UNIVERSITA' DEGLI STUDI DI MODENA
E REGGIO EMILIA

Facoltà di Ingegneria
Sede di Modena

Corso di Laurea Specialistica in Ingegneria Informatica

METODI DI DISAMBIGUAZIONE DEL TESTO ED ESTENSIONI DI WORDNET NEL SISTEMA MOMIS

Relatore:
Chiar.mo Prof. Sonia Bergamaschi

Correlatore:
Ing. Laura Po

Candidato:
Serena Sorrentino

Anno Accademico 2005/2006

PAROLE CHIAVE

WordNet

MOMIS

WordNet Domains

Disambiguazione del testo

Indice

Indice delle Figure

Indice delle Tabelle

Introduzione

Il diffondersi dell'utilizzo del Web, ha portato alla nascita dell'esigenza di poter reperire contenuti informativi provenienti da diverse risorse. Ciò, ha fatto emergere un crescente interesse verso lo sviluppo di sistemi di integrazione di risorse eterogenee. L'integrazione Intelligente delle Informazioni (*I3*) rappresenta la soluzione a tale problema. Il suo scopo è quello di ottenere in maniera automatica, una selezione ragionata dei dati provenienti dalle varie sorgenti e di conseguenza proporne una fusione intelligente. Un sistema che tenta di concretizzare tale obiettivo è MOMIS (***M****ediator* ***E****nvir****O****ment for* ***M****ultiple* ***I****nformation* ***S****ources*), il quale rappresenta un progetto di sistema I3, ideato per l'integrazione di sorgenti di dati testuali, strutturati e semi-strutturati.

Le problematiche legate alla realizzazione di tale sistema sono molteplici. Questa tesi si concentra in particolare, sui problemi derivanti dalla semantica dei dati.

Il problema semantico si intuisce facilmente, se si considera la possibilità che diverse persone possano fornire descrizioni, anche molto diverse tra loro, della stessa porzione di mondo. Anche se si possiede un insieme di conoscenze comuni (per esempio un'ontologia), non è possibile affermare che tali concetti saranno rappresentati, nelle diverse sorgenti, attraverso gli stessi vocaboli. Di conseguenza, nel processo d'integrazione, uno fra i tanti obbiettivi dovrà essere quello di risolvere le differenze semantiche fra le diverse rappresentazioni dei dati. Tale problematica, si realizza in MOMIS, attraverso l'annotazione semantica delle diverse sorgenti. Attualmente, tale annotazione, è realizzata in maniera completamente manuale. L'annotazione di un termine, implica al suo interno, il concetto di disambiguazione del termine stesso.

Il processo di *disambiguazione del testo,* consiste essenzialmente nell'identificazione dei concetti associati ai vari lemmi, ovvero nell'assegnare, ad ogni parola, il senso più corretto in base al contesto nel quale è utilizzato.

La disambiguazione presuppone l'utilizzo di un'ontologia (lessicale), dalla quale sia possibile reperire le informazioni relative ai concetti e alle relazioni che vi intercorrono. Tra le sorgenti d'informazione lessicale, più ampiamente utilizzate riconosciute, vi è WordNet, un database lessicale in lingua inglese. Tuttavia, durante il suo impiego nell'ambito delldisambiguazione

del testo, WordNet ha evidenziato alcune lacune, che gravano sui risultati del processo di disambiguazione.

L'obiettivo di questa tesi, è quindi quello di studiare i metodi e gli algoritmi basati sull'uso di WordNet. Inoltre, si analizzeranno l'estensioni, di tale database lessicale, proposte in letteratura, allo scopo di superare almeno in parte, le sue limitazioni. In particolare, si studierà WordNet Domains, un'estensione di WordNet, che associa ad ogni synset uno o più domini di appartenenza. Si tenterà, quindi, di verificare se l'uso di tale estensione, all'interno di MOMIS, consenta di annotare, almeno parzialmente, i termini in maniera automatica o semi-automatica.

Il contenuto della tesi è così strutturato:

- **Capitolo 1: WordNet.** Il seguente capitolo ha lo scopo di illustrare brevemente quelle che sono le caratteristiche strutturali e di contenuto del database lessicale WordNet.
- **Capitolo 2: Il Sistema MOMIS e le sue interazioni con WordNet.** Si descriverà in maniera sintetica l'architettura del sistema MOMIS e dei suoi componenti fondamentali; inoltre si identificheranno i meccanismi con cui, quest'ultimo, interagisce con il database lessicale WordNet.
- **Capitolo 3: Metodi ed Algoritmi di Disambiguazione del Testo**. In questo capitolo si classificheranno e si analizzeranno i differenti approcci al problema della disambiguazione del testo incontrati in letteratura, soffermandosi, in particolare, su quelli che utilizzano WordNet come risorsa lessicale
- **Capitolo 4**: **Estensioni di WordNet.** Si analizzeranno in maniera dettagliata, due fra le principali estensioni di WordNet proposte in letteratura, allo scopo di identificare possibili soluzioni alle lacune emerse dall'utilizzo di WordNet.
- **Capitolo 5**: **Intergrazione di WordNet Domains in MOMIS.** In questo capitolo si descrive la realizzazione del processo di integrazione fra l'ontologia di dominio *WordNet Domains* (WND) e il Sistema MOMIS. In particolare si descriverà la struttura gerarchica dei domini e le modifiche al database di MOMIS contenente le informazioni relative a WordNet, al fine di poter realizzare un processo di disambiguazione basato sulla polarizzazione di dominio all'interno di una sorgente di dati. Inoltre si riporteranno i risultati dei test effettuati sul sistema, al fine di stabilire l'effettiva utilità di WND.

Capitolo 1

1 WordNet

Il database lessicale WordNet [**1**], è stato sviluppato presso l'università di Princeton sotto la direzione del professore Gorge A.Miller. WordNet è disponibile gratuitamente presso il sito http://www.cogsci.princeton.edu/wn, la licenza d'uso consente l'utilizzo gratuito anche a fini commerciali ed al di fuori della ricerca, a condizione che siano citati gli autori ed il sito ufficiale del progetto.

Il database lessicale WordNet, non è semplicemente un dizionario di termini inglesi, è un sistema lessicale di riferimento il cui disegno è ispirato alle teorie psico-linguistiche contemporanee, sulla memoria lessicale umana. I termini, infatti, non sono disposti seguendo l'ordine alfabetico, ma per affinità di significato. WordNet comprende quattro categorie sintattiche: nomi, verbi, aggettivi ed avverbi. Ogni categoria sintattica è suddivisa in diversi insiemi di sinonimi; ad ognuno di questi insiemi è associato un unico significato, condiviso da tutti i termini ad esso associati.

Un termine, ovviamente, può possedere più di un significato ed essere, quindi, presente in molti di questi insiemi, ed anche in più di una categoria sintattica. Nel gergo utilizzato all'interno del progetto WordNet, un insieme di vocaboli che condivide il medesimo significato, prende il nome di *synset*. Un'altro elemento rilevante, che contraddistingue WordNet da un semplice dizionario di vocaboli, è la presenza di relazioni fra i synset. Vi sono diverse tipologie di relazioni che possono collegare due synset, come, ad esempio, l'iperonimia e l'iponimia, tramite cui si è in grado di creare, all'interno dell'intera categoria sintattica, gerarchie di significato.

1.1 La terminologia di WordNet

In questo paragrafo, si introducono un insieme di significati, propri della terminologia di WordNet. Tali termini verranno utilizzati in questo capitolo, che tratta di WordNet e della sua struttura, e nei capitoli successivi.

- ***Categoria sintattica:*** sono le grandi categorie in cui sono suddivisi i termini (ed anche i file in cui sono contenuti) di WordNet. Le categorie sintattiche trattate sono quattro: nomi, verbi, avverbi ed aggettivi.
- ***Lemma:*** è la parola/termine a cui vengono associati uno o più significati. A volte un lemma può essere costituito da due o più parole, ed, in tal caso, i singoli termini (detti composti) sono uniti dal carattere *underscore* (_).
- ***Synset:*** un synset rappresenta un significato che viene associato ad un insieme di lemmi appartenenti alla stessa categoria sintattica. In pratica risulta corretto affermare, che ad un synset corrispondono più lemmi. Un synset, infatti, può essere rappresentato, oltre che dalla sua glossa, anche dall'insieme dei suoi lemmi.
- ***Glossa:*** rappresenta la descrizione a parole di un significa specifico; ogni synset, oltre a contenere un insieme di sinonimi, possiede anche una glossa.
- ***Relazione Semantica:*** si tratta di una relazione di WordNet, che lega due synset appartenenti alla stessa categoria sintattica; i diversi tipi di relazioni semantiche verranno trattate di seguito.
- ***Relazione lessicale:*** è una relazione che collega due lemmi appartenenti a due synset distinti (ma sempre appartenenti alla stessa categoria sintattica); i diversi tipi di relazioni lessicali verranno trattate di seguito.

1.2 La matrice lessicale

Il punto di partenza per la semantica lessicale, è la comprensione del fatto che esiste un'associazione fra la forma di una parola (dove per forma di una parola intendiamo il modo in cui viene scritta e letta), ed il significato ad essa associato.
La corrispondenza fra forma della parola e significato, non è di tipo univoco, ma rappresenta, invece, una relazione di tipo molti a molti, dando luogo ai concetti di:

- ***Sinonimia:*** proprietà per cui lo stesso significato è esprimibile, tramite l'uso di due o più parole distinte.
- ***Polisemia:*** proprietà per cui ad una stessa parola, sono associati due o più significati distinti. Tale parole sono ambigue dal punto di vista del significato, e vengono dette *polisemiche*, per distinguerle da quelle che viceversa possiedono un solo significato (quindi non ambigue), e vengono dette *monosemiche*.

	W_1	W_2	W_3	W_4	W_5
M_1	$E_{1,1}$				
M_2		$E_{2,2}$			
M_3		$E_{3,2}$	$E_{3,3}$		
M_4					
M_5				$E_{5,4}$	
M_6					$E_{6,6}$

Figura 1.1-La matrice lessicale

Le relazioni fra forma della parola e significato, possono trovare rappresentazione in quella che viene chiamata, *matrice lessicale.* In tale matrice, le righe rappresentano i significati che è possibile attribuire ad una parola, mentre le colonne, rappresentano i diversi termini. In pratica, volendo leggere la matrice lessicale tramite la terminologia di WordNet, ad ogni riga è associato un synset, e ad ogni colonna una lemma. Ogni elemento non nullo che comparare all'interno della matrice, implica che il particolare lemma o termine, situato in quella riga, può essere usato per rappresentare lo specifico significato associato a quella colonna. Se all'interno di una colonna sono contenuti più elementi, si ha un caso di polisemia (il termine associato alla colonna può essere utilizzato per esprimere più di un concetto); se, al contrario, due o più elementi compaiono sulla stessa riga, si è in presenza di una caso di sinonimia (il significato o synset di tale colonna, può essere espresso tramite più parole distinte).

Il concetto di matrice lessicale, viene espresso nel database di WordNet, tramite la separazione fra lemmi e synset (mantenendo cioè separati, termini e significati).

Un synset viene espresso nei file usati in WordNet, tramite l'insieme dei termini che sono ad esso associati. Tuttavia, nella maggioranza dei casi, un insieme di parole di questo tipo, non è sufficiente a descrivere un significato (si pensi al caso in cui si stia trattando un significato particolare che può essere descritto da una sola parola), così viene associata a ciascun synset, anche una descrizione del significato tramite la glossa.

1.3 Le relazioni

WordNet presenta al suo interno, due grandi gruppi di relazioni che si differenziano seconda del tipo di operatori a cui sono applicate. Si hanno così relazioni semantiche quando gli operandi sono synset, viceversa, si hanno relazioni lessicali nel caso in cui gli operandi siano lemmi. Non possono, invece, esistere relazioni tra un lemma ed un synset o fra operandi appartenenti a differenti categorie sintattiche (ad esempio fra un nome ed un verbo). All'interno dei file originali di WordNet tutte le relazioni (eccezion fatta per la relazione di sinonimia), sono rappresentate tramite puntatori e tramite caratteri speciali, che indicano il tipo di relazione specificata. Nei prossimi paragrafi saranno descritte le principali relazioni semantiche e lessicali presenti in WordNet.

1.3.1 Le relazioni semantiche

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

Iponimia

Le relazioni di *iponimia* e *ipernimia* (che rappresenta la relazione inversa), possono essere considerate l'equivalente delle gerarchie di specializzazione/generalizzazione per database relazionali o per l'ereditarietà dei modelli ad oggetti. Una relazione semantica di questo tipo, è valida solamente per le categorie sintattiche dei nomi e dei verbi (ma per i verbi si parla di toponimia). Una relazione di iponimia, lega un concetto (nel nostro caso un synset) ad uno più generale, quello che può essere ritenuto una sua generalizzazione. Trattandosi di un database di lingua inglese è lecito dire che, un synset X è un iponimo di un synset Y, se è corretta

l'affermazione "*X is a kind of Y*". Per quanto riguarda la relazione opposta, quella di ipernimia, essa lega un concetto ad uno più particolare, più specializzato. In pratica si può affermare che un synset *X* rappresenta un ipernimo di un synset *Y*, se *Y* presenta tutte le caratteristiche di *X* più, almeno, una sua caratteristica particolare ed aggiuntiva. Le relazioni di iponimia ed ipernimia, sono le relazioni più numerose presenti all'interno del database lessicale, di WordNet. Un semplice esempio, per comprendere meglio queste importanti relazioni, potrebbe essere: ABETE è in iponimo di ALBERO, ALBERO è a sua volta un iponimo di VEGETALE. Sono, altresì, verificate le razioni opposte: VEGETALE è un ipernimo di ALBERO, ALBERO è un ipernimo di ABETE. La relazione di iponimia (assieme a quella di ipernimia), può essere utilizzata per formare una gerarchia di specializzazione fra i synset di WordNet.

Meronimia

Anche la relazione di *meronimia* lega fra loro due concetti, o synset, e anche in questo caso si è in presenza di una relazione inversa indicata come *olonimia* . Un concetto *X* è detto meronimo di un concetto*Y*, se è lecito per una madrelingua inglese, pronunciare la frase "*X is a part of Y*". Anche la relazione di meronimia, come quella di iponimia, può essere sfruttata per costruire una gerarchia sui synset di WordNet, in cui uno risulta essere una parte dell'altro. Le relazioni di meronimia e olonimia, vengono formulate sulla categoria sintattica dei nomi. Un esempio potrebbe essere rappresentato dai concetti MURA e FONDAMENTA come meronimi di COSTRUZIONE.

Implicazione

La relazione di *implicazione* è posta fra due verbi. Tale relazione può essere ritenuta simile a quella di meronimia posta sui nomi. Questa relazione è verificata se è vera la seguente proposizione: un verbo *X* implica un verbo *Y* se *X* non può verificarsi a meno che non si sia verificato (o non si stia verificando) Y. L'implicazione non è solamente una relazione semantica, ma è possibile avere anche implicazioni lessicali fra verbi (fra singoli termini). Per comprende meglio questo concetto si consideri i verbi DORMIRE e SOGNARE: in base alla definizione precedentemente, SOGNARE risulta implicare DORMIRE (*to dream entails to sleep*), infatti non è possibile sognare senza dormire. L'implicazione lessicale è una relazione univoca: se un verbo *X* implica un verbo *Y* non può essere vero anche il contrario.

Relazione causale

La relazione causale è simile alla relazione di implicazione ma senza inclusione temporale. Un esempio potrebbe essere FORZARE che implica AGIRE.

Raggruppamento di verbi

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

Similarità

La relazione di *similarità* è utilizzata solamente nell'ambito della categoria sintattica riguardante gli aggettivi. Molti synset di questa categoria sono raggruppati in coppie legate da una relazione di antinomia (si pensi, ad esempio, a synset trattanti i concetti di PESANTE e LEGGERO, in netta contrapposizione semantica fra di loro); tali synset, vengono chiamati synset principali (o *head synset*). A questi synset principali sono collegati per similarità, dei synset satelliti, che condividono indirettamente le relazioni di antinomia insieme al significato principale a cui sono legati. Ricapitolando, un aggettivo descrittivo, può avere una relazione di antinomia diretta (si tratta quindi di un synset principale), oppure una indiretta tramite l'ausilio di una relazione di similarità (synset satellite).

Attributo

La relazione di attributo rappresenta il legame che intercorre fra un aggettivo ed un nome di cui esprime il valore. Gli aggettivi in grado di descrivere il valore di un attributo, sono gli aggettivi descrittivi. Per fare un esempio basta pensare ad una frase come: *questa persona è alta.* L'aggettivo descrittivo alta, indica il valore dell'attributo *altezza* riferito a persona. Aggettivi quali *alta* o e *basso,* sono, quindi, legati al nome altezza. WordNet contiene puntatori fra gli aggettivi descrittivi ed i synset, appartenenti alla categoria sintattica dei nomi, che rappresentano gli attributi con cui conferiscono il valore.

Coordinazione

La *coordinazione* non è un tipo di relazione base, ma si potrebbe definire derivata. Due synset sono detti coordinati se possiedono lo stesso ipernimo, se, cioè, risultano essere la specializzazione del medesimo concetto.

1.3.2 Le relazioni lessicali

Le relazioni lessicali, diversamente da quelle semantiche, coinvolgono sempre due lemmi non due synset.

Sinonimia

La sinonimia, anche se rappresenta una relazione lessicale, non è espressa formalmente come le altre relazioni di WordNet: non esiste alcun puntatore che colleghi un termine al suo sinonimo. La relazione è espressa, invece, tramite l'appartenenza, da parte dei due vocaboli sinonimi, allo stesso synset. Ricordiamo, infatti, che lo stesso synset può essere rappresentato dall'insieme di lemmi a cui può essere associato; il termine synset (*set of synonym*) è stato coniato proprio per indicare quest'idea. Per ogni coppia di termini appartenenti allo stesso synset, esiste, dunque, una relazione di sinonimia in maniera implicita. Due possibili definizioni di sinonimia sono :

1. Due termini sono sinonimi se la sostituzione di uno per l'altro non cambia mai il valore della frase in cui è fatta la sostituzione. (Leibniz)
2. Due termini sono sinonimi, all'interno di un contesto linguistico C, se la sostituzione di un termine con l'altro, all'interno di C, non varia il valore della frase (definizione relativa ad un contesto).

La seconda definizione è decisamente più permissiva rispetto alla prima: esistono pochi termini considerati sinonimi nel senso descritto da Leibniz. Infatti, è estremamente difficile trovare due parole da poter interscambiare in ogni genere di contesto. Il database lessicale di WordNet, comunque, adotta, per stabilire la relazione di sinonimia, la seconda definizione: due lemmi sono sinonimi solo all'interno di uno stesso contesto, e di un certo synset. Anche tramite la seconda definizione di sinonimia, appare chiaro che due termini appartenenti a categorie sintattiche differenti, non potranno in nessun caso essere sinonimi. Proprio per questa ragione WordNet è stato diviso nelle categorie sintattiche di nomi, avverbi, verbi e aggettivi.

Antinomia

L'*antinomia* è una relazione lessicale fra due lemmi. Due termini legati da una relazione di antinomia sono l'uno il contrario dell'altro. Non è sempre corretta, comunque, l'affermazione che non *X* è antonimo di *X* . Si pensi, ad esempio, ai termini *ricco* e *povero*: se un individuo non è ricco, non è necessariamente detto che sia povero. Non è corretto considerare l'antinomia come una relazione semantica, quindi fra synset; per esempio i synset {*rise, ascend}* e {*fall, descend*}, pur essendo concettualmente opposti, non rappresentano degli antinomi. Una relazione di antinomia, invece, è presente fra i termini *rise* e *fall*, e *descend* e *ascend*.

Relazione di pertinenza

La relazione di *pertinenza* concerne gli aggettivi relazionali. Un aggettivo relazionale, svolge un ruolo che può essere riassunto in una espressione come: *associato con,* oppure *pertinente a* o semplicemente *di* in relazione ad un nome. L'aspetto di un aggettivo relazionale, risulta molto simile a quello del nome cui è legato, leggermente modificato. Si pensi all'espressione *accuratezza mentale,* l'aggettivo relazionale *mentale,* è associato al nome *mente,* tramite una relazione, appunto, di pertinenza.

Vedi anche

La relazione detta *vedi anche*, è una relazione lessicale e lega singoli lemmi di synset differenti. I motivi di tale relazione possono essere molto differenti fra loro.

Relazione partecipiale

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

Derivato da

Alcuni aggettivi derivano da antichi nomi Greci o Latini. Questa affermazione risulta essere vera, sia per la lingua italiana, che per quella inglese (idioma su cui è costruito WordNet).

L’aggettivo relazionale *verbale*, deriva dal nome neutro latino *verbum* , mentre *lessicale* deriva dal corrispondente nome greco. La relazione *derivato da* lega gli aggettivi ai nomi stranieri da cui derivano.

Capitolo 2

2 Il Sistema MOMIS

Al giorno d'oggi, un problema cui devono far fronte numerose imprese ed organizzazioni, è quello della dispersione del loro patrimonio informativo. Si pensi ai numerosissimi metodi di immagazzinamento di informazioni presenti sul mercato o utilizzabili gratuitamente: DBMS, pagine HTML, pagine XML ecc…

Nel caso in cui un utente voglia reperire informazioni da sorgenti diverse, fatto che accade sempre più frequentemente ogni giorno, si trova di fronte a problemi di non facile soluzione: le sorgenti di conoscenza, infatti, sfrutteranno tecnologie differenti, difficilmente uniformabili, senza contare le possibili contraddizioni ed inconsistenze fra i dati ottenuti da diverse fonti. Un grande aiuto per quanto concerne il problema dello sfruttamento di tecnologie differenti proviene dagli standard esistenti, (come l'ODBC, CORBA ecc…), che risolvono il problema della comunicazione fra moduli diversi. Ciò che rimane irrisolta, è la questione della modellazione delle informazioni: i modelli dei dati, possono differenziarsi gli uno dagli altri, a tal punto da fornire ognuno una propria struttura logica di rappresentazione dei dati da immagazzinare. Tutto ciò crea un'eterogeneità semantica non risolvibile dagli attuali standard. Da quanto descritto in precedenza, si evincono le difficoltà che sorgono nel creare un sistema di integrazione e mediazione delle informazioni eterogenee che sia affidabile, flessibile, modulare (in modo da consentire il riuso delle diverse parti all'evolvere delle tecnologie), e capace di interagire con altri sistemi esistenti. Nel seguito si descriverà una proposta di ARPA (***A****dvanced* ***R****eserch* ***P****roject* ***A****gency*) per un'architettura di integrazione di informazioni, flessibile e riusabile. L'approccio descritto dall'ARPA in [**2**], è stato seguito anche nel progetto MOMIS.

2.1 L’Integrazione Intelligente delle Informazioni

Come viene citato in [**3**], l’integrazione delle informazioni (I_2) si distingue da quella dei dati e dei database, in quanto non cerca di collegare semplicemente alcune sorgenti, ma risultati opportunamente selezionati da esse. Lo scopo dell’integrazione dell’informazione è, quindi, quello di ottenere una selezione ragionata dei dati prelevati dalle varie sorgenti, e produrre una fusione intelligente ed una seguente sintesi degli stessi. Proprio a questo scopo, è stato sviluppato dall’ARPA, un progetto di ricerca atto a fornire un’architettura di riferimento, che realizzi l’integrazione di risorse eterogenee in maniera automatica; il nome di questo progetto è appunto I_3 (**I**ntegrazione **I**ntelligente dell’**I**nformazione). I risultati ottenuti in questo ambito sono molto importanti poiché danno una concreta indicazione su come costruire un sistema di mediazione che sia riusabile, e le cui parti non comportino costi eccessivi di sviluppo. L’integrazione delle informazioni, inoltre, ne aumenta il valore, ma non è semplicemente gestire gli aggiornamenti, le eliminazioni e le sostituzioni fra le varie sorgenti, le loro ontologie e semantiche. L’ARPA ritiene, che una grossa mano a tal proposito, possa essere data dall’utilizzo dell’Intelligenza Artificiale che, essendo in grado di dedurre dagli schemi delle sorgenti informazioni utili, può essere considerata uno strumento prezioso ed in grado di fornire soluzioni flessibili e riusabili. Secondo il programma I_3 è opportuno costruire architetture modulari, in grado di abbassare i costi di sviluppo e mantenimento, eseguite seguendo uno standard che ponga le basi dei servizi necessari all’integrazione. Il paradigma impiegato nel progetto I_3 per la suddivisione dei servizi e delle risorse fra i vari moduli, si basa su due partizionamenti fondamentali:

- Il partizio*namento orizzontale* che fornisce tre sezioni: database, sorgenti, basi di conoscenza intermedie, ed applicazioni utente.
- Il *partizionamento verticale* che distingue i domini in cui raggruppare le sorgenti.

I domini non sono strettamente interconnessi fra loro all’interno di un certo livello, ma si scambiano dati e informazioni. Per facilitare la flessibilità e migliorare le prestazioni del sistema, la fase importantissima della combinazione delle informazioni avviene a livello d’utente. Nel seguito verrà illustrata l’architettura di riferimento per i sistemi I_3

2.1.1 L’architettura dei sistemi I3

L’architettura del programma I3 definita dall’ARPA, deriva la sua forma dal tentativo di far fronte a problemi complessi come quelli citati nei paragrafi precedenti, e che vengono qui brevemente riproposti in un elenco :

1. *Eterogeneità delle sorgenti*: differenze fra i tipi di dato, schemi logici, interfacce per accedere ai dati…
2. *Evoluzione delle sorgenti di dati*: si possono aggiungere nuove fonti o modificare o eliminare quelle vecchie.
3. *Dimensioni delle fonti*: bisogna far fronte all’aumento dei dati di una singola sorgente (ed all’aumento delle sorgenti) ed all’aumento dei tempi di risposta che ne derivano.
4. *Semantica nascosta*: bisogna dedurre regole, dai differenti schemi, per elaborare ed interpretare i dati da integrare.
5. *Necessità di sistemi modulari e riusabili:* questo punto è fondamentale per ridurre i tempi ed i costi di sviluppo delle varie applicazioni, e far fronte ai mutamenti tecnologici, che inevitabilmente si susseguono nel tempo.

Una volta compresi i problemi fondamentali, si può passare all’analisi dell’architettura di riferimento dei sistemi I3 proposta da ARPA. L’architettura del progetto I3 si propone di evidenziare (separando in più moduli), i vari servizi che devono essere svolti ai fini dell’integrazione intelligente d’informazioni. I servizi evidenziati a questo proposito sono cinque:

- *Servizi di coordinamento*
- *Servizi di amministrazione*
- *Servizi di integrazione e trasformazione semantica*
- *Servizi di wrapping*
- *Servizi ausiliari*

Fra tutti quelli elencati in precedenza, i servizi principali sono quelli di coordinamento il cui scopo è, appunto, quello di coordinare le operazioni attuate dai vari servizi sia in fase di progettazione dei vari link di integrazione fra le sorgenti, sia in fase di esecuzione (in tempo reale) su specifiche richieste dagli utenti.

Di seguito si descriveranno brevemente i servizi precedentemente elencati.

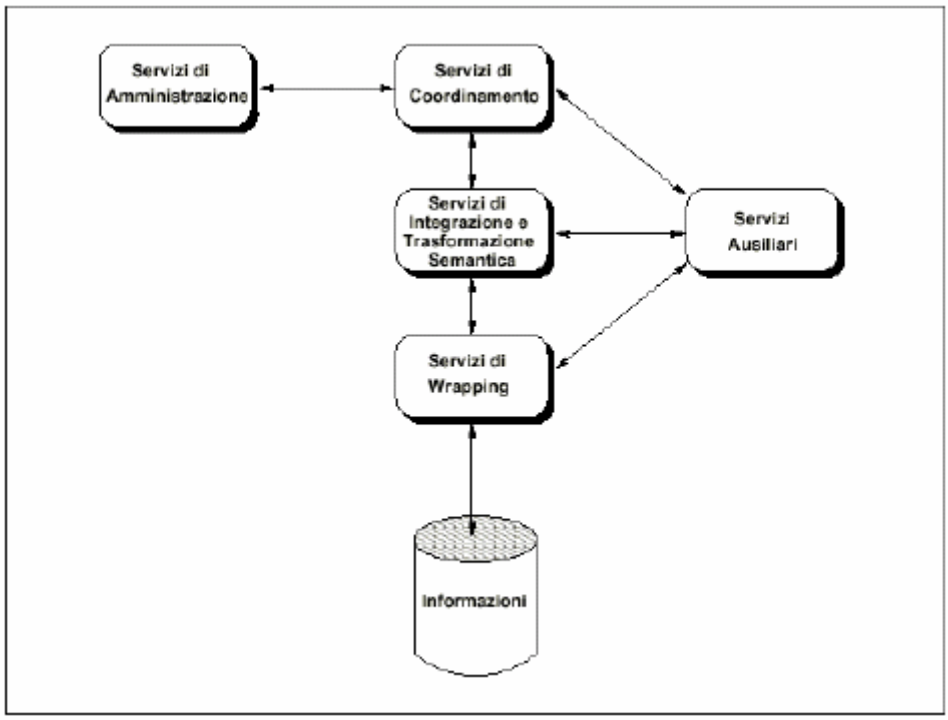


Figura 2.1-Diagramma dei servizi I^3

Servizi di coordinamento

I servizi di coordinamento svolgono i lavori di supporto, sia in fase di progettazione di nuove configurazioni, che a tempo di esecuzione delle richieste dell'utente. Questi servizi sono di alto livello e, oltre ad individuare quali sorgenti possono essere utili per soddisfare una data richiesta, presentano all'utente finale l'intero sistema, diviso fra i suoi vari moduli, come un blocco unico ed omogeneo. Grazie ai servizi di coordinamento, quindi, le divisioni interne di un sistema I^3 , sono trasparenti all'utente. I principali moduli appartenenti a questa sezione sono:

- *Broker*: il suo compito è quello di reperire gli strumenti in grado di trattare le richieste dell'utente. Il broker si occupa di contattare un modulo alla volta.
- *Interative query formulation*: si tratta di un modulo di ausilio nella espressione di una query che ha come oggetto lo schema intergrato. In particolare, è di aiuto se si ha già espresso una query che non ha prodotto risultati interessanti.

- *Primitive di costruzione delle configurazioni***:** servono a scegliere quali servizi e quali strumenti possono essere utilizzati per la costruzione di una configurazione e come collegarli fra loro.

Servizi di Integrazione semantica

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

Servizi di Wrapping

I servizi di *wrapping* fungono da interfaccia tra il sistema integratore e le singole sorgenti, in particolare, rendendo omogenee le informazioni. Si comportano come dei traduttori dai sistemi locali ai servizi di alto livello dell'integratore. Il loro obiettivo è, quindi, quello 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.

Servizi Ausiliari

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.

Il mediatore

Il Mediatore è un modulo intermedio che si pone tra l'utente e le sorgenti d'informazione. Secondo al definizione di Wiederhold in [**3**] " un Mediatore è un modulo software che sfrutta la conoscenza su un certo livello superiore. Dovrebbe essere piccolo e semplice, così da poter essere amministrato da uno o al più, da pochi esperti."

I compiti del 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 tre livelli:

1. *utente*: attraverso un'interfaccia grafica l'utente pone delle query su uno schema globale e riceve risposta, come se stesse interrogando un'unica sorgente d'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 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.

Esistono due approcci fondamentali all'architettura precedentemente descritta:

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

- Approccio semantico: è l'approccio utilizzato in MOMIS, ed è caratterizzato dal fatto che 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, e infine, deve essere possibile un'integrazione (parziale o totale) delle sorgenti di dati. In questo modo il Mediatore può individuare i concetti comuni a più sorgenti e relazioni che li legano.

2.1.2 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 sono dovuti 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*

La nostra tesi si concentrerà principalmente su quest'ultima tipologia di problemi.

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. I livelli di ontologia e le problematiche ad esse associate sono le seguenti:

1. *top-level ontology*: descrive concetti molto generali (spazio, tempo, …), che sono quindi indipendenti da un particolare dominio di appartenenza; 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, o quello relativo a un generico obiettivo, 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.

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 diversi sistemi usino esattamente gli stessi vocaboli per rappresentare questi concetti, ne tanto meno 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 [Bates,86] scrive "*the probability of two person using the same term in describing the same thing is less than 20%*".

Qualche esempio: una persona P1 disegna una fonte d'informazione DB1 e un'altra persona P2, disegna la stessa fonte DB2; sarà molto probabile che le due basi di dati presentino diverse semantiche: le coppie sposate potranno essere rappresentate in DB1 usando 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 [**4**], la causa principale delle differenze semantiche, si può identificare nelle diverse concettualizzazioni del mondo esterno che le 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 del mondo in questione; partendo così dalla stessa concettualizzazione, determinate relazioni fra concetti, avranno strutture diverse a seconda che siano state realizzate, ad esempio, attraverso un modello relazionale o un modello ad oggetti.

L'obiettivo dell'integratore, che ricordiamo è di fornire un accesso integrato ad un insieme di sorgenti, si traduce 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:

- *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;
- *Eterogeneità tra le strutture delle classi*: comprendono le differenze nei criteri di specializzazione, nelle strutture per realizzare un'aggregazione, ed anche le discrepanze schematiche;
- *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.

2.2 L'architettura di MOMIS

MOMIS acronimo di ***M****ediator* ***E****nvir****O****ment for* ***M****ultiple* ***I****nformation* ***S****ources*, è il progetto di un sistema I³, ideato per l'integrazione di sorgenti di dati testuali, strutturati e semi-strutturati. 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.
MOMIS è stato progettato per fornire un accesso intergrato ad informazioni eterogenee, memorizzate sia all'interno di un *database* di tipo tradizionale (e.g. relazionali, *object-oriented*) o *file system* sia in sorgenti di tipo semi-strutturato come quelle descritte in XML.
Seguendo l'architettura di riferimento in [**2**] si possono distinguere i componenti disposti su tre livelli (figura 2.2):

- **Livello dati.** Qui si trovano i *Wrapper*. Posti al di sopra di ciascuna sorgente, sono i moduli che rappresentano l'interfaccia fra il Mediatoree le sorgenti di dati locali. La loro funzione è duplice:
  - In fase d'integrazione forniscono la descrizione dell'informazione in essi contenute. Questa descrizione è fornita attraverso il linguaggio ODLI³.
  - In fase di *query processing* traducono la query ricevuta dal Mediatore(espressa quindi nel linguaggio comune d'interrogazione OQLI³, definito a partire dal linguaggio OQL) in un'interrogazione comprensibile dalla sorgente stessa. Devono, inoltre, esportare i dati ricevuti come risposta all'interrogazione,

presentandoli al mediatore, attraverso il modello comune di dati utilizzato dal sistema.

- **Livello Mediatore.** Il Mediatorerappresenta il cuore del sistema ed è essenzialmente composto da due sottomoduli:
  - ***Global Schema Builder*** (GSB): è il modulo che integra gli schemi locali, il quale partendo dalle descrizioni delle sorgenti espresse, attraverso il linguaggio ODLI$_3$ genera un unico schema globale da presentare all'utente. L'interfaccia grafica di GSB, cioè il *tool* d'ausilio al progettista, è *SI-Designer*.
  - ***Query Manager*** (QM)*:* è il modulo di gestione delle interrogazioni. In particolare, genera la *query* in linguaggio ODLI$_3$ da inviare ai *wrapper*, partendo dalla singola *query* formulata dall'utente sullo schema globale. Servendosi delle tecniche di *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 OQLI3).
  - ***SI-Designer****:* èla GUI (Graphic User Interface) che guida l'utente attraverso le varie fasi dell'integrazione, dall'acquisizione delle sorgenti, fino alla messa a punto del Common Thesaurus. SI-Designer risulta a sua volta composto da quattro moduli:
    - SIM (*Source Integrator Module*): estrae le relazioni inter-schema sulla base della struttura delle classi ODLI$_3$ e delle sorgenti relazionali usando ODB-Tools. Inoltre effettua la "validazione semantica" delle relazioni e ne inferisce delle nuove sfruttando sempre ODB-Tools
    - SLIM (S*ource Lexical Intergrator Module*): estre le relazioni inter-schema tra nomi di attributi e classi ODLI$_3$, sfruttando il database lessicale WordNet.
    - TUNIM (*Tuning of the Mapping Table*): questo modulo gestisce la fase di creazione dello schema globale.

  La GUI di SI-Designer, è una sequenza di finestre, ognuna delle quali relativa ad una fase del processo d'integrazione, e mette a disposizione l'interfaccia per interagire con i moduli SIM, SLIM ed ARTEMIS.
- **Livello Utente.** Il progettista interagisce con il 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 2.2 compaiono inoltre, altri tre tool che accompagnano il Mediatore nella fase di integrazione e sono:

- *ODB-Tolls Engine:* un tool basato sulle *Description Logics* [**5, 6**] che compie la validazione di schemi e l'ottimizzazione di query [**7, 8, 9**].
- *ARTEMIS-Tool Enviroment*: tool basato sulle tecniche di *clustering affinity-based* che compie l'analisi ed il clustering delle classi ODLI3 [**10**].
- *WordNet*: un database lessicale ampiamente descritto nel capitolo 1.

