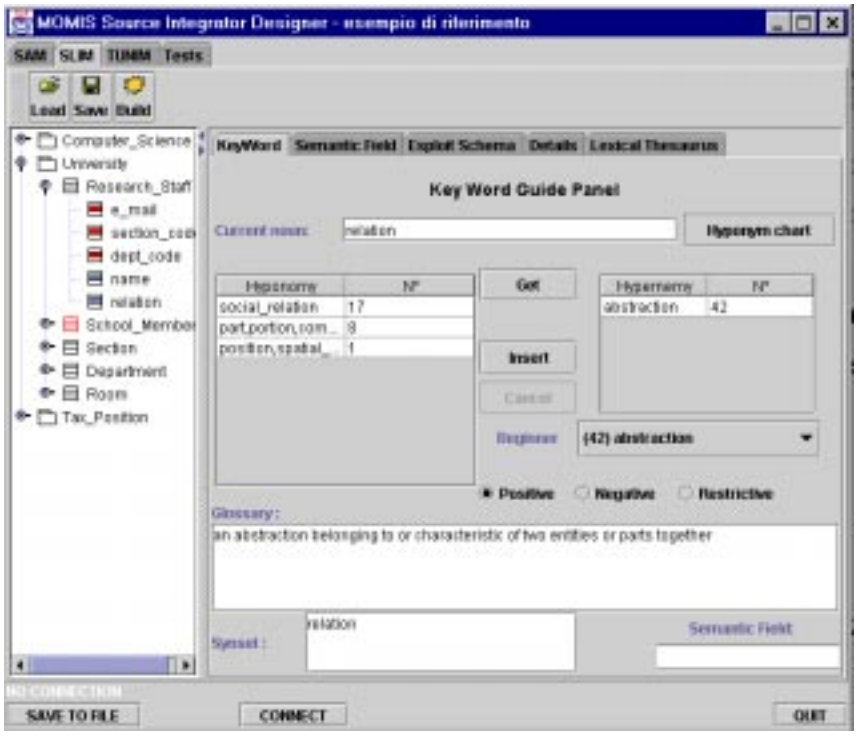


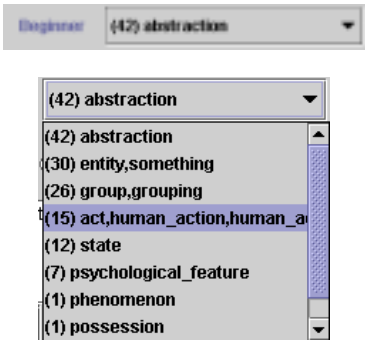
Figura 6.7 Pannello guida per l'inserimento di *KeyWord*

Figura 6.3: SLIM prima dell'integrazione di ARM.

Questo pannello rappresenta il *WordNet* locale, infatti, esso è generato da una *HashTable* che viene creata nel momento in cui vengono acquisiti i termini dagli schemi, e, attraverso il *WordNet* vengono prelevate le informazioni lessicali relative ai termini, e successivamente memorizzate su di essa.

Una volta selezionato un nome, il pannello guida visualizza i suoi *hypernemy*, *hyponomy*, l'insieme dei suoi sinonimi (*synset*) il suo significato e in basso a destra, se è presente, il *semanicfield* relativo al nome.

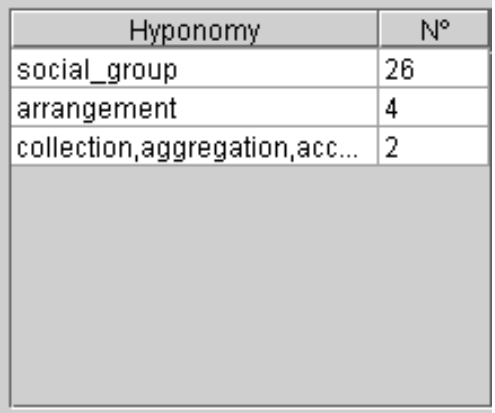
Il primo passo da fare è quello di scegliere un *beginner*, infatti, come visto nelle sezioni 3.2 e 3.2.3.1 il *WordNet* divide tutto l'insieme dei nomi in 9 gerarchie aventi ognuna un unico *beginner*.



Ciccando su di esso apparirà una tendina che visualizza tutti i beginner presenti nel nostro *WordNet* locale, preceduti ognuno da un numero che ne indica l'occorrenza, vale a dire il numero di termini che hanno a loro volta esso come beginner.

Figura 6.3: SLIM prima dell'integrazione di ARM.

Una volta selezionato il *beginner* questo andrà nella casella identificata da *current noun* facendo sì che gli altri elementi del pannello vengano aggiornati con i dati del nome relativo.



Nella tabella *Hyponym* c'è la lista di tutti i diretti *hyponyms*, vale a dire tutti i nomi subordinati ad esso in base alla gerarchia di *WordNet*. Anche qui per ogni nome e mantenuta l'informazione relativa all'occorrenza: il nome *social_goup* avrà per esempio, come subordinati 26 nomi appartenenti alle nostre sorgenti, *arrangment* 4, e cosi via.

Selezionando uno di questi nomi avremo la possibilità di scendere ad un livello inferiore nella gerarchia di *WordNet* andando così a trovare termini sempre più specifici. Una volta trovato un termine, che si pensa poter essere una *keyword* gli si associa dapprima il significato. positivo, negativo o restrittivo e solo successivamente premendo il pulsante "*Insert*" vengono visualizzati (vedi figura 6.8) all'utente i cambiamenti che questa *KeyWord* provocherà, con possibilità di accettare o meno i cambiamenti.

Questo strumento dà la possibilità di rimuovere l'ambiguità su più di un nome contemporaneamente, però sta all'utente avere l'intuito di trovare la chiave giusta. In più, il pannello guida fornisce un'ulteriore strumento, vale a dire la possibilità di visualizzare graficamente tutti gli *Hyponyms* della *keyword* corrente in modo tale da prevedere i cambiamenti che provocherà dando anche la possibilità di individuare in modo più semplice ed efficiente ulteriori *keyword*.

Proviamo ora a fare un esempio, consideriamo di trovarci nel caso in cui abbiamo scelto *abstraction* come *beginner*: e scegliamo "*part, portion, component_part, component*" come *keyword*, e cliccando su "*hyponym chart*" verrà presentato all'utente tutta la gerarchia dei termini subordinati (vedi figura 6.8).

Figura 6.3: SLIM prima dell'integrazione di ARM.

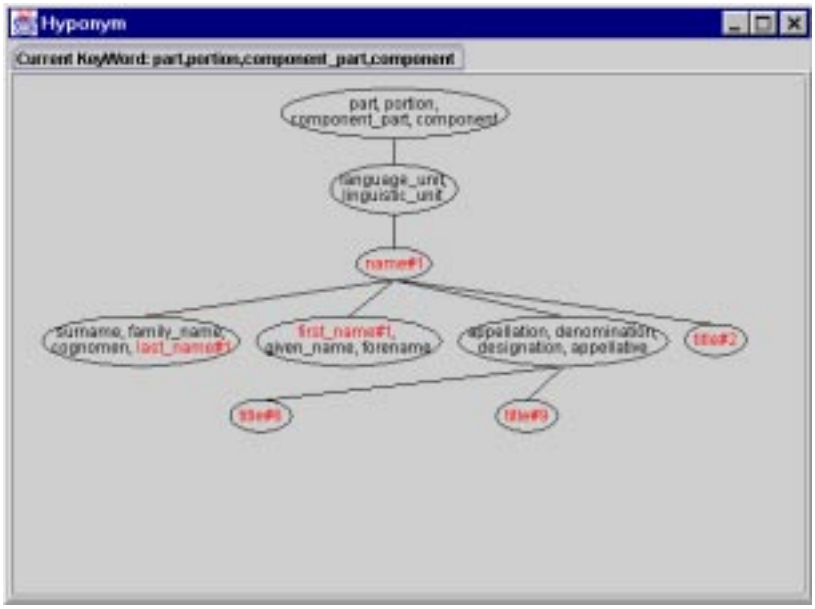


Figura 6.8: Grafo degli *hyponym* di *part portion, ...*

Nel grafo verranno visualizzati in rosso i termini ambigui, in verde quelli non ambigui e in giallo quelli eliminati; per ogni nome presente nelle nostre sorgenti è riportato anche il numero relativo al significato assegnatoli da *WordNet*. Inoltre cliccando su uno degli ovali, rappresentanti i termini, verrà visualizzata una finestra che ne dà il significato.

Osservando la figura 6.8, possiamo pensare di utilizzare come *keyword* positiva *name* (con significato: "*a language unit by which a person or thing is known*"). Dopo aver inserito la *keyword* cliccando sul bottone "*insert"* viene visualizzata una finestra che mostra i cambiamenti che la *keyword* scelta provoca (vedi figura 6.9).

Figura 6.3: SLIM prima dell'integrazione di ARM.

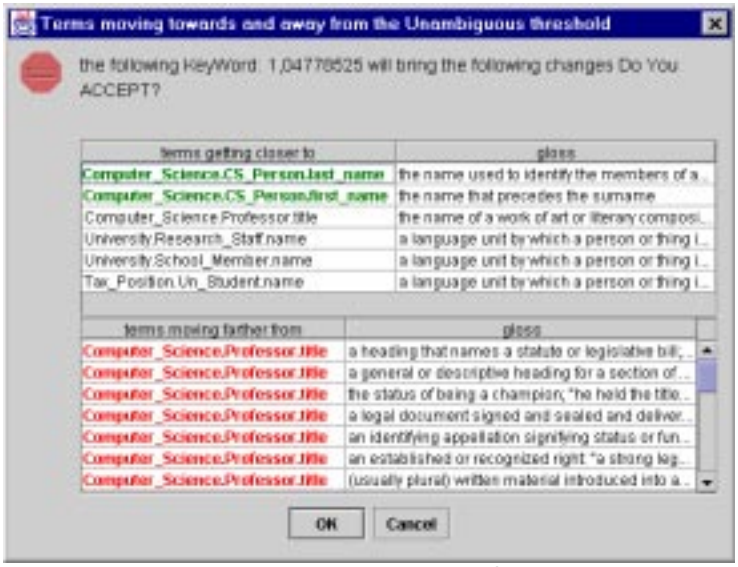


Figura 6.9: Finestra di conferma.

Nella figura 6.9 sono riportate due tabelle, quella in alto mostra i termini che dopo l'inserimento della *keyword* si avvicineranno alla soglia di disambiguazione, ed in particolare quelli in verde la oltrepasseranno, diventando così non ambigui. La tabella in basso mostra i termini che per effetto della *keyword* si allontaneranno dalla soglia di disambiguazione; quelli in rosso verranno eliminati poiché si discosteranno oltre la soglia massima.