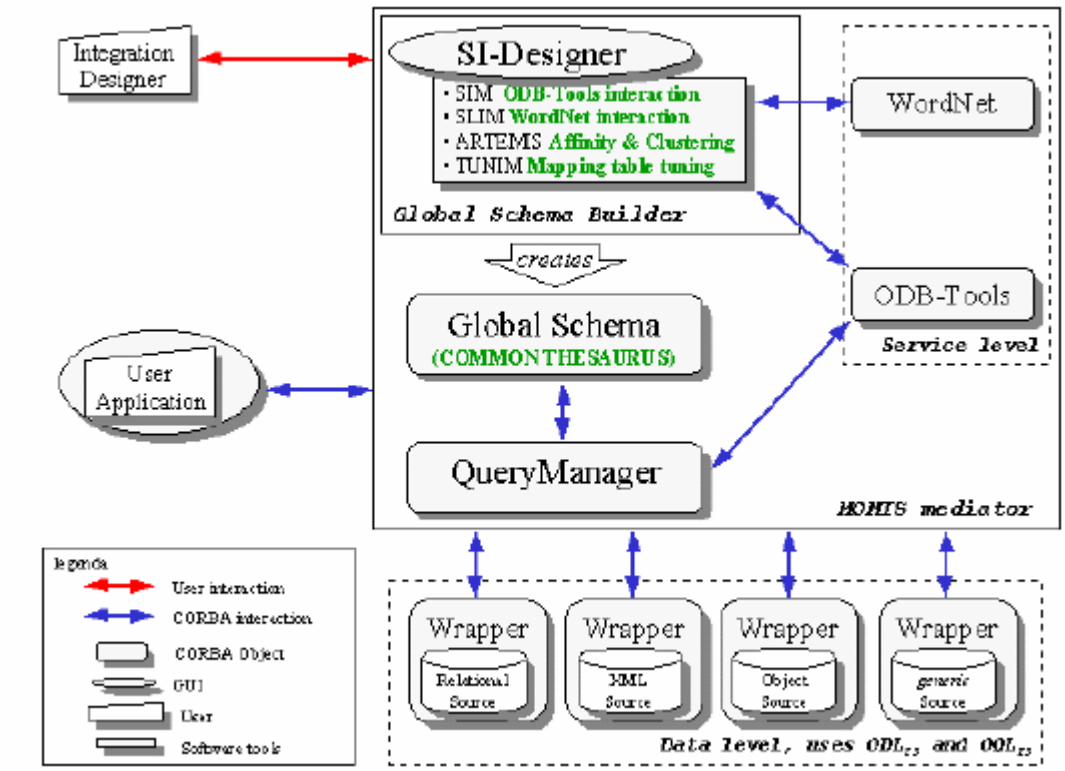


Figura 2.2- Architettura generale del sistema MOMIS

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

2.2.1 Il processo di Integrazione

L'integrazione delle sorgenti informative strutturate e semi-strutturate, è compiuta in modo semi-automatico, utilizzando degli schemi locali in linguaggio ODLI3, e combinando tecniche di *Description Logic* e di *clustering*. Come mostrato in figura 2.3, le attività compiute sono le seguenti:

1. *Generazione del Thesaurus Comune,* con il supporto di ODB-Tool e di WordNet. In questa fase è identificato un Thesaurus comune di relazioni terminologiche. Tali relazioni, esprimono la conoscenza inter-schema su sorgenti diverse e corrispondono alle asserzioni intenzionali utilizzate in [**11**]. Le relazioni terminologiche, sono derivate in modo semi-automatico a partire dalle descrizioni degli schemi in ODLI3, 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 ODLI3* con il supporto dell'ambiente ARTEMIS-Tool. Le relazioni terminologiche contenute nel Thesaurus, sono utilizzate per valutare il livello di affinità tra le classi ODLI3 in modo da identificare le informazioni che devono essere integrate a livello globale. A tal fine ARTEMIS, calcola i coefficienti che misurano il livello di affinità tra le classi, basandosi sia sui nomi delle stesse sia sugli attributi. Le classi con maggiore affinità sono raggruppate utilizzando tecniche di clustering [**12**].
3. *Costruzione dello Schema Globale*. I cluster delle classi ODLI3 affini, sono analizzati per costruire lo schema globale del Mediatore. Per ciascun cluster si definisce una classe globale che rappresenta tutte le classi locali riferite al cluster ed è caratterizzata dall'unione ragionata dei loro attributi e da una *mapping-table*. L'insieme delle classi globali definite, costituisce lo schema globale del Mediatore che sarà usato per porre le query alla sorgenti locali intergrate.

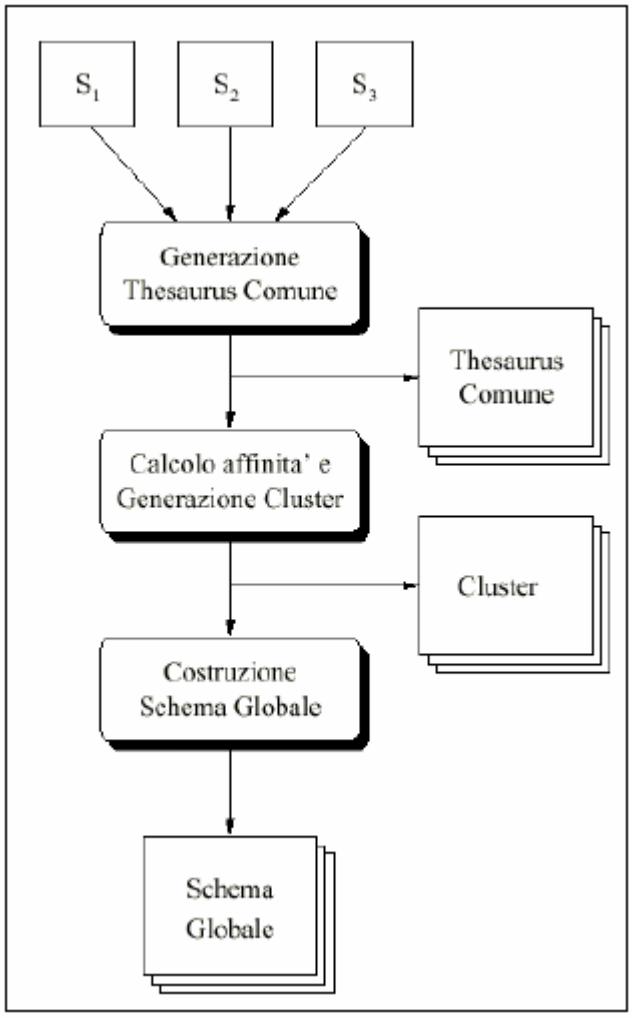


Figura 2.3- Fasi del processo d'integrazione.

2.2.2 Query Processing e Ottimizzazione

Quando un'utente pone una query sullo schema globale, MOMIS la analizza e produce un insieme di sotto-query che saranno inviate a ciascuna sorgente informativa coinvolta. Il processo consiste di due attività principali:

1. *Ottimizzazione Semantica.* L'ottimizzazione semantica è basata sull'inferenza logica a partire dalla conoscenza contenuta nei vincoli d'integrità dello schema globale. La stessa procedura di ottimizzazione semantica, si realizza in termini locali, su ogni sotto-query tradotta dal Mediatore nella formulazione del piano d'accesso: in tal caso ci si basa sui vincoli d'integrità presenti sui singoli schemi locali.

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

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

2.2.3 Il linguaggio ODLI3

Il linguaggio ODL (Object Definition Language) per la specifica di schemi ad oggetti proposto dal gruppo di standardizzazione ODGM-93 [**13**] è universalmente riconosciuto come standard. Le sue caratteristiche peculiari, al pari di altri linguaggi basati i sul paradigma ad oggetti, possono essere così riassunte:

- Definizioni di tipi-classe e tipi valore;
- Definizione fra intenzione ed estensione di una classe di oggetti.
- Definizione di attributi semplici e complessi.
- Definizione di attributi atomici e collezioni.
- Definizione di relazioni binarie con relazioni inverse.
- Dichiarazione delle segnature dei metodi.

Con l'estensione di ODL al linguaggio ODLI3 sono stati raggiunti i seguenti obiettivi:

- Per ogni classe il wrapper, può indicare nome e tipo di 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 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 semi-strutturati.
- Il linguaggio supporta la definizione di grandezze locali e di grandezze globali.

- Il linguaggio supporta la dichiarazione di regole di *mapping* fra grandezze locali e di grandezze globali.
- È data la possibilità di definire regole di integrità sia sugli schemi locali che globali;
- Il linguaggio supporta la definizione di relazioni terminologiche di sinonimia, ipernimia, iponimia, e associazione;
- 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 WordNet Editor

Come abbiamo descritto in precedenza, uno delle problematiche alla base del processo di integrazione delle informazioni, risiede nell'impossibilità di conoscere l'esatto significato dei veri termini contenuti all'interno di una risorsa dati. In MOMIS si adotta, come risorsa lessicale e semantica d'informazione, il database WordNet descritto nel capitolo 1. La scelta di utilizzare Wordnet come risorsa lessicale, va ricercata nel fatto che quest'ultimo è un database lessicale molto conosciuto, tra i più completi e professionali, ed è liberamente disponibile.

Il database WordNet, per poter essere utilizzato all'interno del sistema MOMIS, è stato analizzato, e il suo contenuto informativo è stato riportato all'interno di un database relazionale indicato con "*momiswn*". Tale database contiene al suo interno un insieme di tabelle che consento di accedere velocemente alle informazioni di WordNet.

Tali tabelle, sono mostrate in figura 2.4, e contengono le seguenti informazioni:

- ***wn_synset***: contiene essenzialmente le glosse dei vari synset; tali glosse sono associate ai rispettivi synset attraverso i campi *byte_offset* e *syntactic_category* che consentono di identificare un determinato synset in base alla notazione utilizzata in WN.
- ***wn_relationship_type****:* contiene i tipi di relazioni previsti nella versione di iniziale (1.6) di WN.
- ***wn_relationship_type_new:*** contiene li nuovi tipi di relazioni inserite all'interno della versione 2.0 di WN, comprensive di quelle della versione precedente.
- ***wn_relationship****:* contiene tutte le relazioni fra i synset di WN.

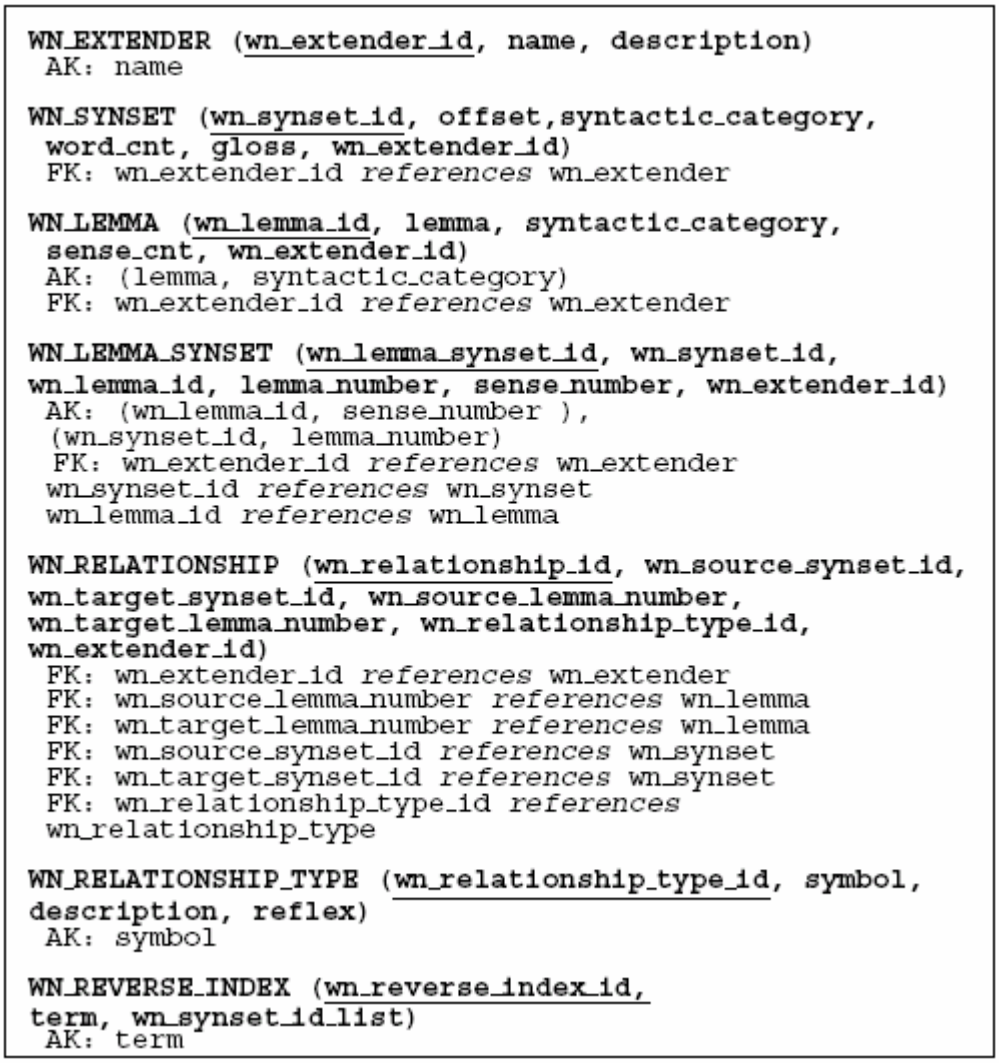


Figura 2.4- La struttura informativa di WordNet all'interno di un DBMS (DataBase Management System) relazionale.

- ***wn_lemma_synset:*** associa ciascun lemma, ai relativi possibili synset.
- ***wn_lemma:*** contiene tutti i lemmi presenti in WN e per ciascuno indica la categoria sintattica di appartenenza.

L'interfaccia MOMIS-WordNet, implementa una tecnica che consente di determinare le relazioni inter-schema. L'obiettivo è scoprire le affinità tra le classi appartenenti a differenti sorgenti di dati.

Avendo a che fare con risorse differenti, è necessario tradurre tutti gli attributi e le classi di tali sorgenti, in linguaggio comune. Tale obiettivo si realizza associando ad ogni termine il

suo senso corretto corrispondente in WordNet. D'ora in poi ci si riferirà a questo processo di *mapping*, come alla *fase di annotazione delle sorgenti di dati.*
L'interfaccia menzionata in precedenza, fornisce l'interazione con il database di WordNet, e risulta integrata all'interno del modulo SLIM del SI-Designer GUI.
Le relazioni fra i termini estratte da WordNet, sono poi sommate al Common Thesaurus.

Durante la fase di annotazione dello schema, al *designer* (utente che interagisce con l'interfaccia MOMIS-WordNet) è richiesto, essenzialmente, di scegliere manualmente il senso corretto di WordNet per ogni elemento dello schema. Nel far ciò, si dovrà, ovviamente, considerare il particolare contesto dettato dagli elementi dello schema da integrare.
La fase di scelta di uno o più significati da attribuire ad ogni termine, viene eseguita in due passi:

- *Scelta del termine*: durante questa prima fase il *WordNet morphologic processor* viene in aiuto al designer realizzando lo *stem* dei termini originali e ottenendo la *word form.* Successivamente la *word form* viene automaticamente cercata in WordNet. Nel caso in cui non sia presente, il designer può inserirla manualmente, senza però che modificare il database.
- *Scelta del significato*: il designer può scegliere di associare a ciascun elemento, zero, uno o più synset.

Una delle limitazioni principali di tale fase di annotazione, oltre ad essere completamente manuale, è quella di non consentire al disegner di modificare il database inserendo nuovi lemmi, synset o relazioni.
Nel 2002 Veronica Guidetti in [**15**], integra all'interno di WordNet il componente *WordNet Editor*, il quale essenzialmente rappresenta una GUI, che sfrutta una libreria Java e rende possibile l'estensione del database di WordNet all'interno di MOMIS. Con "estensione di WordNet", in questo caso, ci riferisce alla possibilità e alla necessità di poter sfruttare nuovi concetti non presenti in WordNet. WordNet infatti, pur essendo popolare e ampiamente utilizzato, presenta alcune limitazioni che saranno descritte in maniera più approfondita nel capitolo 4. In generale, con il termine "estensione di WordNet", si intenderà un processo di arricchimento e/o completamento, delle informazioni, sia semantiche che strutturali, contenute in WordNet.

Con WordNet Editor, si fornisce la possibilità al designer di poter estendere WordNet allo scopo di colmare le sue lacune. Attraverso tale editor, è possibile, infatti, inserire nuovi termini, concetti, o relazioni fra synset nuovi o già esistenti. Per consentire ciò, il database lessicale di MOMIS, è stato modificato aggiungendo la tabella WN_EXTENDER (figura 2.4), la quale tiene traccia delle informazioni riguardanti le modifiche effettuate sul database originale.

Nella realizzazione del WordNet Editor, si è tenuto conto della criticità del processo di estensione, dovuto alla complessità dell'ontologia lessicale. Ciò ha comportato la necessità di consentire al designer, di eseguire le operazioni di estensione passo dopo passo. Ogni relazione di WordNet, come si evidenzia dallo schema del DBMS in figura 2.4, occorre tra un *source synset*, ed un *source target*. Dato un nuovo concetto *X*, questo andrà fissato come *source synset*, e successivamente il designer dovrà essere supportato nel processo di ricerca del target più appropriato fra i vari synset di WordNet che mostrano qualche similarità con *X*.

Sotto l'assunzione che "*definizioni in linguaggio naturale, simili tra loro dovrebbero fornire l'evidenza della similarità tra i concetti che descrivono*", il synset target candidato, può essere individuato applicando, alle glosse dei vari synset, l'euristica conosciuta in letteratura come *Definition Match* [**16**].

La filosofia del WordNet Editor descritta in [**14**], si basa sulla consapevolezza che il designer conosce l'organizzazione del synset nel database lessicale di WordNet così com'è mostrata in figura 2.4. Per comprendere meglio il funzionamento del WordNet Editor riportiamo il seguente esempio: supponiamo che il designer crei un nuovo synset per il nome "*tirocinium*" (figura 2.6), attraverso l'introduzione della glossa "*the period when a student gets practice and learns about a field or activity*". Successivamente al designer, attraverso il *Synset Relationship Editor*, viene chiesto di inserire alcuni termini che il candidato termine target potrebbe contenere all'interno della sua glossa.

Nell'esempio in figura 2.6, si vuole mettere in relazione "*tirocinium*" con i synset già esistente di *learning* e *practice*. Un'altra soluzione possibile, è quella di creare la relazione con il termine esistente *apprenticeship#1: the position of apprentice.*

Nel WordNet Editor è stato implementato un algoritmo di ricerca di similarità detto *approximate string match*, allo scopo di agevolare il processo di connessione del nuovo termine o significato, con uno già esistente.

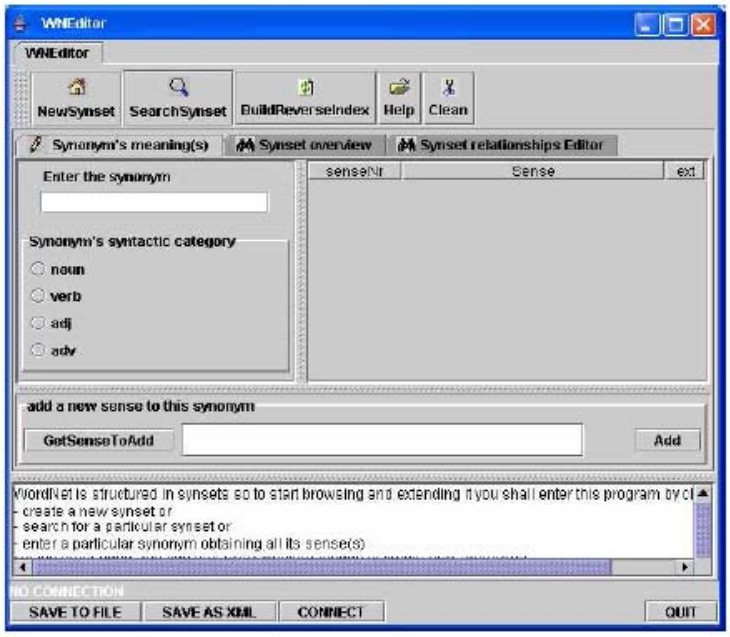


Figura 2.5-La maschera del WordNet Editor

La tabella WN_REVERSE_INDEX è stata creata come supporto a tale algoritmo. Ogni entry di questa tabella è composta da un termine e da una lista di synset contenenti all'interno della loro glossa tale termine. Ovviamente ogni volta che un viene cancellato o introdotto un nuovo synset, tale tabella viene automaticamente aggiornata.

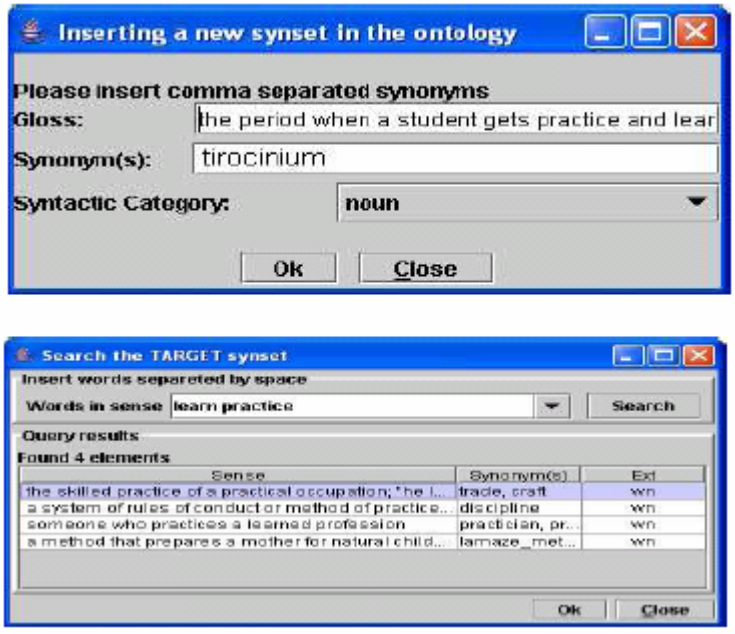


Figura 2.6-Creazione di un nuovo synset con il WordNet Editor

Capitolo 3

3 Metodi e Algoritmi di Disambiguazione del Testo

La disambiguazione del testo è il processo di assegnamento del significato ad un termine definito target. Tale processo si basa principalmente sull'estrarre informazioni dal contesto linguistico all'interno del quale è inserito il termine target. Nella maggior parte dei casi, tale problema si riduce essenzialmente, nel selezionare il senso corretto tra un insieme di sensi possibili, estratti da un dizionario o da un database lessicale come per esempio WordNet.
In realtà, esiste un'ampia varietà di algoritmi e metodologie per la disambiguazione del testo.
Lo scopo di questo capitolo, è quello di classificare e descrivere in maniera completa, le principali metodologie proposte ed applicate in tale ambito.
Il criterio di classificazione dei processi di disambiguazione del testo, non è univoco. Due approcci, infatti, possono differire tra loro per vari aspetti:

- le categorie sintattiche che consentono di disambiguare: per esempio esistono algoritmi che consentono di disambiguare solo nomi, e viceversa altri che possono essere applicati a tutte le categorie sintattiche;
- la struttura dati all'interno della quale memorizzano le informazioni necessarie al processo di disambiguazione: tale struttura può essere, un vettore, un grafo ecc…
- il livello a cui agisce la disambiguazione: si possono averi processi di *disambiguazione globale* o processi di *disambiguazione locale*. La prima si ha nel

caso in cui la metodologia utilizzata consenta di disambiguare contemporaneamente, tutte le parole di un testo o di una porzione di testo; viceversa, il secondo caso si ha quando l'algoritmo disambigua un termine, indipendentemente dai sensi attribuiti alle altre parole del contesto;

- in base alla tipologia ed al numero di relazioni fra i sensi considerate: meronimia, iponimia, ecc…
- l'esigenza di essere supervisionati o meno dall'utente; si distinguono processi di disambiguazione:
  a. Supervisionati: sono approcci che necessitano dell'uso di collezioni di testi etichettati manualmente*;* vengono detti "supervisionati" poiché richiedono l'intervento di una persona che sia in grado di identificare le parole all'interno di una frase come pertinenti ad un senso piuttosto che ad un altro. Tali distinzione diventa fondamentale, poiché lo sforzo richiesto per produrre dei corpus con i sensi etichettati, e sicuramente notevole.
  b. Non Supervisionati: questa tipologia di algoritmi non richiede alcuna supervisione da parte dell'utente. In genere si basa sull'utilizzo di ontologie o dizionari come risorsa lessicale di riferimento.
- l'elemento sul quale viene calcolata la similarità; si distinguono processi di disambiguazione:
  a. *Token-based*: hanno lo scopo di calcolare la similarità o la relazione tra il termine target che deve essere disambiguato e il suo contesto;
  b. *Type_based:* semplicemente assegnano a tutte le istanze di un termine ambiguo, il suo senso più frequente; il senso predominante, è acquisito automaticamente da testi raw senza alcun ricorso all'annotazione manuale dei dati.
- la risorsa di conoscenza utilizzata; in questo caso si distinguono i seguenti algoritmi:
  a. Basati su collezioni di testi annotati manualmente (*hand-tagged corpora*): utilizzano come risorsa frasi all'interno di testi dove sono già stati disambiguati manualmente i sensi dei termini.
  b. Basati su Machine Readable Dictionary (MRD): utilizzano le informazioni contenute all'interno dei dizionari.
  c. Basati su Ontologie: utilizzano la conoscenza contenuta all'interno di un ontologia.

d. Altri approcci: solitamente si ottengono a partire da due o più combinazioni degli approcci precedentemente descritti, o su altri generi di risorse meno sfruttate.

In questa tesi, si è scelto di classificare i vari metodi, in base al criterio fondamentale che li distingue in algoritmi supervisionati ed algoritmi non supervisionati.

Questo capitolo, mira a fornire una visione completa delle varie tecniche proposte, sia per algoritmi supervisionati che per algoritmi non supervisionati, allo scopo di delineare chiaramente quali siano le strade percorribili, nel caso in cui si debba affrontare il problema della disambiguazione del testo. In generale si tenderà a descrivere in maniera più concisa gli approcci di base, mentre ci si concentrerà maggiormente, sulle evoluzioni e combinazioni di quest'ultimi presentate negli ultimi anni.

Durante il seguente capitolo si troveranno spesso riferimenti al Senseval. Prima di iniziare la nostra descrizione, vediamo di chiarire cosa sia: Senseval è un'organizzazione internazionale dedicata alla valutazione dei sistemi di disambiguazione dei sensi delle parole. La sua missione, è quella di fornire i test di riferimento e gli strumenti per la valutazione dei sistemi di disambiguazione, ciò consente di verificare i punti di forza e le debolezze dei sistemi di Word Sense Disambiguation (WSD). Inoltre come obiettivo di fondo vi è la comprensione della semantica del lessico polisemico. Senseval si realizza grazie ad un piccolo comitato a cui fa capo ACL-SIGLEX ovvero il gruppo speciale di interesse comune sul lessico dell'associazione per la linguistica computazionale, la quale fornisce supporto per la ricerca sul lessico. Il successo di tutto il progetto di WSD, è chiaramente legato alla valutazione dei sistemi di disambiguazione. Senseval è nato nel 1997, e fin'ora si sono eseguiti 3 congressi rispettivamente nel '98, '01, e nel '04. La quarta edizione del Senseval è attualmente in corso.

Poiché, in realtà ad oggi non esiste ancora un metodo che consenta di disambiguare in maniera completamente corretta un insieme di termini, durante tale capitolo si farà riferimento alle annotazioni ottenute tramite i vari algoritmi, come ai sensi "più probabili" da attribuire alle varie parole.

In particolare il capitolo è così organizzato: il primo paragrafo tratta gli algoritmi che non richiedono supervisione, distinguendoli in base alla metodologia applicata; il secondo paragrafo descrive, viceversa, i principali algoritmi di disambiguazione supervisionati; infine

l’ultimo paragrafo presenta alcuni algoritmi di disambiguazione composti, ovvero che combinano due o più tecniche di disambiguazione (supervisionata o non).

3.1 Algoritmi non supervisionati

3.1.1 Le catene lessicali

Tra i vari ambiti dove si è tentato di dare una soluzione al problema della disambiguazione del testo, risulta particolarmente importante a mio avviso ricordare quello delle catene lessicali.
La realizzazione delle catene lessicali, è il processo di connessione semantica dei termini in relazione fra loro, il cui risultato è un insieme di catene che rappresentano differenti processi di coesione attraverso il testo. In particolare tale approccio è applicabile solo sui nomi.
Lo studio sulle catene lessicali nasce dalla considerazione che all’interno di un testo scritto sia possibile individuare una proprietà ben definita: la coesione.
Hansan e Halliday nel 1976, descrivono alcuni aspetti della lingua inglese, osservando che esiste un’importante differenza tra un insieme di frasi casuali, e un testo unico.
Un insieme di frasi, infatti, può essere definito testo solo quando il lettore, considerando le frasi nella loro successione, non solo ne comprende il significato, ma si rende anche conto di come queste siano più o meno dipendenti l’una dalle altre. Tale proprietà è definita da Hansan e Halliday, con il termine di *coesione,* e sta ad indicare l’insieme di *feature* grammaticali e lessicali, grazie alle quali una frase all’interno di un testo, viene collegata alle precedenti e alle successive. Tale collegamento si realizza attraverso dei concetti espressi tramite dei termini. Halliday e Hansan suggerirono in [**18**] che un testo caratterizzato da questa proprietà è rappresentabile attraverso un insieme di catene coesive. Ogni termine appartenente a tale catena è relazionato ai suoi predecessori, o ai suoi successori, tramite una relazione di coesione. Per esempio, nel testo riportato di seguito, i termini in corsivo, formano una catena coesiva.

> The major potential complication of total joint replacement is *infection*. *It* may occur just in the area of the wound or deep around the prosthesis. *It* may occur during the hospital stay or after the patient goes home. … *Infections* in the wound area are generally treated with antibiotics.

Uno degli aspetti fondamentali alla base del funzionamento delle catene lessicali, è rappresentato dal problema della disambiguazione dei termini del testo.
Nel calcolo delle catene, le istanze dei termini devono essere raggruppate in base alle relazion che li legano, ma ogni istanza deve appartenere esattamente ad un'unica catena lessicale. Esistono notevoli difficoltà nel determinare a quale catena appartenga una particolare istanza. In particolare, un termine può essere associato a più significati e, un corretto algoritmo di realizzazione di catene lessicali, deve essere in grado di individuare i sensi corretti per ogni istanza del termine. Come vedremo, nel corso degli anni, sono stati proposti diversi algoritmi per la creazione di catene lessicali più o meno efficaci ed efficienti. I primi algoritmi proposti, sono caratterizzati dal fatto di non separare la fase di disambiguazione dei termini da quella di costruzione delle catene lessicali. Solo negli ultimi anni, è stata individuata l'importanza di mantenere separate le due fasi. Ciò, consente una migliore individuazione, ed eventuale risoluzione, delle problematiche. dell'algoritmo, che in tal modo possono essere chiaramente attribuite o alla fase di WSD o a quella di creazione delle catene lessicali.
Tuttavia, una trattazione dettagliata delle ***feature*** delle catene lessicali esula degli scopi della nostra tesi. La trattazione si concentrerà in particolar modo sugli aspetti legati propriamente alla disambiguazione dei termini

Algoritmo di Morris e Hirst.

I primi a cogliere le potenzialità dell'uso delle catene lessicali furono Morris e Hirst, che in [**17**], utilizzarono le proprietà coesive di un testo per determinare i significati in esso contenuti. Se le catene lessicali rappresentano le unità concettuali che contribuiscono ad esprimere il significato di un testo, allora esse possono essere utilizzate per risolvere le ambiguità legate al significato dei termini.
Il contributo fondamentale di Morris e Hirst allo sviluppo della teoria delle catene lessicali, consiste nell'introduzione di un Thesaurus come base di conoscenza per estrarre le relazioni tra i termini.
L'algoritmo proposto si compone di due fasi fondamentali, la prima delle quali è dedicata all'individuazione delle parole candidate da includere all'interno delle catene, e quindi da disambiguare. Nella seconda fase invece, vengono determinate le relazioni tra i termini al fine di costruire le catene lessicali. Morris e Hirst considerarono solo cinque tipi di relazioni scelte

sulle basi dell'organizzazione del Thesaurus utilizzato, ovvero il *Roget's International Thesaurus*. Il codice dell'algoritmo è brevemente descritto di seguito:

```
Repeat
        READ next word
        If word is suitable for lexical analysis then
           CHECK for chains within a suitable span
                CHECK thesaurus for relationships
                CHECK other knowledge sources if available
           If  chain relationship is found then
                INCLUDE word in chain
                CALCULATE chain so far
           End if
           If  there are words that have not formed a
           chain for a suitable number of sentences then
                ELIMINATE words from the span
           End if

           CHECK new word for relevance to existing chains
           that are suitable for checking
           ELIMINATE chains that are not suitable for checking.
        End if
End Repeat
```

L'algoritmo in realtà non fu mai implementato non essendo allora ancora disponibile una versione on line del Roget's Thesaurus.

Con la realizzazione di WordNet, molti ricercatori hanno riconsiderato l'algoritmo di Morris e Hirst e attraverso alcune modifiche, sono stati implementati, differenti algoritmi per la realizzazione delle catene lessicali, che utilizzano come sorgente di relazioni WordNet.

I paragrafi successivi hanno lo scopo di illustrare le evoluzioni più significative di questo algoritmo.

Algoritmo di G.Hirst e D. StOnge

Tra i primi a proporre una variante dell'algoritmo di Morris e Hirts per la realizzazione delle catene lessicali troviamo G.Hirst e D.StOnge che in [**19**], ripresero l'algoritmo di Morris e ne apportarono alcune modifiche. L'algoritmo da loro proposto utilizza come risorsa lessicale WordNet, e per questo essi ridefinirono il concetto di relazione semantica centrandolo sui vari synset di WordNet. A differenza dell'algoritmo di Morris, le relazioni sono individuate non fra termini, ma bensì tra i differenti synset appartenenti a tali termini. Essi definiscono tre differenti tipi di relazioni:

1. **Extra-strong Relation**: si verifica quando una parola viene ripetuta più volte, evento che è indice dell'importanza del termine all'interno del testo;
2. **Strong Relation**: essenzialmente sono dovute a tre fenomeni diversi. Si ha una relazione forte del primo tipo (figura 3.1) quanto due parole condividono un synset, cioè quando hanno, fra tutti i possibili synset, un synset in comune. Una relazione del secondo tipo (figura 3.2) è individuabile quando esiste un link orizzontale fra un synset di una parola e un synset di un altro termine. Tale link può essere ad esempio di antinomia. Infine il terzo tipo (figura 3.3) riguarda l'eventualità in cui non esiste un collegamento di alcun genere fra due synset se una parola è una combinazione dell'altro termine. E' un caso abbastanza frequente nella lingua inglese, in cui una parola viene scritta come se si trattasse di un solo termine ma in realtà è formata da più elementiseparati da un trattino (es : brown-eyed…).
3. **Medium-Strong Relation**: si verificano quando esiste un percorso che college due synset. Un percorso è una sequenza di un minimo di due fino ad un massimo di cinque link fra i synset.

Il peso associato a ciascun percorso varia in base alla sua lunghezza e al numero dei "cambi di direzione" secondo la formula:

$$weight = C - PathLenght - K * NumChangeOfDirection$$

dove C e K sono due parametri costanti.

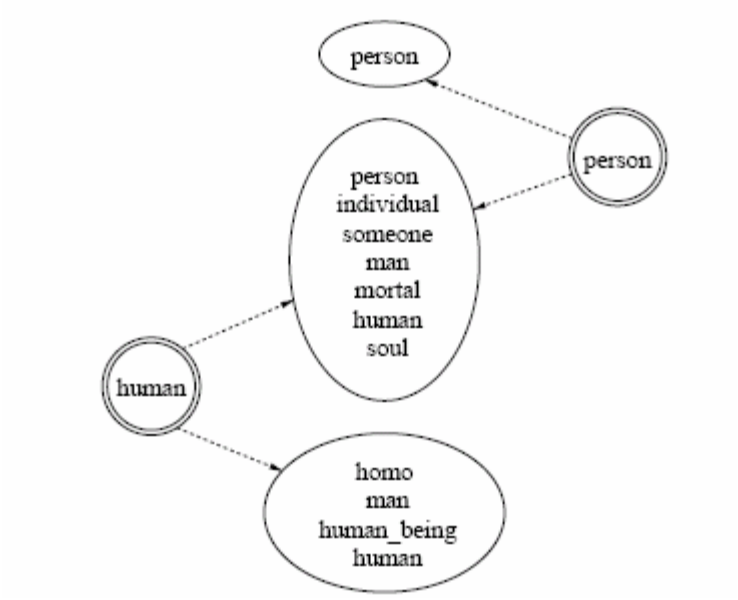


Figura 3.1-Strong Relation: synset in comune

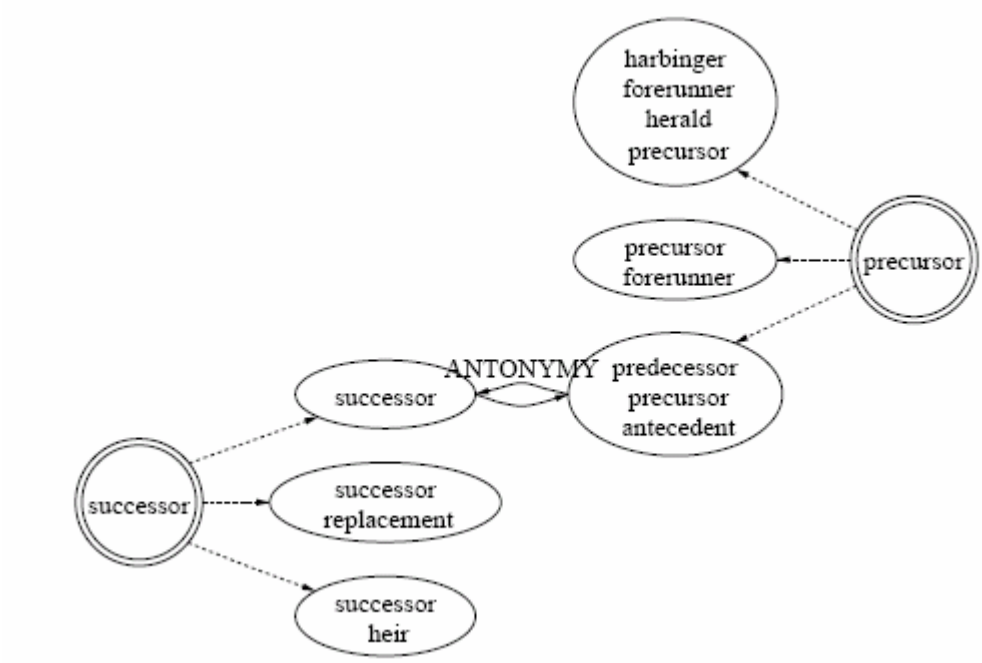


Figura 3.2-Strong Relation: link orizzontale

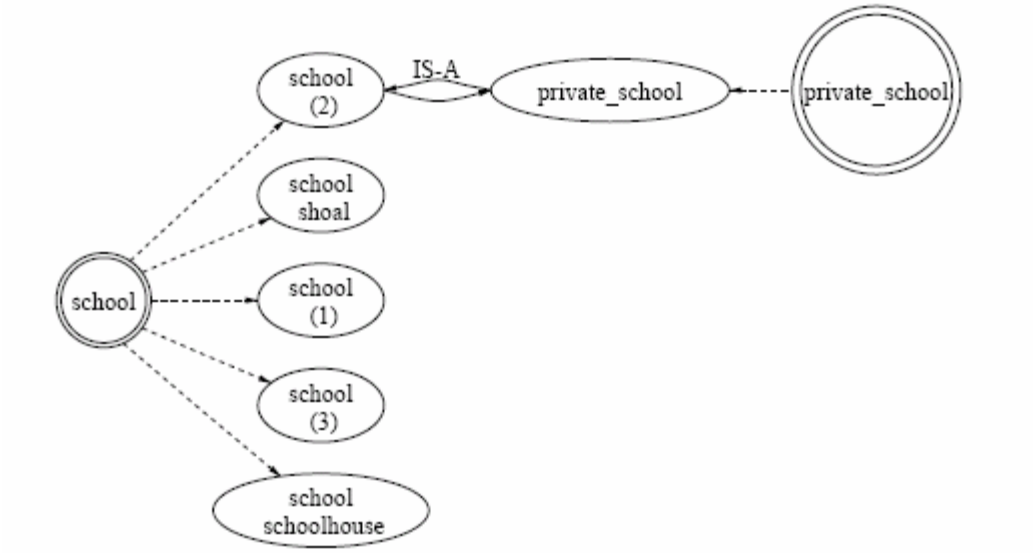


Figura 3.3-Strong Relation: composizione di termini

Ciò implica che, più lungo è il percorso e maggiore è il numero di cambiamenti di direzione, minore sarà il peso ad esso associato. Hirst e St-Onge affermano che se esiste un percorso multi-link fra due synset, allora esiste una prossimità o vicinanza semantica fra loro, per cui la semantica di ogni relazione lessicale deve essere presa in considerazione. In particolare, una direzione dal basso verso l'alto corrisponde ad una generalizzazione.
Consideriamo ora un esempio di costruzione di una catena lessicale che contenga i termini economy, sector, economic_system (figura 3.4).

Per ogni elemento da inserire viene creato un record che viene aggiunto ad una lista costruita dinamicamente. Il primo termine (*economy*) è il primo elemento della lista. Il secondo termine viene aggiunto in testa alla lista, ed il record creato per memorizzare il lemma, contiene anche un puntatore verso l'elemento con cui è posto in relazione. Infine, il termine *economic_system* viene riferito ad *economy*. L'ordine con cui i termini compaiono all'interno della lista non è significativo delle relazioni lessicali: esse possono essere ricostruite grazie ai puntatori.
Un lemma, però, può essere incluso in più synset, per cui la gestione dei puntatori deve essere accurata. L'inserimento di un secondo termine grazie ad una relazione extra-strong comporta il collegamento fra tutti i synset attivi; quando la relazione è forte si connettono tutte le coppie dei relativi synset; infine, quando la relazione è medio-forte, la coppia o le coppie di synset il cui peso è maggiore fra tutte le coppie, vengono connessi.

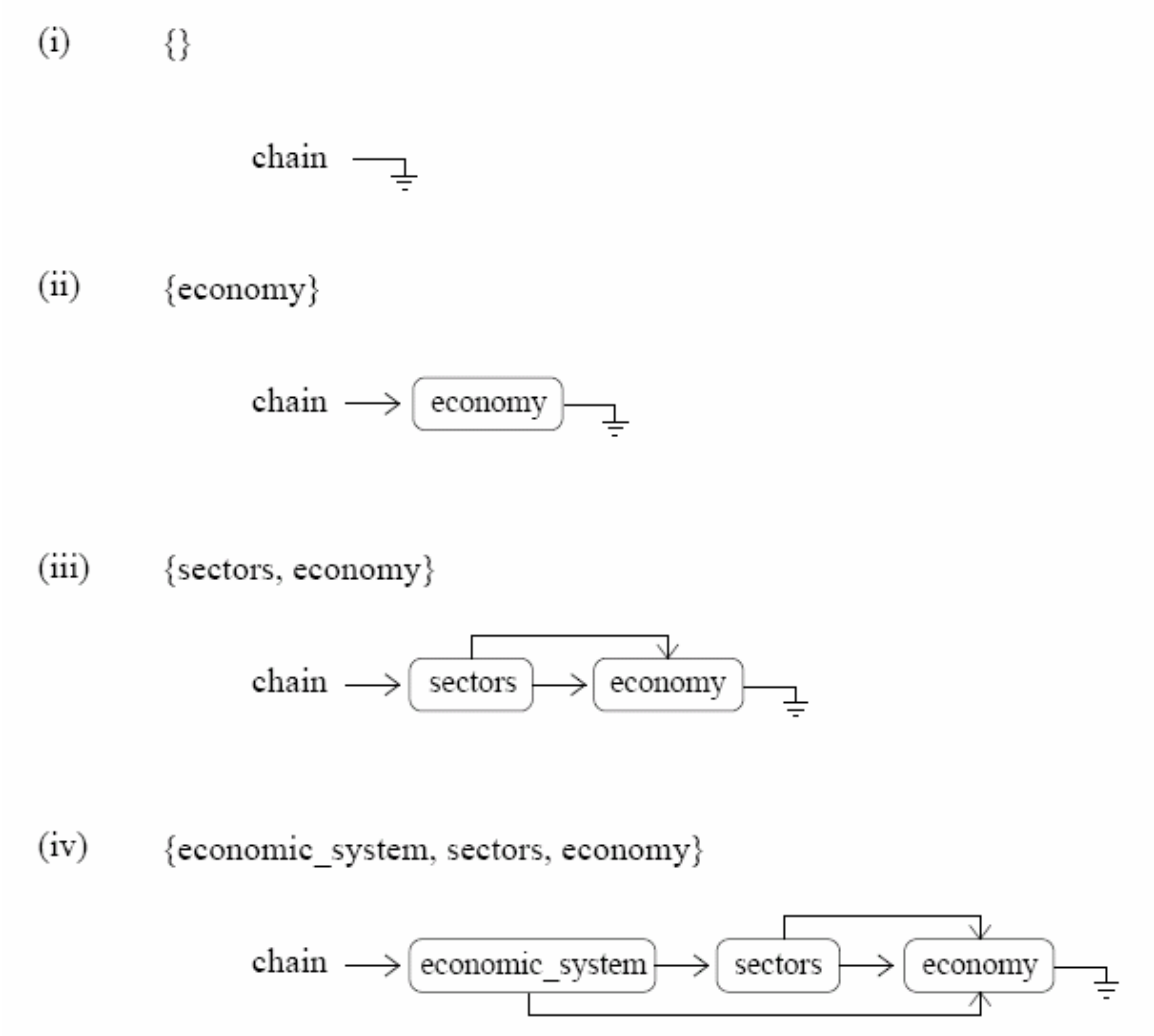


Figura 3.4-Esempio di costruzione di una catena lessicale

Dopo aver connesso le parole,ogni synset non collegato della nuova parola è cancellato, e la catena è scandita per rimuovere, dove possibile, ogni altro synset. La rimozione dei synset permette di determinare progressivamente il senso corretto di ciascun termine. E' quindi in questa fase che si conclude il vero processo di disambiguazione dei termini del testo.

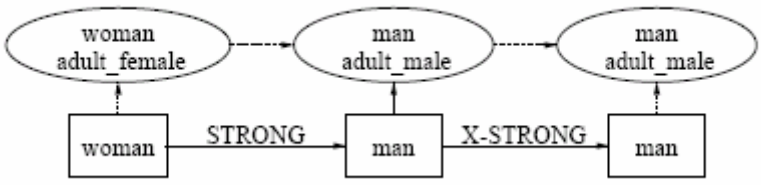


Figura 3.5-Risultato finale dell'esempio sull'algoritmo di Hirst e StOnge

Algoritmo di R.Barzilay e M. Elhadad

Nel 1997 due ricercatori israeliani, R.Barzilay e M. Elhadad, in [**20, 21**], proposero una variante dell'algoritmo delle catene lessicali, applicandolo alla problematica della *Text Summarization.*

L'algoritmo proposto si differenzia dai precedenti in quanto applica una strategia meno "avida"(nel senso che non disambigua i termini al primo passaggio) per il processo di disambiguazione dei termini. Anche in questo caso tuttavia, il processo di disabiguazione viene eseguito in maniera parallela a quello della creazione catene lessicali, e quindi i termini disambiguati si ottengono solo al termine dell'intero algoritmo.

Di seguito si descriverà brevemente l'algoritmo attraverso un esempio.

Consideriamo il seguente testo:

Mr. Kenny is the person that invented an anesthetic machine which uses micro-computers to control the rate at which an anesthetic is pumped into blood. Such machines are nothing new. But his device uses two micro-computer to achieve much closer monitoring of the pump feeding the anesthetic into patient.

L'algoritmo di Hirst e St-Onge, in questo caso, prevedrebbe la creazione di una nuova catena per la parola Mr.. Il termine *Mr.* è associato ad un solo synset, ovvero è un termine monosemico, e quindi è già disambiguato. La parola *person* è in relazione con questa catena quando assume il significato di "a human being". Il secondo lemma viene pertanto aggiunto ad una catena grazie ad una relazione medium-strong. La terza parola *machine*, ha anche il significato di "efficient person", che essendo in relazione di omonimia con *person*, può essere aggiunta alla catena. Fin'ora quindi la disambiguazione di *machine* non è ancora stata realizzata in maniera corretta, in quanto tale termine dovrebbe essere associato al significato di *device*. Pertanto, la disambiguazione non deve avvenire al primo passaggio dell'algoritmo e la decisione su quali synset rimuovere, deve essere presa in un momento successivo. L'algoritmo proposto da Barzilay e Elhadad elabora il testo in esame nel seguente modo:

1. Si crea il primo nodo contenente il termine *Mr*.;
2. Poichè il termine person ha due significati, la scelta sul significato viene rimandata, viene creata una nuova catena (s*plitting*), così come mostrato in figura 3.6. In particolare si definisce *componente* una lista di possibili interpretazioni mutuamente

esclusive fra loro. Le parole contenute nei componenti si influenzano l'una con l'altra, durante la determinazione dei significati.

3. Il termine *anesthetic*, non essendo in relazione con nessuno dei componenti precedenti, richiede la creazione di una nuova catena.
4. Il termine *machine* è collegato in WordNet a cinque sensi differenti. Nel caso in cui assuma il significato di "efficient person" è in relazione con la parola *person* appartenente alla prima catena, e di conseguenza viene inserita come in figura 3.7. I rimanenti significati, vengono memorizzati all'interno della seconda catena, senza essere quindi eliminati (figura 3.8).
5. L'inserimento dei termini *micro-computer*, *device e pump*, provoca un incremento elevato del numero di possibili alternative. Ipotizzando che il testo goda della proprietà di coesione, viene scelta la migliore fra tutte le possibili interpretazioni. Per migliore interpretazione Barzilay e Elhadad, intendono la catena che possiede il punteggio maggiore calcolato in termini di numero e tipo di relazioni ricostruite al suo interno. Le catene migliori, quindi, sono quelle mostrate.

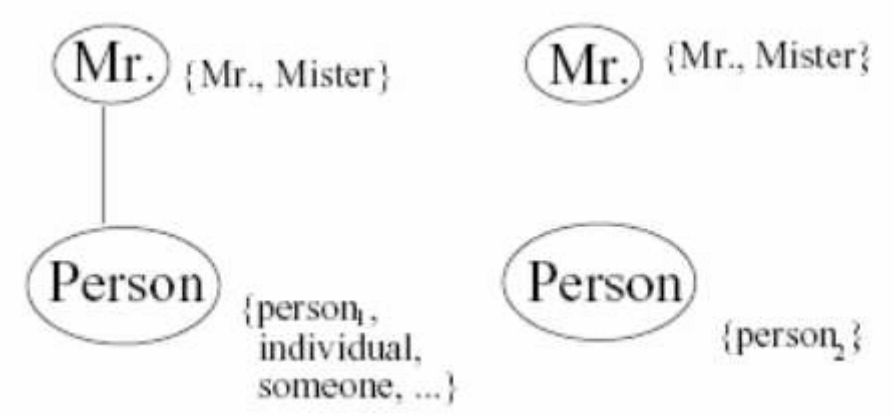


Figura 3.6-Split di una catena

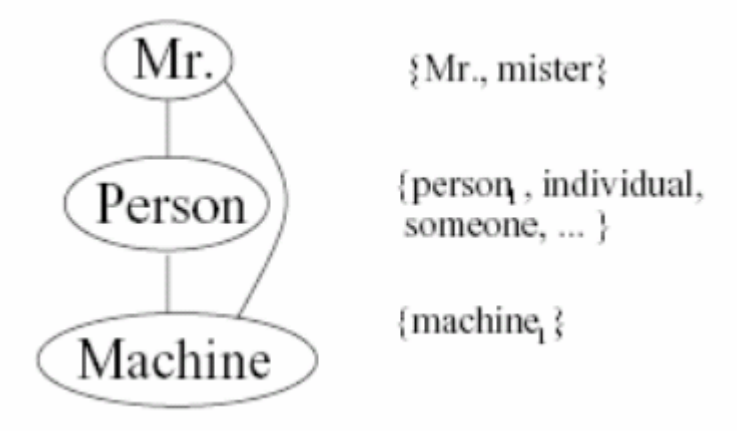


Figura 3.7-Inserimento di un nuovo termine in relazione con quelli esistenti

Per la precisione, si osserva che la lunghezza della catena, è un ottimo parametro per discriminare fra catene forti, e catene meno forti. Se un termine compare più volte, all'interno della medesima catena, significa probabilmente che il concetto ad esso associato è rappresentato all'interno del testo, in maniera forte.

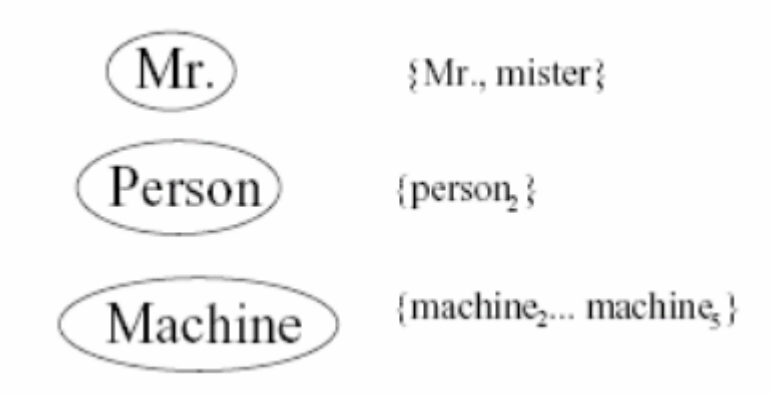


Figura 3.8-Inserimento di un termine con più significati

Viceversa se la catena risulta composta da termini tutti differenti fra loro, questo significherà che non è possibile individuare un concetto prevalente. Si definisce a tale riguardo l'omogeneità di una catena, in base alla seguente formula:

$$HomogeneityIndex = 1 - N_{do} / N_{cw}$$

dove N_{do} corrisponde al numero di parole distinte e N_{cw} corrisponde invece, al numero totale di termini di cui è composta una catena.
Il punteggio associato ad una determinata catena viene calcolato nel seguente modo:

$$Score(Chain) = Lenght * HomogeneityIndex$$

dove *Length* il numero dei termini che compongono la catena.
Le catene indicate come forti, sono quelle per le quali vale la relazione:

$$Score(Chain) > Average(Scorse) + 2 * StandardDeviation(Scores)$$

Algoritmo di Silber e McCoy

Nel 2002 in [**22**] due ricercatori dell'università di Delaware, Gregory Silber e Kathleen McCoy, svilupparono ulteriormente l'algoritmo delle catene lessicali prendendo spunto da quello di Barzilay e Elhadad. Uno dei limiti di quest'ultimo algoritmo è appunto quello di comportare un'elevata complessità computazionale. Rimandare la scelta dei synset da rimuovere, provoca un proliferare di possibili interpretazioni, la cui gestione rende esponenziale la complessità computazionale del metodo.
L'approccio di Silber e McCoy si pone l'obiettivo di rendere il metodo delle catene lessicali tempo-lineare, e quindi applicabile anche in contesti come il Web.
Allo scopo di rendere l'algoritmo, tempo-lineare, invece di calcolare ogni possibile interpretazione di un documento sorgente, come nell'algoritmo di Barzilay e Elhadad, essi realizzano una struttura che immagazzina implicitamente ogni interpretazione, senza crearla. Questo fa si che sia il tempo, che lo spazio utilizzati dall'algoritmo, siano lineari.
Anche il nuovo approccio utilizza WordNet come risorsa lessicale. In una fase preliminare, il metodo di accesso a WordNet viene tuttavia modificato, in particolare il database dei nomi (poiché il metodo delle catene lessicali è applicato solo ai nomi) viene compilato anche in formato binario. Tale formato viene poi memorizzato, per consentire di recuperare un elemento come se si trattasse di accedere ad un elemento di un grande vettore.
A differenza degli altri algoritmi, prima di iniziare l'esecuzione dell'algoritmo di disambiguazione, ogni documento viene elaborato attraverso un *part of speech tagger* per identificare i potenziali nomi da inserire all'interno delle catene. L'elaborazione del

documento coinvolge la creazione di un vettore di grande dimensioni indicato come "*Meta-Chains"*. Tale nome deriva dal fatto che esso rappresenta tutte le possibili catene dove la prima parola è già disambiguata. Le dimensioni di tale vettore dipendono dal numero dei synset e dal numero dei nomi contenuti all'interno del documento.
L'algoritmo proposto può essere sintetizzato come segue:

1. Per ogni nome contenuto nel documento sorgente, si identificano tutte le possibili catene lessicali, recuperando le relazioni di sinonimia, iponimia, iperonimia e fratello (due nodi con lo stesso padre). Tali informazioni sono poi memorizzate all'interno di un vettore, indicizzato in base alla posizione che il significato del lemma ha in WordNet.
2. Per ogni nome contenuto nel documento sorgente, si utilizza l'informazione ottenuta nel passo precedente, al fine di inserire il termine in ciascuna *meta-chain.* L'inserimento di un termine all'interno di una *meta-chain*, consente di aumentare il punteggio nella misura in cui il nuovo elemento, è semanticamente correlato a quelli già presenti.

L'algoritmo prosegue, continuando a cercare di individuare la migliore interpretazione per tutti i termini. Ogni meta-catena è, dal punto di vista della rappresentazione, un grafo chiuso i cui i vertici sono condivisi. La migliore interpretazione del testo, sarà data dall'insieme di grafi all'interno dei quali vi è il maggior grado di connessione. In sintesi, la disambiguazione dei termini avviene secondo il seguente pseudo-codice:

for(∀ termine del documento) do

for(∀ ogni catena a cui il termine appartiene) do

1. *Individua la catena il cui punteggio varierà in modo maggiore in seguito alla rimozione del termine stesso;*
2. *Inizializza a zero il punteggio del termine corrente in ogni altra catena alla quale appartiene, e aggiorna il punteggio di tutte le catene nelle quali il termine riflette quello rimosso.*

In tal modo si individuano l'insieme delle catene lessicali che massimizzano il punteggio complessivo senza averle costruite tutte esplicitamente. Questo è sicuramente l'aspetto più

interessante. L'estrarre l'interpretazione del testo (ovvero l'insieme indipendente di catene non intersecanti) con il punteggio più alto senza aver costruito nessuna altra interpretazione, consente all'algoritmo di risolvere la disambiguazione in un tempo lineare.

M.Galley- K.McKeown

Nel 2003 in [**23**] McKeown e Galley, propongono una nuova versione dell'algoritmo delle catene lesicali. Essi partono dalla considerazione che i precedenti algoritmi per il calcolo delle catene lessicali presentano notevoli lacune nell'accuratezza del processo di disambiguazione del testo e inefficienze computazionali, in particolare accusano l'algoritmo tempo-lineare proposto da Hirst e St-Onge di eseguire un WSD in accurato, poiché la loro strategia di disambiguare in un unico passaggio dell'algoritmo, disambigua immediatamente un termine. Per quanto concerne l'algoritmo di Barzilay e Elhadada, fanno osservare come pur alleviando significativamente la problematica precedente, ciò è realizzato a scapito dei tempi di esecuzione, che diventa quadratico; tale inefficienza computazionale viene attribuita all'elaborazione di tutte le possibili combinazioni di sensi dei termini, allo scopo di determinare quello più probabile. L'algoritmo di Silber e McCoy, più recente, è tempo-lineare e quindi efficiente nel calcolo delle catene lessicali, ma risulta anch'esso avere un processo di WSD non accurato.

L'algoritmo presentato adotta l'assunzione di *one sense per discorse* [**24**], ovvero assegna il medesimo senso a tutte le occorrenze di un determinato lemma. Inoltre separa il processo di WSD da quello di realizzazione della catena vera e propria.

L'algoritmo utilizza come risorsa lessicale WordNet, per costruire le catene di termini candidati che sono in relazione semantica tra loro, e viene testato ed indirizzato alla WSD di termini appartenenti ad un documento di testo. Anche in questo caso il metodo si applica solo ai nomi. Viene assegnato un peso ad ogni relazione semantica, in particolare si considerano solo le relazioni: iperonimia/iponimia e i loro fratelli. Si introducono, inoltre, i fattori di distanza (Tabella 3.1) per ogni tipo di relazione semantica, allo scopo di evitare collegamenti tra termini troppo lontano all'interno del testo.

Semantic relation	1 sent.	3 sent.	1 par.	other
synonym	1	1	0.5	0.5
hypernym/hyponym	1	0.5	0.3	0.3
sibling	1	0.3	0.2	0

Tabella 3.1-Fattori di distanza per tipo di relazione

La tabella sopra riportata, mostra i pesi di ciascuna relazione in base al tipo di relazione e alla distanza, calcolata in numero di frasi o paragrafi, tra due nodi.

L'algoritmo può essere decomposto in tre passi fondamentali:

1. costruzione della rappresentazione di tutte le possibili interpretazioni del testo;
2. disambiguazione di tutti i termini;
3. costruzione delle catene lessicali.

Come in [**22**], anche in questa variante dell'algoritmo si elabora l'intero testo e si calcolano tutte le relazioni semantiche tra i termini candidati, per ciascuno dei loro possibili sensi. In questa fase non viene realizzata alcuna disambiguazione, il solo proposito è quello di costruire una rappresentazione dei dati da poter utilizzare nella fase successiva dell'algoritmo. Inoltre, tale fase è l'unica che prevede l'accesso al testo; tutte le fasi successive, infatti, lavorano solo su questa implicita rappresentazione delle possibili interpretazioni, chiamata grafo di disambiguazione (Figura 3.9).

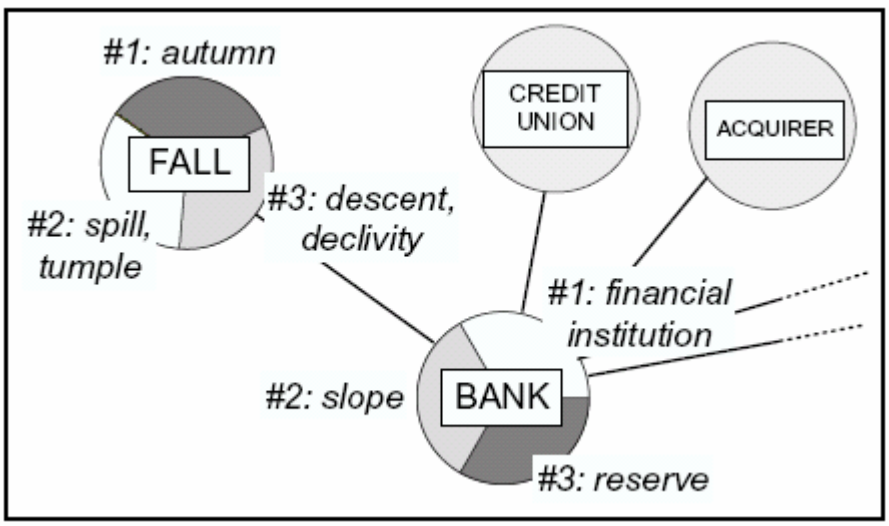


Figura 3.9-Esempio di grafo di disambiguazione

In questo tipo di grafo, i nodi rappresentano le istanze dei termini, e gli archi le relazioni semantiche. Poiché WordNet non individua relazioni tra termini, ma bensì fra sensi, ogni nodo è composto da un nome identificativo del lemma, e dall'insieme di sensi associabili a tale lemma. Da qui, ogni arco collega esattamente due sensi di nomi.

Tale rappresentazione, è costruita in un tempo lineare, e può essere a sua volta ulteriormente suddivisa nei seguenti passi:

a. Si definisce un vettore che indicizza i vari sensi di WordNet e si elabora il testo in maniera sequenziale, inserendo una copia di ogni termine candidato all'interno di ciascuna *entry* del vettore corrispondente ad un senso valido per quel termine. Per esempio, in figura 3.10 le istanze *car* ed *auto* sono state inserite in corrispondenza dello stesso senso all'interno del vettore.

b. Si controlla che l'istanza del nome appena inserita, sia legata da relazioni semantiche, ad altri nomi già presenti all'interno del vettore. Ciò è realizzato attraverso il controllo dei suoi ipernimi, iponimi e dei suoi fratelli; nel caso in cui i termini ad esso collegati non siano ancora presenti all'interno del vettore, viene comunque creato un link appropriato all'interno del grafo. Per esempio, sempre in figura 3.20, l'algoritmo individua una relazione di iperonimia, tra i nomi *car* e *auto* e il nome taxi (avente un unico senso all'interno del vettore), quindi vengono aggiunti gli opportuni link all'interno del grafo tra i due precedenti nomi e la parola taxi. Elaborando tutti i nomi secondo questa metodologia, è possibile creare tutti i link semantici nel grafo di disambiguazione, in un tempo *O(n)*, dove *n* rappresenta il numero dei termini da disambiguare.

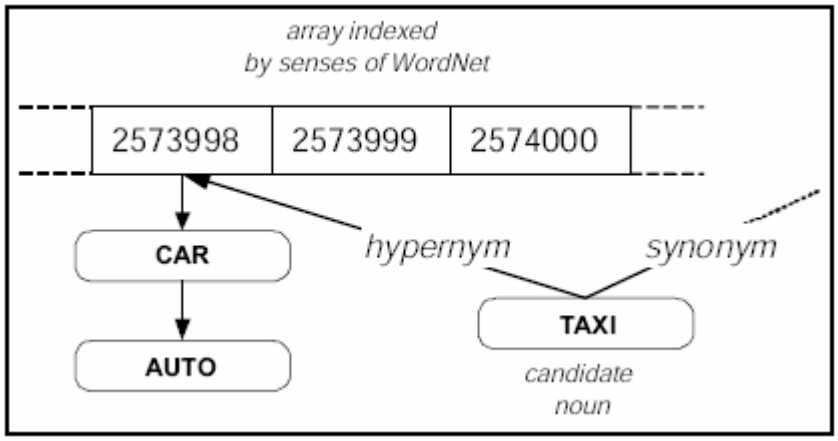


Figura 3.10-Primo passo dell'algoritmo: costruzione del grafo attraverso un vettore

Il secondo passo consiste nell'utilizzare il grafo appena creato, per eseguire il processo di disambiguazione vero e proprio. A differenza di dei precedenti algoritmi sulle catene lessicali, dove si disambiguava ogni occorrenza di un termine, in questo metodo, attraverso il vincolo di *one sense per discourse*, si disambiguano tutte le occorrenze di un termine contemporaneamente. Ciò si realizza sommando tutti i pesi degli archi che partono dal nodo del termine ambiguo, per ognuno dei suoi sensi. Il senso del termine che avrà accumulato il punteggio maggiore (somma dei pesi dei suoi archi), è quindi considerato come il senso più probabile da attribuirgli. Per esempio, in figura 3.9, il senso di *bank* con il punteggio più alto è *financial institution.* Tale senso verrà poi assegnato a tutte le occorrenze del termine *bank.*
Nel caso in cui si verifichi, tra due sensi, solamente una piccola differenza di punteggio, l'algoritmo seleziona tra i due, il senso che appare per primo all'interno di WordNet, ovvero il più frequente.
Durante il passo conclusivo dell'algoritmo, vengono costruite le vere e proprie catene lessicali, attraverso l'elaborazione dell'intero grafo di disambiguazione. In questo caso si applica la strategia introdotta da Barzilay in [**20**], detta di *strong chian sense disambiguation* dove un termine può apparire all'interno di una sola catena, in conseguenza del vincolo *one sense per discourse*. Avendo già assegnato i sensi più probabilmente corretti ai vari termini, tale fase consiste essenzialmente nella rimozione, all'interno del grafo, dei link inutilizzati, ovvero degli archi che connetto tra loro termini con senso scorretto. Una volta completato il processo di rimozione, ciò che rimane nel grafo, corrisponde alla più corretta interpretazione del testo, ed è da qui che vengono ricavate le catene lessicali. M. Galley e K.McKeown sottolineano l'importanza della separazione tra il processo di disambiguazione dei termini e quello di determinazione delle catene lessicali, in quanto in tal modo, è possibile analizzare tutte le relazioni semantiche tra i termini, corrette o no, senza rischiare di creare relazioni semantiche sbagliate all'interno delle catene lessicali.
L'algoritmo di disambiguazione del testo utilizzato, è stato poi testato applicandoli a 74 documenti estratti da SemCor, i quali in totale contenevano all'incirca 35000 nomi, ottenendo un livello di accuratezza pari al 62,09%. In figura 3.11 viene mostrato un grafico che mette in relazione l'algoritmo di Galley e McKeown con i due algoritmi sulle catene lessicali, precedentemente presentati, mostrando miglioramenti per quanto riguarda l'accuratezza nella disambiguazione dei lemmi polisemici.

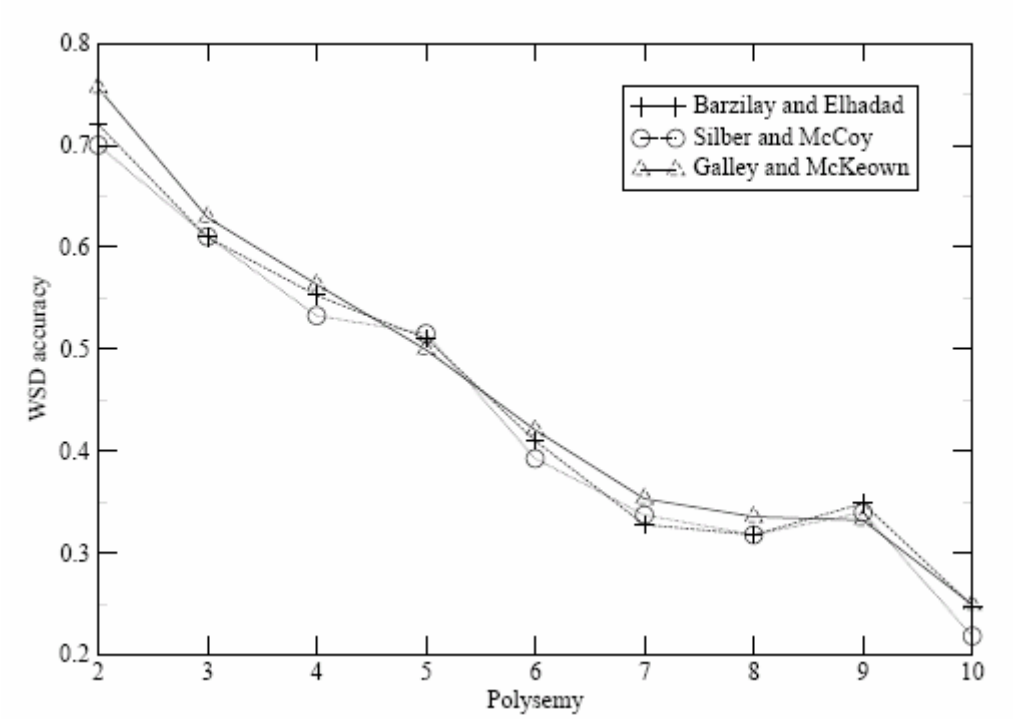


Figura 3.11-Confronto del livello di accuratezza di diversi algoritmi sulle catene lessicali

Algoritmo di TUCUXI

TUCUXI è un In**T**elligent H**U**nter Agent for **C**oncept **U**nderstanding and Le**X**ical Cha**I**ning, realizzato dall'Università degli Studi di Modena e Reggio Emilia, da R.Benassi S.Bergamaschi e M.Vincini. Esso in [**25**] sviluppa il meccanismo delle catene lessicali, allo scopo di realizzare una metodologia alternativa alle tradizionali tecniche di Information Retrieval che, nella ricerca di pagine nel web, utilizzano query basate su parole chiave, dove non si considera né la possibile polisemia dei termini, né le relazioni lessicali che li legano.

L'obiettivo è quello di effettuare un processo di ricerca dei documenti nel Web, basato sul contenuto semantico associato ai termini che esprimono le richieste dell'utente. Come vedremo, tale algoritmo mantiene separata le fase di disambiguazione dei termini da quella di creazione delle catene lessicali.

L'algoritmo può essere riassunto nei seguenti passi:

1. *Selezione dei termini candidati a formare le catene lessicali*. Si assume, come nei precedenti algoritmi, che i concetti siano espressi in modo rilevate, principalmente dai

nomi, di conseguenza, attraverso un *part-of-speech tagger* vengono individuati e selezionati solo i nomi come parole candidate. Inoltre un *parser* riconosce i termini composti, ovvero formati da più parole. In questo caso si decide di mantenere il termine composto come un'unica unità, catturando così più informazioni semantiche, rispetto a quelle che si otterrebbero considerando i termini separatamente.

2. *Word SenseDisambiguation*. Il processo di disambiguazione dei termini utilizza WordNet come risorsa lessicale. L'idea chiave è quella di risolvere (anche solo parzialmente) l'ambiguità dei termini, attraverso un processo incrementale guidato dalle proprietà di coesione. Si definiscono delle unità di base come delle associazioni tra un termine candidato e uno dei suoi synset. Successivamente, in base alla tipologia di relazione lessicale e alla posizione che il termine ha all'interno del testo, ogni unità di base, esprime un voto di coesione per se stesso e per le altre unità di base con cui ha un synset in relazione semantica. L'algoritmo utilizzato è descritto brevemente in figura 3.12. L'algoritmo si conclude selezionando per ogni parola candidata, solo le unità di base con il più alto punteggio di preferenza, mentre le rimanenti vengono conseguentemente eliminate. In figura 3.13 viene mostrato un esempio di applicazione dell'algoritmo di disambiguazione.
3. Generazione delle catene lessicali. Il processo di creazione delle catene lessicali, si realizza attraverso un algoritmo di *clustering*, che riceve come input le unità di base selezionate al passo 2, e produce come output un insieme di cluster composti da i termini disambiguati. In questo algoritmo si è deciso di selezionare solo relazioni di coesione *strong* fra i termini, ottenendo quindi delle catene lessicali cosiddette strong anch'esse. A tal fine,viene adottato il criterio di identificazione delle catene strong proposto da Barzillay e Elhadad, applicando tuttavia ulteriori filtri nel caso in cui si abbia a che fare con documenti ricchi d'informazione testuale. In figura 3.14 viene mostrato l'algoritmo di definizione delle catene.