Si nota però dalla figura in questione che nel terzo rigo della tabella in alto viene preso il termine *title* con un significato non relativo al nostro contesto. A questo punto conviene annullare l'inserimento della *keyword* ed eliminare di conseguenza il significato di *title* (il tutto attraverso l'inserimento di una *keyword* restrittiva o usando SLIM), una volta eliminato il significato non attinente ("*the name of a work of art or literary composition ecc.*") si può reinserire la *keyword name.* Come risultato finale avremo disambiguato 6 nomi:

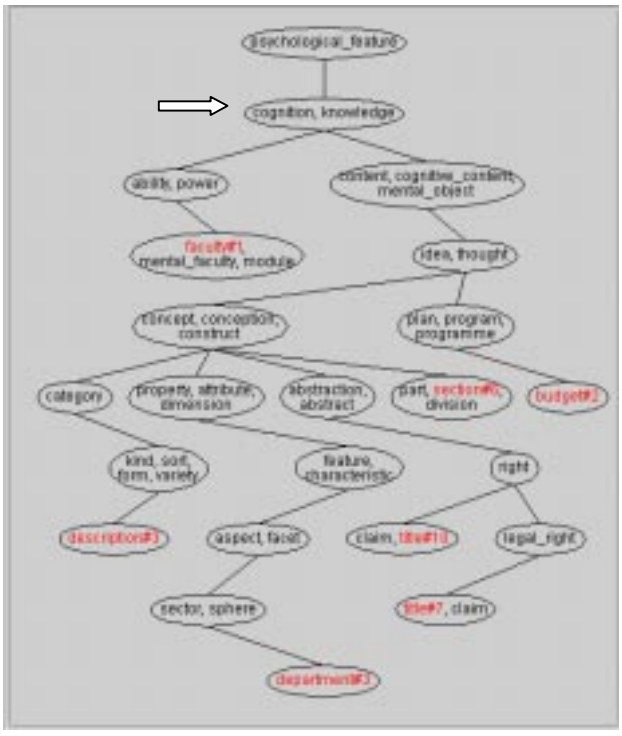
- `Computer_Science.CS_Person.last_name` – "*the name used to identify the members of a family"*
- `Computer_Science.CS_Person.first_name` – "*the name that precedes the surname"*

Figura 6.3: SLIM prima dell'integrazione di ARM.

- `University.Research_Staff.name` - *"a language unit by which a person or thing is known"*
- `University.Schol_Member.name` - *"a language unit by which a person or thing is known"*
- `Tax_Position.Un_Student.name` - *"a language unit by which a person or thing is known"*
- `Computer_Science.Professor.title` - *"an identifying appellation signifying status or function; e.g. Mr. Or General;"*

Abbiamo visto che con solo due *keyword* siamo riusciti a disambiguare 6 nomi. Un ulteriore miglioramento da tenere in considerazione per sviluppi futuri, consiste nell'inserire di inserire per ogni termine - nelle tabelle della finestra di conferma - dei *check box* permettendo cosi di selezionare direttamente i termini che la *keyword* non dovrà considerare.

Vediamo ora un esempio usando una *keyword* negativa. Prendiamo "*cognition, knowledge*" come *keyword*, che ha come *beginner* "*psychological_feature*".



Osservando i suoi *hyponomy* si nota che non sono attinenti al nostro contesto, infatti, *faculty* è inteso come facoltà mentale e, budget, *section*, *desription*, *title*, department derivano turri da idea e pensiero.

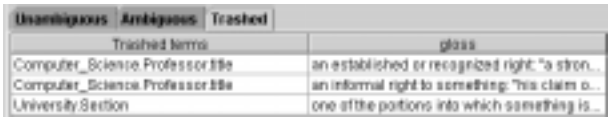
Figura 6.3: SLIM prima dell'integrazione di ARM.

University.School_Member.faculty (=) 2.0	one of the inherent cognitive or perc...
University.School_Member.faculty (=) 2.0	the body of teachers and administrat...
University.Department.budget (=) 2.0	a sum of money allocated for a parti...
University.Department.budget (=) 2.0	a summary of intended expenditures
Tax_Position.Un_Student.faculty_name (=) 2.0	one of the inherent cognitive or perc...
Tax_Position.Un_Student.faculty_name (=) 2.0	the body of teachers and administrat...
Computer_Science.Office.description (=) 3.0	a statement that represents somethi...
Computer_Science.Office.description (=) 3.0	a act of describing something
Computer_Science.Office.description (=) 3.0	sort or variety; "every description of b...
University.Department (=) 3.0	a specialized division of a large orga...
University.Department (=) 3.0	the territorial and administrative divis...
University.Department (=) 3.0	a specialized sphere of knowledge; "...
Computer_Science.Professor.title (=) 10.0	a heading that names a statute or le...
Computer_Science.Professor.title (=) 10.0	the name of a work of art or literary c...
Computer_Science.Professor.title (=) 10.0	a general or descriptive heading for ...
Computer_Science.Professor.title (=) 10.0	the status of being a champion; "he ...
Computer_Science.Professor.title (=) 10.0	a legal document signed and seale...
Computer_Science.Professor.title (=) 10.0	an identifying appellation signifying ...
Computer_Science.Professor.title (=) 10.0	an established or recognized right; "...
Computer_Science.Professor.title (=) 10.0	(usually plural) written material intro...
Computer_Science.Professor.title (=) 10.0	an appellation signifying nobility; "yo...
Computer_Science.Professor.title (=) 10.0	an informal right to something; "his c...
University.Section (=) 13.0	a self-contained part of a larger com...
University.Section (=) 13.0	a very thin slice (of tissue or mineral ...
University.Section (=) 13.0	a distinct region or subdivision of a t...
University.Section (=) 13.0	one of several pieces or parts that fit ...
University.Section (=) 13.0	a small team of policemen working ...
University.Section (=) 13.0	one of the portions into which somet...
University.Section (=) 13.0	a land unit of 1 square mile measuri...
University.Section (=) 13.0	(geometry) the area created by a pla...
University.Section (=) 13.0	a division of an orchestra containing ...
University.Section (=) 13.0	a small army unit usually having a s...
University.Section (=) 13.0	a specialized division of a large orga...
University.Section (=) 13.0	a segment of a citrus fruit; "he ate a ...
University.Section (=) 13.0	the cutting of or into body tissues or ...

Prima di inserire la keyword vediamo, nella tabella a sinistra, i pesi che questi termini hanno.

Al momento i pesi coincidono con il loro grado di polisemia poiché nessuna operazione è stata fatta

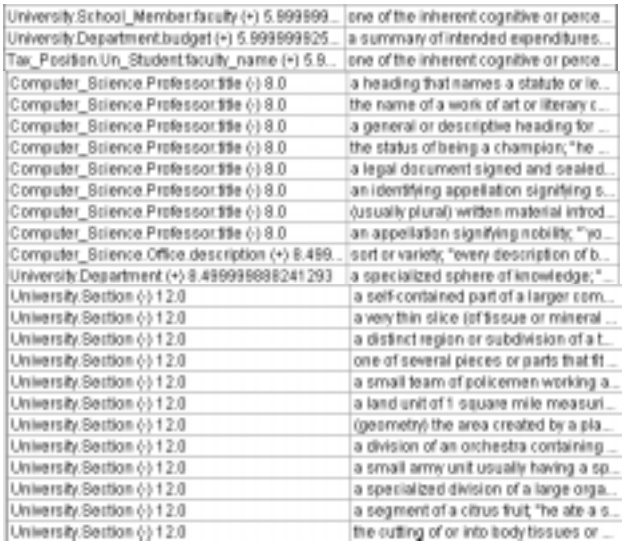
Vediamo ora i cambiamenti provocati dalla *keyword*: negativa "*cognition, knowledge*"



Unambiguous | Ambiguous | Trashed

Trashed terms	gloss
Computer_Science.Professor.title	an established or recognized right; "a stron...
Computer_Science.Professor.title	an informal right to something; "his claim o...
University.Section	one of the portions into which something is...

I termini eliminati sono 3.



University.School_Member.faculty (+) 5.999999...	one of the inherent cognitive or perce...
University.Department.budget (+) 5.999999925...	a summary of intended expenditures...
Tax_Position.Un_Student.faculty_name (+) 5.9...	one of the inherent cognitive or perce...
Computer_Science.Professor.title (-) 8.0	a heading that names a statute or le...
Computer_Science.Professor.title (-) 8.0	the name of a work of art or literary c...
Computer_Science.Professor.title (-) 8.0	a general or descriptive heading for ...
Computer_Science.Professor.title (-) 8.0	the status of being a champion; "he ...
Computer_Science.Professor.title (-) 8.0	a legal document signed and sealed...
Computer_Science.Professor.title (-) 8.0	an identifying appellation signifying s...
Computer_Science.Professor.title (-) 8.0	(usually plural) written material introd...
Computer_Science.Professor.title (-) 8.0	an appellation signifying nobility; "yo...
Computer_Science.Office.description (+) 8.499...	sort or variety; "every description of b...
University.Department (+) 8.499998888241293	a specialized sphere of knowledge; "...
University.Section (-) 12.0	a self-contained part of a larger com...
University.Section (-) 12.0	a very thin slice (of tissue or mineral ...
University.Section (-) 12.0	a distinct region or subdivision of a t...
University.Section (-) 12.0	one of several pieces or parts that fit ...
University.Section (-) 12.0	a small team of policemen working a...
University.Section (-) 12.0	a land unit of 1 square mile measuri...
University.Section (-) 12.0	(geometry) the area created by a pla...
University.Section (-) 12.0	a division of an orchestra containing ...
University.Section (-) 12.0	a small army unit usually having a sp...
University.Section (-) 12.0	a specialized division of a large orga...
University.Section (-) 12.0	a segment of a citrus fruit; "he ate a s...
University.Section (-) 12.0	the cutting of or into body tissues or ...

Nella tabella di fianco riporto i termini che hanno subito una variazione del peso.

Figura 6.3: SLIM prima dell'integrazione di ARM.

Vediamo come la *keyword* negativa ha influito sui pesi. La sua distanza (k_j) dal *beginner* è 2, quindi il suo contributo (*K*) è:

$$K = \frac{1}{kj + 0.5} = \frac{1}{2.5} = 0.4 \qquad \text{(per le } \textit{keyword} \text{ positive } K=(k_j + 0.5))$$

Inizialmente tutti i termini hanno il peso (*w*) coincidente con il grado di polisemia (*p*) e, quindi, numeratore (*num*) e denominatore (*den*) uguali ad uno.

Il peso dei significati individuati dalla *keyword* si calcola nel seguente modo:

*den=den*K* (per le *keyword* positive invece *num=num*K*)

$$w = \frac{p * num}{den} + 1$$

di conseguenza il peso dei significati *faculty, budget* diventerà:

$w = \frac{2*1}{0.4} + 1 = 6$, con *den*=1*0.4=0.4

quello di *department, description*: $w = \frac{3}{0.4} + 1 = 8.5$, con *den*=0.4

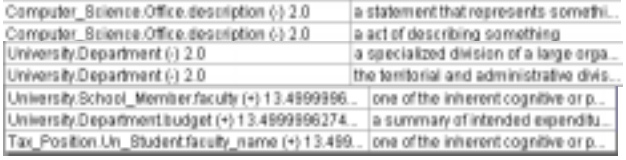
quello di *title*: $w = \frac{10}{0.4} + 1 = 26$, con *den*=0.4

mentre il peso di *section* diventerà: $w = \frac{13}{0.4} + 1 = 33.5$, con *den*=0.4

Si nota che *title* e *section* oltrepassano la soglia superiore che è 19, quindi come visto anche sopra verranno eliminati. L'eliminazione di questi 3 significati dei nomi *title* e *section,* provocherà negli altri un decremento del grado di polisemia, quindi, per i significati rimanenti di *section* la polisemia da 13 passerà a 12, mentre per gli altri di *title* la polisemia passera da 10 a 8, poiché i significati di *title* eliminati sono due.