Algorithm 1 TUCUXI's Word Sense Disambiguation

Input: *WordNet lexical Database (WNx) and its extensions if any*
S={s_i: s_i is one of the 1..n possible synsets contained in the text}, an ordered set CW={w_j: w_j is one of the 1..k candidate words in the text}, WS_j={ws_l: ws_l is one of the 1..t possible meanings of w_j}, $j = 1..k$, scoring criteria C.

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

Figura 3.12-Codice dell'algoritmo di disambiguazione dei termini di TUCUXI

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

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

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

(a) *Word Sense disambiguation*

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

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

Figura 3.13-Esempio di disambiguazione di alcune frasi estratte da www.cs.stanford.edu/Courses/index.html

Algorithm 2 TUCUXI's Lexical Chaining Process

Input: $BU=\{bu_j : bu_j$ *represents the* w_j *word in* CW *and the most reasonable meanings* $ws_l{}^{best}$ *in* $WS_j\}$, *a list of preserved synsets* PS *and their related ones, scoring criteria* C

Create an empty array L;
for all $bu_j \in BU$ **do**
  add bu_j to the chain in L whose basic units establish the strongest connection with it (through the bu_j synset or the related ones),
  if no chains are suitable **then**
    create a new chain in L with bu_j
  else
    update the score of the selected chain, according to C;
  end if
end for
Calculate the $avg(score)$ of chains and the standard deviation $stDev$;
Delete (not strong) chains (Barzilay and Elhadad' criterion: $score_{chain} \leq avg(score) + 2 * stDev$, other pruning criteria if necessary).

Output: *The survived lexical chains.*

Figura 3.14- Algoritmo di creazione delle catene lessicali implementato da TUCUXI

3.1.1 Algoritmi di Gloss Overlap

L'algoritmo di Lesk

Il primo ad implementare l'algoritmo di Gloss Overlap fu Lesk nel 1986 in [**26**]. L'algoritmo originale di Lesk, disambigua termini di tutte le categorie sintattiche, all'interno di brevi frasi. Data la parola da disambiguare e la sua definizione nel dizionario o la sua glossa, per ognuno dei sensi ad esso associati, viene confrontato con le glosse degli altri termini che compaiono all'interno della frase. Successivamente, al termine viene associato il significato la cui glossa presenta il maggior numero di parole in comune con le glosse appartenenti alle altre parole. L'algoritmo prosegue, eseguendo un processo iterativo per ogni termine, e non considerando i sensi precedentemente assegnati. Di seguito si riporterà il codice dell'algoritmo:

```
for every word w[i] in the phrase
  let BEST_SCORE = 0
  let BEST_SENSE = null
  for every sense sense[j] of w[i]
    let SCORE = 0
    for every other word w[k] in the phrase, k != i
      for every sense sense[l] of w[k]
        SCORE = SCORE + number of words that occur in the gloss of
                        both sense[j] and sense[l]
      end for
    end for
    if SCORE > BEST_SCORE
      BEST_SCORE = SCORE
      BEST_SENSE = w[i]
    end if
  end for
  if BEST_SCORE > 0
    output BEST_SENSE
  else
    output "Could not disambiguate w[i]"
  end if
end for
```

Riportiamo ora un esempio di applicazione dell'algoritmo sulla parola *pine cone*. Utilizzando *l'Oxford Advanced Learner's Dictionary*, si ritrovano i seguenti due sensi per la parola *pine:*

- Senso 1: kind of **evergreen tree** with needle-shaped leaves
- Senso 2: waste away through sorrow or illness

Per la parola *cone* troviamo invece tre sensi:

- Senso 1: solid body which narrows to a point
- Senso 2: something of this shape whether solid or hollow
- Senso 3: fruit of certain **evergreen tree**

Ognuno dei sensi della parola *pine* è confrontato con tutti i sensi della parola *cone*. Come si nota facilmente il senso 1 di *pine* ha in comune la parola *evergreen tree* con il senso 3 di *cone.* Di conseguenza questi due sensi sono scelti come i più appropriati da assegnare alle parole *pine* e *cone*, nel caso in cui quest'ultime compaiano insieme.

L’algoritmo di Lesk, quindi, non fa altro che contare il numero di elementi che le glosse hanno in comune e assegna un punteggio pari al numero di parole in comune.
L’algoritmo di Lesk utilizza una *window of context* intorno al termine target, da sottomettere come input all’algoritmo. In particolare è stata definita una finestra di 2n + 1 parole intorno al termine target, che include ovviamente la parola target e gli n termini subito alla sua destra e gli n subito alla sua sinistra. In particolare si è fissata la dimensione della finestra a 11.

Algoritmo di Disambiguazione Globale di Banerjee e Pedersen

Nel 2002 in [**27**] Banerjee e Pedersen proposero un’estensione dell’algoritmo proposto da Lesk. Essi hanno modificato l’algoritmo originario sotto diversi aspetti allo scopo di realizzare un algoritmo *baseline*. Come nell’implementazione dell’algoritmo di Lesk, l’unità di base da disambiguare continua ad essere una breve finestra di contesto centrata esattamente nella parola target, tuttavia la finestra viene ridotta a soli 3 termini.
L’algoritmo proposto, invece di adottare il dizionario *Oxoford Advanced Learner*, utilizza come risorsa letterale WordNet, visto la ricchezza di relazioni tra i termini che quest’ultimo offre.
La scelta di WordNet, ha portato ad una delle modifiche sicuramente più rilevanti all’algoritmo originale: Banerjee e Pedersen hanno intuito l’importanza dell’informazione associata alle relazioni fra i termini fornite da WordNet. In particolare il loro algoritmo, utilizza come risorsa non solo le glosse dei termini, ma bensì anche le relazioni che legano il termine target con gli altri termini del contesto, ottenendo così maggiore accuratezza nel processo di disambiguazione.
Nella fase di confronto tra le glosse, hanno considerato quindi, non solo le glosse dei termini presenti nel contesto, ma anche le glosse delle parole in relazione con il termine target. Le relazioni considerate sono mostrate in tabella 3.2.
Il processo di confronto fra le glosse, può essere scomposto in due fasi: la prima riguarda la selezione delle glosse da confrontare, e la seconda la valutazione dei pesi da dare al risultato di ciascun confronto. Vediamo ora come viene scelta la coppia di glosse da comparare.
D’ora in poi si indicherà con *word#pos#sense*, rispettivamente la parola#categoria sintattica#senso. Per esempio *sentence#n#2*, sta ad indicare il secondo senso del nome *sentence*. Notiamo come ogni parola#categoria_sintattica#senso, appartenga esattamente ad un unico synset di WordNet.

Noun	Verb	Adjective
Hypernym	Hypernym	Attribute
Hyponym	Troponym	Also see
Holonym	Also see	Similar to
Meronym		Pertainym of
Attribute		

Tabella 3.2- Relazioni considerate nell'algoritmo

Banerjee e Pedersen hanno sperimentato due possibili schemi di selezione delle coppie di glosse, uno così detto *schema eterogeneo* e l'altro *schema omogeneo*.
Nello schema eterogeneo, ogni glossa associata con il senso in considerazione della prima parola, viene paragonata con ogni glossa associata alla seconda parola. Un esempio di schema eterogeneo è mostrato in figura 3.15, dove si eseguono 6 confronti comparando a turno ognuna delle tre glosse associate a *sentence#n#2* con ognuna delle due glosse associate a *bench#n#2*.

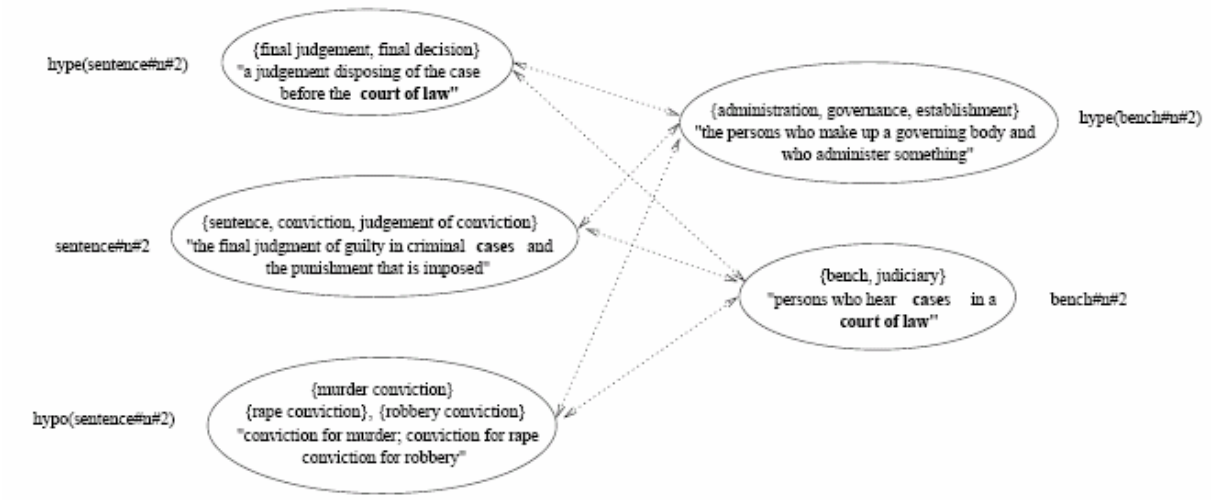


Figura 3.15- Esempio di schema eterogeneo per la selezione delle coppie di glosse confrontare.

Gli schemi omogenei, invece, per confrontare i sensi di due parole, si cercano le sovrapposizioni tra le loro glosse, e le sovrapposizioni tra le glosse dei synset ad esse

relazionati, attraverso la medesima tipologia di relazione. In figura 3.16 è mostrato un esempio di schema omogeneo dove si eseguono solo due confronti: la glossa di *sentence#n#2* con la glossa di *bench#n#2* e la glossa dell' ipernomo di *sentence#n#2* con la glossa dell'ipernomo di *bench#n#2.*

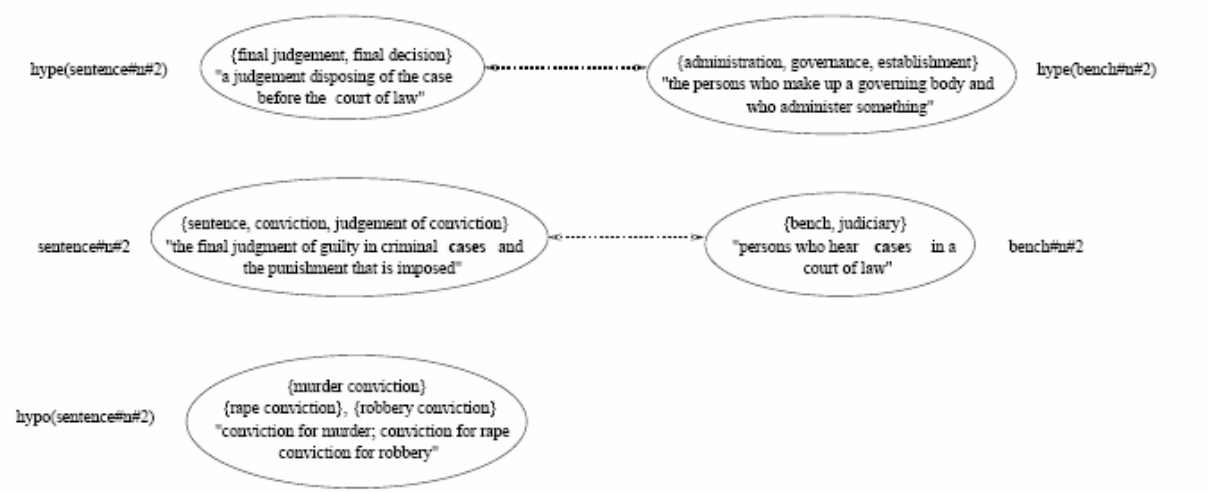


Figura 3.16- Esempio di schema omogeneo per la selezione delle coppie di glosse da confrontare

Successivamente, una volta stabilite le coppie di glosse da confrontare, vengono individuati gli *overlap,* ovvero gli insiemi di una o più parole consecutive (non considerando pronomi, preposizioni, articoli o congiunzioni), che compaiono in entrambe le glosse. Ovviamente tra due glosse potranno esistere più overlap. Per esempio considerando le due stringhe :

1- This is the first string we will compare

2- *I will compare the first string with the second*

esisteranno due overlap *(first string* e *will compare),* ognuno composto da due parole consecutive. Indichiamo con *n* il numero di parole da cui è composto un overlap.

Gli overlap sono successivamente usati, per calcolare la misura di relazione tra i synset . A differenza dell'algoritmo di Lesk, nel calcolo del punteggio, non si considerano le singole parole ma i gruppi di parole consecutive (gli overlap) attribuendo un peso maggiore a quelli composti da più parole. Cioè si vuole attribuire un peso diverso in base al fatto che due glosse abbiano in comune due termini consecutivi o viceversa due termini separati.

Tale assunzione è ispirata dalla *legge di Zipf* in [**28**] la quale afferma che la lunghezza di una frase è inversamente proporzionale alla sua frequenza all'interno di un corpus di testi. Le frasi

con lunghezza maggiore saranno meno frequenti delle frasi a lunghezza minore.Quindi, riuscire a trovare più termini consecutivi in comune è un evento raro e deve essere conseguentemente pesato diversamente. In particolare l'algoritmo calcola il punteggio tra due glosse elevando al quadrato il contributo di ogni singolo overlap, in tal modo il contributo dato dall'overlap di più termini, è sempre maggiore del contributo dato sall'overlap dei termini presi singolarmente, essendo la somma dei quadrati dei singoli valori, maggiore del quadrato delle singole somme. Per esempio, se due glosse hanno tre termini consecutivi in comune, gli verrà assegnato un peso pari a (3 x 3) = 9, viceversa verranno semplicemente sommati i contributi unitari dei singoli termini, se tali termini non compaiono consecutivamente (1 + 1 + 1) = 3.

Pedersen e Banerjee in questo algoritmo, utilizzano un approccio globale, in cui si disambiguano contemporaneamente tutte le parole di una finestra di contesto. In tal caso, in opposizione al metodo locale utilizzato nell'algoritmo di Lesk, il senso scelto per un termine andrà ad influire sul senso da scegliere per i termini appartenenti alla stessa finestra.

L'algoritmo calcola tutte le possibili combinazioni tra i vari sensi di tutti i termini di una finestra. Successivamente per ogni coppia di sensi, è calcolato il punteggio attraverso il confronto tra le glosse ad essi associate (usando o lo schema omogeneo o quello eterogeneo). I singoli punteggi sono poi sommati fra loro. La combinazione dei sensi con il punteggio più alto è scelta come più appropriata per disambiguare i termini.

Questo algoritmo tuttavia presenta alcune limitazioni. La scelta di disambiguare tutti i termini di una finestra contemporaneamente, può essere ritenuta corretta solo quando le parole che devono essere confrontate, sono fortemente in relazione fra loro. Tale caratteristica è in generale ritrovabile nelle *query* sottoposte nei sistemi di *Information Retrieval* ma non in molte altre applicazioni.

Un'altra problematica è legata la fatto che, l'approccio di confrontare ogni senso di un termine con ciascun senso di tutti gli altri termini all'interno della finestra di contesto, rende l'algoritmo di complessità esponenziale dal punto di vista computazionale. Indicando con *s* il numero medio di sensi per ciascun termine, e con *N* il numero di termini all'interno di una finestra di contesto, si avranno:

$$s^2 \times \frac{N \times (N-1)}{2}$$

coppie di synset da confrontare. Si noti come tale valore vari in maniera esponenziale in base ad *N*.

Questi due svantaggi sono alla base della motivazione per il quale l'algoritmo è stato implementato con una finestra pari a tre termini. In tabella 3.3 sono mostrati i risultati della valutazione dell'algoritmo. Si notano risultati migliori se comparati a quelli ottenibili tramite l'algoritmo di Lesk originale. Inoltre, appare come l'applicazione di schemi eterogenei nell'individuazione delle glosse da confrontare, in generale consenta di ottenere valori più alti sia di recall che di precision.

	POS	Precision	Recall	F–Measure
Homo–genous	Noun	0.364	0.364	0.364
	Verb	0.183	0.183	0.183
	Adjective	0.276	0.276	0.276
	Overall	0.273	0.273	0.273
	POS	Precision	Recall	F–Measure
Hetero–genous	Noun	0.406	0.406	0.406
	Verb	0.190	0.190	0.190
	Adjective	0.324	0.324	0.324
	Overall	0.301	0.301	0.301

Tabella 3.3- Prestazioni dell'algoritmo nei due schemi differenti

Algoritmo di Disambiguazione Locale di Banerjee e Pedersen

L'algoritmo presentato nel precedente paragrafo, può essere applicato solo nel caso in cui i termini da disambiguare siano relazionati fortemente fra loro. Tale caratteristica, ne limita notevolmente la possibilità di utilizzo. Conseguentemente, Banerjee e Pedersen, negli stessi anni in [**27**], proposero la versione di disambiguazione locale del precedente algoritmo.

Nella maggiorparte dei casi, infatti, si ha a che fare con frasi complete, dove molti termini non sono in relazione col termine target. Di conseguenza, il confronto con le loro glosse risulterebbe inutilmente dispendioso . Questo nuovo algoritmo si pone l'obiettivo di eliminare tutti i confronti superflui tra le glosse. Grazie all'analisi del precedente algoritmo, essi scelsero di utilizzare solo lo schema eterogeneo per la selezione delle glosse da confrontare, in quanto migliora le prestazioni.

L’algoritmo locale, considera un senso del termine target e confronta la sua glossa e tutte le glosse dei synset ad esso relazionati, attraverso una delle relazioni riportate in tabella 3.2, con le glosse di tutti i sensi di tutte le parole non-target nella finestra del contesto e le glosse degli altri synset relazionati a quest’ultimi attraverso le relazioni sempre in tabella 3.2. Per ciascun confronto viene calcolato il punteggio applicando gli stessi principi descritti nel paragrafo precedente. Alla fine, il senso del termine target che avrà accumulato il punteggio maggiore, verrà selezionato come più appropriato.

Per esempio, consideriamo una finestra di tre termini, *sentence, bench e offered*, dove *bench* è il termine target. Assumiamo inoltre di considerare solo le relazioni di iperonimia e iponimia. Sia *sentence* che *bench* possiedono due sensi differenti, mentre *offender* è un termine monosemico. Riportiamo di seguito le glosse di ciascun synset:

```
gloss(sentence#n#1) = a string of words satisfying the grammatical rules
of a language
gloss(sentence#n#2) = a final judgment of guilty in a criminal cases and
the punishment that is imposed
gloss(bench#n#1) = a long seat for more than one person
gloss(bench#n#2) = persons who hear cases in a court of law
gloss(offender#n#1) = a person who transgresses law
```

Nell’algoritmo ad approccio locale, si ottiene un punteggio per ognuno dei due sensi del termine target. La figura 3.17 mostra il processo di confronto per la determinazione del punteggio del senso #2 di *bench.*

In maniera del tutto simile vengono effettuati i confronti per il senso #1 di *bench.*

La tabella 3.4 riporta i risultati dei calcoli effettuati per determinare il punteggio associato al senso #2.

L’algoritmo ad approccio locale, eliminando i confronti fra le glosse di tutti i termini della finestra del contesto, riduce notevolmente la complessità computazionale che diventa:

$$s \times s \times (N - 1).$$

Quindi ,la complessità varia in maniera lineare in base ad *N*. Ciò consente all’algoritmo di poter essere implementato ed eseguito, utilizzando finestre di dimensione anche superiore a tre. In tabella 3.5 sono mostrate le prestazioni dell’algoritmo in termini di recall, precision e

F-Measure (che combina in un'unica misura i singoli valori di precision e recall), per varie dimensioni della finestra.

Dalla tabella 3.5, si nota come l'algoritmo locale, oltre ad essere più performante, risulta anche più efficiente dell'algoritmo globale, ottenendo valori migliori sia di recall che di precision.

Figura 3.17-Confronti necessari per determinare il punteggio di *bench#2*

In generale, inoltre i nomi vengono disambiguati meglio, rispetto alle altre categorie sintattiche, questo probabilmente, è dovuto al fatto che in generale ai nomi in WordNet sono associati meno sensi rispetto ai verbi, aggettivi ecc…

First Gloss	Second Gloss	Overlap String	Normalized Score
hype(bench#n#2)	hype(offender#n#1)	person	0.015
hype(bench#n#2	offender#n#1	person	0.022
hype(bench#n#2)	hypo(offender#n#1)	person	0.006
bench#n#2	hype(sentence#n#2)	court of law, case	0.111
bench#n#2	sentence#n#2	cases	0.008
bench#n#2	hype(offender#n#1)	person	0.018
bench#n#2	offender#n#1	person, law	0.055
bench#n#2	hypo(offender#n#1)	person	0.097
Total score for sentence#n#2 – bench#n#2 – offender#n#1			0.332

Tabella 3.4- Calcoli effettuati nella determinazione del punteggio di *bench#2*

	Window Size	3	5	7	9	11
Noun	Precision	0.411	0.411	0.415	**0.420**	0.419
	Recall	0.411	0.411	0.415	**0.420**	0.419
	F–measure	0.411	0.411	0.415	**0.420**	0.419
Verb	Precision	0.203	0.220	0.220	0.226	**0.227**
	Recall	0.203	0.220	0.220	0.226	**0.227**
	F–measure	0.203	0.220	0.220	0.226	**0.227**
Adj	Precision	0.329	0.334	**0.339**	0.331	0.338
	Recall	0.329	0.334	**0.339**	0.331	0.338
	F–measure	0.329	0.334	**0.339**	0.331	0.338
Overall	Precision	0.310	0.318	0.320	0.323	**0.324**
	Recall	0.310	0.318	0.320	0.323	**0.324**
	F–measure	0.310	0.318	0.320	0.323	**0.324**

Tabella 3.5- Prestazioni dell'algoritmo in base a differenti dimensioni di finestra

3.1.2 Algoritmi basati sulle misure di relazione semantica

Molti algoritmi di disambiguazione del testo, stabiliscono il senso corretto di un termine, calcolando la misura di relazione semantica che tale termine ha nei confronti delle parole che fanno parte del suo contesto. Molto spesso in letteratura, si ha la tendenza ad utilizzare senza distinzione i termini "relazione semantica" e "similarità semantica". In realtà esiste una differenza fondamentale tra i due termini: Budanisky e Hirst in [**29**] affermano che la similarità semantica è un particolare tipo di relazione semantica tra due parole che ne definisce una somiglianza. La relazione semantica copre, invece, un più ampio insieme di relazioni tra concetti come l'iponimia, meronimia ecc…

I successivi paragrafi, hanno lo scopo di riassumere brevemente le principali metodologie di calcolo della relazione semantica. Nel far ciò, si utilizzerà la classificazione utilizzata da Michelizzi in [**30**] che distingue le misure di relazione semantica in: misure di similarità semantica basate sul *path* (ovvero del percorso all'interno della gerarchia che unisce due concetti), misure di similarità basate sul contenuto informativo dei concetti e misure di relazione semantica.

Misure di similarità basate sul Path

La misura di Wu e Palmer

Wu e Palmer presentano una misura di similarità semantica basata sulla profondità di una tassonomia. La misura prende in considerazione la distanza tra due synset S_1 ed S_2 e il loro *last common subsumer* (LCS) S_3, e la distanza tra l'LCS e la radice della tassonomia all'interno della quale risiede il synset.

La formula di similarità è la seguente:

$$Sim_{wup}(s_1, s_2) = \frac{2 \cdot dist_{node}(s_3, Root)}{dist_{node}(s_1, s_3) + dist_{node}(s_2, s_3) + 2 \cdot dist_{node}(s_3, Root)}$$

dove $dist_{node(s1, s3)}$ rappresenta la distanza tra il synset S_1 ed il synset S_3 espressa in termini di nodi.

La misura di Leacock- Chodorow

Anche la misura di Leacock e Chodorow proposta nel 1998 in [**31**] si basa sulla lunghezza del percorso esistente tra due concetti all'interno di una gerarchia *is-a.* Il percorso più breve tra due concetti sarà quello che include, al suo interno, il numero minore di concetti intermediari. Questo valore deve essere tuttavia valutato in base alla profondità della gerarchia, dove per profondità si intende la lunghezza del percorso più lungo dal nodo radice ad un nodo foglia della gerarchia.

Di conseguenza la misura di relazione è definita come segue:

$$related_{lch}(c_1, c_2) = max[-log(ShortestLength(c_1, c_2)/(2 \cdot D))]$$

dove *ShortestLength(c1, c2)* è la lunghezza del percorso più breve (avente cioè il minimo numero di nodi), tra due concetti, e *D* è la profondità massima della tassonomia. Considerando WordNet come gerarchia di domini, il valore di *D* diventa una costante pari a 16 per tutti i nomi, in quanto in WordNet la lunghezza del percorso dal nodo radice ad un nodo foglia di un nome, è sempre pari a 16.

Misure di similarità basate sul contenuto informativo dei concetti

La misura di Resnik

Resnik in [32] nel 1995 ha introdotto una misura di relazione basata sulla sua formula di calcolo del contenuto informativo, la quale consente di assegnare un valore ad ogni concetto all'interno di una gerarchia basata su evidenze ricavate da un corpus.

Prima di descrivere tale misura, tuttavia è opportuno chiarire la nozione di contenuto informativo*:* esso rappresenta semplicemente una misura della specificità di un concetto. Un concetto con un alto valore di contenuto informativo, è particolarmente specifico nei confronti di un particolare argomento, mentre, viceversa, i concetti con un basso valore sono associati, in maniera più generica, ad argomenti meno specifici.

Per esempio, i termini *carving fork* hanno un alto contenuto informativo, mentre *entity* viceversa ha un basso contenuto informativo.

Il contenuto informativo di un concetto, può essere stimato attraverso il calcolo della frequenza del medesimo in un ampio corpus e, quindi,in base alla probabilità massima di occorrenza del lemma. In particolare la formula è la seguente:

$$IC(\text{concetto}) = -\log(P(\text{concetto}))$$

Se il testo è già *sense-tagged,* il calcolo della frequenza dei concetti può essere ottenuto direttamente grazie al fatto che, ad ogni concetto, è associato un unico senso. Viceversa, se il testo non è *sense-tagged* (come accade nella maggior parte dei casi), sarà necessario adottare uno schema di calcolo alternativo. Resnik, in tal caso, suggerisce di calcolare il numero di occorrenze di una parola in un corpus, e successivamente dividere il calcolo per il numero di sensi differenti associati al termine. Tale valore viene poi associato a ciascun concetto. Per esempio, supponiamo che il termine *bank* compaia 20 volte all'interno di un corpus, e che esistano due sensi ad esso associati all'interno della gerarchia, uno per "*river bank*" e uno per "*financial bank*". Ognuno di questi concetti riceve inizialmente un valore di occorrenze pari a 10. Una possibile alternativa può essere quella di assegnare ad ogni senso l'intero punteggio iniziale (nell'esempio 20).
La misura di similarità semantica di Resnik utilizza il contenuto informativo dei concetti e la loro posizione all'interno della gerarchia *is-a* di WordNet, per calcolare e determinare un valore di relazione semantica. L'idea chiave della sua misura, consiste nel fatto che due concetti sono in relazione semantica fra loro in proporzione alla quantità d'informazione che hanno in comune. La quantità d'informazione comune ai due concetti, è determinata dal contenuto informativo del concetto più basso all'interno della gerarchia che sussume entrambi i concetti dati. Questo concetto viene indicato come *lowest common subsumer (lcs)* dei due concetti. Di conseguenza, la misura di similarità di Resnik è definita come:

$$sim_{res}(c_1, c_2) = IC(lcs(c_1, c_2))$$

Si noti come questa misura non considera il contenuto informativo del concetti stessi, e non considera neppure la lunghezza del percorso. Una potenziale limitazione di questo approccio, è che i concetti devono condividere il medesimo ultimo sussunto, e devono avere lo stesso valore di similarità con esso. Per esempio, in WordNet il concetto *vehicle* è l'ultimo sussunto

comune di *jumbo jet, tank, hourse trailer,* e *ballistic missile.* Tuttavia, nessun di questi possiede lo stesso valore di similarità con *vehicle.*

La misura di Jiang-Conrath

Jang e Conrath nel 1997 in [34] utilizzano il contenuto informativo definito da Resnik integrandolo con il valore della lunghezza del percorso tra due concetti. Come risultato si ottiene, un approccio ibrido al calcolo della relazione semantica tra due concetti. Tale approccio include sia il contenuto informativo dei concetti stessi, sia il contenuto informativo dei loro ultimi sussulti all'interno della gerarchia. La misura è quindi determinabile attraverso la seguente formula:

$$dist_{jcn}(c_1, c_2) = IC(c_1) + IC(c_2) - 2 \times IC(lcs(c_1, c_2))$$

Tale formula, restituisce una misura della "distanza" tra i due concetti. Concetti più fortemente relazionati dal punto di vista semantico, avranno un punteggio più basso rispetto a coppie di concetti relazionati più debolmente. Allo scopo di mantenere una certa continuità con le varie misure presentate, viene definita anche la misura inversa, rappresentante la relazione semantica :

$$related_{jcn}(c_1, c_2) = \frac{1}{dist_{jcn}(c_1, c_2)}$$

La misura di Lin

La misura di Lin in [36], si basa sul suo teorema sulla similarità. Esso afferma che la similarità fra due concetti è misurabile attraverso il rapporto tra la quantità di informazione a livello di *commonality* tra i due concetti e la quantità d'informazione richiesta per descriverli
La *commonality* tra due concetti si ottiene a partire del contenuto informativo dei loro più bassi sussunti comuni e dal contenuto informativo portato dai due stessi concetti. La misura di similarità è quindi la seguente:

$$related_{lin}(c_1, c_2) = \frac{2 \times IC(lcs(c_1, c_2))}{IC(c_1) + IC(c_2)}$$

Nella tabella 3.6 è mostrato un breve confronto in termini di accuratezza fra le varie misure precedentemente descritte.

Misure di relazione semantica

All'interno degli algoritmi basati su misure di relazione semantica, rientrano anche gli algoritmi descritti nei precedenti paragrafi di Gloss Overlap e di Hirst e Orange per le catene lessicali. Tuttavia si è preferito riportarli separatamente in quanto il primo individua una casistica ben definita di algoritmi, e il secondo è più propriamente classificabile sotto gli algoritmi legati alle catene lessicali. Di seguito si descriveranno alcuni algoritmi di disambiguazione che utilizzano misure di relazione semantica

Massimizzazione della misura di relazione sematica

Nel 2004, Patwardlhan Pedersen e Banerjee riprendono gli algoritmi proposti in [**37**] osservando che il processo di disambiguazione del testo, può essere ottenuto non solo attraverso la misura di similarità proposta dall'algoritmo di Lesk ma bensì, attraverso qualsiasi tipologia di misura di relazione semantica che sia in grado di valutare la similarità tra due sensi.

Essi si basano sull'assunzione che i termini che formano una frase sono necessariamente relazionati fra loro. Il problema della disambiguazione si concentra quindi, sulla scelta della misura di relatedness più adatta ed efficace.

Essi generalizzano l'algoritmo di disambiguazione locale di Banerjee e Pedersen rendendolo indipendentemente dalla misura di similarità scelta. Ciò che si viene a realizzare è quindi una completa separazione tra quello che è l'algoritmo di disambiguazione e la misura di relatedness scelta. La formula che riassume il loro algoritmo è quindi la seguente:

$$argmax_{i=1}^{m_t} \sum_{j=1, j\neq t}^{n} max_{k=1}^{m_j} relatedness(s_{ti}, s_{jk})$$

dove *relatedness(Sti, Sjk*) è la misura di relazione semantica scelta e *Sti e Sjk* sono due sensi dei termini all'interno della finestra di contesto.

Word	Instance count	Senses		Measures of Relatedness					
		WN	cand	Res	Jcn	Lin	Lch	Hso	Lesk
art	98	4	14	0.41	0.54	0.42	0.44	*0.40*	**0.61**
authority	92	7	9	*0.14*	0.16	0.17	0.19	0.20	**0.27**
bar	151	13	21	0.21	0.23	**0.25**	*0.18*	**0.25**	0.21
bum	44	4	4	0.20	**0.73**	0.41	0.59	0.31	*0.13*
chair	69	4	7	0.37	0.33	0.46	*0.21*	0.44	**0.84**
channel	73	7	13	0.15	0.15	0.16	**0.23**	0.20	*0.10*
child	63	4	5	0.27	0.43	0.38	*0.02*	0.16	**0.62**
church	64	3	9	*0.37*	0.41	*0.37*	0.41	**0.48**	0.38
circuit	85	6	15	0.43	0.51	0.48	*0.34*	0.41	**0.53**
day	134	10	18	*0.12*	**0.43**	0.32	0.28	0.19	0.15
detention	32	2	5	0.61	0.81	0.61	*0.52*	0.63	**0.88**
dyke	28	2	2	0.73	0.86	0.77	*0.46*	0.61	**0.89**
facility	58	5	7	0.24	**0.34**	0.29	*0.21*	0.23	0.29
fatigue	43	4	6	*0.16*	0.42	0.22	**0.77**	0.44	**0.77**
feeling	51	6	7	*0.22*	**0.55**	0.27	0.53	0.26	0.49
grip	42	7	8	**0.22**	0.19	0.19	0.17	0.18	*0.12*
hearth	32	3	3	0.43	0.72	0.59	*0.38*	0.42	**0.75**
holiday	31	2	3	**0.55**	*0.16*	0.32	**0.55**	**0.55**	*0.16*
lady	53	3	8	0.36	*0.17*	0.19	**0.42**	0.36	*0.17*
material	69	5	10	0.44	**0.55**	0.44	0.40	0.38	*0.29*
mouth	57	8	8	0.12	0.12	0.11	*0.05*	0.20	**0.46**
nation	37	4	6	*0.18*	0.35	0.26	0.22	0.26	**0.59**
nature	44	5	6	0.10	0.11	*0.05*	0.11	**0.18**	0.16
post	78	8	12	0.16	**0.35**	0.19	*0.09*	0.15	0.31
restraint	45	6	7	0.31	**0.40**	0.33	0.36	0.30	*0.16*
sense	50	5	11	0.49	*0.40*	0.51	*0.40*	0.43	**0.50**
spade	33	3	4	**0.70**	*0.15*	0.56	0.21	0.40	0.59
stress	39	5	5	0.32	0.38	0.33	**0.44**	0.39	*0.31*
yew	28	2	3	0.66	0.79	0.73	*0.57*	0.70	**0.86**
Total	1723			*0.295*	0.380	0.331	0.305	0.316	**0.391**

Tabella 3.6-Accuratezza delle varie misure di similarità

La formula restituisce quindi l'indice del senso del termine target che è più relazionato con i sensi delle altre parole del contesto.

Tale algoritmo è stato implementato, e ha portato alla realizzazione del software WordNet::Similarity. Tale *package* essenzialmente fornisce la possibilità di calcolare la similarità tra due termini attraverso sei formule di misure di similarità semantica e tre di misure di relazione semantica, molte delle quali sono state descritte nei precedenti paragrafi e tutte basate sul database lessicale di WordNet. WordNet:: Similarity è disponibile liberante on-line al sito htto:// wn-similarity.sourcefource.net .

Il codice del generico algoritmo di disambiguazione è riportato di seguito.

```
foreach sense s_ti of target word w_t

{
   set score_i = 0
   foreach word w_j in window of context
   {
      skip to next word if j == t
      foreach sense s_jk of w_j
      {
         temp_score[j] = relatedness(s_ti, s_jk)
      }
      winning_score = highest score in array temp_score[]
      if (winning score > threshold)
      set score_i = score_i + winning_score
   }
}
return i, such that score_i ≥ score_j, ∀j, 1 ≤ j ≤ n, n =
number of words in sentence
```

Misura di relazione semantica basata sul Vettore-Contesto

Patwardhan nel 2006 in [**35**] introduce una nuova misura di relazione semantica ispirandosi all'algoritmo di Gloss Overlap proposto da Banerjee e Pedersen e all'*Harris Distributional Hypothesis* [**39**]. L'ipotesi distributiva suggerisce che termini simili tendono ad essere presenti in contesti linguistici simili. Laudauer e Dumains nel 1997 in [**40**] descrivono un metodo basato su un "vettore-contesto" che simula l'apprendimento del significato dei termini all'interno di un testo "grezzo". Schutze nel 1998 in [**41**], mostrò inoltre come i vettori costruiti a partire da un contesto di un termine siano un'utile rappresentazione del significato del termine stesso.