Reinserendo nuovamente la stessa *keyword* negativa si ottiene:

Avremo 2 termini che verranno eliminati.



Mentre gli altri varieranno secondo la tabella riportata di fianco.

Figura 6.3: SLIM prima dell'integrazione di ARM.

Le variazioni vengono calcolate nel seguente modo:

Per *faculty* e *budget* abbiamo: $den=den*K=0.4*0.4=0.16$, $w = \frac{2}{0.16} + 1 = 13.5$

Mentre *description,* e *department*: $den=0.16$, $w = \frac{3}{0.16} + 1 = 19.75$, che porta i significati oltre la soglia massima, questo provocherà l'eliminazione degli stessi, e la diminuzione di 1 del grado di polisemia degli altri significati, che diventerà a 2.

Inserendo nuovamente la stessa *keyword,* elimineremo i 3 significati rimanenti, infatti: $den= den*K =0.16*0.4 =0.064$, $w = \frac{p*num}{den} + 1 = \frac{2}{0.064} + 1 = 32.25$, oltrepassa la soglia provocando l'eliminazione del significato, e la diminuzione di 1 del grado di polisemia dei restanti significati, che quindi, da 2 passa a 1 oltrepassando così la soglia minima (1.6), e diventando così non ambigui.

University.School_Member.faculty	the body of teachers and administrators at ...
University.Department.budget	a sum of money allocated for a particular p...
Tax_Position.Un_Student.faculty_name	the body of teachers and administrators at ...

Questo riportato sopra è solo un esempio per vedere come variano, e come vengono calcolati i pesi in funzione delle *keyword*. Infatti, lo stesso risultato si sarebbe ottenuto inserendo la stessa *keyword* una volta sola ma con significato restrittivo. Per quanto riguarda le *keyword* positive i calcoli sono pressoché simili, l'unica differenza è che mentre quelle negative agiscono solo su significati che esse individuano, quelle positive agiscono su tutti i significati dei nomi individuati, vale a dire positivamente su quelli sottostanti le *keyword* e negativamente su gli altri.

Figura 6.3: SLIM prima dell'integrazione di ARM.

6.3.2 Semantic Field

Il panello guida dei *SemanticField* (vedi figura 6.10), mostra una tabella di tutti i *semanticfield* presenti nei significati dei nostri termini con a fianco un numero che rappresenta l'occorrenza. Selezionando uno di essi sono visualizzati nella tabella in basso i termini che contengono il *semanticfield*, e un *check box* che permette di selezionarli. Per ogni termine viene riportato il significato e la forma base, anche qui come nel pannello delle *keyword* sono presenti i 3 bottoni che permettono di selezionare il tipo di influenza (positiva, negativa, restrittiva) che si vuole assegnare al *semanticfield*.

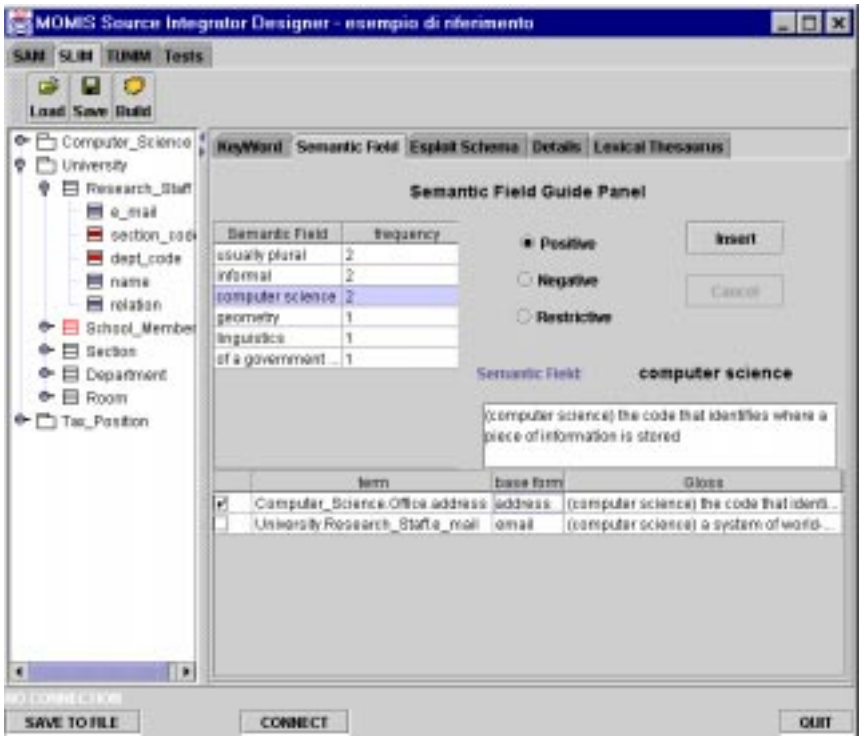


Figura 6.10 Pannello guida per i *SemanticField*

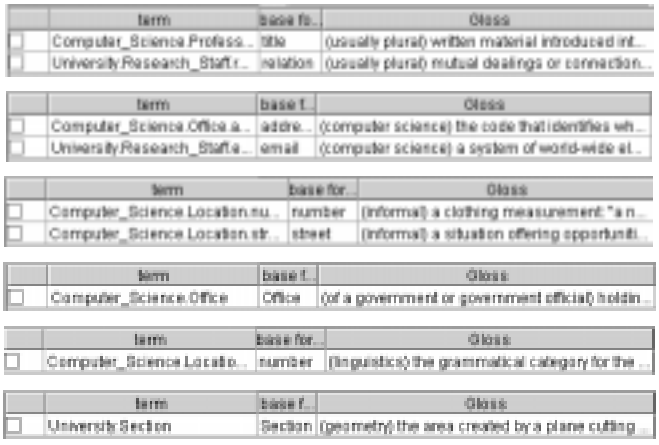
Figura 6.3: SLIM prima dell'integrazione di ARM.

La figura 6.9 ci mostra il pannello guida per l'inserimento dei *semanticfield* ed in particolare ci fa vedere cosa succede quando se ne seleziona uno, in questo caso quello selezionato è *computer science*, nella tabella in basso sono invece visualizzati i termini in cui è presente, che nel nostro esempio sono:

`Computer_Science.Office.address`
`University.Research_Staff.e_mail.`

Per fare in modo che il *semanticfield* abbia effetto sul termine bisogna selezionarlo e poi premere il bottone *insert*, anche qui apparirà una finestra di conferma che chiede se accettare o meno l'inserimento.

Di seguito riporto la lista dei termini presenti nelle nostre sorgenti contenenti *semanticfield*.



Osservando i termini notiamo che tutti possono essere eliminati ad eccezione di:

`University.Research_Staff.e_mail`
`University.Research_Staff.relation`

Potremmo accettare cosi `e_mail` e `relation`, usando il *semanticfield* con significato positivo, ed eliminare gli altri usandolo con significato restrittivo. Tralasciamo il caso di `e_mail`, poiché ha polisemia 1 e quindi sarebbe automaticamente disambiguato, analizziamo `relation` che ha grado di polisemia 5.

Inizialmente il suo peso coinciderà con il grado di polisemia, mentre numeratore e denominatore saranno uguali ad 1. Il contributo (s) che il *semanticfield* ha sui termini si è posto, in base a prove sperimentali, uguale a 4. Passiamo a vedere

Figura 6.3: SLIM prima dell'integrazione di ARM.

come variano i pesi inserendo il *semanticfield* "*usually plural*" con significato positivo:

il numeratore del significato, di *relation*, individuato dal *semanticfield* diverrà: *den*=*den***s*=1*4=4;

mentre il peso: $w = \frac{p * num}{den} + 1 = \frac{5 * 1}{4} + 1 = 2.25$;

Per i quatto significati rimanenti di *relation* il numeratore diventerà:

num=*num***s*= 1*4=4;

e il peso: $w = \frac{p * num}{den} + 1 = \frac{5 * 4}{1} + 1 = 21$;

Quest'ultimo peso, maggiore del valore di soglia massima (che è 19), porterà l'eliminazione dei quattro significati, e quindi la diminuzione del grado di polisemia del significato scelto il quale diventerà cosi non ambiguo

Dall'esempio di riferimento, purtroppo, non è possibile elogiare l'uso di questo strumento. Infatti, compaiono dei *semanticfield* che sono loro stessi "ambigui", nel senso che non specificano un contesto (come "*usual plural*" e "*informal*"), inoltre non compare un gruppo di nomi legati da un unico contesto su cui è possibile fare una disambiguazione di gruppo, questo è dovuto soprattutto alla scarsa presenza di queste *label* nel *WordNet* (contenute solo nel 3,5% dei nomi).

I *semanticfield* ci permettono di individuare il significato di un nome in modo intuitivo, infatti, nel nostro esempio vedendo "*address*" associato a "*computer science*" e facile pensare che è inteso come "*the code that identifies where a piece of information is stored*" invece di "*the place where a person or organization can be found or communicated with*". In più ci permettono di organizzare i nomi in gruppi aventi lo stesso contesto, e quindi è possibile lavorare per rimuovere l'ambiguità su un gruppo e non su un singolo nome.

6.3.3 Exploit Schema

In questo pannello sono presenti 4 bottoni ognuno accompagnato da una piccola descrizione (figura 6.11), ogni bottone ricerca se una particolare relazione

Figura 6.3: SLIM prima dell'integrazione di ARM.

strutturale tra due termini sia anche lessicale, quando la relazione viene trovata sono proposti all'utente i termini completi di significato.

Dati due termini, in relazione dal punto di vista strutturale, se troviamo tra essi una relazione lessicale è probabile che i significati con cui compaiono nella relazione siano quelli relativi al nostro contesto.

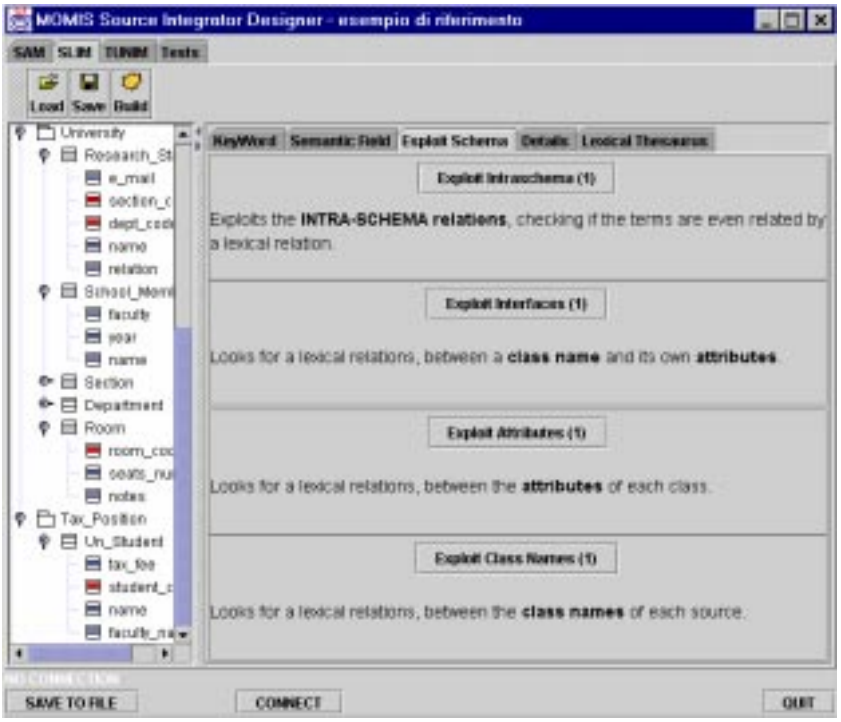


Figura 6.11: Pannello guida per sfruttare le relazioni strutturali.