La misura di relazione semantica di *Gloss Vector* proposta da Patwardhan si basa su un vettore di secondo ordine di co-occorrenze creato in combinazione con la struttura e il contenuto di WordNet. Tale misura cattura l'informazione semantica dei concetti, separandola dall'informazione testuale che questi possiedono all'interno di un testo.

La misura proposta è estremamente flessibile e può essere applicata ad ogni categoria sintattica.

I vettori-contesto sono strumenti ampiamente utilizzati negli ambiti del *Natural Language Procesing* e dell'*Information Retrieval*. Molto spesso essi sono utilizzati per rappresentare le co-occorrenze di primo ordine, ovvero le occorrenze delle singole parole che compaiono all'interno di un testo. Per esempio, *police* e *car* saranno co-occorrenze di primo ordine, poiché generalmente compaiono insieme. Un vettore di co-occorrenze di primo ordine per una data parola indicherà, cioè, semplicemente tutte le co-occorrenze di primo ordine del termine all'interno di un testo.

Consideriamo ora per esempio, *car* e *mechanic*, se sono co-occorrenze di primo ordine, allora *mechanic e police* saranno co-occorrenze di secondo ordine poiché entrambe sono legate a *car* come co-occorrenze di primo ordine.

Il metodo di Schultze, si fonda sulla creazione di un *Word Space*, che essenzialmente rappresenta una matrice di co-occorrenze, in cui ogni riga può essere vista come un vettore di co-occorrenze di primo ordine. Ogni cella di tale matrice, inoltre, rappresenta la frequenza con la quale i termini compaiono gli uni vicino agli altri all'interno del corpus. Lo spazio delle parole, di solito è molto ampio e sparso, poiché all'interno di un corpus esistono molti termini diversi e molti di essi non compaiono vicino ad altri. Per ridurre le dimensioni le dimensioni di tale matrice i termini come *the, for, a,* ecc… non vengono considerati. Dato un *Word Space*, un contesto può essere rappresentato attraverso un vettore di co-occorrenze di secondo ordine. Ciò è realizzato a partire dal vettore di primo ordine.

Per esempio, supponiamo di avere il contesto seguente:

The painting were displayed in the art gallery.

Il vettore di secondo ordine, si otterrà sommando i vettori di primo ordine di *painting, display, art, e gallery.* Le parole *the, were* e *in* saranno ovviamente escluse. La rappresentazione di tale vettore è mostrata in figura 3.18.

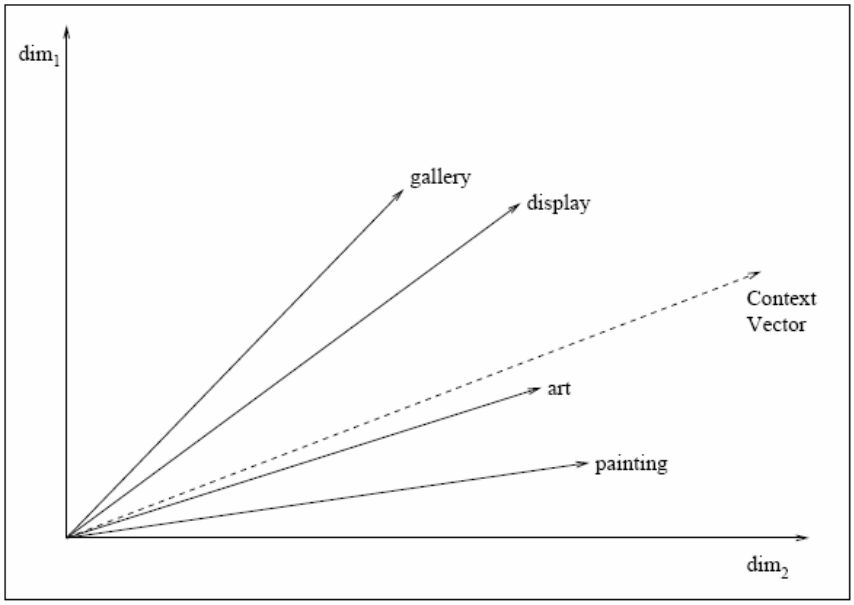


Figura 3.18-Creazione di un vettore di contesto a partire dai vettori di primo ordine dei singoli termini.

Intuitivamente, si può osservare come l'orientamento del vettore dia un'indicazione riguardo al dominio o alla topica associata al contesto stesso. Da qui due vettori di contesto che formano un angolo simile con l'asse delle ascisse conterranno termini pertinenti e simili nel significato.

Patwardlhan essenzialmente, misura la relazione semantica tra due sensi, *s1* ed *s2*, attraverso il confronto dei vettori rappresentanti le loro glosse (*Gloss Vector*).

Ogni *Gloss Vector* viene creato a partire dal singoli vettori di co-occorrenze. Successivamente seguendo l'algoritmo di Pedersen, Patwardhan si estende tale vettore con le glosse dei termini ad essa relazionati.

Infine, si calcola la relazione tra i due sensi *s1* ed *s2* come il coseno dell'angolo tra il vettore di glossa di *s1* e quello di *s2*:

$$Relatedness(s_1, s_2) = \frac{\vec{s_1} \cdot \vec{s_2}}{|\vec{s_1}||\vec{s_2}|}$$

Uno dei vantaggi dell'approccio a *Gloss Vector*, rispetto per esempio all'algoritmo di *Gloss Overlap* consiste nel fatto che il metodo vettoriale non ha il limite di dover individuare i *matching* esatti tra due glosse.

3.1.3 Algortimo Graph-Based di Mihalcea

L'algoritmo proposto da R. Mihalcea in [**42**], rappresenta un metodo non supervisionato basato sui grafi, che consente di disambiguare una sequenza di termini contemporaneamente, sfruttando le relazioni tra i termini stessi. L'algoritmo essenzialmente, data una sequenza di parole $W= \{w1, w2, \ldots, wn\}$, dove ad ogni parola wi è associata una etichetta $L_{wi} = \{ l_{wi,1}, l_{wi,2}, \ldots, l_{wi,Nwi} \}$, crea un grafo etichettato $G = (V, E)$, tale che esiste un vertice v appartenente a V per ogni possibile etichetta $l_{wi,j}$ con $i = 1, .., n$ $1 j = 1, \ldots, N_{wi}$. Le dipendenze esistenti tra coppie di etichette sono rappresentate tramite, archi appartenenti all'insieme E, definito l'insieme delle coppie di vertici $V x V$. La figura 3.19 mostra un esempio di struttura a grafo, ottenuto da un insieme di etichette per una sequenza di quattro termini.

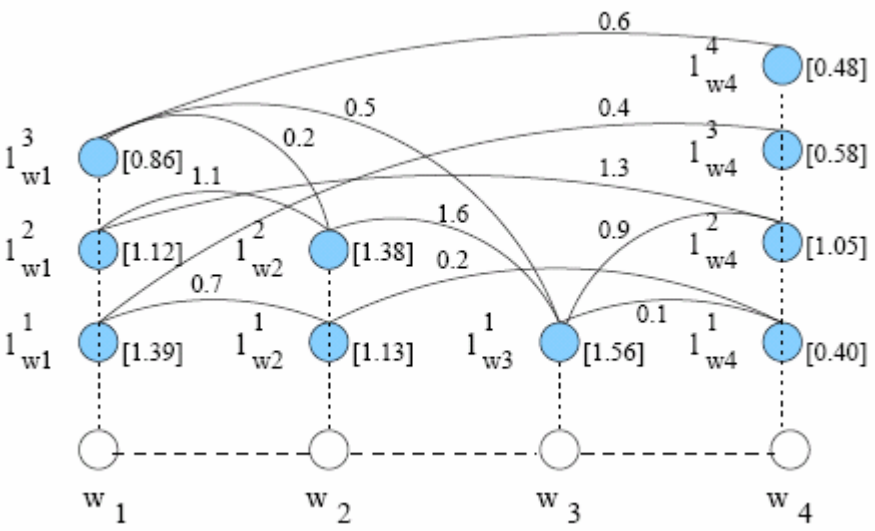


Figura 3.19- Esempio di grafo costruito sulle possibili etichette associate a quattro termini

Si noti come il grafo non sia totalmente connesso, e non tutte le coppie di etichette siano legate da una relazione.

Dato un grafo di etichette associato ad una sequenza di termini, l'algoritmo modella, in maniera iterativa, un percorso casuale guidato da una distribuzione stazionaria della probabilità delle etichette, rappresentata attraverso il punteggio che viene associato ad ogni vertice del grafo. Questi punteggi sono utilizzati per identificare le etichette più probabilmente corrette per ciascun termine, allo scopo di annotare tutti le parole della sequenza in input. Per esempio, se consideriamo il grafo in figura 3.19, il termine *w1* sarà assegnato all'etichetta $l_{w1,}$

1, poiché il punteggio associato a tale etichetta (1.39) rappresenta il valore massimo tra i punteggi di tutte le etichette associabili a tale termine.

Una proprietà importante di questo algoritmo, che lo rende particolarmente adatto per disambiguare sequenze di termini, è il fatto che esso considera l'informazione totale all'interno del grafo, realizzando così un processo di disambiguazione globale. Attraverso percorsi casuali compiuti nel grafo etichettato, l'algoritmo tenta di sfruttare le dipendenze tra tutte le etichette. Ciò rende, a parere di Mihalcea, tale algoritmo superiore rispetto ad altri approcci che eseguono il processo di disambiguazione considerando un termine alla volta.
L'idea di base implementata attraverso questo algoritmo, è quella di sfruttare un processo di "votazione". Quando un vertice è connesso ad un altro, esso esprime un voto nei confronti dell'altro vertice con cui è collegato. Più alto è il numero di voti assegnati da un vertice e maggiore sarà l'importanza di quest'ultimo.
Dato un grafo *G = (V, E)*, si definisce *In (Va)* come l'insieme dei vertici che puntano al vertice *Va* (ovvero i suoi predecessori), e *Out (Va)* l'insieme dei vertici a cui punta il vertice *Va* (ovvero i suoi successori). Il punteggio associato al vertice *Va*, è definito attraverso la seguente formula:

$$P(V_a) = (1-d) + d * \sum_{V_b \in In(V_a)} \frac{P(V_b)}{|Out(V_b)|}$$

dove *d* è un parametro fissato e compreso tra 0 e 1. Per applicare tale algoritmo ai processi di disambiguazione del testo, è necessario avere una risorsa da cui poter estrarre le informazioni relative ai sensi dei termini e alle relazioni di similarità che li legano. Tale risorsa può essere WordNet. Le dipendenze tra i vari sensi possono essere derivate in vari modi, in base al tipo di risorsa disponibile. La similarità tra due significati, se consideriamo come risorsa WordNet, può essere determinata attraverso una variante dell'algoritmo di Lesk.
L'algoritmo riceve come input un testo. A partire da questo viene costruito un grafo etichettato, dove ogni vertice corrisponde ad un possibile senso dei termini. Successivamente, si calcolano i pesi degli archi che collegano synset posti ad una distanza non superiore a *MaxDist* (ottenuta sperimentalmente), utilizzando la misura di similarità basata sull'algoritmo

di Lesk. Una volta costruito il grafo, si applica l'algoritmo a quest'ultimo, e si calcola un punteggio per ciascun nodo del grafo, in base alla formula vista precedentemente. Infine, per ogni termine da disambiguare, si seleziona il vertice (ovvero il senso), con il punteggio più alto associato e si etichetta la parola con tale synset. Di seguito si riporta il codice dell'algoritmo:

Input: Sequence $W = \{w_i | i = 1..N\}$
Input: Admissible labels $L_{w_i} = \{l^t_{w_i} | t = 1..N_{w_i}\}, i = 1..N$
Output: Sequence of labels $L = \{l_{w_i} | i = 1..N\}$, with label l_{w_i} corresponding to word w_i from the input sequence.

Build graph G of label dependencies

for $i = 1$ to N **do**
  for $j = i + 1$ to N **do**
    if $j - i > MaxDist$ **then**
      $break$
    end if
    for $t = 1$ to N_{w_i} **do**
      for $s = 1$ to N_{w_j} **do**
        $weight \leftarrow Dependency(l^t_{w_i}, l^s_{w_j}, w_i, w_j)$
        if $weight > 0$ **then**
          $AddEdge(G, l^t_{w_i}, l^s_{w_j}, weight)$
        end if
      end for
    end for
  end for
end for

Score vertices in G

repeat
  for all $V_a \in Vertices(G)$ **do**
    $WP(V_a) = (1 - d) + d * \sum_{V_b \in In(V_a)} w_{ba} WP(V_b) / \sum_{V_c \in Out(V_b)} w_{bc}$
  end for
until convergence of scores $WP(V_a)$

Per comprendere meglio il funzionamento dell'algoritmo consideriamo il seguente esempio: supponiamo di dover disambiguare la seguente frase " *The church bells no longer rung on Sunday"*. Per semplicità, non si considereranno più di tre sensi per termine. I sensi considerati ed estratti da WordNet sono mostrati in figura 3.20.

The **church bells** no longer **rung** on **Sundays**.

church
1: one of the groups of Christians who have their own beliefs and forms of worship
2: a place for public (especially Christian) worship
3: a service conducted in a church

bell
1: a hollow device made of metal that makes a ringing sound when struck
2: a push button at an outer door that gives a ringing or buzzing signal when pushed
3: the sound of a bell

ring
1: make a ringing sound
2: ring or echo with sound
3: make (bells) ring, often for the purposes of musical edification

Sunday
1: first day of the week; observed as a day of rest and worship by most Christians

Figura 3.20- Sensi associati ai termini dell'esempio

Tutti i sensi dei termini sono inseriti all'interno del grafo come vertici, e le relazioni fra di essi come archi con i rispettivi pesi indicanti i valori di similarità tra i sensi. Ovviamente, nel caso in cui la similarità sia uguale a zero non viene realizzato un alcun arco. Come risultato, si ottiene di conseguenza, il grafo in figura 3.21.

Con l'esecuzione dell'algoritmo, si determinano i punteggi associati ai vari nodi, e successivamente si annotano i termini con il senso più probabile. Di conseguenza si ottengono le seguenti annotazioni: *church#2 bell#1 no_longer_rung#3 on_Sunday#1*, le quali risultano essere effettivamente corrette.

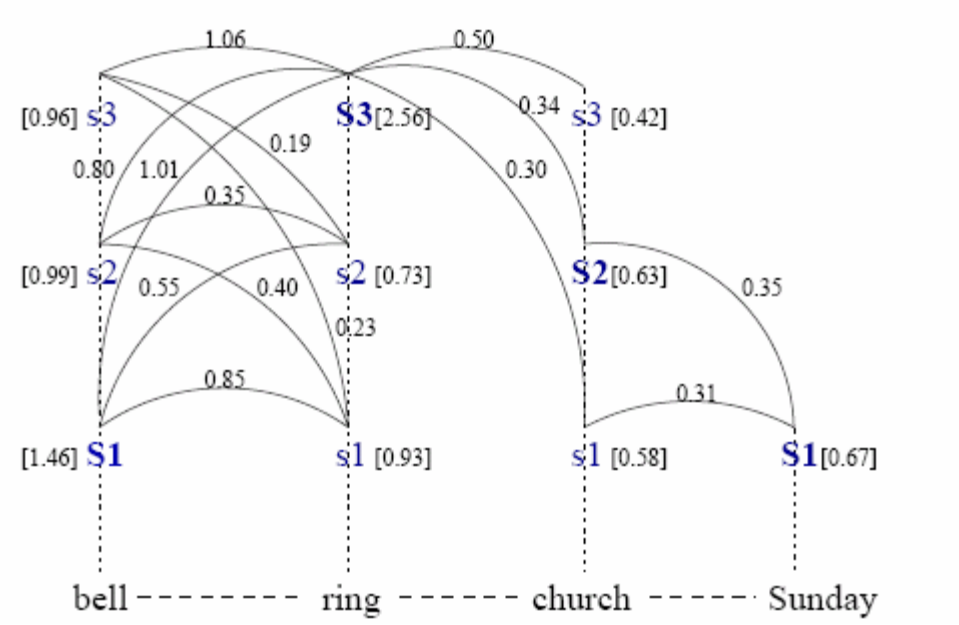


Figura 3.21- Grafo risultante dall'esempio considerato

Tale algoritmo è stato presentato e sperimentato durante il Senseval-2 ottenendo i seguenti risultati:

Part-of speech	Random baseline		Individual (Lesk)		Sequence (graph-based)	
	P	R	P	R	P	R
Noun	41.4%	19.4%	50.3%	23.6%	57.5%	27.0%
Verb	20.7%	3.9%	30.5%	5.7%	36.5%	6.9%
Adjective	41.3%	9.3%	49.1%	11.0%	56.7%	12.7%
Adverb	44.6%	5.2%	64.6%	7.6%	70.9%	8.3%
ALL	37.9%	37.9%	48.7%	48.7%	**54.2%**	**54.2%**

Tabella 3.7 Risultati ottenuti dall'algoritmo di Sequenze Graph-Based durante il Senseval-2

3.2 Algoritmi Supervisionati

Gli algoritmi di disambiguazione del testo supervisionati, si basano su risorse di conoscenza lessicale ottenute da grandi corpus hand-tagged, ovvero contenenti già le informazioni relative al significato di ciascun termine e alle relazioni che li legano. I metodi supervisionati, si

basano sul fatto che, una volta che si possiede l'informazione relativa alla lista dei sensi necessari, per disambiguare i termini all'interno di una frase, basterà confrontarli con il proprio modello dei sensi. Per esempio, per disambiguare il termine *pages* all'interno della domanda *"Can you get me a short version of this dissertation, of 20 pages more or less?"*, basterà confrontare l'informazione estratta da questa frase, con i dati che si hanno a disposizione per ogni senso di page nei corpus etichettati.

Proprio per questa loro caratteristica, tali algoritmi per molto tempo sono stati applicati solo in contesti limitati, in quanto nuovi ambiti avrebbero richiesto dei dati di *training* non disponibili. Tuttavia, motivata dalle migliori prestazioni di questa tipologia di algoritmi rispetto a quelle ottenibili con algoritmi non supervisionati, negli ultimi anni la ricerca si è concentrata verso l'individuazione di meccanismi che consentano di acquisire in maniera automatica le risorse necessarie al processo di disambiguazione.

Prima di proseguire, tuttavia si ritiene opportuno riportare brevemente le collezioni di testi, utilizzate negli algoritmi che verranno descritti successivamente. Tra i principali corpus *hand-tagged* oggi liberamente disponibili ricordiamo:

- **SemCor Corpus** : fu ideato da Miller nel 1993, ed essenzialmente consiste in un sottoinsieme del *Corpus di Brown*. Esso contiene un numero di testi comprendenti circa 200000 parole dove quest'ultime sono state tutte manualmente etichettate con il senso corretto di WordNet 1.6. SemCor è stato prodotto dagli stessi creatori di WordNet. Esso, attualmente, rappresenta uno dei corpus più utilizzati per testare sia algoritmi supervisionati che non supervisionati.
- **DSO Corpus**: Il corpus ***D****efence* ***S****cience* ***O****rganization* è composto da 191 termini polisemici estratti dal *Wall Street Journal* e dal C*orpus di Brown*. Esso etichetta 192800 occorrenze di questi termini attraverso i sensi di WordNet 1.6, ottenendo in media quindi più di 1000 istanze per ogni termine.
- **Corpus dei Senseval**: durante i Senseval 1, 2, 3 sono stati prodotti una serie di corpus *hand-tagged* in inglese per consentire di testare i vari algoritmi presentati.

Il contenuto del seguente paragrafo può essere suddiviso in due parti: nella prima si descriveranno brevemente i primi approcci di disambiguazione completamente supervisionati, nella seconda parte, invece, si descriveranno in maniera più dettagliata le evoluzioni di tali metodi verso una metodologia minimamente supervisionata.

3.2.1 Metodi completamente supervisionati

Metodo del senso più frequente

Tale metodo consiste semplicemente nel valutare, all'interno dei training-data il numero di assegnazioni per un termine di ogni suo possibile senso , e assegnare al termine il suo senso più frequente. In caso di parità di frequenza, l'algoritmo sceglie un senso a caso.
Tale metodo, è implicitamente contenuto all'interno di WordNet, il quale ordina i sensi in base alla loro frequenza di utilizzo.

Liste di Decisione

Una Lista di Decisioni (DL), consiste in un insieme di regole ordinate, nella forma [*feature*-valore; senso; peso]. All'interno di questo contesto, l'algoritmo delle DL consiste nei seguenti passi: i dati di *training* sono utilizzati per determinare le *feature* che consentono di distinguere due sensi fra loro. Queste vengono successivamente pesate attraverso una misura presentata da Yarowsky in [**43**], la quale indica la probabilità di ottenere un determinato senso considerando una particolare valore della *feature*. La lista di tutte le regole, è quindi ordine decrescente rispetto al valore dei suoi pesi. Quando si esegue l'algoritmo su nuovi testi, viene controllata la lista di decisioni, e le *feature* con il peso più alto vengono confrontate con il testo, e selezionano il senso più probabile.
La formula originale di Yarowsky, può essere adattata allo scopo di gestire il in cui si verifichino più sensi egualmente probabili. In questo caso, il peso di un senso quando si verifica una determinata *feature* all'interno del contesto, è calcolato come il logaritmo della probabilità di ottenere il senso tramite la *feature f*, diviso per la somma della probabilità degli altri sensi dati sempre da *f*. Riassumendo il peso di un senso *sk* è ottenibile tramite la seguente formula:

$$weight(s_k) = \arg\max_f \ \log\left(\frac{P(s_k|f)}{\sum_{j\neq k} P(s_j|f)}\right)$$

Tale formula spesso necessita l'uso di alcuni tipi di *smoothing* come per esempio quello di sostituire il valore del denominatore con 0,1 nel caso in cui si ottenga una frequenza pari a zero. Il valore 0,1 è stato determinato empiricamente attraverso degli esperimenti.
Può accadere che, per una determinata *feature*, esista una sola occorrenza all'interno del contesto, oppure che il peso di tutti i sensi ad essa associati sia minore di zero. In tal caso,

bisogna valutare se convenga o meno includere tale *feature* all'interno della lista di decisioni. Il parametro che esprime questo tipo di decisioni viene indicato come *pruning* . Tale parametro viene utilizzato soprattutto in fase sperimentale e se applicato indica la decisione di non considerare le *feature* con le caratteristiche precedenti. Ovviamente utilizzare *pruning* significa comunque perdere in termini di copertura del processo di disambiguazione disambiguazione a vantaggio di una maggior precisione.

Il metodo Naive Bayes

Il metodo di Naive Bayes, si basa sulla probabilità condizionale che si presenti il senso *sk* data la *feature fi* all'interno del contesto. Esso assume che le varie *feature* siano fra loro indipendenti. Tale supposizione in realtà non è mai verificata, ma è stato dimostrato in (Mooney, 1996; Ng 1997; Leocock 1998) che consente di ottenere comunque buoni risultati. Al suo termine l'algoritmo restituisce il senso *sk* che massimizza la probabilità ottenuta tramite la seguente formula:

$$weight(s_k) = P(s_k) \ \prod_{i=1}^{m} P(f_i|s_k)$$

Il valori $P(s_k)$ e $P(f_i \mid s_k)$ si ottengono a partire dai dati di *training*, utilizzando le frequenze relative di ciascun senso. Anche tale formula tuttavia richiede, tuttavia, alcuni interventi di *smooting* alla scopo di evitare che possa ritornare zero a causa di una sola *feature*. Un metodo utilizzato in (Ng, 1997; Escudero 2000) può essere quello di rimpiazzare lo zero con $P(sk)/ N$ dove N è il numero totale di contesti contenuti nei dati di training.

Modello dello Spazio Vettoriale

Uno spazio vettoriale rappresenta, tramite dei vettori, tutte le occorrenze dei termini all'interno di un contesto, dove ogni *feature* può assumere il valore di 0 oppure 1, a seconda del fatto che sia rispettivamente, presente o assente nel contesto stesso. Per ogni senso all'interno dei dati di *training*, viene ricavato un *centroide* nello spazio vettoriale indicato con C_{sk}. Tali centroidi, sono poi confrontati con i vettori di contesti da disambiguare, attraverso il calcolo del coseno della funzione di similarità riportata di seguito:

$$weight(s_k) = cos(\vec{C_{s_k}}, \vec{f}) = \frac{\vec{C_{s_k}} \cdot \vec{f}}{|\vec{C_{s_k}}||\vec{f}|}$$

Il centroide con peso maggiore assegnerà il senso al termine target. In questo metodo, non viene richiesta l'applicazione di alcun *smoothing*.

Metodo AdaBoost

Il metodo AdaBoost, è un metodo generale per ottenere una classificazione altamente accurata delle regole delle DL attraverso la combinazione lineare di varie classificazioni che consentono di ottenere un'accuratezza solo parziale. Tale algoritmo è capace di lavorare in maniera efficiente in spazi di *feature* con un alto livello di dimensionalità, ed ha ottenuto buoni risultati nell'ambito della disambiguazione (Esculero 2000). Si utilizza un parametro di *smoothing* che viene fissato ad un valore di *default* e l'algoritmo viene eseguito circa 200 volte per ciascun termine. L'algoritmo si conclude assegnando al termine il senso con la predizione più alta.

Metodo IRST-Kernel

Tale metodo è stato proposto da Strapparava nel 2004 in [**44**]. Egli utilizza una funzione di kernel per integrare sorgenti d'informazione eterogenee. I kernel, nell'ambito della disambiguazione dei sensi, sono funzioni matematiche in grado di modellare i sensi dei termini. In particolare esistono varie tipologie di kernel, ciascuna definisce una *feature* per la distinzione dei sensi. Per esempio, esistono kernel rappresentanti la *feature* di dominio di appartenenza dei sensi, kernel modellanti la categoria sintattica ecc…. Questi possono poi essere combinati fra loro per realizzare una misura di confronto fra due concetti più completa. Il kernel di Strapparava rappresenta un'estensione di due kernel fondamentali:

- **Il Kernel *syntagmatic*** : valuta la similarità fra sequenze di parole. L'idea chiave consiste nel fatto che le relazioni *syntagmatic* tra due contesti possono essere modellate in base al numero di sequenze di termini che essi hanno in comune. Il *Word Similarity Kernel*, valuta la similarità tra i termini attraverso il calcolo delle sottosequenze di caratteri comuni, all'interno di una finestra di contesto di un termine. Questo principio è stato modificato suddividendo ulteriormente il kernel in un "*collocation kernel* "(basato sulla sequenza dei lemmi) e un "*PoS kernel"* (basato sulla sequenza della categorie sintattiche). Viene inoltre applicata una soglia di similarità basata su con lo scopo di includere i *matching* di termini equivalenti: si è

stabilito che, i termini che superano questa determinata soglia, sono da ritenersi equivalenti.

- **Il kernel *paradigmatic***: valuta la similarità fra contesti. Tale kernel introduce l'informazione di dominio ed è realizzato in addizione ad altri due: un kernel "*bag of word"* e un kernel "*Latent Semantic Indexing "(*LSI).

GAMBL-AW

GAMBL-AW (Decadt 2004) è un sistema ispirato dai metodi kernel, che realizza la fusione tra differenti corpus come per esempio SemCor, i corpus utilizzati e forniti durante i vari Senseval ecc..Dagli esempi contenuti in questi corpus vengono estratte due *feature*: il contesto locale, e le parole chiave del contesto.

3.2.2 Metodi minimamente supervisionati

SenseLearn

SenseLearn, è un sistema realizzato nel 2004 da R.Mihalcea e E.Faruque e descritto in [**45**]. Esso rappresenta un tentativo di ridurre al minimo la supervisione richiesta dai tradizionali algoritmi di disambiguazione supervisionati. SenseLearner ha l'obiettivo di utilizzare solo parzialmente dati etichettati, e di rendere l'algoritmo sufficientemente generale per poter essere applicato a qualsiasi tipo di disambiguazione. Esso utilizza sia SemCor che WordNet, quest'ultimo durante il processo di *generalizzazione semantica* dei termini non presenti all'interno del corpus SemCor. L'input dell'algoritmo di disambiguazione consiste in una testo non annotato. L'output è l'annotazione di tutte i termini all'interno del testo indipendentemente dalla loro categoria sintattica. L'algoritmo prevede una fase iniziale di pre-elaborazione, durante la quale si determina, in maniera automatica, la categorie sintattica di ciascun termine all'interno del testo di input.
Dopo questa prima fase iniziale, l'algoritmo si suddivide nei seguenti due passi:

- ***Creazione del modello semantico del linguaggio***. Il primo passo consiste nella creazione del modello semantico di linguaggio per ogni categoria sintattica, partendo dal copus annotato. Tali modelli sono poi successivamente usati, per annotare le

parole all'interno del testo da disambiguare. Questo passo è quindi applicabile alle sole parole che compaiono almeno una volta all'interno dei dati di *training* (in questo caso di SemCor).

- ***Generalizzazione semantica***. Tale fase si realizza utilizzando le dipendenze sintattiche all'interno della rete concettuale considerata (in questo caso WordNet). Questa fase, viene viceversa applicata a tutte le parole che non compaiono all'interno di SemCor .

Di seguito vedremo in dettaglio le due fasi principali dell'algoritmo.

L'obiettivo della prima fase è quello di realizzare un modello globale per ogni categoria sintattica, che verrà utilizzato per disambiguare i termini contenuti nel testo di input.

Tale modello tuttavia può essere utilizzato solo per disambiguare termini che compaiono almeno una volta all'interno del corpus SemCor.

Durante questa fase si separano i vari termini all'interno del corpus annotato, in base alla categoria sintattica a cui appartengono, e si creano i modelli, seguendo particolari regole descritte in [**45**]. Successivamente, ogni termine che compare all'interno del corpus, viene associato un vettore di *feature*. Ogni vettore risulta quindi identificato tramite la parola target e il senso corrispondente.

Per l'annotazione del testo di input, invece, vengono creati vettori simili per tutte le parole del testo da disambiguare. I vettori sono immagazzinati in file differenti, in base alla loro categoria sintatticata. Per l'annotazione si utilizza l'algoritmo di predizione di Timbl [**46**], la cui applicazione è stata dimostrata essere utile per i processi di disambiguazione in [**47**].

Successivamente, ogni vettore di termine (contenente le *feature* relative a quel termine in quel contesto), viene etichettato con il senso predetto attraverso tale metodo. Se il termine predetto dall'algoritmo di apprendimento coincide con il termine target all'interno del vettore di *feature* del testo di input, allora il senso predetto è usato per annotare i termine target. Viceversa se la parola predetta non coincide con il termine target, non viene prodotta alcuna annotazione, e il termine verrà annotato nella fase successiva.

La fase di generalizzazione semantica si ispira all'algoritmo di Lin [47], considerando la dipendenza sintattica tra i termini, e prendendo in considerazione la gerarchia concettuale ottenibile tramite la rete semantica di WordNet. Tale fase dell'algoritmo è in grado di disambiguare solo i termini non presenti all'interno del corpus considerato (SemCor). L'algoritmo di generalizzazione semantica, si divide a sua volta in due parti principali:

1. *Fase di training*. Allo scopo di combinare la dipendenza sintattica tra le parole e la gerarchia concettuale ottenuta attraverso le relazioni di iperonimia di WordNet, vengono eseguiti i seguenti passi:
    I. Vengono rimosse le etichette di annotazione dai termini all'interno del SemCor, e se ne ricava un testo grezzo formato da una frase per riga.
    II. Viene effettuato il *parse* delle frasi attraverso l'uso del *parse di Link*, e vengono salvate tutte le coppie di termini relazionate tramite dipendenze (per esempio dipendenza oggetto-verbo).
    III. Si somma la categoria sintattica e l'informazione sul senso (come viene fornita da SemCor), ad ogni parola all'interno della coppia di dipendenza.
    IV. Per ogni nome o verbo in una coppia di dipendenze, si ricava l'albero di ipernimi del termine. Si costruisce un vettore composto dalla parole stesse, dalla loro categoria sintattica, dal loro senso in WordNet e da un riferimento a tutti i synset ipernimi in WordNet.
    V. Per ogni coppia di dipendenza, si genera un vettore di *feature* per i sensi che compaiono nei dati di training, e un vettore negativo per tutti i restanti sensi possibili.
2. *Fase di Test.* Successivamente alla *fase di training*, viene utilizzato un vettore di *feature* generalizzato, allo scopo di assegnare un senso ai termini nuovi contenuti nel testo. Tale fase si scompone nei seguenti passi:
    I. Viene effettuato il *parse* di ogni frase all'interno del file attraverso il *parser di Link*, e vengono salvate le conseguenti coppie di dipendenze.
    II. Si parte dal termine più a destra nella frase e si ricercano tutte i termini ad esso connessi.
    III. Per ogni coppia di dipendenze, si crea un vettore di *feature* per tutte le combinazioni possibili di sensi. Per esempio, se la prima parola in una coppia ha due sensi possibili, e la seconda ne possiede tre, verranno creati in tutto sei vettori.
    IV. Infine, i vettori creati vengono tutti passati attraverso un motore di apprendimento basato sulla memoria, il quale etichetterà ogni vettore di *feature* con un'etichetta in base all'informazione appresa dai dati di *training.*

Per comprendere meglio il funzionamento dell'algoritmo consideriamo il seguente esempio. Consideriamo la frase estratta da SemCor: *" The fulton Country Grand Jury, said Friday an investigation of Atlanta's recent similarity election produced " no evidence" that any irregularities took place."* . Nell'esecuzione del primo passo, per semplicità, considereremo solo la relazione verbo-oggetto tra *produce* ed *evidence*. Si estraggono, quindi, i sensi associati alle due parole, dal SemCor. A questo punto, combinando la conoscenza sintattica derivante dal *parser* con la conoscenza semantica, estratta da Semcor, si ottiene una relazione oggetto-verbo, tra *produced#v#4* e *evidence#n#1*.

Successivamente si identifica l'albero degli ipernimi per ciascuna delle due parole, e si crea il seguente vettore di *feature,*:

Os, produce#v#4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, produce#v#4, expose#v#3, show#v#4, evidence#n#1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, evidence#n#1, information#n#3, cognition#n#1, psychological_feature#n#1

dove "Os" indica una relazione oggetto-verbo, e gli zeri sono utilizzati per riempire gli elementi nulli poiché il vettore di *feature* ha una dimensione costante pari ad
20 elementi per ogni parola.

SenseLearn è stato proposto e testato durante il Senseval-3, ottenendo i seguenti risultati:

Class	Precision	Fraction of Recall
Nouns	69.4	31.0
Verbs	56.1	20.2
Adjectives	71.6	12.2
Total	64.6	64.6

Tabella 3.8- Risultati ottenuti dall'algoritmo di SenseLearn durante il Senseval-3

Si noti come, anche in questo caso, confermando i risultati ottenuti nei precedenti algoritmi di disambiguazione (sia supervisionati che non), si ottengono risultati discreti ma che confermano l'idea di come un processo di completa disambiguazione sia praticamente irrealizzabile, in particolar modo per categorie sintattiche come i verbi, in cui il numero di sensi diversi è elevato.

Algoritmo Structural Semantic Interconnections

L'algoritmo di ***S**tructural **S**emantic **I**nterconnection* (SSI), è stato ideato e realizzato da un gruppo di ricercatori del dipartimento d'Informatica dell'Università "La Sapienza" di Roma, a cui fanno capo R.Navigli e P.Velardi.

Tale algoritmo descritto in [**48**], nasce dalle considerazioni, che gli approcci alla disambiguazione del testo sono basati su conoscenze semantiche estratte manualmente o automaticamente da risorse lessicali come per esempio WordNet.

Più recentemente, l'uso di tecniche di *machine learning* e di metodi statistici ed algebrici, ha prevalso sui metodi classici basati sulla conoscenza di base. Tali metodi si basano spesso su dati di *training* estratti da archivi di documenti o pagine web.

SSI è un approccio *knowledge-based* al problema della disambiguazione del testo. Essenzialmente utilizza dei grafi allo scopo di descrivere gli oggetti da analizzare, ovvero i sensi delle parole, e una "*grammatica*" per determinare i percorsi tra i grafi rilevanti dal punto di vista semantico. Il criterio di classificazione dei sensi, si basa sul numero e sulla tipologie di interconnessioni determinate. La rappresentazione dei vari sensi delle parole all'interno di un grafo viene automaticamente realizzata attraverso diverse risorse disponibili come ontologie lessicali, glossari ecc… poi combinati in parte manualmente e in parte automaticamente.

L'approccio alla WSD si realizza attraverso una *structural pattern recognition* .

Tale approccio è stato provato essere efficace quando gli oggetti che devono essere rappresentati, contengono un'intrinseca ed identificabile organizzazione interna. Per questa tipologia di oggetti infatti, una rappresentazione così detta "*flat*", come ad esempio quella vettoriale, può causare una perdita considerevole di informazione, avendo quindi impatti negativi sulle prestazioni della classificazione. Il compito della classificazione, in un sistema di *structural pattern recognition*, viene implementata attraverso l'uso di grammatiche che incarnano precisi criteri di discriminazione tra le differenti classi.

In questo caso, i sensi delle parole ricadono sotto la categoria di oggetti, la quale può potenzialmente descrivere nel modo migliore le caratteristiche strutturali in essi contenuti. Il campo della linguistica computazionale, offre una vasta varietà di risorse lessicali di base, le quali rappresentano un'ideale punto d'inizio per la costruzione di strutture che rappresentino i sensi dei vari termini.

La rappresentazione grafica dei sensi dei termini, viene automaticamente generata da conoscenze lessicali di base. Quest'ultime vengono costruite integrando diverse risorse on-line. In particolare sono state utilizzate le seguenti risorse: WordNet 2.0, testi annotati, i quali forniscono esempi di contesti per determinati sensi dei termini, e dizionari di collocazioni dove le collocazioni sono liste di parole che appartengono ad un dato dominio semantico. Il risultato è quindi una *Lexical Knowledge Base* (LKB) la quale include relazioni semantiche, esplicitamente codificate in WordNet, e relazioni semantiche estratte da corpus annotati.
Tale LKB è utilizzata per generare un grafo direzionato dove ogni nodo rappresentante i sensi delle parole ed è opportunamente etichettato. Tali grafi prendono il nome di *grafi semantici*, poiché rappresentano una concettualizzazione alternativa di un determinato elemento lessicale.
La figura 3.22, mostra un esempio di grafo semantico generato per il senso #1 (*vehicle*) ed il senso #2 (*connector*) del lemma *bus.* I nodi rappresentano i concetti, ovvero i synset di WordNet, mentre gli archi rappresentano le varie relazioni semantiche esistenti tra i vari concetti. In ogni grafo vengono inclusi solo i nodi con una distanza massima di tre dal nodo centrale. Tale distanza è stata determinata sperimentalmente come ottimizzante per il processo di disambiguazione. L'insieme di relazioni semantiche incluse in un grafo sono: iperonimia, iponimia, meronimia, olonimia,pertinenza, attributo, similarità, glossa, contesto e dominio.

Gli input dell'algoritmo sono:

- *T* ovvero la lista dei termini appartenenti al contesto considerato;
- *t* ovvero il termine appartenente a *T* che deve essere disambiguato;
- $S_{1,t}, S_{2,t}, \ldots\ldots, S_{n,t}$ i quali rappresentano le specifiche strutturali dei possibili concetti per *t* (ovvero il grafo semantico);
- *I* ovvero il contesto semantico, cioè una lista di specifiche strutturali dei concetti associati ad o ad alcuni, dei termini inclusi in *T*.
- *G* è una grammatica descrivente le relazioni rilevanti tra le specifiche strutturali, ovvero le interconnessioni semantiche tra i grafi.

Lo scopo dell'algoritmo sarà quindi quello di determinare e quantificare come le specifiche strutturali in *I*, coincidano con ogni elemento di *G.*
L'output dell'algoritmo, è quindi l'elemento di *G*, *S* che realizza il miglior *matching.*
L'algoritmo SSI può essere suddiviso in una fase di inizializzazione, ed un fase iterativa.

In un generico passo iterativo dell'algoritmo, l'input è una lista di termini co-occorrenti *T* e una lista di sensi associati *I,* ovvero l'interpretazione semantica di *T.*

Viene inoltre mantenuto in memoria un insieme di termini *P* definiti "pendenti"; *I* come già detto, è il contesto semantico di *T,* ed è utilizzato ad ogni passo, per disambiguare nuovi termini in *P.*

L'algoritmo lavora in maniera iterativa, nel senso che ad ogni passo o uno dei termini è rimosso da *P* (ovvero un termine pendente è stato disambiguato) o la procedura termina, in quanto nessun altro termine può essere disambiguato. L'output è una lista modificata di sensi associati con i termini di input di *T.* Inizialmente la lista I contiene solo i termini monosemici appartenenti a *T,* o alcuni synset forniti inizialmente. Se non è presente alcun termine monosemico o se nessun synset viene inizialmente fornito, l'algoritmo seleziona il senso più probabile (per esempio quello indicato in WordNet con frequenza maggiore di utilizzo), del termine *t* meno ambiguo (ovvero con il numero minore di sensi associati).

Durante una generica iterazione l'algoritmo, si selezionano i termini *t* in *P,* i quali mostrano un'interconnessione fra almeno un senso *S* di *t* e uno o più sensi in *I.* La probabilità che un senso *S* ha di essere un'interpretazione corretta per *t,* dato un contesto semantico *I,* è determinabile attraverso la funzione :

$$f_I(S,t) = \begin{cases} \rho(\{\varphi(S,S') | S' \in I\}) & if\ S \in Senses(t), \\ 0 & \text{otherwise,} \end{cases}$$

dove *Senses(t)* è il sottoinsieme di concetti *C* di *O* associati al termine t

$$\varphi(S,S') = \rho'(\{w(e_1 \cdot e_2 \cdot \ldots \cdot e_n) | \\ S \xrightarrow{e_1} S_1 \xrightarrow{e_2} \cdots \xrightarrow{e_{n-1}} S_{n-1} \xrightarrow{e_n} S'\}),$$

è una funzione ρ' di peso *w* per ogni percorso che connette *S* con *S'*, dove *S* e *S'* sono grafi semantici. Un percorso semantico tra *S* ed *S'* è rappresentato da una sequenza di archi etichettati. Una possibile scelta per ρ e ρ' potrebbe essere la funzione somma o la funzione di media delle somme. Una similarità *G = (E, N, Sg, Pg)* racchiude e rappresenta tutti i percorsi

semantici significativi. Il simbolo *E* rappresenta l'insieme delle etichette degli archi, mentre il simbolo *N* contiene i sotto-percorsi tra i concetti; *Sg* è il simbolo iniziale di *G* mentre *Pg* è l'insieme dei suoi prodotti. L'algoritmo associa un peso ad ogni produzione $A \rightarrow \alpha$ in *Pg*, dove *A* appartiene ad *N*, ed α appartiene ad $(N \cup E)$, cioè α è una sequenza di simboli terminali e non. Se la sequenza delle etichette degli archi, appartiene ad *L(G)*, ed il linguaggio generato dalla grammatica e determinante *G* non è ambiguo, allora il peso *w* è dato dalla somma dei pesi delle produzioni applicate in derivazione. Infine l'algoritmo seleziona
$S = argmax\ f_I(S, t)$ come la più probabile interpretazione di *t* e modifica la lista *I* aggiungendovi il concetto scelto. Può essere ulteriormente applicata una soglia minima alla funzione $f_I(S, t)$ per migliorare la robustezza delle scelte del sistema.
Al termine di un'iterazione generica, un certo numero di termini sono stati disambiguati, e ognuno di essi è rimosso dall'insieme *P* dei termini pendenti.

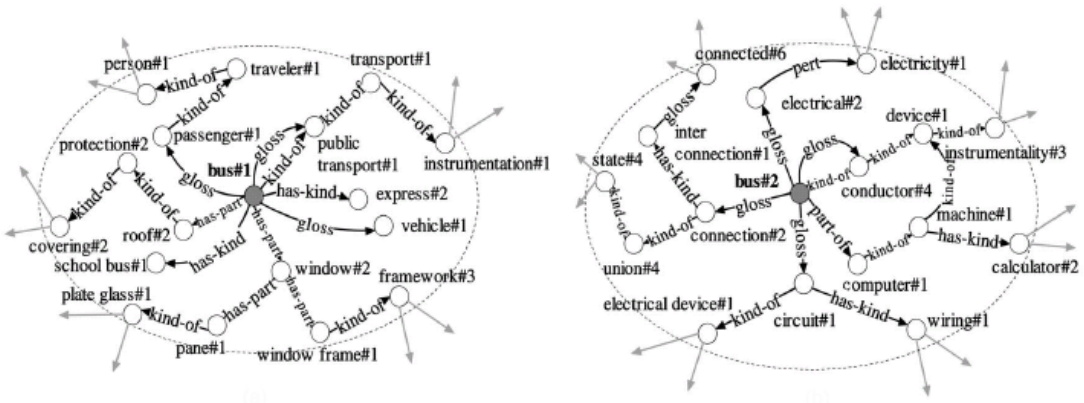


Figura 3.22-Grafo di rappresentazione dei sensi #1 e #2 di *bus*

L'algoritmo termina restituendo come output l'insieme *I*, quando per i rimanenti termini in *P* non può essere determinato alcun senso *S* tale che $f_I(S, t)>0$, ovvero l'insieme *P* non può essere ulteriormente ridotto. Durante ogni iterazione, le interconnessioni possono essere determinate solamente tra il senso di un termine pendente *t*, e i sensi già disambiguati durante le precedenti iterazioni.
Un caso speciale di applicazione dell'algoritmo di SSI si ha quando si verifica che $I = [\_, \_, \ldots, \_]$.Questo accade quando, non è disponibile alcun contesto semantico iniziale, cioè, nell'insieme *T*, non sono presenti termini monosemici. In tal caso, si applica una politica di selezione del termine *t* e l'esecuzione dell'algoritmo viene suddivisa in tanti processi quanti

sono i sensi possibili del termine *t*. Detto *n* tale numero, per ogni processo *i* con *i=1,…,n* l'input è dato dall'insieme $I_i = [\_, \_, \ldots, S_{i,t}, \ldots, \_]$, dove $S_{i,t}$ è l'i-esimo senso di *t* contenuto in *Senses(t).* L'output di ogni esecuzione, sarà un parziale o completo, contesto semantico di *Ii.* Infine, il contesto *Im* sarà dato da :

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

Come già detto in precedenza, tale algoritmo utilizza una determinata grammatica *G*, la quale ha lo scopo di descrivere il significato dei percorsi di interconnessione tra i grafi semantici, rappresentanti concetti all'interno dell'ontologia *O*. Vediamo ora di chiarire il significato e l'utilizzo di tale grammatica.

Un percorso di interconnessione è definito come una sequenza di relazioni semantiche consecutive *e1, e2, …, en*, dove *ei* appartiene all'insieme dei simboli terminali *E*, ovvero il vocabolario delle relazioni concettuali in *O*. Un percorso significativo tra due sensi *S* ed *S'* è quindi una sequenza *e1, e2, …,en* appartenente all'insieme *L(G).*

Nell'ultima versione dell'algoritmo SSI, la grammatica *G* è stata definita manualmente da coppie di sensi di termini disambiguati manualmente anch'essi co-occorrenti in differenti domini. Alcune delle regole di *G* sono state ispirate da progetti precedenti come eXended WordNet (che verrà trattato nel prossimo capitolo). I simboli terminali *ei* altro non sono che relazioni concettuali estratte da WordNet e da altre risorse lessicali on-line. *G* è definita quindi dalla quadrupla (*E, N, Sg, Pg),* dove:

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

$$N=[S_G, S_S, S_g, S_1, S_2, S_3, S_4, S_5, E_1, E_2, \ldots]$$

e l'insieme *Pg* include circa 40 produzioni. Un estratto della grammatica è mostrato in figura 3.23.

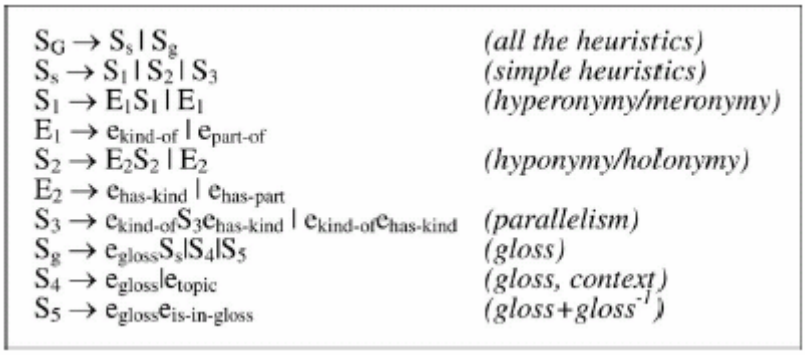


Figura 3.23- Estratto della grammatica G per l'individuazione delle interconnessioni tra i sensi

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

weight (percorso j)= αj + βj (1/ lunghezza_percorsoj)

dove αj è il peso della rule *j* in *G* e βj è il parametro di *smoothing* inversamente proporzionale alla lunghezza del percorso. Due esempi di relazioni con un elevato valore di α sono l'iperonimia e la meronimia.

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

L'algoritmo SSI è stato testato durante lo svolgimento del Senseval-3 tenutosi nel Marzo del 2004. La versione standard dell'algoritmo, mira ad ottimizzare in particolar modo la precisione dei risultati: nel caso in cui non si determinasse alcun percorso all'interno del grafo, o nel caso in cui il peso di una connessione fosse inferiore ad una determinata soglia, non viene scelto alcun senso. Tuttavia, durante lo svolgimento di tali test, si è forzata la scelta

di un senso per tutti i termini da disambiguare. In particolare si è eliminata la soglia e, nel caso in cui non si riuscisse ad individuare alcun percorso all'interno del grafo per un dato termine, si è scelto di selezionare il primo senso indicato da WordNet. Di seguito si riporteranno le tabelle contenenti i risultati dei test valuti in termini di recall e precision, per l'algoritmi SSI standard e per quello con l'aggiunta dell'euristica di scegliere il primo senso come ultima scelta.

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

Tabella 3.9- Risultati nella disambiguazione delle glosse

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

Tabella 3.10- Precisione e Recall in base alla categoria sintattica considerata

Dalle tabelle si notano buoni valori di precision, ma bassi di recall. In questo caso viene fatto notare come i valori di recall dipendano dal livello di "chiusura" del contesto dei termini utilizzati.

3.3 Algoritmi composti di disambiguazione del testo

Gli algoritmi di disambiguazione del testo analizzati fino ad ora, hanno mostrato come in realtà non esista una soluzione totale e unica al problema. Ogni metodo ha ottenuto risultati più o meno efficienti se applicato singolarmente, evidenziando lacune o in un verso o nell'altro. Negli ultimi anni si è andata sempre più diffondendo l'idea che attraverso un algoritmo combinato, ovvero utilizzante due o più metodologie viste precedentemente, si potessero ottenere risultati complessivi migliori.

I metodi combinati oggi rappresentano una delle vie più promettenti nell'ambito della disambiguazione del testo, e sono, per la maggior parte, in grado di sviluppare prestazioni migliori rispetto all'utilizzo dei singoli metodi. Di seguito si descriveranno una serie di algoritmi e metodi composti che hanno dimostrato, come attraverso la combinazione di due o

più delle tecniche descritte in precedenza, sia possibile ottenere risultati migliori per quanto concerne la problematica della disambiguazione del testo.

3.3.1 Algoritmo di Navigli

Nel 2002 Novischi in [**49**], ha presentato un algoritmo composto come soluzione al problema della disambiguazione del testo. Questo in particolare si basa un insieme di regole estratte da dati di *training*. Come già sappiamo, ogni algoritmo di disambiguazione, può assegnare un senso corretto o errato ad un determinato termine, oppure decidere di non effettuare alcuna annotazione.

L'algoritmo compone ed utilizza i seguenti metodi di disambiguazione:

- *Parallelismo lessicale*: identifica termini appartenenti alla stessa categoria sintattica separati da comma o congiunzioni, e gli assegna i sensi appartenenti alla stessa gerarchia (per nomi e verbi), o allo stesso cluster (per gli aggettivi). Nel caso di nomi e verbi, questo metodo può individuare più synset per ciascun termine, appartenenti alla stessa gerarchia.
- *SemCor previous Word* : data un termine e un contesto, questo metodo forma una coppia con la parola subito precedente il termine target, e ricerca questa coppia all'interno del corpus SemCor. Se si verifica che in tutte le occorrenze di tale coppia, il termine target sia associato sembre al medesimo synset, tale euristica allora associa tale synset al termine target.
- *SemCor next Word:* seleziona il senso secondo la stessa euristica del metodo precedente, ma utilizzando come coppia il termine target e la parola subito successive all'interno del contesto.
- *Gloss Overlap:* tale metodo come abbiamo già visto, determina il numero degli elementi in comune tra la glossa del termine target e le glosse dei termini che compaiono all'interno del suo contesto.
- *Dominio Comune:* come vedremo nel capitolo 4 questo rappresenta una via importante per disambiguare i termini, assegnando alla parola target il synset associato al dominio principale del contesto.
- *First Sense Restricted:* tale metodo assegna il primo senso al termine target (nome o verbo) se tale synset corrisponde a quello con il numero minore di antenati, all'interno

di una gerarchia ISA, fra tutti i sensi possibili, ovvero rappresenta il synset più generico per il termine. Per quanto concerne gli aggettivi, tale metodo seleziona il primo senso del termine se questo possiede il maggior numero di relazioni di similarità con gli altri sensi.

Questo approccio, in realtà, è stato proposto per il processo di disambiguazione delle glosse, ma in pratica può essere utilizzato per qualsiasi applicazione di disambiguazione del testo, ed inoltre, è indipendente dal numero e dai criteri di disambiguazione da combinare.

Il funzionamento dell'algoritmo è molto semplice: ogni metodo di disambiguazione restituisce per ciascun senso della parola target una valutazione di senso CORRECT o viceversa INCORRECT. In alcuni casi, il metodo può decidere di non restituire alcuna annotazione definendo il senso UNKNOWN. In generale quindi dopo l'esecuzione dei singoli metodi, ciò che si ottiene è un insieme di *tag* CORRECT o INCORRECT associati a ciascun senso. E quindi possibile attribuire, per esempio, un valore positivo ad ogni senso per ogni *tag* CORRECT ad esso associato, viceversa un valore negativo per ogni *tag* INCORRECT. Tali valori possono essere poi eventualmente pesati in base al livello di accuratezza dei singoli metodi utilizzati, o ancora, possono valutarsi altre caratteristiche dei vari sensi e attribuirvi un punteggio in base al valore che assumono. L'algoritmo quindi calcola la somma dei contributi dei singoli metodi e il senso con punteggio maggiore viene selezionato come più probabile.

3.3.2 Algoritmo Composto di Brody, Navigli e Lapata

L'algoritmo proposto da Brody, Navigli e Lapata in [**50**] rappresenta un metodo non supervisionato al problema della disambiguazione del testo, il quale propone ed investiga su diverse strategie di combinazione, di differenti sistemi di algoritmi.

L'algoritmo combinato, da loro proposto, non fa uso di alcun dato di training, e non ricorre neppure all'utilizzo del primo senso più frequente in WordNet.

L'algoritmo proposto, è stato realizzato attraverso la selezione di diversi metodi in base alle seguenti caratteristiche:

a. A seconda che l'algoritmo sia *token-based* o *type_based*.
b. La struttura di rappresentazione (grafo, orientato alla parola, a vettore…) e la dimensione del contesto del termine ambiguo.
c. Il numero e il tipo di relazioni semantiche utilizzate nel processo di disambiguazione.

L'algoritmo utilizza come sorgente lessicale dei sensi WordNet, tuttavia potrebbe essere adattato ad una qualsiasi altra risorsa ontologica.
Vengono considerati i seguenti metodi di disambiguazione:

- *Extended Gloss Overlap*: considerano l'algoritmo di Lesk (1986) esteso da Banrjee e Pedersen nel 2003.
- *Distributional and WordNet Similarity*: considerano l'algoritmo proposto da McCarthy nel 2004 in [**51**], basato sul calcolo della similarità tra i termini.
- *Metodo delle catene lessicali*: considerano l'algoritmo delle catene lessicali così come modificato da Galley e McKeon 2003.
- *Structural Semantic Interconnection*: considerano l'algoritmo proposto da Navigli e Velardi nel 2005.

Nella seguente tabella vengono riassunte le proprietà dei metodi precedentemente elencati.

Method	WSD	Context	Relations
LexChains	types	document	first-order
Overlap	tokens	sentence	first-order
Similarity	types	corpus	higher-order
SSI	tokens	sentence	higher-order

Tabella 3.11- Caratteristiche degli algoritmi considerati

I metodi di disambiguazione precedenti, sono applicati ad un differente livello di dimensione dei dati: gli algoritmi SSI e Gloss Overlap disambiguano a livello di frase, mentre gli altri due utilizzano e considerano l'intero documento. In quest'ultimo caso non possono essere fatte distinzioni su ciascuna occorrenza di un termine. LexChain e Overlap, considerano nella loro esecuzione, un limitato insieme di relazioni semantiche che possono occorrere tra due termini, mentre l'SSI e il Similarity usano un ampio insieme di relazioni.

Consideriamo i metodi precedentemente elencati, d'ora in poi indicheremo ciascun metodo con *Mi*. Si definisce la funzione *Score (Mi, sj)*, la quale restituisce il valore normalizzato del punteggio che il metodo Mi attribuisce al senso *sj* di un termine.
Il senso scelto dal metodo *Mi* per il termine *w*, è determinato attraverso il seguente calcolo:

$$PS(M_i, w) = \underset{s_j \in senses(w)}{\text{argmax}} \; Score(M_i, s_j)$$

I metodi vengono poi combinati tra loro ed ogni combinazione viene indicata con il nome del metodo e l'insieme dei singoli algoritmi *Mi* che riceve come input.
Di seguito si presenteranno i metodi per combinare gli algoritmi considerati.

Metodo della Votazione Diretta

Ogni componente da un voto al senso che lui ritiene più probabilmente corretto, il senso con il numero maggiore di voti viene scelto. La funzione che calcola il punteggio totale ottenuto dal senso *S*, è la seguente:

$$Score(Voting(\{M_i\}_{i=1}^{k}), s)) = \sum_{i=1}^{k} eq[s, PS(M_i, w)]$$

dove

$$eq[s, PS(M_i, w)] = \begin{cases} 1 & \text{if } s = PS(M_i, w) \\ 0 & \text{otherwise} \end{cases}$$

Mistura di Probabilità

Ogni metodo fornisce una funzione di distribuzione di probabilità dei sensi. Queste probabilità vengono normalizzate e sommate tra loro. Infine viene scelto il senso con punteggio più alto:

$$Score(ProbMix(\{M_i\}_{i=1}^{k}), s)) = \sum_{i=1}^{k} Score(M_i, s)$$

Rank-Based combination

Ogni metodo fornisce un elenco ordinato di sensi per un dato termine ambiguo. Per ogni senso si verifica la sua posizione in base al metodo. Vince il senso con collocazione generale più bassa all'interno della lista. Il punteggio viene così calcolato

$$Score(Ranking(\{M_i\}_{i=1}^{k}), s)) = \sum_{i=1}^{k} (-1) \cdot Place_i(s)$$

Place$_i$ (s) rappresenta il numero di punteggi distinti maggiori o uguali al punteggio *Score* (*Mi,s).*

Arbiter-based Combination

Un metodo di disambiguazione può agire come arbitro per giudicare i termini su cui esiste disaccordo tra i vari algoritmi. In questo caso si seleziona l'algoritmo SSI come arbitro, poiché quest'ultimo ha ottenuto risultati più accurati durante dei test eseguiti in precedenza. Per ogni termine di disaccordo *w,* e per ogni senso *s* di *w,* assegnato da ciascun sistema nella composizione *[Mi],* viene calcolato il seguente punteggio:

$$Score(Arbiter(\{M_i\}_{i=1}^{k}), s) = SSIScore^{*}(s)$$

dove *SSIScore*(s*) è una versione modificata dello *Score* calcolato nell'algoritmo standard di SSI. In questo caso si sfrutta come contesto per *s* l'insieme dei sensi disponibili e le restanti parole per ogni frase. Si esclude però dal contesto usato da SSI, i sensi di *w* non scelti da nessun sistema composto. In tal modo si riduce il numero di sensi da considerare dall'arbitro e si può influenzare in maniera positiva la prestazione dell'algoritmo, poiché in tal modo viene eliminato il rumore rappresentato dai sensi sicuramente indicati dai metodi come errati.

I vari sistemi composti sono stati testati sullo stesso insieme di dati, in particolare su dei nomi, contenuti in SemCor.

I risultati dei test, sono riassunti nella tabella 3.12.

Come si nota dal grafico in figura 3.24, per tutti i metodi composti si ottengono risultati migliori rispetto all'esecuzione individuale degli algoritmi. Il metodo tra quelli composti che ottiene prestazioni peggiori è quello basato sull'arbitro. Ciò viene attribuito al fatto che nel circa 30% dei casi, nessuno dei sensi suggeriti dall'algoritmo in disaccordo è corretto. In tali casi il metodo basato sull'arbitro non possiede nessun criterio per poter scegliere il senso corretto.

Method	Acc_{ps}	$Acc_{wsd/ps}$
Similarity	54.9	46.5
SSI	53.5	47.9
Voting	$57.3^{\dagger\$}$	$49.8^{\dagger\$}$
PrMixture	$57.2^{\dagger\$}$	$50.4^{\dagger\$\ddagger}$
Rank-based	$58.1^{\dagger\$}$	$50.3^{\dagger\$\ddagger}$
Arbiter-based	$56.3^{\dagger\$}$	$48.7^{\dagger\$\ddagger}$
UpperBnd	100	68.4

Tabella 3.12-Risultati ottenuti per ciascun metodo composto

Nella tabella successiva, invece, viene mostrato il confronto tra i due migliori algoritmi singoli e tutti gli algoritmi composti, in termini di accuratezza del processo di disambiguazione, e in funzione della frequenza dei nomi nel SemCor. Si nota che per ogni banda di frequenza almeno uno dei metodi composti esegue con maggior accuratezza dei singoli algoritmi.

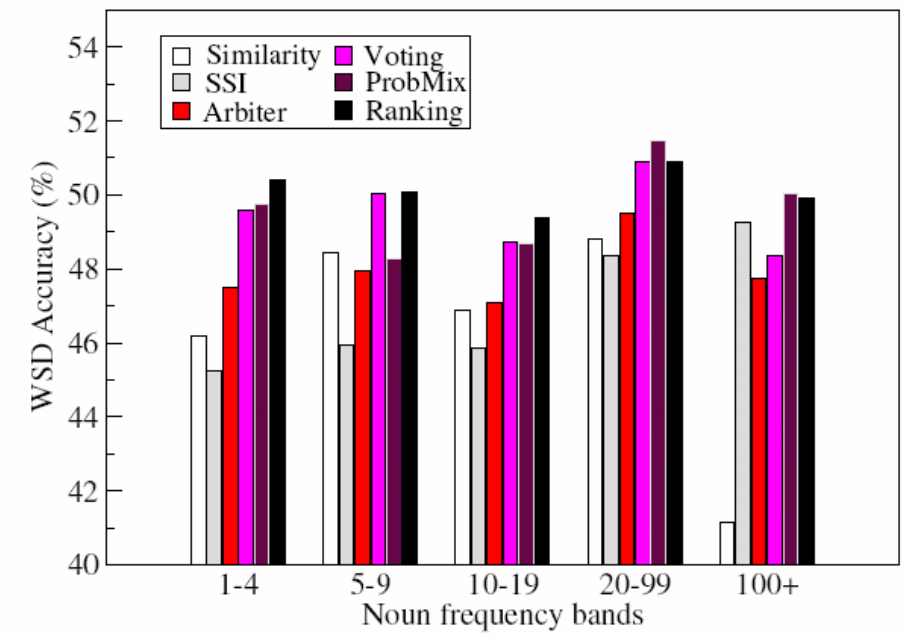


Figura 3.24- Accuratezza del processo di WSD in funzione della frequenza dei nomi in SemCor

3.3.3 Algoritmo di Mandreoli, Martoglia e Ronchetti

Mandreoli, Martoglia e Ronchetti in [**52**], presentano un approccio al problema della disambiguazione basato sull'uso dei grafi.

L'algoritmo è stato inserito nel paragrafo relativo ai metodi composti, poiché rappresenta una via per disambiguare i termini, utilizzando l'informazione data dalle relazioni dei vari sensi del termine ambiguo con i sensi dei termini del suo contesto, più l'informazione ottenibile dalle glosse dei synset del termine target, più l'informazione legata alla frequenza di utilizzo di ciascun synset all'interno della lingua inglese.

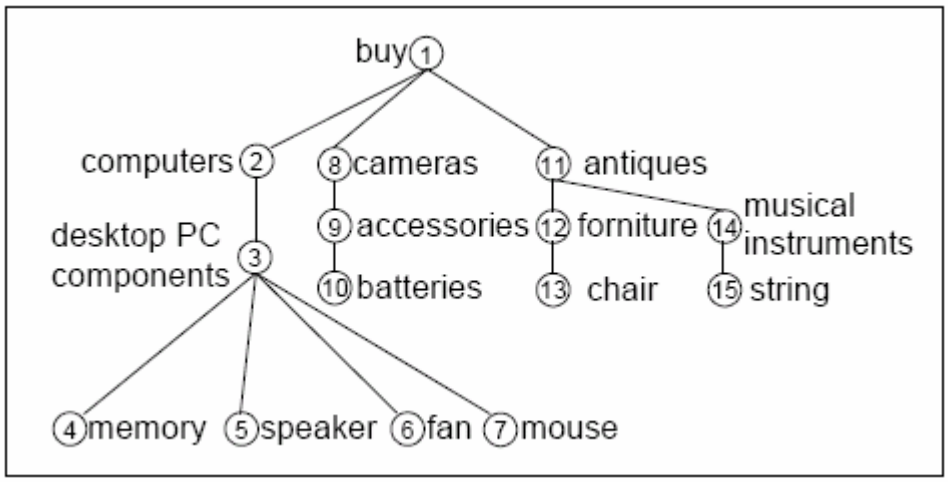


Figura 3.25- Una porzione delle categorie di *eBay*

Esso rappresenta un algoritmo di disambiguazione locale, applicabile però solamente a sorgenti d'informazione strutturate a grafo, in particolare ad albero, come per esempio dati provenienti da schemi XML, da *Web directory* oppure Ontologie. Un esempio di tali sorgenti d'informazione è riportato in figura 3.25 .Essi in particolare presentano un'architettura funzionale, che realizza il servizio di disambiguare le etichette di una generica struttura ad albero (figura 3.26).

L'algoritmo presentato si basa su un approccio che sfrutta sia il contenuto informativo legato al contesto (e quindi agli altri nodi dell'albero non rappresentanti il termine target) sia le informazioni estratte da una risorsa di conoscenza come WordNet. Infatti, il contesto di un termine target non viene considerato solamente come un insieme di termini, ma da esso sono

estratte altre informazioni come la distanza degli altri nodi dal termine target e le relazioni semantiche con le quali sono collegati a quest'ultimo.

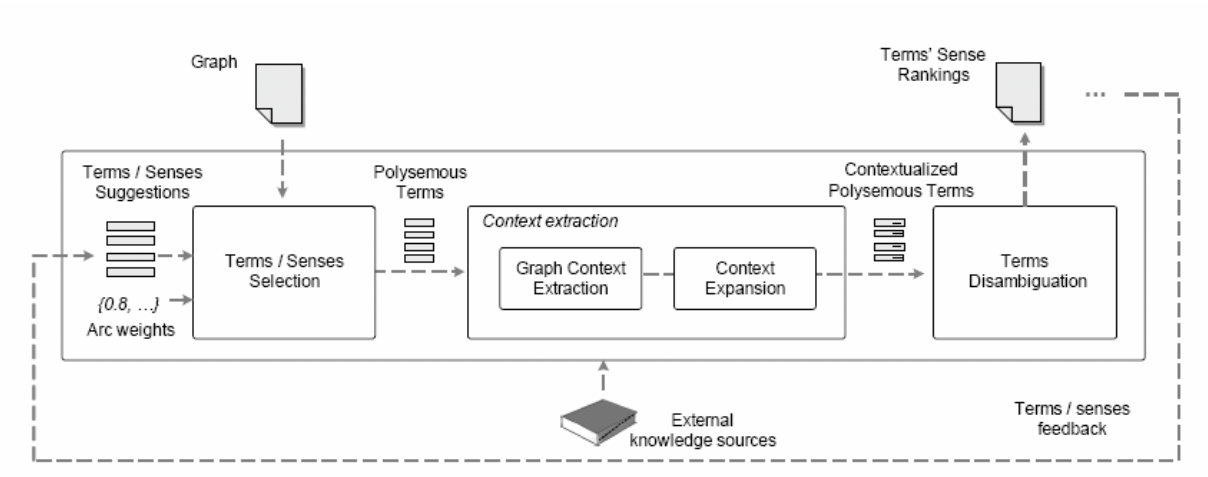


Figura 3.26-Servizio di disambiguazione di un grafo generico

Inoltre, si utilizzano informazioni estratte da Thesauri come le relazioni di iperonimia e iponimia tra i sensi, le glosse degli stessi synset e la loro frequenza di utilizzo. L'approccio di Mandreoli e al. non richiede l'utilizzo di alcun dato di *training.* Come abbiamo già detto l'algoritmo è ideale per la disambiguazione dei termini contenuti in una struttura dati ad albero, come per esempio le sorgenti di XML Schemas, le quali appunto esplicitano le relazioni strutturali tra gli elementi coinvolti, e fornendo così un contesto per l'elemento target. Le strutture dati ad albero, contengono in particolare, un insieme di nodi, le cui etichette devono essere disambiguate (è importante sottolineare come a differenza de altri algoritmi basati sui grafi, ciascun nodo rappresenti esattamente un termine e non una sua singola istanza), ed un insieme di archi che connetto i nodi e possono essere etichettati a loro volta. L'individuazione del senso corretto, può essere ottenuta attraverso l'analisi del contesto all'interno del quale sono inseriti i termini con il supporto di una sorgente di conoscenza esterna. Gli archi sono elementi particolarmente importanti, poiché essi collegano ogni etichetta con il suo contesto. Ad ogni etichetta di ciascun arco, saranno associati rispettivamente due pesi, calcolati uno per ogni direzione dell'arco. Tali pesi serviranno poi per calcolare la distanza fra due nodi all'interno di un grafo, in particolare più basso sarà il peso dell'arco, più chiusi (in termini semantici) saranno fra loro i due nodi che congiunge. Il processo di selezione "senso/termine" in figura 3.26, considera le etichette di ogni nodo *N*

dell'albero, vi estrae il termine *t* contenuto (o i termini contenuti nel caso di parole composte), e associa ognuno di questi termini *(t, N)* (indica il termine *t* associato al nodo *N*), una lista di sensi *Senses(t, N) = [s1, s2, …, sk]*. Tale lista si ottiene tramite il Thesaurus utilizzato (WordNet). Ogni termine da disambiguare viene associato al suo contesto. Tale contesto, viene ovviamente estratto dall'albero, ma non coincide necessariamente con l'intero albero. Fanno parte del contesto solo i nodi raggiunti o raggiungibili dal termine *t* attraverso un arco. Il realtà il contesto da considerare, dipenderà dalla specifica applicazione. Per esempio, per selezionare contesti più specifici, è possibile imporre vincoli come la distanza massima tra due nodi, la tipologia di relazione che lega i due nodi ecc..Se consideriamo solo gli archi con direzione diretta, considereremo nel contesto solo i nodi discendenti ed i suoi sotto alberi. Viceversa nell'esempio di *eBay* in figura 3.25, è conveniente considerare tutti i discendenti, i padri e i fratelli, di ciascun nodo, poiché l'intera struttura è utile nel disambiguare un termine.
Dato un insieme di archi, il componente "*graph context extraction*" in figura 3.26, riesce a contestualizzare ciascun nodo/termine estraendo, dall'insieme dei termini appartenenti ai nodi raggiungibili da ciascuno, il proprio sotto-grafo di contesto *Gcontext(t, N*). Ovviamente non tutti i nodi contribuiranno con lo stesso peso nel processo di disambiguazione, perciò ad ognuno di essi si associa un peso *weight (Nc)*calcolato nel seguente modo:

$$weight(N_c) = 2 \cdot \frac{e^{-\frac{d^2}{8}}}{\sqrt{2\pi}} + 1 - \frac{2}{\sqrt{2\pi}}$$

Ogni elemento del grafo di contesto sarà quindi caratterizzato dalla seguente tripla di valori *((tc, Nc), Senses(tc, Nc), weight (Nc)).*
Il contesto di ogni termine può essere poi espanso attraverso il contesto *Scontext (s),* di ogni senso *Senses (t,N).* Tale espansione, è utile in particolare quando il grafo di contesto fornisce poche informazioni. Il modulo *" context expansion"* definisce *Scontext (s)* come l'insieme dei nomi contenuti all'interno della glossa del synset.
Infine, ogni termine *t,* viene disambiguato utilizzando i contesti precedentemente descritti. Il risultato è una lista di sensi ordinati *s* appartenenti ad *Senses(t, N),* dove a ciascun senso è associato un valore di "confidenza" $\phi(s)$.
Di seguito analizzeremo in maniera dettagliata il funzionamento dell'algoritmo, il cui codice è descritto in figura 3.27.