Dalla figura 6.11 possiamo notare che su ogni bottone compare anche un numero, questo perché ogni volta che un pulsante viene premuto la ricerca della relazione lessicale viene estesa a termini più generici, e il peso che verrà associato dipenderà dal numero di livelli in cui si effettua la ricerca della relazione lessicale.

Quando viene trovata una relazione lessicale, vengono visualizzati i termini attraverso una finestra di conferma, essa oltre a dare l'opportunità di accettare i significati (utilizzando il peso con significato positivo) o ignorarli, permette anche di eliminarli.

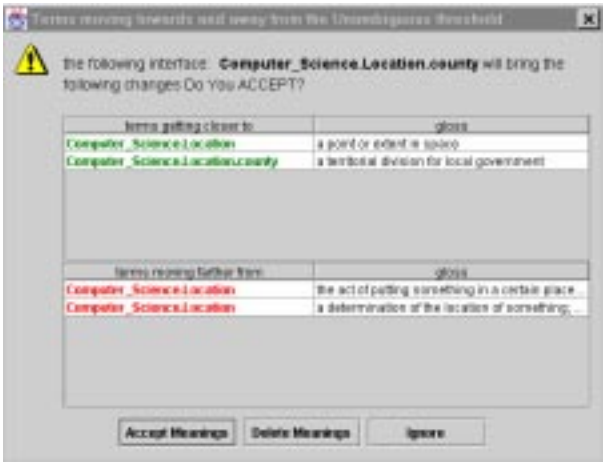
Passiamo ora a vedere le relazioni estratte per ogni singolo pulsante.

Figura 6.3: SLIM prima dell'integrazione di ARM.

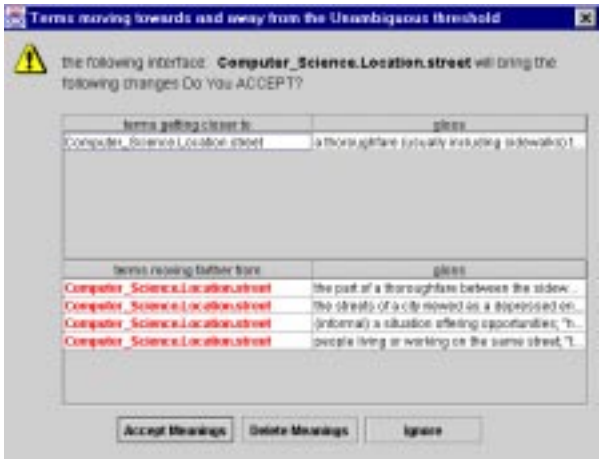
6.3.3.1 *Exploits Interfaces*

Le *Exploits Interfaces* ricerca per ogni coppia, nome di classe e attributo, se esiste una relazione lessicale.

Ritornando al nostro esempio si può notare che alla terza pressione del pulsante appare la finestra che ci propone i significati per *location* e *city*. Quello relativo a location è attinente al nostro contesto mentre quello relativo a *city* non è del tutto adatto; a questo punto quindi ignoriamo i risultati.

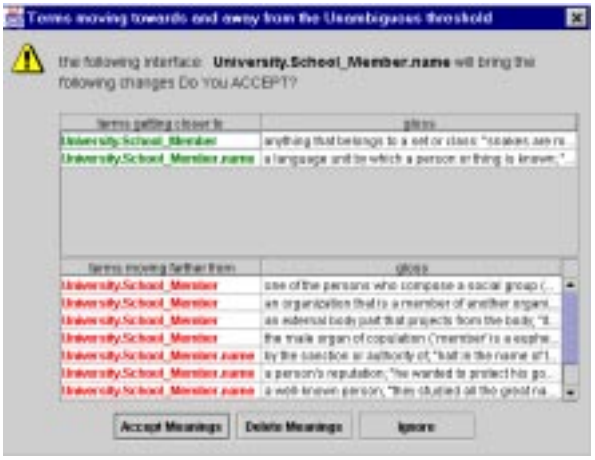


La finestra successiva ci propone *location* e *county*, in questo caso entrambi possono essere accettati.



Quella successiva invece ci propone per *street* "*a thoroughfare (usually including sidewalks) that is lined with building*". Questo significato è attinente, e viene quindi accettato.

Figura 6.3: SLIM prima dell'integrazione di ARM.



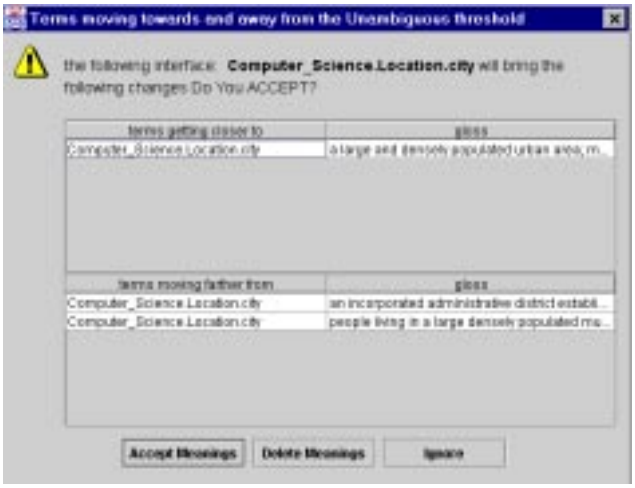
Lo stesso vale per *Member* e *name*, entrambi sono attinenti, e quindi vengono accettati.

Mentre in quest'ultima i significati *section* e *length*, non essendo attinenti al contesto vengono eliminati.

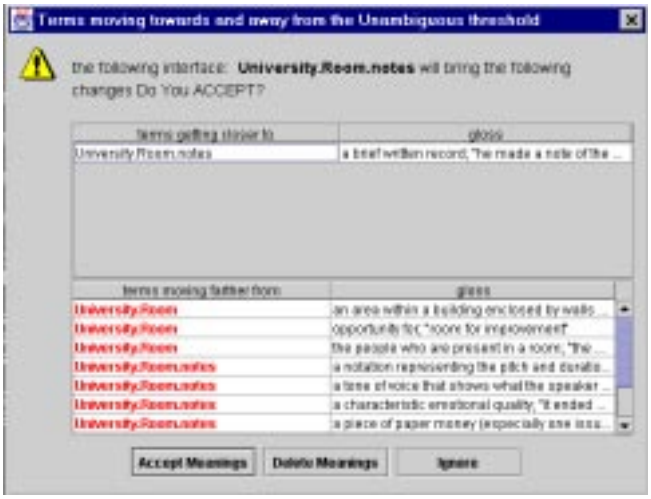
I termini disambiguati dopo questa fase sono : `Computer_Science.Location`, `Computer_Science.Location.county`, `Computer_Science.Location.street`, `University.School_Member`, `University.School_Member.name`.

Premendo ulteriormente il pulsante che permetterà, cosi facendo, di estendere ulteriormente la ricerca di relazioni lessicale si ottiene:

Figura 6.3: SLIM prima dell'integrazione di ARM.



city è attinente e quindi viene accettato,



così come notes.

Con quest'ultimi due termini estratti, `Computer_Science.Location.city`, e `University.Room.notes`, abbiamo disambiguato un totale di sette nomi. Se questo doveva essere fatto con SLIM occorreva per ognuno di questi nomi andare a leggere ogni significato per poi scegliere quello esatto. Inoltre questo strumento facilità la deduzione del significato visualizzando i nomi in coppia.

6.3.3.2 Exploits Attributes

Questo pulsante invece ricerca, per ogni classe, se una coppia di attributi appartenenti alla medesima classe sono legati da una relazione strutturale. Riporto di seguito solo i termini che sono stati proposti con esito positivo.

Figura 6.3: SLIM prima dell'integrazione di ARM.

Dopo aver cliccato la terza volta sul pulsante compare la finestra di conferma. In questo caso i significati proposti e accettati sono *first_name* e *last_name*.

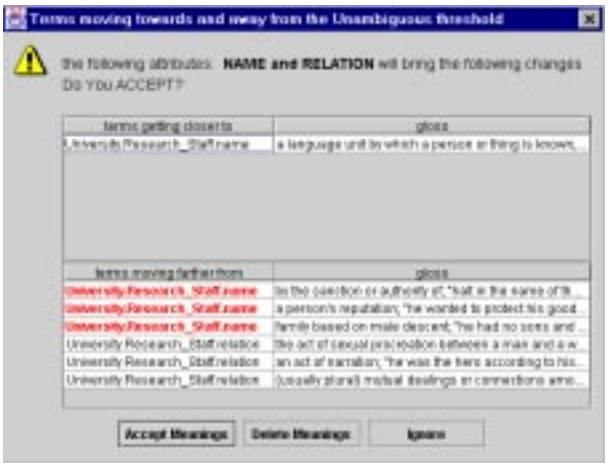
city; è un altro termine proposto che può essere accettato.

Altri due termini che sono attinenti al nostro contesto sono: *faculty* e *year*

I termini fin qui disambiguati sono: `Computer_Science.CS_Person.last_name`, `Computer_Science.CS_Person.first_name`, `Computer_Science.Student.year`, `Computer_Science.Location.city`, `University.School_Member.faculty`, `University.School_Memeber.year`.

Figura 6.3: SLIM prima dell'integrazione di ARM.

Un ulteriore pressione del pulsante ci darà:



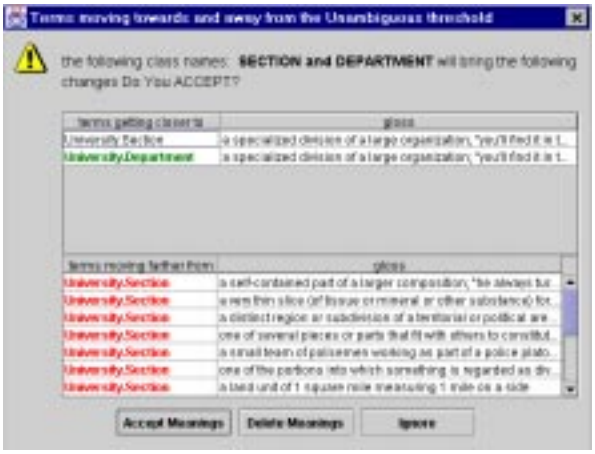
il termine *name*, con significato attinente.

Con `University.Research_Staff.name`, abbiamo estratto sei termini sfruttando le relazioni tra attributi.

6.3.3.3 Exploits Class Names

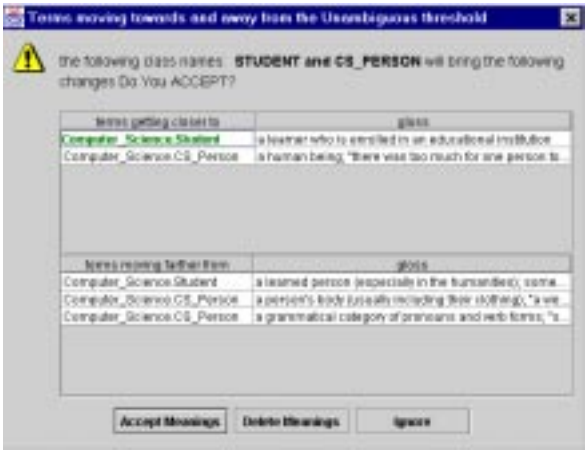
La ricerca effettuata schiacciando il pulsante "*Exploits Class Names*" controlla per ogni sorgente, se una coppia di nomi di classi appartenenti alla stessa sorgente sono in relazione lessicale.

Si riportano di seguito i risultati ottenuti.

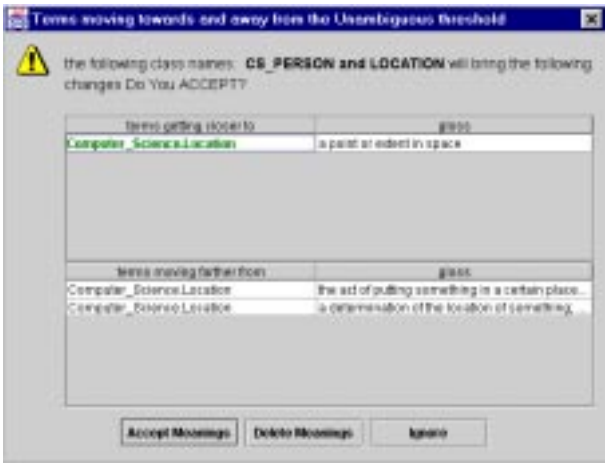


I primi termini proposti sono *section* e *department*, entrambi possono essere accettati.

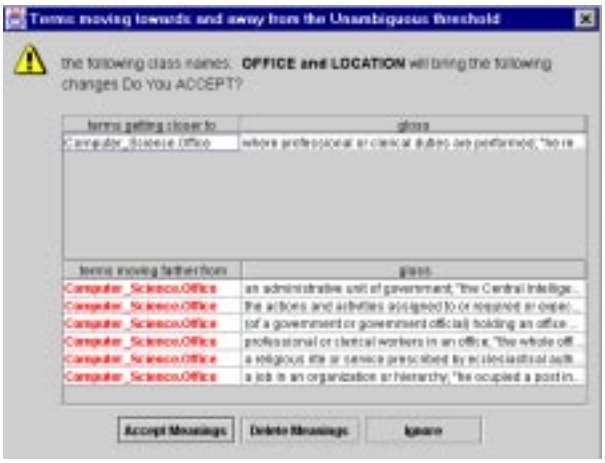
Figura 6.3: SLIM prima dell'integrazione di ARM.



Altri due termini che verranno accettati sono: *student* e *person*



location, è un altro che può essere accettato



così come è *office*.

Figura 6.3: SLIM prima dell'integrazione di ARM.

è *course*.

In totale avremmo estratto, sfruttando le relazioni tra classi sette nomi: `University.Section, University.Department, Computer_Science.Student, Computer_Science.CS_Person, Computer_Science.Location, Computer_Science.Office, Computer_Science.Course.`

6.3.3.4 *Exploits Intraschema*

Exploits Intraschema controlla per ogni coppia di termini appartenenti ad una relazione INTRA-SCHEMA se esiste una relazione lessicale che li lega.

Le relazioni INTRA-SCHEMA estratte da SIM sono riportate in figura 6.12

Source	Type	Destination	Produ...	Valid
University.Department	RT	University.Research_Staff	900	☑
University.Section	RT	University.Research_Staff	900	☑
University.Room	RT	University.Section	900	☑
Computer_Science.Student	NT	Computer_Science.CS_Person	900	☑
Computer_Science.Professor	NT	Computer_Science.CS_Person	900	☑
tax_position_xml.Student	RT	tax_position_xml.ListOfStudent	900	☑
Computer_Science.Course	RT	Computer_Science.Student	900	☑
Computer_Science.Location	RT	Computer_Science.Office	900	☑
Computer_Science.Professor	RT	Computer_Science.Course	900	☑
Computer_Science.Office	RT	Computer_Science.Professor	900	☑

Figura 6.12: Relazioni INTRA-SCEMA relative all'esempio di riferimento

Sfruttando le relazioni INTRA-SCHEMA di figura 6.12, riusciremo a disambiguare i termini sotto riportati.

Figura 6.3: SLIM prima dell'integrazione di ARM.

I primi due termini proposti sono *student* e *person* entrambi vengono accettati perché relativi al nostro contesto.

Altri due termini proposti sono *office* e *location*, entrambi possono essere accettati.

Sfruttando le relazioni INTRA-SCHEMA siamo riusciti a disambiguare quattro termini e sono: `Computer_Science.Student`, `Computer_Science.CS_Person`, `Computer_Science.Office`, `Computer_Science.Location`.

6.3.4 Details

L'ultimo pannello rimasto è *Details* che non è uno strumento per rimuover l'ambiguità, ma può essere utile per osservare l'influenza che ha avuto un'operazione sui significati. Esso è diviso in due cartelle, *Terms* e *Relations*, ed ognuna di esse e divisa in altri tre pannelli, *Unambiguous*, *Ambiguous*, *Trashed*, in cui sono visualizzati i termini (o relazioni), ordinati in base al peso e con il loro rispettivo significato (vedi figura 6.13). Nel pannello relativo ai termini (o le relazioni) ambigui viene anche visualizzato il peso e un carattere tra parentesi tonde che ci indica

Figura 6.3: SLIM prima dell'integrazione di ARM.

l'ultima variazione che il peso ha subito, in base all'ultima operazione fatta, più precisamente il carattere '-' indica che il peso ha subito una variazione che lo ha avvicinato alla soglia di disambiguazione, il carattere '=' indica che il peso è rimasto costante, mentre il '+' indica che si è allontanato.

Il pannello delle relazioni per ora non ha nessuna utilità, in quanto si è ritenuto effettuare l'estrazione delle relazioni solo sui termini disambiguati. Quindi le relazioni estratte sono tutte prive di ambiguità.

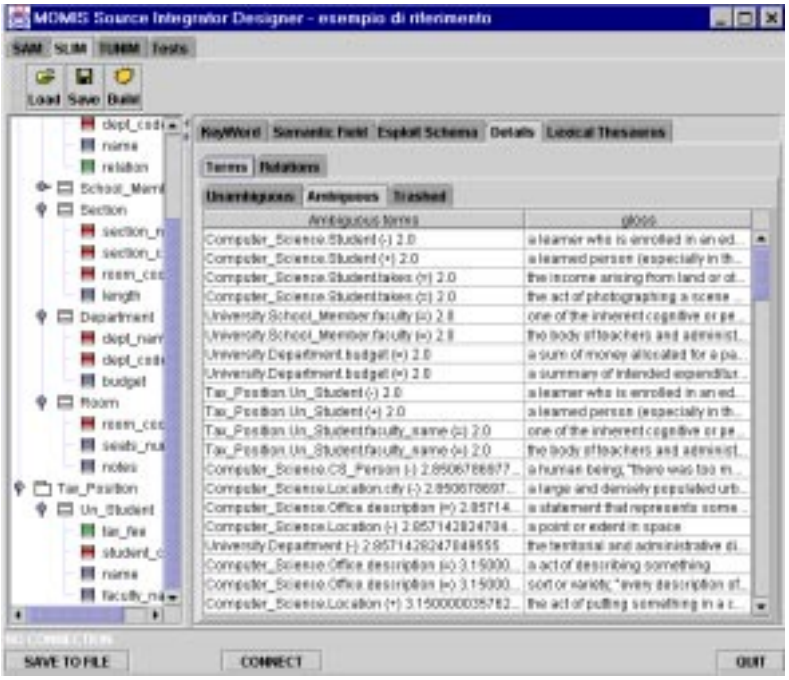


Figura 6.13: Panelo *Details*

Selezionando un termine nel pannello *Details* e successivamente andando nel pannello *KeyWord* troveremo che la *keyword* corrente risulta essere il termine selezionato in *Details*, questo può essere utile nel caso si decidesse di accettare o eliminare un termine (vedi figura 6.14).

Figura 6.3: SLIM prima dell'integrazione di ARM.

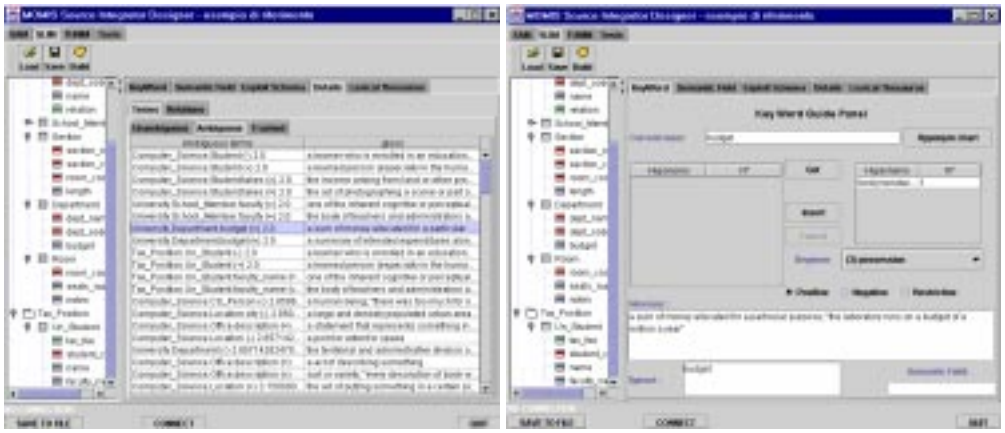


Figura 6.14: Esempio di come eliminare direttamente un significato

6.4 Analisi di ARM come strumento di rimozione dell'ambiguità

Abbiamo fin qui visto singolarmente gli strumenti che ARM fornisce nel processo di rimozione dell'ambiguità, nella tabella 6.1 sono riportati i termini estratti.