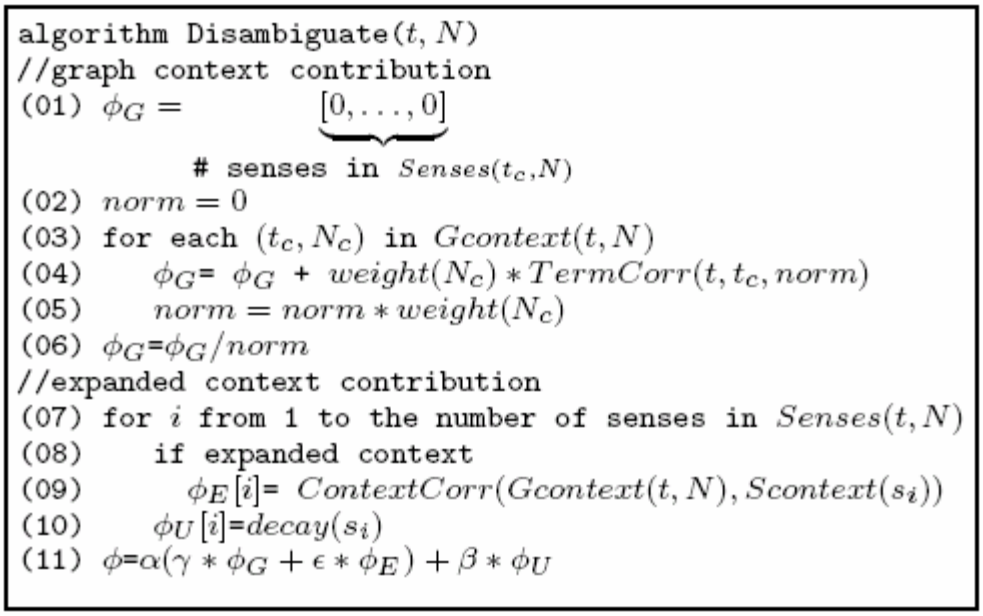


Figura 3.27- Algoritmo di disambiguazione

L'algoritmo riceve in input un termine *(t, N)* da disambiguare, e produce un vettore ϕ di valori di confidenza. In particolare, dato un insieme di sensi *Senses (t, N) = [s1, s2, …, sk],* ϕ è un vettore di *k* elementi dove $\phi[i]$ esprime la confidenza nella scelta di *si* come senso per *(t,N).* Il vettore di confidenze è ottenuto tramite due contributi, (linea 11 dell'algoritmo):

1. quello del contesto di *t*, il cui peso è espresso attraverso una costante α e si suddivide in un grafo di contesto (vettore di confidenza ϕ_G, peso γ) ed un contesto espanso (vettore di confidenza ϕ_E, peso ε), tale che $\gamma + \varepsilon = 1$;
2. quello della frequenza di utilizzo dei sensi all'interno del linguaggio inglese (vettore di confidenza ϕ_U) con peso β e tale che $\beta + \alpha = 1$.

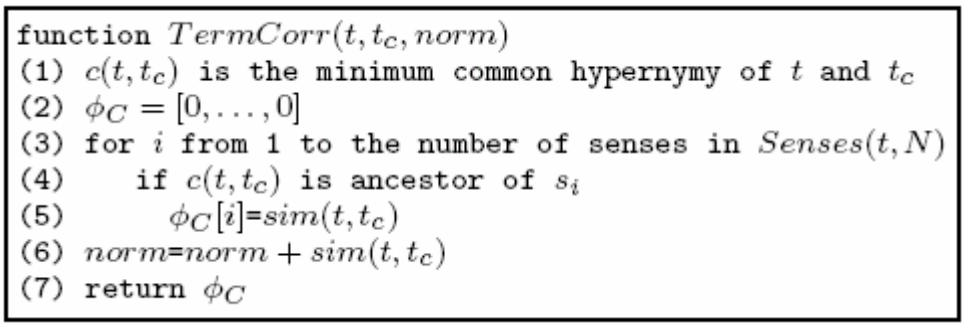


Figura 3.28-La funzione TermCorr

Come abbiamo visto in precedenza, i termini costituenti il contesto della parola target, forniscono importanti informazioni allo scopo di disambiguare quest'ultimo. Il contributo fornito dal grafo di contesto è calcolato dal passo 1 al passo 6 dell'algoritmo. In particolare ϕ_G è la somma dei valori che calcolano il livello di correlazione semantica tra *t* e un altro termine all'interno del grafo di contesto *Gcontext (t, N)* (passo 4). Il contributo di ogni termine del contesto*(tc, Nc)* è pesato in base alla sua posizione all'interno del grafo. Infine, al passo 6, l'intero vettore ϕ_G, è diviso per il valore *norm* allo scopo di ottenere un valore di confidenza normalizzato.

```
function ContextCorr([t_1,...,t_n],[t^s_1,...,t^s_m])
(1) φ_C = [0,...,0]
(2) for i from 1 to n
(3)     φ_T = [0,...,0]
(4)     norm = 0
(5)     for j from 1 to m
(6)         φ_T= φ_T + TermCorr(t_i, t^s_j, norm)
(7)     φ_C[i]=max(φ_T/norm)
(8) return mean(φ_C)
```

Figura 3.29-La funzione ContextCorr

La funzione *TermCorr(),* deriva da un approccio al concetto di similarità semantica che sfrutta le gerarchie di iperonimia di WordNet. In particolare sfrutta la misura proposta da Leacock e Chodorow,(descritta al paragrafo 3.1.3), definendo inoltre un minimo comune ipernomo, tra il termine target e una parola del suo contesto, come il più specifico (ovvero più basso all'interno della gerarchia) tra gli ipernimi che essi hanno in comune. La funzione *TermCorr()* incrementa il valore di confidenza dei sensi *Senses (t, N)* che sono discendenti di tale minimo comune ipernomo (linea 3-4 della figura 3.29), in particolare lo incrementa di un valore proporzionale al contenuto informativo di tale ipernomo (line 5 figura 3.29).

Come abbiamo già affermato, oltre al contributo del grafo di contesto, può essere sfruttato anche il contributo fornito dal contesto espanso (linea 7-9, figura 3.27). In tal caso, l'obiettivo principale è quello di quantificare la correlazione semantica tra il contesto del termine target e ciascuna glossa dei synset ad esso associati. In particolare la confidenza nella scelta di un senso *s* è proporzionale al valore di similarità (linea 9 figura 3.27). Il codice della funzione *ContextCorr()* è mostrato in figura 3.29. Esso essenzialmente calcola la similarità semantica tra ogni termine del grafo *ti* e i termini all'interno del contesto di sensi *Scontext(s)* (linea3-7,

figura 3.29). La funzione, alla linea 8, restituisce il significato del valore di similarità, calcolato per i termini in *Gcontext (t, N).*

L'ultimo contributo all'algoritmo, è dato dalla funzione *decay(),* la quale sfrutta la frequenza dell'utilizzo dei sensi all'interno del lingua inglese (linea 9, figura3.27). In particolare, WordNet, ordina i suoi synset in base alla frequenza del loro utilizzo. La confidenza di un senso *s* è incrementata in maniera inversamente proporzionale alla frequenza del synset stesso. In particolare:

$$decay(s_i) = 1 - \rho \frac{pos(s_i) - 1}{|WNSenses(t)|}$$

dove $0 < \rho < 1$ è un parametro solitamente fissato a 0.8 e |*WNSenses(t)*| è la cardinalità di *WNSenses(t).* Tale aggiustamento, ha lo scopo di emulare il comportamento della mente umana nella scelta del corretto significato di un nome, quando il contesto fornisce poche informazioni per supportare tale decisione.

Capitolo 4

4 Estensioni di WordNet

Nel capitolo precedente, sono stati analizzati le varie metodologie di disambiguazione del testo proposte in letteratura. Tali metodologie nella maggior parte dei casi, hanno in comune l'utilizzo di un database lessicale che fornisce informazioni riguardo ai significati dei termini e alle relazioni che intercorrono fra quest'ultimi. WordNet si è dimostrata essere la conoscenza di base preferita nella maggior parte dei casi, tanto da poter quasi essere ritenuta uno standard per molti processi di disambiguazione. Nonostante il grande successo riscosso da questo database lessicale, il suo utilizzo ne ha evidenziato lacune, specialmente in quei casi in cui WordNet viene utilizzata come unica risorsa all'interno del processo di disambiguazione. Essenzialmente tali lacune possono essere attribuite a vari aspetti di WordNet. Uno è rappresentato dall'elevato livello di granularità, nella distinzione fra i sensi dei termini. Per esempio, in WordNet esistono verbi ai quali vengono associati più di 40 synset differenti. E' facile immaginare, quindi, quanto complessa sia la disambiguazione di tali termini, specialmente in ambienti in cui il contesto informativo non fornisce sufficiente contributo per poter discernere fra tali sensi. Nella maggior parte delle applicazioni reali, è sufficiente, se non addirittura necessario, un livello di distinzione dei sensi a granularità molto inferiore. Un altro limite di WordNet risiede nella possibilità di individuare relazioni solo fra termini appartenenti alla stessa gerarchia sintattica. Ovvero la gerarchia di WordNet non da alcuna informazione riguardo ai legami tra synset appartenenti per esempio ad un verbo ed altri appartenenti ad un nome. Inoltre tali relazioni, anche quando sussistono tra istanze di termini appartenenti alla medesima categoria sintattica, risultano spesso insufficienti per

delineare in maniera completa un processo di disambiguazione basato, appunto, sulle relazioni fra i termini.

Tali lacune, hanno portato al crescere di un sempre maggiore interesse, nei confronti di tecniche e metodologie che consentano di estendere la risorsa WordNet, arricchendola di nuovi contenuti informativi, di varia natura, che consentano di superare anche solo parzialmente, le mancanze di WordNet.

Di conseguenza sono state analizzate alcune possibili risorse lessicali di conoscenza che potessero ampliare WordNet o affiancarsi ad esso. Tra quelle effettivamente implementate la nostra analisi si è concentrata su ***W**ordN**e**t **D**omains* (WND), ed *e**X**tended **W**ordN**e**t* (XWN).

Questo capito ha lo scopo di illustrare e descrivere le possibili estensioni di WordNet incontrate in letteratura, e di analizzare brevemente i conseguenti algoritmi di disambiguazione utilizzati per la creazione di tali estensioni.

4.1 WordNet Domains

I domini rappresentano un'area comune della discussione umana, come l'economia, la politica, legge ecc…, le quali contribuiscono in maniera rilevante a dimostrare l'esistenza di una coerenza lessicale. Una sostanziale porzione della terminologia del linguaggio, può essere caratterizzata come un dominio di parole, i cui significati, si riferiscono a concetti appartenenti a domini specifici, e che spesso compaiono in testi che discutono del dominio corrispondente.

I domini hanno trovato applicazione prevalentemente attraverso due ruoli, all'interno della descrizione linguistica:

1. Un ruolo caratterizza i sensi dei termini, come il campo semantico di un senso di un termine in un dizionario. Per esempio il termine *crane* (gru) ha senso se utilizzato in domini come la zoologia e le costruzioni. WordNet Domains descritto in [**56, 62**], è una risorsa lessicale la quale mira ad estendere WordNet, sommando a quest'ultimo l'informazione relativa al dominio di appartenenza di ciascun synset.
2. Un secondo ruolo è quello di caratterizzare i testi, tipicamente attraverso un generico livello di categorizzazione del testo.

Dal punto di vista della disambiguazione dei termini, un dominio può essere considerato sotto due differenti punti di vista:

1. Primo, l'informazione di dominio rappresenta un'informazione sostanziale e fondamentale per disambiguare un termine. Molte delle caratteristiche che contribuiscono al processo di disambiguazione identificano un dominio che caratterizza un particolare senso o un particolare sottoinsieme di sensi. Per esempio i termini economici, forniscono aspetti caratteristici per sensi di termini finanziari come, *bank* ed *interest*, mentre termini legali caratterizzano i sensi giudiziari di *senstence* e *court*. Il metodo di disambiguazione basato sui domini mira a catturare tale informazione di dominio relativamente ad ogni senso di ciascun termine, e può richiedere parecchi esempi di training, allo scopo di ottenere sufficienti caratteristiche di tal genere per ogni senso [**55**]. Tuttavia i domini rappresentano una nozione linguistica del discorso, la quale non dipende dal senso specifico di un termine.
2. I domini possono fornire un utile livello di granularità di distinzione dei sensi. Molte applicazioni non beneficiano degli alti livelli di granularità di distinzione dei sensi, come ad esempio quella fornita da WordNet, le quali spesso rendono dificile il processo di disambiguazione del testo nelle applicazioni reali. Si consideri per esempio come alcuni verbi di WordNet posseggano più di 40 sensi differenti.

Di seguito si descriverà WordNet Domains sviluppato presso il centro di ricerca ITC-irst, un database che associa a ciascun synset di WordNet un'etichetta corrispondente al dominio ad esso associato. Per primo si descriverà la struttura e il processo di realizzazione di WordNet Domains, mentre successivamente si presenterà un algoritmo di disambiguazione del testo detto Domains Driver Disambiguation (DDD), il quale utilizza come risorsa l'estensione di WordNet, WND.

4.1.1 Il ruolo dei domini nella disambiguazione del testo

Nel 2002 Magnini e Strapparava in [**54**], investigano sul ruolo dell'informazione di dominio nell'ambito della disambiguazione del testo. L'ipotesi è che l'etichette di dominio (come medicina, sport ecc…), forniscano una via potente per stabilire le relazioni semantiche tra i termini, le quali possono successivamente essere utilizzate per disambiguare il testo. In particolare essi assumono che i domini costituiscono una fondamentale proprietà semantica sulla quale si basa la coerenza del testo, così che i sensi delle parole che compaiono

all'interno di una porzione di testo, tendono a massimizzare l'appartenenza ad uno stesso dominio.

La figura 4.1, mostra un esempio estratto *dall'English lexical Sample task* durante il Senseval-2. Il termine target è la seconda occorrenza di "*chairs*". Si suppone che per la maggior parte delle parole all'interno dell'esempio, e per ogni senso ad esse associato, sia disponibile in WordNet un'etichetta di dominio, e che tali parole siano già disambiguate all'interno del testo. Si può notare come diverse parole come "*sofà", "living room", "dinner table"*, siano associate al dominio FURNITURE; altre poche parole come "*games", "chess"* e "*backgammon*" sono invece associate al dominio PLAY, mentre solo un termine è associato al domino LETTERATURA. Allo scopo di disambiguare il termine "*chair*", sembra naturale non poter non considerare il fatto che il dominio prevalente nel testo sia FORNITURE. Questo porta a scegliere, come senso corretto per il termine, quello più strettamente legato a tale dominio.

Tuttavia, per rendere possibile il meccanismo di disambiguazione precedentemente dedotto, si ha bisogna di poter usufruire di una risorsa lessicale la quale sia in grado di associare, ad ogni senso di ogni termine, il dominio corrispondente. A tale scopo è stata sviluppata una versione estesa di WordNet, denominata WordNet Domains (WND), la quale fornisce l'annotazione di dominio per ciascun synset. Inoltre, attraverso questa risorsa, è possibile effettuare alcune analisi del testo che consentono di individuare l'orientamento di dominio del testo. Per esempio, è possibile determinare il dominio prevalente di un testo o di una sua porzione, allo scopo di determinare come questo influisca sulla determinazione dei sensi delle parole in esso contenute. Inoltre, è possibile calcolare una misura di coerenza del testo, sulle basi dell'ipotesi di *one domain of discourse*, in opposizione a quella di *one sense of discorse*. Un risultato rilevante di tale analisi, è che un numero piuttosto limitato di termini contribuiscono a determinare il dominio prevalente di una porzione di testo. Tali parole rappresentano i "centroidi" del processo di disambiguazione basato sull'etichette di dominio. Magnini e Strapparava, con il termine dominio, intendono un insieme di parole tra le quali esistono relazioni semantiche forti. WND estende il database lessicale WordNet, attraverso delle etichette associate a ciascun synset che ne indicano il dominio o i domini di appartenenza. Tali domini sono selezionati da un insieme di una gerarchia organizzata composta da circa 200 etichette (descritta in maniera dettagliata nel capitolo 5). Le etichette di dominio, rappresentano un'informazione complementare a ciò che è già presente in WordNet.

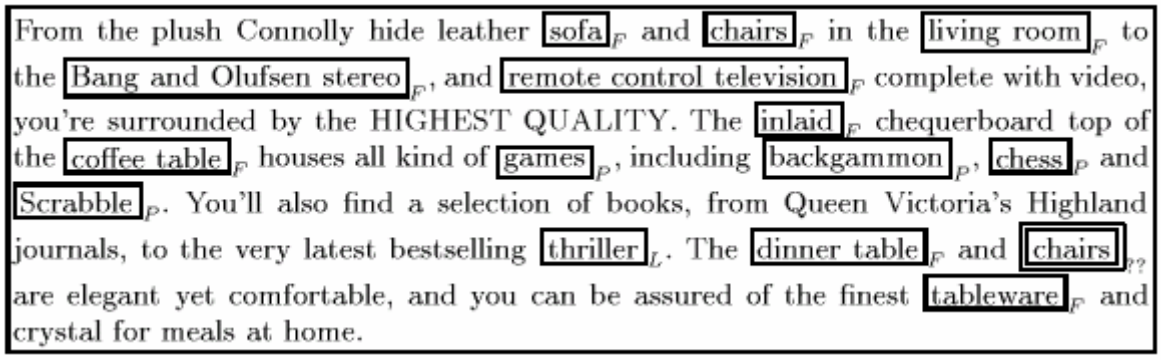


Figura 4.1 Esempio di disambiguazione dei termini di basato sui domini ad essi associati

Un dominio può includere synset appartenenti a differenti categorie sintattiche: per esempio, il dominio MEDICINE raggruppa insieme sensi di nomi come *doctor#1* e *hospital#1*, e di verbi come *operate#7*. Inoltre, un dominio può includere sensi appartenenti a differenti sotto-gerarchie di WordNet: per esempio, il dominio SPORT contiene sensi come *atlete#1*, derivante da *life_form#1*, *game_equipment#1* derivante da *Physical_object#1*, *sport#1* derivante da *act#2,* e *playing_field#1* derivante da *location#1*. Infine, i domini possono raggruppare sensi di una stessa parola all'interno di differenti *cluster* tematici, i quali hanno l'importante effetto di ridurre il livello di ambiguità quando si sta disambiguando attraverso il dominio.

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

Tabella 4.1- Synset associati al termine bank

La tabella 4.1 mostra un esempio: la parola "*bank*" possiede dieci sensi differenti all'interno di WordNet, tre di questi possono essere raggruppati sotto il dominio di ECONOMY, mentre altri due appartengono ai domini GEOGRAPHY e GEOLOGY.
La metodologia di creazione di WND è stata principalmente manuale e basata su un criterio lessico-semantico, il quale trae vantaggio dalle relazioni già esistenti in WordNet.
Sono state selezionate circa 200 etichette di dominio da un insieme di dizionari. Successivamente, tali etichette sono state strutturate in una gerarchia in base alla *Dewey Decimal Classification* [**64**].
L'associazione dei domini ai vari synset di WordNet, è stata effettuata, in base alla classificazione DDC. Per primo, un piccolo numero di synset di alto livello, vengono annotati manualmente con il loro dominio di pertinenza. Successivamente, attraverso una procedura automatica che sfrutta alcune relazioni di WordNet, come l'iponimia, la meronimia, l'antinomia, ecc…, si estende l'assegnamento manuale a tutti i synset raggiungibili. Per esempio, attraverso tale procedura è possibile marcare i synset [*beak, bill, neb, nib*] con il dominio ZOOLOGY, iniziando dal synset *bird*, e proseguendo attraverso l'utilizzo delle relazioni di *part-of*. Tuttavia esistono casi in cui tale procedura, necessita di essere bloccata, attraverso delle eccezioni, allo scopo di evitare propagazioni incorrette. Infatti, esistono synset in WordNet, che non appartengono ad alcun dominio specifico, ma piuttosto possono comparire in testi associati ad un qualsiasi dominio. Per questa ragione, è stata inserita un'etichetta FACTOTUM la quale, essenzialmente, raggruppa due tipologie di synset:

1. **synset generici**, i quali sono difficilmente classificabili all'interno di un particolare dominio di appartenenza, come per esempio man#1, cioè "an adult male person".
2. **synset stop sense**, i quali appaiono frequentemente in contesti differenti, come numeri, giorni della settimana, colori ecc…

La realizzazione di WND, ha richiesto in tutto due anni-uomo di lavoro. La lista completa dei domini, e il numero di annotazioni in WordNet per ciascun dominio è riportata in tabella 4.2.
Le parole all'interno di un testo, non si comportano in maniera omogenea, in particolare è possibile individuare tre ruoli differenti che una parola può assumere all'interno di un testo (la stessa parola può giocare ruoli differenti a seconda del testo in cui appare):

- *Text Related Domain words* (TRD): rappresentano parole che possiedono almeno un senso che contribuisce a determinare il dominio dell'intero testo; per esempio, il

termine *bank* all'interno di un testo riguardante l'ECONOMY, è probabilmente un TRD.

- *Text Unrelated Domain words* (TUD): rappresentano parole che possiedono sensi appartenenti a domini specifici, ma che non contribuiscono a determinare il dominio predominante del testo; per esempio, il termine "*church*" all'interno di un testo, riguardante l'ECONOMY probabilmente non riguarderà l'argomento principale del testo.
- *Text Unrelated Generic words* (TUG): rappresentano parole che non portano con se alcuna informazione di dominio; tali synset sono etichettati con *factotum* e rappresentano la maggior parte dei sensi annotati. Un esempio è rappresentato dal verbo *to be*.

Domain	#Syn	Domain	#Syn	Domain	#Syn
Factotum	36820	Biology	21281	Earth	4637
Psychology	3405	Architecture	3394	Medicine	3271
Economy	3039	Alimentation	2998	Administration	2975
Chemistry	2472	Transport	2443	Art	2365
Physics	2225	Sport	2105	Religion	2055
Linguistics	1771	Military	1491	Law	1340
History	1264	Industry	1103	Politics	1033
Play	1009	Anthropology	963	Fashion	937
Mathematics	861	Literature	822	Engineering	746
Sociology	679	Commerce	637	Pedagogy	612
Publishing	532	Tourism	511	Computer_Science	509
Telecommunication	493	Astronomy	477	Philosophy	381
Agriculture	334	Sexuality	272	Body_Care	185
Artisanship	149	Archaeology	141	Veterinary	92
Astrology	90				

Tabella 4.2-Distribuzione dei synset di WordNet tra i domini scelti della gerarchia DDC

Allo scopo di fornire una stima quantitativa della distribuzione delle tre tipologie differenti di termini, è stato effettuato un esperimento sul corpus SemCor, utilizzando WordNet Domains come sorgente di annotazioni di domini. In questo esperimento, Magnini e Strapparava, hanno considerato solo 42 domini disgiunti, escludendo quello *factotum* (per esempio, si è usato solamente il dominio SPORT al posto di domini come VOLLEYBALL, BASKETBALL ecc…). Tale insieme, consente un buon livello di astrazione, senza una perdita rilevante d'informazione e, in più, evita il problema dell'applicazione di tecniche di apprendimento a domini non rappresentati abbastanza all'interno dei testi disponibili.

Per ogni testo di SemCor, è assegnato un punteggio a ciascuno di questi 42 domini, in base alla loro frequenza fra i sensi delle parole dei testi. I tre domini con punteggio più alto sono

stati considerati come domini prevalenti all'interno del testo. Successivamente, ogni parola del testo è stata assegnata ad una delle tre tipologie di parole precedentemente descritte, in base al fatto che:

i. Almeno un dominio tra quelli a cui il termine è associato, coincida con uno dei tre termini (parole TRD).
ii. La maggior parte dei sensi del termine siano associati ad un dominio specifico ma nessuno di questi corrisponda ad uno dei tre domini individuati (parole TUD).
iii. La maggior parte dei sensi associati ai termini, siano etichettati come *factotum* e nessuno dei sensi rimanenti appartenga ad uno dei tre domini principali (parole TUG).

Ogni gruppo di parole, viene analizzato ulteriormente in base alla categoria sintattica di appartenenza, e viene calcolata la media dei termini polisemici in base a WordNet.
I risultati dell'esperimento sono riportati in tabella 4.3: circa il 21% dei termini contribuisce a determinare i domini prevalenti; tra questi circa l'80% è rappresentato da nomi. I termini TUG, come ci si aspettava, sono sia i più frequenti (circa il 64%), sia quelli più polisemici. A tale risultato contribuiscono principalmente i verbi con l'83%, i quali si caratterizzano per essere altamente polisemici e, quindi, per il non contributo sostanziale alla determinazione dei domini.
L'ipotesi di O*ne Sense per Discorse* (OSD), implica l'importante tendenza, nell'uso multiplo dei termini, di avere sempre il medesimo senso, all'interno di un discorso. Di conseguenza, l' ipotesi il o*ne domain per discourse* (ODD), porta ad assumere che, l'uso multiplo di un termine all'interno di una porzione coerente di testo, tende a mostrare lo stesso dominio. Dimostrando l'assunzione di ODD, si rafforzerebbe l'ipotesi alla base dell'uso di WordNet Domain nell'ambito della disambiguazione del testo, ovvero che il dominio prevalente di un testo, rappresenta una caratteristica importante nella determinazione del senso corretto dei termini stessi.
Per dimostrare la validità di tale approccio, è stato eseguito un test, utilizzando sempre ovviamente WordNet Domains come sorgente di informazione di dominio. Secondo Krovetz in [**57**], per invalidare l'ipotesi di OSD, è sufficiente che almeno un termine all'interno dello stesso testo, non rispetti tale assunzione. Seguendo questa osservazione, per effettuare il test, è stato estratto da SemCor un insieme di 23,877 parole ambigue con occorrenze multiple

all'interno dello stesso documento, e si è contato il numero di termini annotati con sensi differenti.

Successivamente per ciascuno dei vari sensi dei termini così individuati si è determinato il dominio associato da WordNet Domains. La differenza tra OSD e ODD, può essere facilmente intuita considerando il termine *bank* il quale compare tre volte all'interno di un testo e con tre sensi differenti (*bank#1, bank#3, bank#8*). In questo caso viene dimostrata l'inconsistenza dell'ipotesi di OSD, ma rimane consistente quella di ODD poiché tutte le tre occorrenze del termine sono etichettate sotto lo stesso dominio, ECONOMY.

Pos	Cases [a]	Exceptions to OSD [b]	Exceptions to ODD [c]
All	23877	7469 (31%)	2466 (10%)
Nouns	10291	2403 (23%)	1142 (11%)
Verbs	6658	3154 (47%)	916 (13%)
Adjectives	4495	1100 (24%)	391 (9%)
Adverbs	2336	790 (34%)	12 (1%) [d]

Tabella 4.3-One Sense per Discourse vs. One Domain per Discourse

I risultati del test sono riportati in tabella. Essi mostrano che l'ipotesi di ODD è verificata insieme a quella che all'interno di un testo esista solo un numero limitato di domini rilevanti.

Le poche eccezioni al ODD possono essere dovute a variazioni di dominio all'interno dei testi di SemCor, caratterizzati dal fatto di essere abbastanza lunghi (in media sono composti da 2000 termini per testo). In tali casi, infatti, alcune parole possono appartenere a domini differenti, in differenti porzioni dello stesso testo.

La figura 4.2, ottenuta dopo aver disambiguato i termini in base ai loro domini possibili, mostra come la rilevanza dei due domini, PEDAGOGY e SPORT, vari all'interno del medesimo testo.

Di conseguenza, il concetto di dominio predominante, ha senso se applicato all'interno di una porzione di testo, piuttosto che rispetto all'intero testo. Supponiamo, per esempio, di dover disambiguare il termine *acrobatics*. Considerando solo la porzione di testo rappresentata dai termini intorno alla parola target, *acrobatics* viene correttamente disambiguato assegnandogli il senso associato al dominio SPORT.

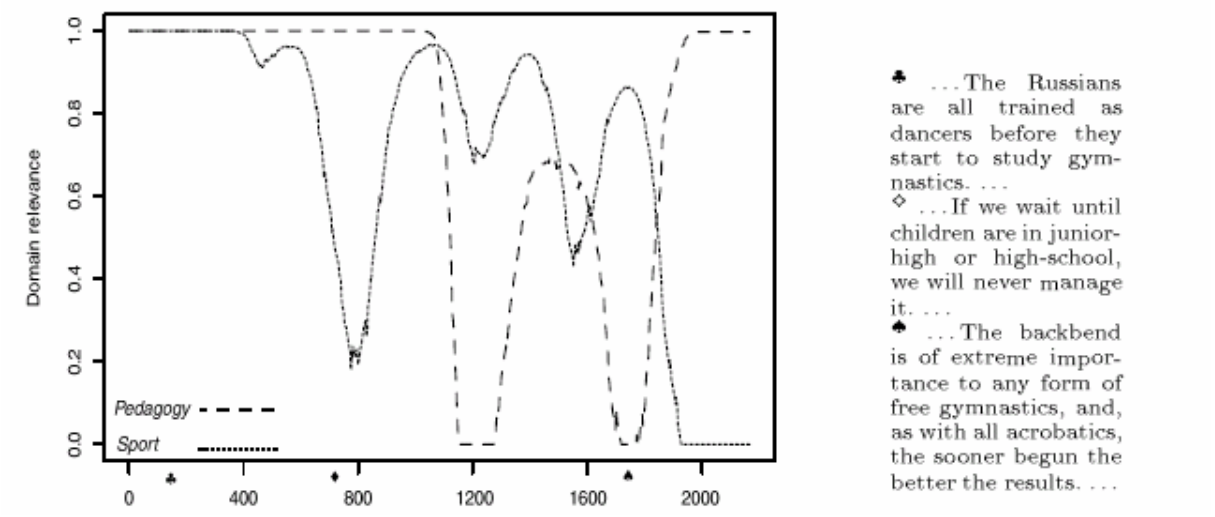


Figura 4.2-Variazioni di dominio all'interno del testo br-e24 del corpus SemCor

4.1.2 Domains Driver Disambiguation

Magnini e Strapparava in [**56**], sulla base delle affermazioni precedentemente dimostrate, propongono un algoritmo di disambiguazione del testo chiamato Domain Driver Disambiguation (DDD). L'idea di base è che il processo di disambiguazione di un termine all'interno del suo contesto, è principalmente un processo di comparazione tra il dominio del contesto e i domini associati ai sensi del termine.

L'algoritmo richiede, in input, porzioni di testo, e fornisce una struttura per integrare l'informazione di dominio acquisita tramite testi annotati.

La struttura dei dati che consente di raccogliere le informazioni di dominio è chiamata vettore di dominio, e la sua lunghezza è determinata dal numero di domini considerati. Tale approccio è stato sperimentato considerando solo, 42 domini e quindi un vettore egualmente lungo.

I vettori di dominio consentono di fondere e gestire con facilità le informazioni riguardanti i sensi dei termini, per porzioni di testo. In particolare si sono utilizzate due tipologie di vettori:

i. **Vettore di Testo**: rappresenta la rilevanza di una porzione di testo calcolata rispetto ad ogni dominio considerato.
ii. **Vettore di Sensi**: rappresenta la rilevanza di un senso di un dato termine, calcolata rispetto ad ogni dominio considerato.

Per disambiguare l'occorrenza di un termine *w* all'interno di una porzione di testo *T*, devono essere calcolati sia il vettore di testo per la porzione di testo *T* intorno al termine *w*, sia tutti i vettori per i sensi *s1, s2, s3… sk,* di *w* . Successivamente il sistema sceglierà il senso il cui vettore, ha massima similarità con il vettore della porzione di testo *T*. *T* può essere rappresentato come una lista di coppie <*lemma, POS*> ottenute tramite un *tagger*. I lemmi sono indicizzati in base alla loro posizione nel testo. D'ora in poi la notazione *Tp,* sarà utilizzata per riferirsi ad una parola collocata in posizione *p* all'interno del testo *T*.

Magnini e Strapparava, rappresentano la rilevanza di un dominio rispetto ad un testo, con un numero reale positivo compreso fra [0, 1]. Dato un dominio, esso avrà rilevanza pari ad 1 per un testo, se rappresenta l'argomento principale,mentre avrà rilevanza 0, se non è relazionato in nessun modo al testo. Per esempio, il testo il cui argomento è "*September 11th attack on the Twin Towers*" può avere una rilevanza pari ad 1, rispetto ai domini POLITICS o MILITARY, e rilevanza 0 per domini come SPORT.

Per calcolare i domini rilevanti per una data porzione di testo, intorno ad una parola in posizione Tp, l'algoritmo prima di tutto identifica la sottosequenza delle parole da *T(p-c)* a *T(p+c)*, dove *2c* è la dimensione del contesto fornita all'algoritmo come parametro. Tale approccio è stato testato su SemCor. La sperimentazione ha dimostrato come le prestazioni dell'algoritmo diminuiscano quando il valore di *2c* supera 50. Il secondo passo dell'algoritmo raccoglie tutte le annotazioni di dominio corrispondenti ai vari synset dei termini, e calcola la frequenza di ogni dominio, all'interno di questo insieme. Tuttavia, Strapparava e Magnini, non credono che la frequenza di un dominio in un testo implichi necessariamente la sua rilevanza all'interno del testo stesso. Per esempio, può accadere che POLITICS sia il dominio più frequente all'interno di un articolo giornalistico, perfino se in realtà l'argomento dell'articolo corrisponda al dominio VETERINARY. Questo può accadere perché le parole legate al dominio VETERINARY, sono meno frequenti dei termini legati al dominio POLITICS. La loro ipotesi è che un dominio è rilevante per un testo, se la sua frequenza nel testo è significativamente più alta rispetto alla frequenza che tale dominio ha in testi con cui non è in relazione.

Per stimare la relazione fra frequenza e rilevanza, Magnini e Strapparava, assumono che all'interno di un generico corpus bilanciato, il numero di testi rilevanti per un certo dominio *D*, è distribuito egualmente. In fase di sperimentazione, ciò significa dover determinare la

deviazione standard per ogni dominio di WordNet Domains all'interno del LOB Corpus [**56**], considerandolo come generico corpus bilanciato per l'Inglese. La rilevanza di dominio è valutata utilizzando teoremi riguardanti la distribuzione normale: se la frequenza di un dominio *D*, calcolata nel testo *T*, è significativamente più alta della frequenza di *D* nel corpus (cioè eccede più del doppio la deviazione standard), allora *D* è rilevante rispetto a *T*.
Per esempio, supponiamo di voler valutare la rilevanza di ECONOMY nella frase " *Today I draw money from my bank*". L'algoritmo individuerà tutti i domini associati da WND ad ogni senso di ogni parola. Il nome *bank* ha 5 occorrenze associate al dominio ECONOMY su 10, il nome *money* ne ha 3 su 3 e il verbo *draw* ha un'occorrenza sola associata ad ECONOMY su un totale di 33. Da qui la frequenza totale di ECONOMY è 1.53. Supponiamo che la frequenza di ECONOMY nel corpus LOB sia di 0.2, e che la deviazione standard sia 0.1. Tale valore rappresenta la distribuzione della frequenza di ECONOMY in testi non correlati. Di conseguenza, ECONOMY non sarà da considerarsi come dominio prevalente,in testi in cui la sua frequenza è compresa nel *range* [0, 0.4]; viceversa sarà considerato rilevante, se avrà una frequenza significativamente più alta, come nel caso in esempio (1.53).

Di seguito, si descriveranno più in dettaglio i due vettori utilizzati in tale algoritmo.
Un vettore di testo è un vettore di dominio, estratto da una porzione di testo. Dato un insieme di domini *D=[D1, D2, …, Dn]*, un testo *T* e una posizione *p*, il vettore di testo *Tp* è il vettore n-dimensionale, il cui componente i rappresenta la rilevanza del dominio i-esimo per *T* alla posizione *p*. Dato un contesto, intuitivamente, *Tp* rappresenta i domini rilevanti per un punto *p* del testo. I vettori di testo calcolati su differenti posizioni dello stesso testo, possono essere diversi, e per lo stesso testo, possono esistere più domini rilevanti.
Un vettore di senso, è un vettore di dominio ottenuto a partire da un synset. Esso fornisce due informazioni importanti: la sua lunghezza, rappresentante la frequenza di occorrenze dei sensi, e le sue direzioni, rappresentanti il vettore "significato" dei testi dove il senso appare generalmente. La via più naturale per costruire i vettori dei sensi, è l'applicazione di tecniche supervisionate a dei dati di *training.* Tuttavia, in questi casi i dati di training non sono quasi mai disponibili. Una via alternativa, consiste nell'ottenere i vettori dei sensi sfruttando le informazioni contenute in WordNet Domains. Questa possibilità, la quale è stata applicata durante gli esperimenti del Senseval-2, rende l'approccio di Magnini e Strapparava, molto flessibile.

Nel caso in cui siano disponibili i dati di *training*, il vettore dei sensi è costruito attraverso la somma dei vettori di testo, considerando la direzione del vettore significato dei testi all'interno del quale compaiono tipicamente i synset. Questo metodo è, inoltre, particolarmente efficace per sensi generici (cioè classificati come factotum), i quali in genere compaiono in vari tipi di testo e producono vettori senza una dimensione dominante. Tuttavia, dati dei testi generici, spesso questi presentano pochi domini prelevanti, ed è necessario un elevato numero di dati di *training*, per produrre vettori di senso generico.
Nel caso in cui non si abbiano a disposizione i dati di *training*, il vettore dei sensi deve essere costruito utilizzando WordNet Domains e SemCor. In questi casi, il vettore dei sensi ha un 1 nelle rispettive posizioni di appartenenza di dominio di WordNet Domains e 0 altrimenti. La sua lunghezza è proporzionale alla frequenza del senso in SemCor. Per esempio, il vettore di *bank#1* nell'esempio in tabella 4.4, sarà ((ECONOMY 20)(ARCHITECTURE 0)…(SPORT 0)). Se il senso viene annotato con *factotum*, il suo vettore dei sensi ha la direzione di un vettore di 1 per ogni componente.