Termini	***interface***	***attributes***	*class names*	**INTRA-SCHEMA**
Computer_Science.Location	**X**		**X**	**X**
Computer_Science.Location.county	**X**			
Computer_Science.Location.street	**X**			
Computer_Science.Location.city	**X**	**X**		
Computer_Science.Student			**X**	**X**
Computer_Science.Student.year		**X**		
Computer_Science.CS_Person.last_name		**X**		
Computer_Science.CS_Person.first_name		**X**		
Computer_Science.CS_Person			**X**	**X**
Computer_Science.Office			**X**	**X**
Computer_Science.Course			**X**	
University.School_Member	**X**			
University.School_Member.name	**X**			
University.School_Member.faculty		**X**		
University.School_Memeber.year		**X**		
University.Research_Staff.name		**X**		
University.Section			**X**	
University.Department			**X**	
University.Room.notes	**X**			
totale	**7**	**7**	**7**	**4**

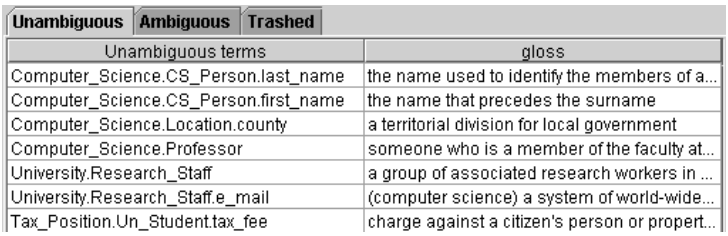
Tabella 6.1 termini estratti sfruttando le informazioni degli schemi

Figura 6.3: SLIM prima dell'integrazione di ARM.

Analizzeremo ora una simulazione di ARM nel rimuovere l'ambiguità. Consideriamo l'esempio di riferimento, supponiamo che alcune forme basi siano state già risolte, come: `e_mail`, `CS_Person`, `notes`, `seats_num`, `faculty_name`, `Tax_fee`, `Un_Student`. In totale avremo 37 termini e 163 significati.

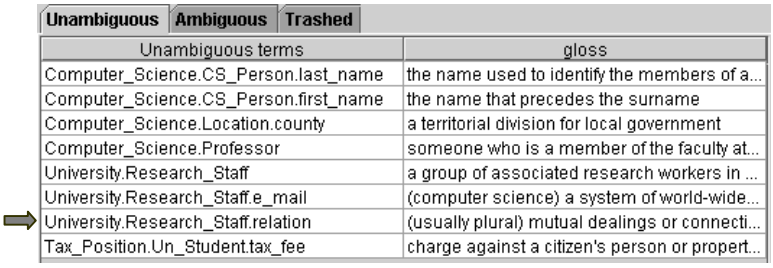
Analizziamo ora la fase di disambiguazione:

1. Inizialmente se controlliamo nel pannello relativo i dati non ambigui troveremo sette termini che sono stati considerati non ambigui poiché hanno polisemia 1



Unambiguous terms	gloss
Computer_Science.CS_Person.last_name	the name used to identify the members of a...
Computer_Science.CS_Person.first_name	the name that precedes the surname
Computer_Science.Location.county	a territorial division for local government
Computer_Science.Professor	someone who is a member of the faculty at...
University.Research_Staff	a group of associated research workers in ...
University.Research_Staff.e_mail	(computer science) a system of world-wide...
Tax_Position.Un_Student.tax_fee	charge against a citizen's person or propert...

2. Come secondo passo si usa il pannello dei *SemanticField*, in modo restrittivo per i termini non relativi al contesto, e in modo positivo per quelli da accettare. L'unico termine che và nel pannello dei non ambigui è `University.Research_Staff.relation`, mentre nei termini eliminati né vanno 11.



Unambiguous terms	gloss
Computer_Science.CS_Person.last_name	the name used to identify the members of a...
Computer_Science.CS_Person.first_name	the name that precedes the surname
Computer_Science.Location.county	a territorial division for local government
Computer_Science.Professor	someone who is a member of the faculty at...
University.Research_Staff	a group of associated research workers in ...
University.Research_Staff.e_mail	(computer science) a system of world-wide...
University.Research_Staff.relation	(usually plural) mutual dealings or connecti...
Tax_Position.Un_Student.tax_fee	charge against a citizen's person or propert...

3. Osservando il pannello *Details* ci sono dei nomi (come *name*, *rank* ecc.) che compaiono in più di una classe, sfruttando le *keyword* riusciamo a disambiguarli contemporaneamente.

   Come visto nella sezione 6.3.1 prima di inserire una *keyword* positiva su *name* occorre eliminare il significato di *title*, subordinato ad che non è attinente al contesto, fatto questo inseriamo prima come *keyord name* (*"a language unit by*

Figura 6.3: SLIM prima dell'integrazione di ARM.

wich a person or thing is known"), e successivamente *rank* ("*relative status*") ottenendo cosi altri 6 nomi disambiguati e un totale di 42 significati eliminati

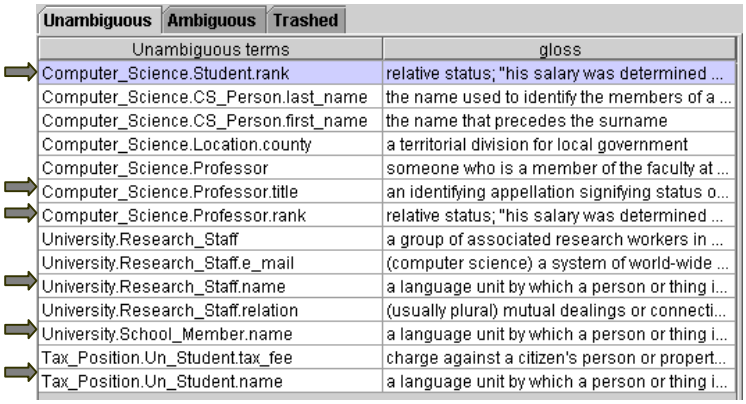
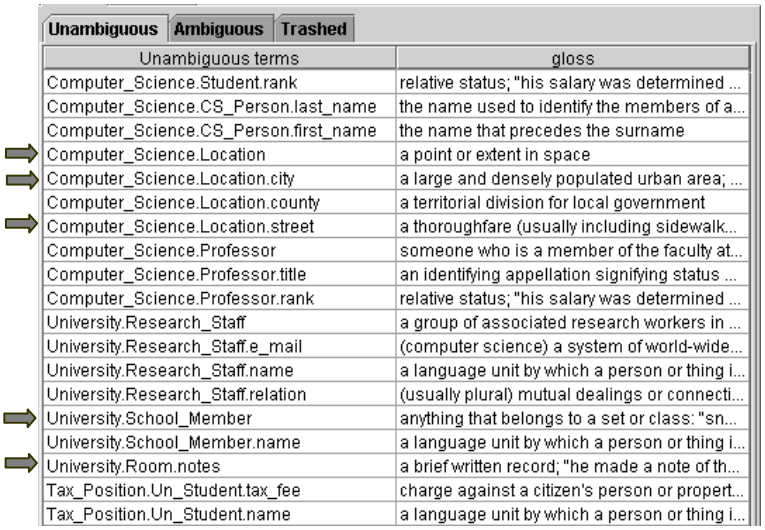


Figura 6.3: SLIM prima dell'integrazione di ARM.

4. Nella fase successiva utiliziamo l'*Exploit Schema*. Dopo aver sfruttato le *Interface* il pannello dei termini disambiguati ne contiene 19 mentre i significati eliminati sono 49.



5. Dopo le *Interface* usiamo il pulsane relativo agli *Attributes.* Come risultato avremo ulteriori tre termini disambiguati.

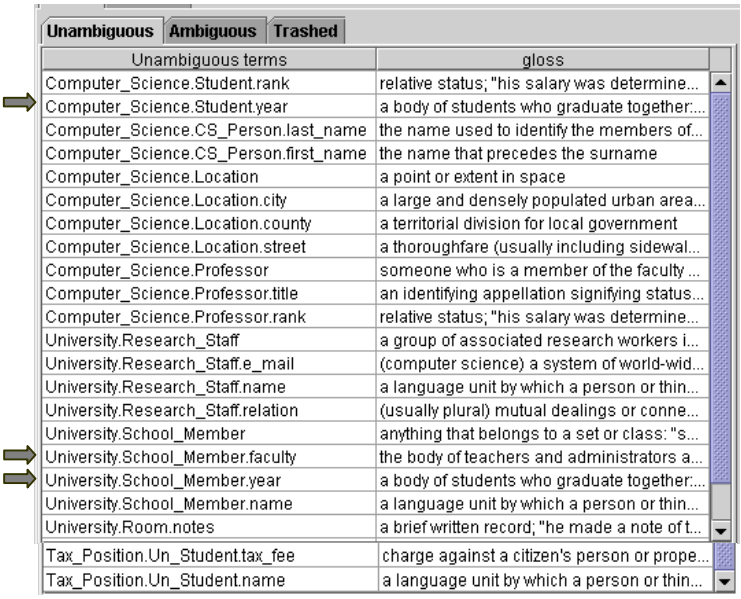
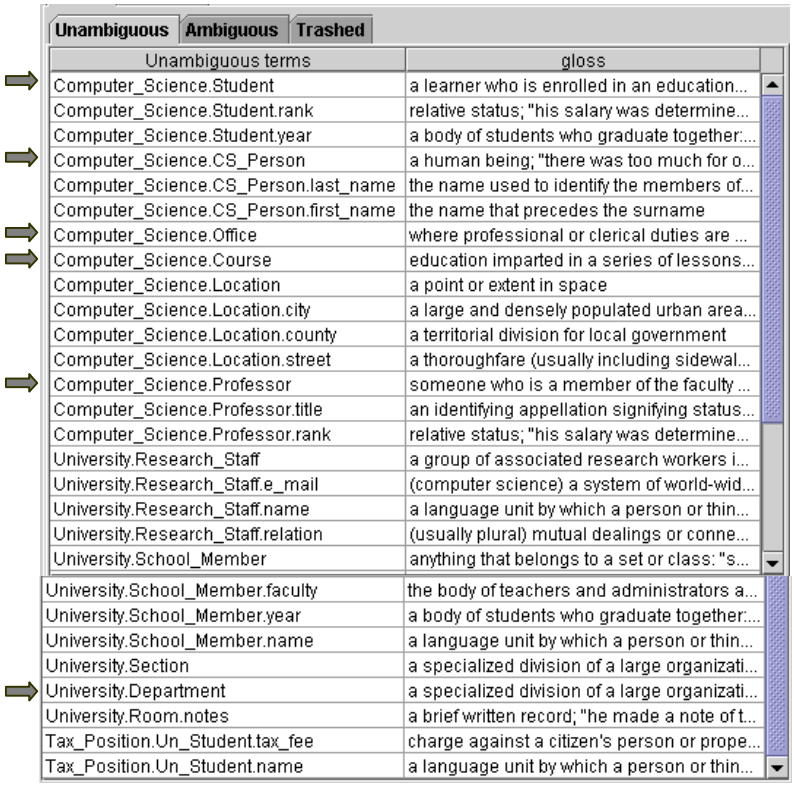


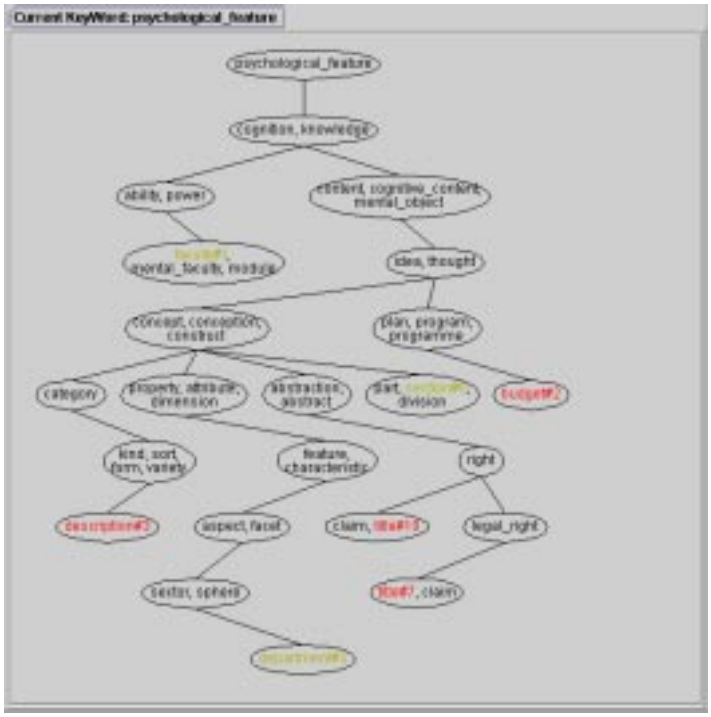
Figura 6.3: SLIM prima dell'integrazione di ARM.

6. Successivamente sfruttando i *Class Names* riusciamo a disambiguare altri 6 termini, ottenendo un totale di 28 termini disambiguati e 105 significati eliminati.

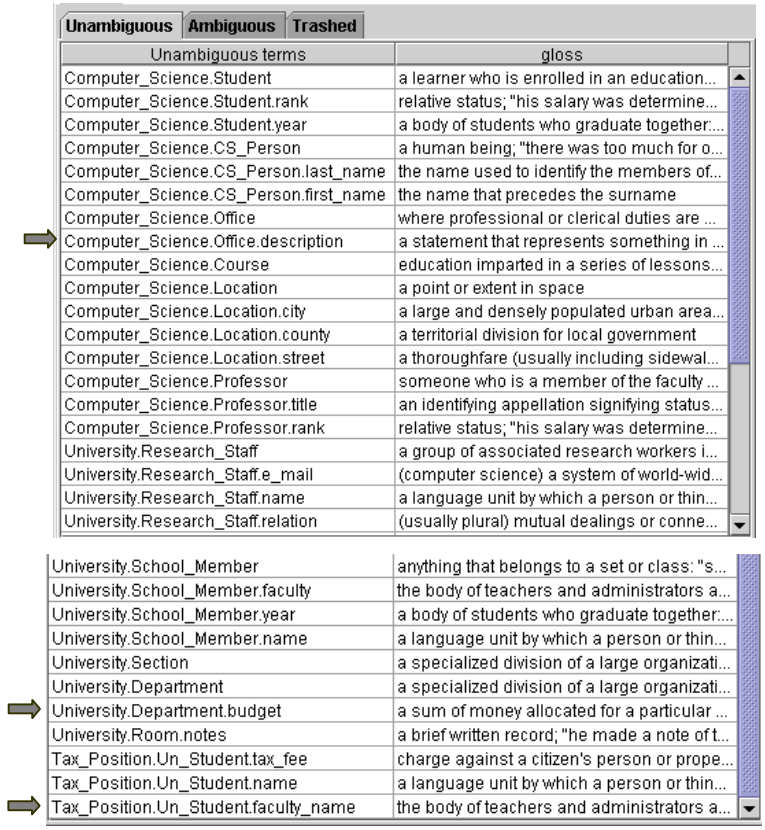


7. A questo punto inseriamo *psycological_feature* come *kwyword* restrittiva, analizzando i suoi *hyponomy* insieme ai pesi che essi hanno in questo momento, si riesce a prevedere che tre termini verranno disambiguati, infatti, `Computer_Science.Description`, `University.Department.budget` e `Tax_Position.Un_Student.faculty_name` sono presenti ognuno con due significati e con peso 2, l'inserimento della *keyword* negativa porta la cancellazione dei significati individuati, e di conseguenza viene decrementato di uno il grado di polisemia dei restanti, ricalcolando il peso avremo che i tre termini risulteranno disambiguati.

Figura 6.3: SLIM prima dell'integrazione di ARM.



Dopo l'inserimento della *keyword* restrittiva *pshycological_feature*, avremo eliminato altri 3 termini e ne avremo disambiguati altri 3, arrivando ad un totale di 108 eliminati e 31 disambiguati.



Utilizzando ARM siamo riusciti a rimuovere l'ambiguità dal 84% dei termini. In virtù di come ora appaiono le gerarchie dei nomi converrebbe continuare a rimuovere l'ambiguità dei 6 termini rimanenti utilizzando SLIM.

Alla fine di questo processo possiamo affermare che il risultato raggiunto soddisfacente, infatti, con questo strumento siamo riusciti a disambiguare 31 termini in maniera efficace.

Conclusioni

L'area di ricerca in cui si inserisce il progetto è l'Integrazione Intelligente di Informazioni (sistemi I^3), area abbastanza recente ma si avvale di tecniche di Intelligenza Artificiale ampiamente consolidate. L'obiettivo è lo sviluppo di un sistema in grado di fornire un accesso integrato e facilitato ad un insieme di sorgenti eterogenee distribuite.

Lo studio compiuto in questa tesi ha arricchito il sistema **MOMIS** di un componente importante per la costruzione delle relazioni lessicali del *Thesaurus* comune. Si è provveduto alla progettazione e realizzazione del *tool* ARM (***A**mbiguity **R**emoving **M**odule*) d'ausilio al progettista nella fase di rimozione dell'ambiguità data dalle parole, nomi di classi e attributi, presenti negli schemi delle sorgenti da integrare.

Lo strumento utilizzato per l'estrazioni delle relazioni lessicali è il *WordNet*, un *database* monolingue che consente di lavorare solo con sorgenti scritte nella lingua inglese. Questa limitazione sarà presto colmata grazie alla collaborazione con l'IRST (Istituto di Ricerca Scientifica e Tecnologica di Trento), il quale fornirà il *MultiWordNet,* un *database* lessicale multilingue. Altro limite è costituito dalla mancanza di un componente in grado di rilevare in modo intelligente le forme basi, tuttora inserite manualmente attraverso il *JTree* di SLIM.

I risultati sono stati raggiunti grazie ad un lavoro bene articolato. Inizialmente si è studiato il sistema MOMIS al fine di capirne il funzionamento e l'ambiente in cui il progetto si inserisce. Successivamente ci si è dedicati all'apprendimento di *WordNet*, e del linguaggio di programmazione *Java*. Fatto questo è stato possibile passare alla fase di progettazione e realizzazione del modulo *software*. Come ultimo si è provveduto ad integrare strutturalmente e graficamente i moduli SLIM ed ARM.

Considero estremamente formativa l'esperienza che ha portato a questo lavoro di tesi, non solo per l'importanza e l'attualità dei temi trattati, ma soprattutto per la disponibilità, serenità e professionalità riscontrate all'interno del gruppo di lavoro.

Il progetto **MOMIS**, in cui si inserisce questa tesi, è stato presentato all'ultima conferenza internazionale *Very Large DataBase* {VLDB2000}, Cairo (Egitto), 10-14 settembre 2000.

APPENDICE A

Glossario *WordNet*

In questa appendice sono riportati i termini usati da WordNet con la spiegazione. Durante la trattazione della tesi si è sempre cercato di utilizzare la traduzione in italiano del termine invece che l'originale inglese; nel glossario appaiono in tutte e due le lingue, per chiarezza.

adjective cluster Un gruppo di *synset* di aggettivi organizzati intorno a coppie o triplette di antinomi. Un cluster di aggettivi contiene due o più *synset* principali *(head synset*) che rappresentano i concetti opposti. Ogni *synset* principale ha uno o più *synset* satellite.

attribute Un nome per il quale degli aggettivi esprimono un valore. Esempio il nome `weight` è un *attribute* per il quale `light` e `heavy` esprimono un valore.

base form La forma base di un termine è la parola privata di declinazioni o coniugazioni.

collocation (locuzioni) Una locuzione in WordNet è una stringa di due o più parole connesse da spazi o trattino. Esempio: `man-eating shark`, `blue-collar`, `depend on`, `line of product`s. Nei file del database gli spazi sono sostituiti con il carattere di sottolineatura ('_').

coordinate Termini coordinati sono parole che condividono uno stesso iperonimo.

cross-cluster pointer Un puntatore semantico da un cluster di aggettivi ad un altro.

cousin Significati i cui iponimi generano una specifica relazione tra di loro.

direct antonyms Una coppia di parole tra le quali vi è una relazione di antinomia (contrari). Nei cluster degli aggettivi le relazioni di antinomia sono solo tra *synset* origine.

entailment Un verbo *X* implica *(entail*) *Y* se *X* non può essere svolto a meno che *Y* sia, o sia stato, eseguito.

exception list Trasformazione morfologica per le parole che non sono regolari e non possono essere processate in maniera algoritmica.

group Significati simili per via di relazioni di tipo *cousi*n, *sister* o *twin*.

gloss Definizione e/o sentenza d'esempio per un *synset*.

head synset *synset* in un cluster di aggettivi che contiene almeno una parola come diretto antinomo.

holonym Il "tutto" in una relazione di aggregazione. *Y* è un olonomo di *X* se *X* è una parte di *Y*.

hypernym Il termine sovraordinato in una gerarchia di specializzazione. *Y* è un iperonimo di *X* se *X* è un *(is a*) *Y*.

hyponym Il termine sottoordinato in una gerarchia di specializzazione. *X* è un iponimo di *Y* se *Y* è un *(is* a) *X*.

indirect antonym Un aggettivo in un *synset* satellite che non ha antinomi diretti, ma mediati attraverso il *synset* principale.

lemma rappresentazione tramite forma scritta o suoni di una parola.

lexical pointer Una relazione tra due singoli lemmi.

monosemous Un lemma che è contenuto in un solo *synset* in una categoria sintattica.

meronym La "parte" in una relazione di aggregazione. *X* è un olonomo di *Y* se *Y* è una parte di *X*.

part of speech Categoria sintattica: nomi, verbi, aggettivi ed avverbi.

participial adjective Un aggettivo derivato da un verbo.

pertainym Un aggettivo relazionale. Un aggettivo "pertinente" è usualmente definito da una frase come "di o pertinente a" e non ha antinomi. Può puntare ad un nome o ad un altro *pertainym*.

polysemous Un lemma contenuto in più *synset* in una categoria sintattica.

polysemy count Conteggio dei significati di un lemma in una categoria sintattica in WordNet.

postnominal Un aggettivo che si trova unicamente dopo il nome che modifica.

predicative Un aggettivo che può essere usato solo in una posizione predicativa. Se *X* è un aggettivo predicativo, può essere usato in una frase come "esso è *X*".

prenominal Un aggettivo che si trova unicamente prima del nome che modifica.

satellite synset Un *synset* in un cluster di aggettivi che rappresenta un concetto che è simile in significato al concetto del suo *synset* principale.

semantic pointer Un puntatore semantico che indica una relazione tra *synse*t.

sense Un significato di una parola in WordNet. Ogni senso di una parola è in un *synset* diverso.

sense key Informazione necessaria per trovare un senso in WordNet. Un *sense key* è formato da lemma, informazioni lessicografiche e, per gli aggettivi, informazioni a riguardo del *synset* principale. È chiave univoca ed immutabile con le diverse versioni di WordNet.

sister Un coppia di *synset* che siano iponimi dello stesso diretto sovraordinato.

synset Insieme di sinonimi; un insieme di parole che possono essere scambiate in certi contesti.

troponym Un verbo che esprime una maniera specifica per compiere un'azione di un altro verbo. *X* è un troponimo di *Y* se fare *X* è fare *Y* in una certa maniera.

twin *synset* che hanno almeno tre parole in comune.

unique beginner Un *synset* dei nomi che non ha iperonimi.

Appendice B

Esempio di riferimento in ODL$_{I3}$

Di seguito è riportata la descrizione, attraverso il linguaggio ODL$_{I3}$, dell'esempio di riferimento citato nella trattazione della tesi.

Sorgente `University` (a volte abbreviata in UNI):

```
interface Research_Staff
 ( source relational University
   extent Research_Staff
   key (name)
   candidate_key ke_mail (e_mail)
   foreign_key (dept_code)
references Department
   foreign_key (section_code)
references Section  )
{       attribute string name;
        attribute string
relation;
        attribute string e_mail;
        attribute integer
dept_code;
        attribute integer
section_code; };

interface School_Member
( source relational University
   extent School_Member
   key (name) )
{       attribute string name;
        attribute string faculty;
        attribute integer year;
};
```

```
interface Department
 ( source relational University
   extent Department
   key (dept_code) )
{       attribute string
dept_name;
        attribute integer
dept_code;
        attribute integer
budget;  };

interface Section
 ( source relational University
   extent Section
   key (section_code)
   foreign_key ( room_code )
references Room )
{       attribute string
section_name;
        attribute integer
section_code;
        attribute integer
length;
        attribute integer
room_code;  };

interface Room
 ( source relational University
   extent Room
   key (room_code) )
{       attribute integer
room_code;
        attribute integer
seats_number;
        attribute string notes;
};
```

Sorgente `Computer Science` (a volte abbreviata in CS):

```
interface CS_Person
 ( source object
Computer_Science
   extent CS_Persons
   key (first_name, last_name) )
{  attribute string first_name;
   attribute string last_name;
};

interface Professor : CS_Person
 ( source object
Computer_Science
   extent Professors )
{  attribute string title;
   attribute Office belongs_to;
   attribute string rank;     };

interface Student : CS_Person
 ( source object
Computer_Science
   extent Students )
{  attribute integer year;
   attribute set<Course> takes;
   attribute string rank;     };

interface Office
 ( source object
Computer_Science
   extent Offices
   key (description) )
{  attribute string description;
   attribute Location address;
};
```

```
interface Location
 ( source object Computer_Science
   extent Locations
   key (city, street, county,
number) )
{  attribute string city;
   attribute string street;
   attribute string county;
   attribute integer number;  };

interface Course
 ( source object Computer_Science
   extent Courses
   key (course_name) )
{  attribute string course_name;
   attribute Professor taught_by;
};
```

Sorgente `Tax Position` (a volte abbreviata in TP):

```
interface Un_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;   };
```

Bibliografia

[1] R. Hull and R. King et al. Arpa I^3 reference architecture, 1995. Available at http://www.isse.gmu.edu/I3_Arch/index.html.

[2] F. Saltor and E. Rodriguez. On intelligent access to heterogeneous information. In *Proceedings of the 4th KRDB Worksho*p, Athens, Greece, August 1997.

[3] N.Guarino. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. Technical report, Summer School on Information Extraction, Frascati, Italy, July 1997.

[4] N.Guarino. Understanding, building, and using ontologies. A commentary to 'Using Explicit Ontologies in KBS Development', by van Heijst, Schreiber, and Wielinga.

[5] Gio Wiederhold et al. *Integrating Artificial Intelligence and Database Technology*, volume 2/3. Journal of Intelligent Information Systems, June 1996

[6] Arpa I^3 refererence architecture. Avaliable at http://www.isse.gmu.edu/I3_Arch/index.html.

[7] Gio Wiedherhold. Mediators in the architecture of future information system. *IEEE Computer*, 25: 38 – 49, 1992.

[8] S. Chawathe, Garcia Molina, H., J. Hammer, K.Ireland, Y. Papakostantinou, J.Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *IPSJ Conference, Tokyo, Japa*n, 1994. ftp://db.stanford.edu /pub/chawathe/1994/ tsimmis-overview.ps.

[9] H. Garcia-Molina et al. The TSIMMIS approach to mediation: Data models and languages. In *NGITS worksho*p, 1995. ftp://db.stanford.edu/pub/garcia/1995/tisimmis-models-languages.ps.

[10] Y. Papakonstantinou, H. Garcia-Molina, and J. Ullman. Medmaker: A mediation system based on declarative specification. Technical report, Stanford University, 1995. ftp://db.stanford.edu/pub/papakonstantinou/1995/medmaker.ps.

[11] Y.Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In *VLDB Int. Conf.*, Bombay, India, September 1996.

[12] M.J.Carey, L.M. Haas, P.M. Schwarz, M. Arya, W.F. Cody, R. Fagin, M. Flickner, A.W. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J.H.

Williams, and E.L. Wimmers. Object exchange across heterogeneous information sources. Technical report, Stanford University, 1994.

[13] M.T. Roth and P. Scharz. Don't scrap it, wrap it! a wrapper architecture for legacy data sources. In *Proc. of the 23rd Int. Conf. on Very Large Database*s, Athens, Greece, 1997.

[14] Y. Arens, C.Y. Chee, C. Hsu, and C. A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information System*s, 2(2):127–158, 1993.

[15] Y. Arens, C. A. Knoblock, and C. Hsu. Query processing in the sims information mediator. *Advanced Planning Technolog*y, 1996.

[17] D. Beneventano, S. Bergamaschi, S. Lodi, and C. Sartori. Consistency checking in complex object database schemata with integrity constraints. Technical Report 103, CIOC{CNR, Viale Risorgimento, 2 Bologna, Italia, September 1994.

[18] S. Bergamaschi and B. Nebel. Acquisition and validation of complex object database schemata supporting multiple inheritance. Applied Intelligence: The International Journal of Artifcial Intelligence, Neural Networks and Complex Problem Solving Technologies, 4:185-203, 1994.

[19] Domenico Beneventano, Sonia Bergamaschi, Claudio Sartori, and Maurizio Vincini. Odb-qoptimizer: a tool for semantic query optimization in oodb. In Proc. of Int. Conf. on Data Engineering, ICDE'97, Birmingham, UK, April 1997.

[20] D. Beneventano, S. Bergamaschi, C. Sartori, and M. Vincini. a description logics based tool for schema validation and semanti query optimization in object oriented databases. Technical report, sesto convegno AIIA, 1997.

[21] Domenico Beneventano, Sonia Bergamaschi, Claudio Sartori, and Maurizio Vincini. Odb-tools: a description logics based tool for schema validation and semantic query optimization in object oriented databases. In Proc. of Int. Conference of the Italian Association for Artificial Intelligence (AI*IA97), Rome, 1997.

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

[23] A.G. Miller. Wordnet: A lexical database for english. Communications of the ACM, 38(11):39-41, 1995.

[24] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. Journal of Intelligent and Cooperative Information Systems, 2(4):375{398, 1993.

[25] B. Everitt. *Computer-Aided Database Design: the DATAID Projec*t. Heinemann Educational Books Ltd, Social Science Research Council, 1974.

[27] R. G. G. Cattell, editor. *The Object Database Standard: ODMG9*3. Morgan Kaufmann Publishers, San Mateo, CA, 1994.

[28] R.G.G. Cattell and others., editors. *The Object Data Standard: ODMG 2*.0. Morgan Kaufmann Publishers, San Francisco, CA, 1997.

[29] Domenico Beneventano, Sonia Bergamaschi, Claudio Sartori, and Maurizio Vincini. ODB-QOPTIMIZER: A tool for semantic query optimization in oodb. In *Int. Conference on Data Engineering - ICDE9*7, 1997. http://sparc20.dsi.unimo.it.

[30] D. Beneventano, A. Corni, S. Lodi, and M. Vincini. ODB: validazione di schemi e ottimizzazione semantica on-line per basi di dati object oriented. In *Quinto Convegno Nazionale su Sistemi Evoluti per Basi di Dati - SEBD97, Verona*, pages 208–225, 1997.

[31] W. A. Woods and J. G. Schmolze. The KL-ONE family. In F. W. Lehman, editor, *Semantic Networks in Artificial Intelligenc*e, pages 133–178. Pergamon Press, 1992. Pubblished as a Special issue of *Computers & Mathematics with Application*s, Volume 23, Number 2-9.

[32] J. P. Ballerini, D. Beneventano, S. Bergamaschi, C. Sartori, and M. Vincini. A semantics driven query optimizer for OODBs. In A. Borgida, M. Lenzerini, D. Nardi, and B. Nebel, editors, *DL95 - Intern. Workshop on Description Logic*s, volume 07.95 of *Dip. di Informatica e Sistemistica - Univ. di Roma "La Sapienza" - Rapp. Tecnic*o, pages 59–62, Roma, June 1995.

[33] J. J. King. QUIST: a system for semantic query optimization in relational databases. In *7th Int. Conf. on Very Large Database*s, pages 510–517, 1981.

[34] J. J. King. *Query optimization by semantic reasonin*g. PhD thesis, Dept. Of Computer Science, Stanford University, Palo Alto, 1981.

[35] M. M. Hammer and S. B. Zdonik. Knowledge based query processing. In *6th Int. Conf. on Very Large Database*s, pages 137–147, 1980.

[36] G. P. Grifa. Analisi di Affinità Strutturali fra classi ODL I3 nel Sistema MOMIS. Tesi di Laurea, Univeristà di Modena, Facoltà di Ingegneria, corso di laurea in Ingegneria Informatica, 1999.

[37] N.V. Findler, editor. *Associative Network*s. Academic Press, 1979.

[38] S. Bergamaschi, S. Castano, S. De Capitani di Vimercati, S. Montanari, and M. Vincini. Exploiting schema knowledge for the integration of heterogeneous sources. In *Sesto Convegno Nazionale su Sistemi Evoluti per Basi di Dati - SEBD98, Ancon*a, pages 103–122, 1998.

[39] G. Malvezzi, Tesi di Laurea, Estrazione di relazioni lessicali con *WordNet* nel sistema MOMIS. Tesi di Laurea Univeristà di Modena, Facoltà di Ingegneria, corso di laurea in Ingegneria Informatica, 1999.

[40] A. Artale, A. Goy, B. Magnini, E.Pianta & C. Strapparava. Coping with WordNet Sense Proliferation. *Irst, Istituto per la Ricerca Scientifica e Tecnologica.*

[41] B. Magnini, C. Strapparava, F. Ciravegna and E. Pianta. Multilingual Lexical Knowledge Bases: Applied WordNet Prospects. *Irst, Istituto per la Ricerca Scientifica e Tecnologica.*