Il processo di disambiguazione di un termine Tp, consiste in un semplice confronto tra, il vettore di testo *Tp* ed i vettori di tutti i sensi di tale parola. Allo scopo di prendere in considerazione sia le direzioni (cioè il dominio) sia la lunghezza (ovvero la frequenza), dei vettori di senso, viene calcolato il prodotto vettoriale tra *Tp* e ogni vettore di senso. Il risultato è un lista ordinata di sensi di *Tp* e la selezione finale si basa sul confronto rispetto ad una soglia fissata.
Da qui derivano tre possibili output:

1. Se il *match* di un senso eccede in maniera significativa la soglia fissata, il senso viene selezionato come più probabile.
2. Se si raggiunge più di un buon *match* (ovvero si hanno due sensi appartenenti allo stesso dominio rilevante), la strategia prevede semplicemente di non assegnare alcun senso al termine, come se non vi fosse nessun altra informazione disponibile.
3. Nel caso in cui non si individui alcun *match*, come può accadere per esempio, con termini altamente polisemici, non si seleziona alcun senso.

Supponiamo per esempio di voler disambiguare la parola *bank*, all'interno della frase "*Today I have draw money from my bank*" rispetto ai sensi *s1: bank#1* ed *s2*: *bank#2*. Tale situazione

è riportata in tabella 4.4, dove sia il vettore dei sensi sia il vettore del testo, sono rappresentati per una sotto insieme di domini. Il prodotto vettoriale tra *T8* ed *s1* da come risultato 1.7356, mentre quello tra *T8* ed *s2* da 0.06185. Di conseguenza si seleziona *bank#1*.

	SPORT	MEDICINE	ECONOMY	GEOGRAPHY
$\vec{s}_1$ $(Bank\#1)$	0.02	0.08	1.73	0.04
$\vec{s}_2$ $(Bank\#2)$	0.005	0.03	0.04	0.69
$\vec{T}_8$	0.2	0.005	1	0.03

Tabella 4.4-Risultati tra bank#1 e bank#2

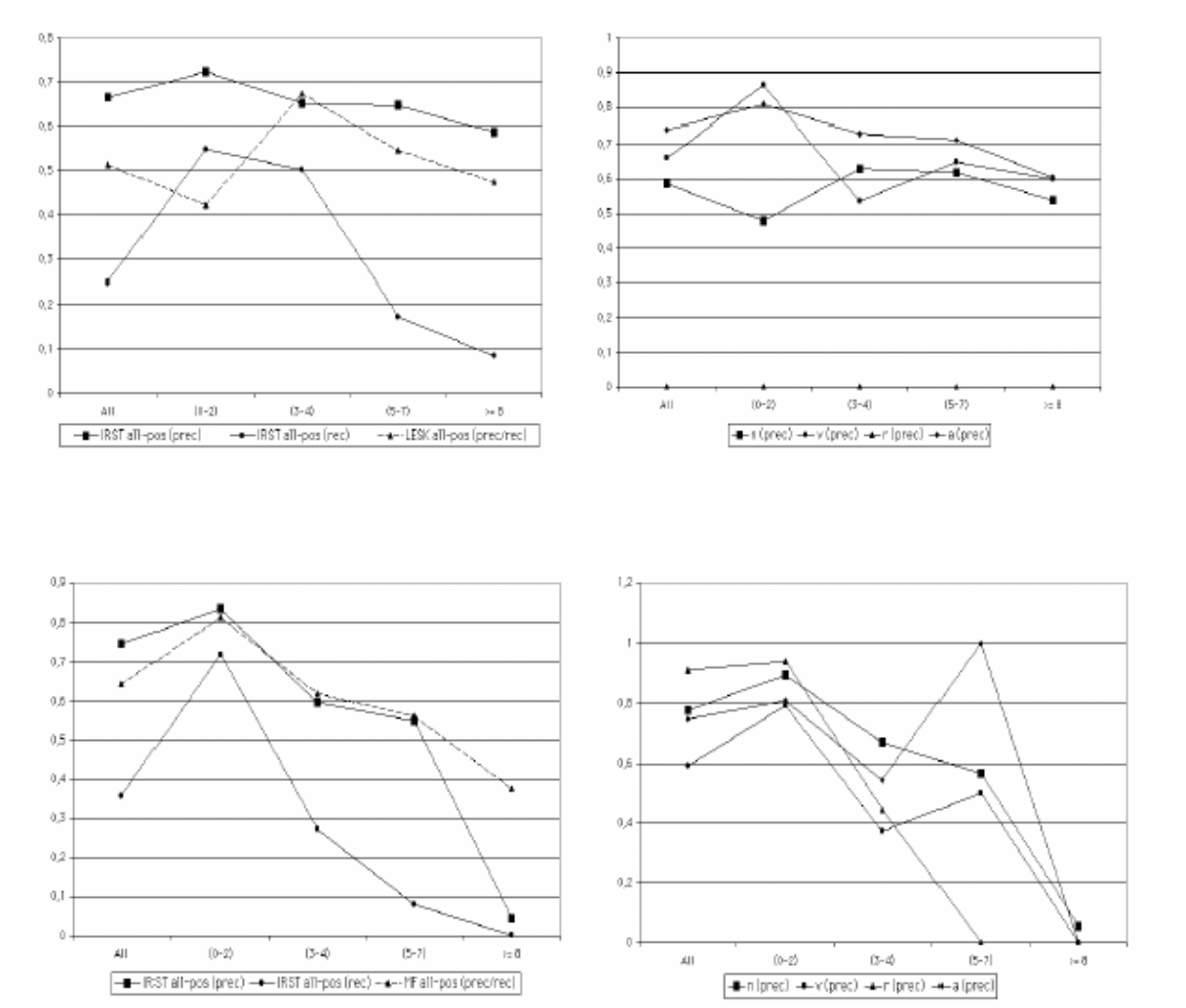


Figura 4.3- Prestazioni dell'algoritmo DDD

Nei grafici in figura 4.3 sono riportati i risultati ottenuti da tale metodo di disambiguazione, durante il Senseval-2. Tali risultati vanno interpretati considerando che, oltre all'informazione di dominio, non si è utilizzata nessun altra risorsa ne sintattica ne semantica. I risultati sono valutati in termini di Recall e Precision così determinate: P= annotazioni corrette/(annotazioni sbagliate + annotazioni corrette), R = (annotazioni corrette / (annotazioni sbagliate + annotazioni corrette + annotazioni non eseguite).

4.2 Extended WordNet

Come abbiamo già detto in precedenza, WordNet è un database lessicale in inglese, il quale trova notevoli ed ampie applicazioni in ambiti come l'Intelligenza Artificiale, la Linguistica Computazionale ecc…Tuttavia, proprio grazie alle sue applicazioni, sono state individuate alcune limitazioni, in particolar modo per quanto concerne il suo utilizzo in applicazioni riguardanti la disambiguazione dei sensi. Spesso si ha la necessità di individuare termini legati tra loro attraverso relazioni particolari, ma la suite di relazioni offerta da WN può non essere sufficiente a tale fine. Da qui nasce quella che rappresenta l'idea chiave alla base del progetto eXended WordNet (XWN), il quale essenzialmente sfrutta il ricco potenziale informativo contenuto all'interno delle definizioni dei termini, ovvero delle glosse dei synset. Tali glosse, in precedenza, erano utilizzate esclusivamente per identificare il significato corretto delle parole.

Il principale intento di XWN, è quindi quello di effettuare in maniera automatica il *parse* sintattico delle glosse per poter trasformare quest'ultime in forma logica, e distinguere semanticamente attraverso un *tagger* i nomi, i verbi, gli aggetti e gli avverbi contenuti nelle glosse stesse. Tutto ciò ha come effetto principale quello di incrementare la connessione tra i synset e fornire l'accesso ad un contesto per ogni concetto.

4.2.1 Realizzazione di XWN

XWN è un progetto descritto in [**58**], che coinvolge le glosse dei synset di WordNet diverse fasi di elaborazione. La fase di disambiguazione delle glosse, richiede una *pre-processing* allo scopo di individuare i concetti e le varie categorie sintattiche dei termini. Un altro elemento fondamentale, è la verifica automatica dell'accuratezza del parsing. L'obiettivo in XWN è quello di costruire un tool che sia in grado di etichettare in maniera automatica, le parole all'interno delle glosse sia da un punto di vista morfologico che semantico.

Uno dei principali requisiti di XWN è la correttezza, che oggi rappresenta ancora, nei processi di disambiguazione, uno dei più difficili obiettivi da raggiungere. Moldovan e Mihalcea, hanno delineato un "compromesso" tra l'etichettare in maniera automatica le glosse, e l'alta precisione richiesta in tale processo. La precisione diviene un requisito fondamentale, in quanto l'obiettivo di XWN, è essenzialmente di fornire strumenti più approfonditi ed efficaci per le applicazioni di elaborazione del testo, in particolare per la disambiguazione dei termini.

Realizzando una risorsa caratterizzata da un mediocre livello di precisione, sarebbe impossibile, ottenere buoni risultati dai metodi che si basano su di essa.

XWN è stato introdotto per la prima volta nel 1999 da Harabagiu in [59]. Il principale contributo alla versione originale di WordNet si ritrova nella disambiguazione delle glosse dei suoi synset e di conseguenza, nelle nuove relazioni che possono essere inferite tra di esse. Tale progetto si è realizzato seguendo le seguenti fasi:

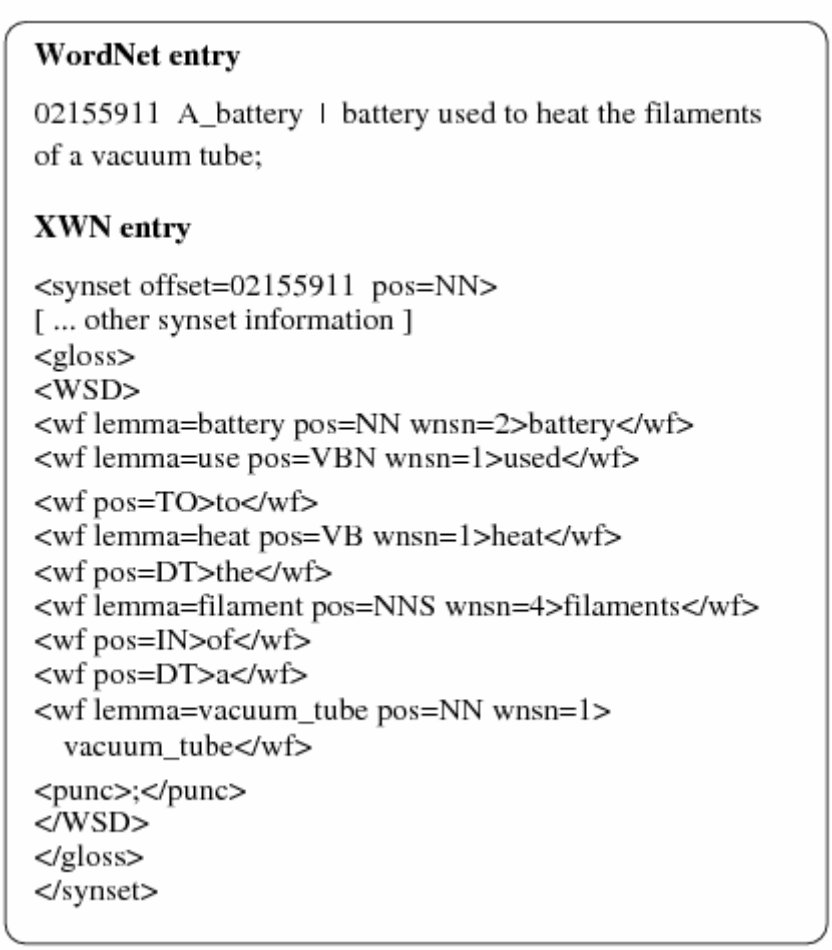
1. *Pre-processing e Parsing.* Questa fase implica la separazione delle glosse nelle due componenti: la definizione vera e propria e gli esempi correlati. Durante questa fase si realizza, inoltre, l'identificazione delle categorie sintattiche di ciascun termine e l'individuazione dei termini composti.
2. *Disambiguazione dei termini*. Tutti i termini all'interno delle glosse sono etichettati con il loro senso appropriato di WordNet. Le parole vengono quindi collegate ai rispettivi synset, ponendo in tal modo le basi per la derivazione delle relazioni topiche che verranno identificate nella fase 4.
3. *Trasformazione in forma logica*. Le glosse sono trasformate in forma logica, consentendo così di essere utilizzate in applicazioni come le inferenze di testo o prove assiomatiche.
4. *Relazioni topiche.* In seguito alla disambiguazione delle glosse, possono essere inferite una serie di nuove relazioni tra i termini, indipendentemente dalla categoria sintattica a cui appartengono, basandosi sulla loro associazione ad un particolare contesto o ad una particolare topica. Tali informazioni possono essere utilizzate in ambiti come *l'Information Retrieval*, per l'estrazione di informazione dai testi, per calcolare la coerenza di un testo e per altre applicazioni.

Una delle prime problematiche affrontate durante la realizzazione di XWN, ha riguardato il formato del suo database. I requisiti più importanti sono l'esigenza di una struttura flessibile e al contempo scalabile: la struttura scelta dove consentire, eventuali future estensioni di XWN stesso, senza compromettere o dover modificare le informazioni in esso già inserite.

Il formato, scelto alla fine, fa uso di *tag* e ricorda la notazione utilizzata in SemCor. In particolare:

- ogni parola deve includere al suo interno l'informazione relativa alla sua categoria sintattica;

- le parole definite in WordNet devono includere un lemma (ovvero la loro forma base), e il campo per il senso;
- gli elementi di puntualizzazione devono essere marcati adeguatamente;
- le varie fasi di elaborazione devono essere separate da simbologie specifiche.



WordNet entry

02155911 A_battery | battery used to heat the filaments of a vacuum tube;

XWN entry

<synset offset=02155911 pos=NN>
[... other synset information]
<gloss>
<WSD>
<wf lemma=battery pos=NN wnsn=2>battery</wf>
<wf lemma=use pos=VBN wnsn=1>used</wf>
<wf pos=TO>to</wf>
<wf lemma=heat pos=VB wnsn=1>heat</wf>
<wf pos=DT>the</wf>
<wf lemma=filament pos=NNS wnsn=4>filaments</wf>
<wf pos=IN>of</wf>
<wf pos=DT>a</wf>
<wf lemma=vacuum_tube pos=NN wnsn=1>
vacuum_tube</wf>
<punc>;</punc>
</WSD>
</gloss>
</synset>

Figura 4.4-Esempio del formato di eXtended WordNet

La figura 4.4 mostra l'esempio di come viene rappresentata una glossa all'interno di XWN dopo aver subito il processo di disambiguazione. Si noti come, in questo caso, in figura 4.4, vengano mostrati solo i risultati della fase di disambiguazione come risulta indicato attraverso i due tag <DISAMBIGUAZIONE DEL TESTO>. I risultati delle fasi successive, sono strutturati e delineati all'interno dei tag <LFT> per quanto concerne la fase di trasformazione logica, e con il tag <TR> per quanto riguarda le risultanti relazioni topiche.

L’idea di combinare diversi classificatori per poter raggiungere un maggior livello di accuratezza, non è nuova in letteratura. Tuttavia, essa non ha mai trovato applicazione, prima di XWN, in ambito di verifica della correttezza dei *parsing*. L’idea è quella di formalizzare il limite inferiore e superiore della precisione che può essere raggiunta nel combinare differenti classificatori. Dati due *parser,* indicati rispettivamente con *T1* e *T2*, e aventi una precisione stimata pari a *PT1* e *PT2,* se i due *parser* concordano su un numero di elementi denotato con *cov*, allora è possibile determinare un limite superiore ed inferiore, alla precisione ottenuta nell’insieme di elementi dove i due *parser* si sono dimostrati concordi. Tali limiti possono essere determinati attraverso una funzione di accuratezza per ciascuno dei *parser* e conoscendo il numero *cov* di elementi di concordanza. Se indichiamo la precisione raggiunta per i *cov* elementi con *Pcov*, i limiti sono così determinati:

$$minP_{cov} = \frac{P_{T_1} + P_{T_2} - 1 + cov}{2 * cov}$$

$$maxP_{cov} \simeq \frac{P_{T_1} + P_{T_2} - 1 + cov + (1 - P_{T_1})(1 - P_{T_1})}{2 * cov}$$

La validità di tale formule, è stata ampiamente dimostrata sperimentalmente. Da qui, può poi essere ricavata una generalizzazione di tale equazione, nel caso si vogliano considerare più di due *parser.*

Moldovan e Mihalcea, ritengono tali formule particolarmente utili nella determinazione della correttezza morfologica e semantica della fase di *parsing*. Essenzialmente, dati due o più *parser,* e ricavando il numero di casi su cui i *parser* concordano sul risultato, è possibile determinare la precisione e la correttezza degli insieme di elementi identificati.

Se tale livello di accuratezza è sufficientemente alto per i propositi dell’applicazione, allora l’utente dovrà preoccuparsi di controllare manualmente solo gli elementi rimanenti, ovvero quelli su cui i *parser* si sono trovati in disaccordo. Se invece, il livello di precisione non è soddisfacente, si può decidere di aumentare il numero di *parser* da combinare insieme per aumentare il livello di accuratezza.

Le fasi iniziali del processo di realizzazione di XWN, necessitano di un livello elevato di accuratezza, per evitare che eventuali imprecisioni si diffondano e ripercuotano sull’intero processo di realizzazione.

Dallo stato dell'arte riguardante i *parser*, deriva che, attualmente, quest'ultimi raggiungano valori di accuratezza intorno al 93-94% del totale dei dati. Sebbene tali valori, possono essere ritenuti buoni, non sono sufficienti per i propositi di XWN. Tale problematica trova soluzione, nei risultati riportati in (Mihalcea e Bunescu 2000), nella determinazione di una combinazione di *parser* che consenta di raggiungere un'accuratezza minima del 98%. In questo modo, e attraverso l'intervento umano per il controllo degli elementi di disaccordo, è possibile raggiungere una accuratezza pari al 100%.

Per quanto concerne l'identificazione dei termini composti, Moldovan e Muhalcea utilizzano un meccanismo semi-automatico realizzato attraverso un *tool* chiamato *xwnPreprocessing.* Tale *tool*, esegue sia il *parsing* che l'individuazione dei termini composti, richiedendo l'intervento umano solo se strettamente necessario. Le funzionalità eseguite da *xwnPreprocces* sono riassunte in figura 4.5.

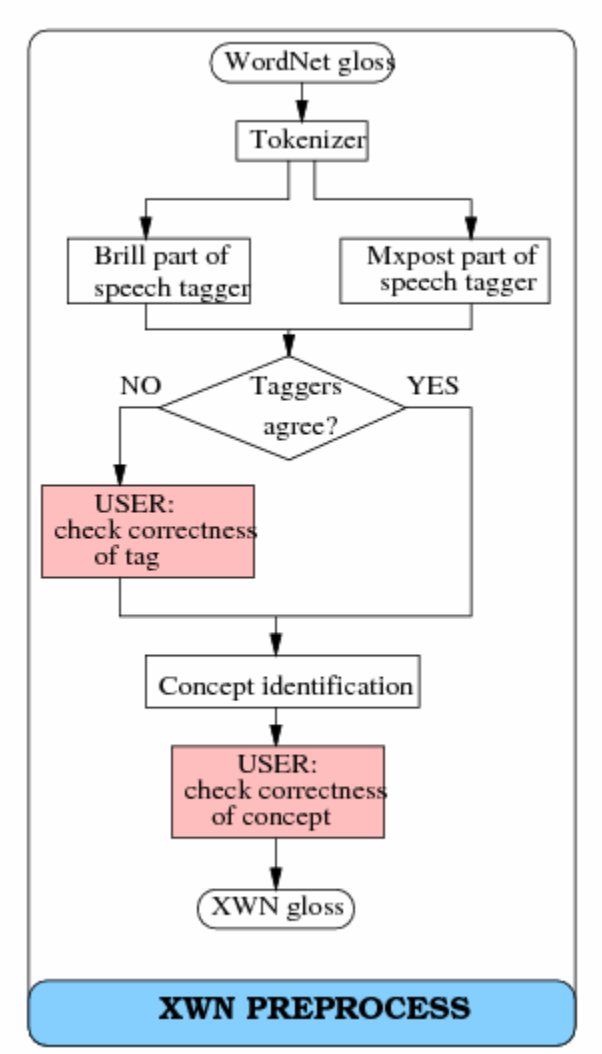


Figura 4.5- Funzionalità del tool xwnPreprocess

La tabella 4.5, riassume le informazioni riguardanti le categorie sintattiche identificate per un campione di 1000 glosse, indicando, inoltre, il numero di termini composti identificati in maniera automatica e quali fra questi sono stati ritenuti corretti dall'utente.
Dalla tabella 4.5, si delinea come l'utente sia dovuto interagire con il *tool* in due situazioni: per indicare la correttezza della categoria sintattica dei termini, nel caso in cui i *parser* si trovassero in disaccordo, e per controllare la correttezza dei termini composti individuati automaticamente. Tali interventi hanno richiesto un tempo di circa quattro ore considerando il campione di 1000 glosse. Tuttavia, tale sforzo risulta non esagerato se si considera che consente di raggiungere una correttezza pari al 100%.

No.	Noun glosses	Verb glosses
Words	5539	3120
Nouns	1699	773
Verbs	758	719
Adjectives	381	181
Adverbs	95	112
Concepts identified	413	191
correct	232	104

Tabella 4.5-Risultati della fase di pre-processing di XWN

Come si evince anche dall'analisi del precedente capitolo, ad oggi non esiste un unico criterio di applicazione nella problematica della disambiguazione dei termini. Gli ultimi algoritmi proposti, mirano a combinare una serie di aspetti e metodologie, proposte negli anni precedenti, allo scopo di superare i limiti dei singoli approcci e ottenere risultati e prestazioni migliori.
La procedura utilizzata in XWN, nel disambiguare i termini delle glosse, combina nuove e vecchie tecniche allo scopo di disambiguare con precisione elevata. Ciò che si richiede, nel caso di XWN, è di ottenere valori di recall pari al 100% e di precision pari al 90%. Tuttavia, tali valori, risultano difficilmente raggiungibili attraverso il solo ricorso ai mezzi automatici. Come abbiamo visto lo stato dell'arte degli algoritmi di disambiguazione non eccede il 70-80% di precisione.

Bisogna inoltre considerare che il processo di disambiguazione semantica delle glosse, non può essere ridotto ad un semplice problema di disambiguazione di testo, in quanto in realtà si traggono vantaggi, dalle informazioni aggiuntive fornite dalla conoscenza del concetto a cui la glossa appartiene.
Il modulo della fase di disambiguazione, è chiamato *xwnDisambiguation*, e il suo input è costituito dalla glossa, nel formato di XWN, ottenuta tramite il *tool xwnPreprocessor*.
Le formule predentemente descritte, ed utilizzate nella fase di valutazione dell'accuratezza dei *parser*, sono riutilizzate per valutare anche l'accuratezza della fase di disambiguazione delle glosse. In questo caso, tuttavia, non si hanno a disposizione due metodi di disambiguazione ad elevata accuratezza da combinare.

Per superare tale limitazione, si decide di assegnare un livello numerico di confidenza ad ogni termine, compreso fra 0 e 2, in base alle seguenti regole:

0- se la parola non viene disambiguata da alcun metodo;
1- se la parola viene disambiguata in maniera concorde da uno o due metodi ma il suo livello di accuratezza non eccede il 90%;
2- se la parola ha un livello di accuratezza superiore al 90% grazie ad un singolo metodo accurato, o grazie a due o più metodi che concordano sul senso da attribuirgli.

Alla fine, vengono ritenuti corretti solo i termini con un livello di confidenza pari a 2; mentre gli altri termini richiederanno l'intervento dell'utente. Il problema diventa, quindi, il criterio di combinazione dei vari metodi. Per la risoluzione di tale problema si utilizzata la formula precedente.
Di seguito si descriveranno brevemente i singoli metodi utilizzati per disambiguare le glosse.

Metodo dei Termini monosemici

Tale metodologia, semplicemente identifica i termini monosemici in base a WordNet, e gli assegna il senso appropriato corrispondente. Per esempio la glossa di *abbey#3* è " *a monastery ruled by an abbot*". In questo caso la parola *abbot* possiede in WordNet un solo senso, conseguentemente non è ambigua e viene semplicemente etichetta con il senos#1.

Relazione di stessa gerarchia

Identifica i termini della glossa appartenenti alla medesima gerarchia all'interno di WordNet. Tale metodo, fu introdotto per la prima volta in (Harabagiu e al., 1999), e si riferiva al solo primo termine della glossa. Moldovan e Mihalcea, generalizzano tale approccio a tutti i termini della glossa. Esempio: la glossa di *devote#1*, è "*pass on or delegate to another*". *delegate#2* appartiene al suo synset hypernymy, di conseguenza tale verbo viene disambiguato assegnandogli il senso #2.

Relazione di parallelismo lessicale

Identifica le parole della glossa collegate tramite una relazione parallelismo lessicale. Tale criterio, fu introdotto in (Harabagiu e al., 1999). Le relazioni di parallelismo vengono determinate semplicemente come coppie di termini separati da una congiunzione o da un comma. Tali termini, di conseguenza, apparterranno alla stessa gerarchia, nel caso siano nomi o verbi, e allo stesso *cluster*, nel caso siano aggettivi o avverbi. Se si verifica che uno dei termini appartenenti a tale coppia è già disambiguato tramite uno dei metodi precedenti, allora il senso associato al termine già disambiguato, rappresenta una restrizione alle possibili combinazioni di sensi fra le due parole. Esempio 1: la glossa del termine aba#2 è "*a fabbric woven from goat and camel hair*" . In tale esempio le due parole legate da una relazione di parallelismo lessicale sono *goat e camel* ; tuttavia *camel* è già semanticamente disambiguato grazie al primo metodo, essendo un termine monosemico. Di conseguenza il solo senso di goat appartenenti alla medesima gerarchia di *camel#1* è *goat#1*. Esempio 2: la glossa di *exert#3* è "*make a great effort at mental or physical task*". *Mental e physical,* sono entrambi termini ambigui, ma esiste un cluster di aggettivi al quale appartengono entrambi con i rispettivi sensi *mental#1 e physical#1*.

Metodo dei Bigrammi di SemCor

Tale metodo sfrutta l'informazione fornita da SemCor. Dato un termine *Wi,* collocato alla posizione i nella glossa, si formano due coppie, una con il termine subito precedente *Wi* all'interno della glossa (coppia *Wi-1 ; Wi*), e una con il termine subito successivo nella glossa (coppia *Wi ; Wi+i*). Successivamente tale metodo va a ricercare tutte le occorrenze di tali coppie all'interno di SemCor. Se in tutte le occorrenze tali coppie, *Wi* si presenta sempre con il medesimo senso *#k*, ed il numero di occorrenze supera una determinata soglia, allora tale

senso è assegnato come più probabile a *Wi*. Esempio: Consideriamo il termine *approval* all'interno del frammento di testo "*committee approval of*", e fissiamo la soglia a 3 occorrenze. Le coppie saranno quindi: "*committee approval* " e "*approval of*". Per la prima coppia non viene ritrovata alcuna occorrenza all'interno del corpus, mentre per la seconda coppia si ritrovano quattro occorrenze:

"with the *approval#1 of* the Credit Association"

"subject to the *approval#1 of* the Secretary of State"

"administrative *approval #1 of* the reclassification "

"recommended *approval#1 of* 1-A classification"

Avendo in tutte le quattro occorrenze (quindi viene superata la soglia) sempre il medesimo senso, il termine viene annotato con il senso #1.

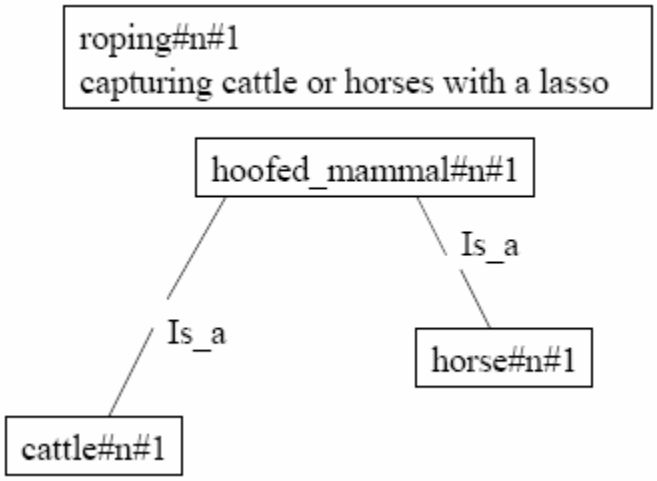


Figura 4.6-Esempio di parallelismo lessicale

Riferimenti incrociati

Dato un termine ambiguo *W* di una glossa *G*, associata al synset *S*, tale metodo consente di determinare, se esiste, un riferimento dalla definizione di uno dei sensi di *W* alle parole nel synset *S*. Se esiste, tale relazione viene considerata un "riferimento incrociato" tra le parole di *S* e un particolare senso di *W*. Tale senso è selezionato come corretto. Esempio1: il synset *{ agora#3, forum#3, public_square#2}* è *"a place of assembly for the people in ancient Greece"*. Il senso #14 di *place* è "*a public square with room for pedestrians";* di conseguenza esiste un riferimento incrociato fra la glossa di *place#14 e agora#3.*

Esempio 2: la glossa del synset { *alarm_clock#1, alarm#4*} è "*wakes sleeper at present time"*. Esistono 10 possibili sensi da associare time, e il suo senso #6 è "*the time as given by a clock"*. Ancora una volta abbiamo trovato un riferimento incrociato fra la definizione di tempo e quella di *alarm*, e possiamo di conseguenza etichettare *time* con il senso #6.

Riferimento incrociato inverso

Dato un termine ambiguo *W* all'interno di una glossa *G* appartenente al synset *S*, tale metodo trova, se esistono, i riferimenti dalla definizione *G* ad un termine dei vari synset associati a *W*. Se esiste tale relazione è considerata come un riferimento incrociato inverso tra la glossa *G* e il synset di un particolare senso di *W*. Tale senso è etichettato come corretto per *W*.

Esempio 1: la glossa di *start#10* è "*begin a work or acting in a certain capacity, office or job"*. Il nome *work* possiede sette significati ed il synset del suo quarto senso è {*job, employment, work}*. Di conseguenza esiste un riferimento incrociato inverso tra i due e il termine viene etichettato.

Esempio 2: la glossa di *withdraw#2* è "*withdraw from active partecipation"*. Dai 17 sensi possibili di *active*, il suo quarto senso appartiene al synset {*active, partecipanting}*, e quindi il quarto senso viene selezionato come corretto.

Distanza fra le Glosse

Dato un termine ambiguo *W* all'interno di una glossa *G*, tale metodo determina il numero di termini in comune tra le glosse di ciascun senso. Tale approccio è una variante dell'algoritmo di *Gloss Overlap* di Lesk (capitolo 3). Nel caso in cui più sensi abbiano lo stesso numero di sovrapposizioni con la glossa del termine target, si sceglie di non associare alcun senso, ma di lasciare che tale termine venga successivamente disambiguato tramite altri metodi.

Esempio 1: il nome *filament* possiede quattro sensi possibili, ma solamente il senso #4 ha la parola *heat* in comune con la glossa data.

Esempio 2: *abacus#1* è associato alla glossa " *a table placet horizontally on topo of the capital of a column as an aid in supportino the architrave"*. La parola ambigua *capital*, ha come glossa per il suo quinto senso " *the upper part of a column that support the entablature"*, quindi ha due parole in comune con la glossa corrente (*support, column)* e tale senso è utilizzato per disambiguare il termine; con tale metodo, in questo esempio, si disambigua anche il termine *architrave#2*.

Dominio Comune

Riprendendo le considerazioni presentate nel precedente paragrafo, Moldoval e Mihalcea, sfruttano anche il metodo legato all'informazione di dominio, limitandosi però ad utilizzare solo i domini forniti all'interno di WordNet.

Esempio: la glossa di *mental#3* è "(*biology) of or relating to the chin- or lip-like structure in insect and certain molluscks*". In questa definizione, *insect* è un termine ambiguo, ma solo uno dei due sensi ad esso associati è legato al dominio (*biology).* Tale senso è selezionato per disambiguare il termine.

Tuttavia, a causa del fatto che in realtà WordNet non associa un dominio in tutti i synset, il dominio di un senso sarà propagato, per questo utilizzo, a tutti i suoi sensi iponomi. In tal caso l'utilizzo di WordNet Domains risulterebbe molto più appropriato.

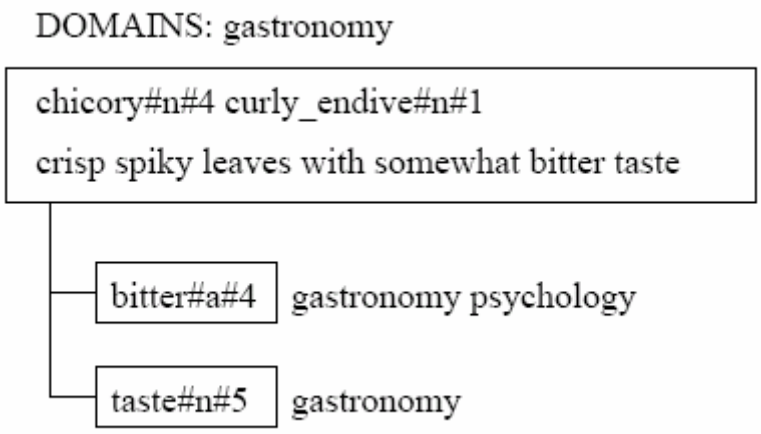


Figura 4.7 -Esempio di Dominio Comune

Moldovan eMihalcea, hanno implementato e testato i primi sette metodi descritti, ottenendo i risultati nella tabella 4.6.

Una volta calcolata la precision e la recall di ciascun metodo, è possibile analizzare una possibile combinazione di quest'ultimi che gli consenta di ottenere le migliori prestazioni in termini di accuratezza. Per realizzare questo, si contano le sovrapposizioni in termini di parole annotate tra i vari metodi, e successivamente si determina la precisione raggiunta su questi elementi etichettati in comune, attraverso la formula iniziale. La tabella 4.7 mostra alcune possibili combinazioni e le relative precisioni ottenibili.

Procedure	Recall	Precision
P.4.2.1	21.3%	100%
P.4.2.2	13.2%	99%
P.4.2.3	11.9%	85.7%
P.4.2.4	16.2%	92.2%
P.4.2.5	4.2%	80%
P.4.2.6	5%	79%
P.4.2.7	17.9%	89.2%

Tabella 4.6-Prestazioni dei primi sette metodi presentati

Proc.	P.4.2.3	P.4.2.4	P.4.2.7	P.4.2.5	P.4.2.6
P.4.2.3	-	80/96	80/98	-	-
P.4.2.4	-	-	85/95	74/98	71/99
P.4.2.7	-	-	-	74/96	69/99
P.4.2.5	-	-	-	-	66/94

Tabella 4.7- Precisione di alcune combinazioni di metodi

Nella nostra spiegazione si sono considerati, per semplicità solo alcuni dei metodi in realtà utilizzati per la realizzazione di XWN. Tuttavia lo scopo era semplicemente di dare un'idea di base del meccanismo che ha portato alla nascita di XWN. Di seguito si daranno informazioni più dettagliate riguardanti la sua struttura e altre caratteristiche.

4.2.2 Informazioni tecniche su XWN

La versione attuale di XWN è la 2.0-1 e deriva direttamente dalla versione 2.0. Tuttavia, essendo un progetto in fase di realizzazione non si escludono future versioni. XWN 2.0-1 può essere scaricato gratuitamente on-line sul sito http://xwn.hlt.utdallas.edu . XWN 2.0-1 software è un database concesso dall'Università del Texas a Dallas attraverso licenza la quale garantisce il permesso di utilizzo, copia, modifica e distribuzione del software, del database e della relativa documentazione per ogni proposito e senza compenso o diritti di autore, con rispetto ai diritti di copyright

Gli utilizzi e le possibili applicazioni di XWN, sono varie, e includono, ovviamente, tutti gli ambiti di applicazione di WordNet. Le glosse, infatti, contengono una parte della conoscenza del mondo, in quanto definiscono, in linguaggio inglese, i concetti più comuni.Il progetto di XWN, è stato fondato dalla *National Science Foundation*. Il responsabile del progetto è il Prof. Dan Moldovan, assistito dalla Prof.ssa Sanda Harabagiu.

Fase di parsing delle glosse

XWN, come abbiamo già descritto in precedenza, esegue il *parsing* delle glosse. La fase di *parsing*, consente l'identificazione delle varie componenti sintattiche della glossa. Tuttavia la glossa, subisce inizialmente una fase di *pre-processing*. Le glosse di WN infatti, sono composte oltre che dalla definizione vera e propria anche da specifiche tra parentesi e da esempi indicati tra virgolette.

Quest'ultimi due elementi vengono ignorati, e solo la definizione vera e propria subisce l'elaborazione. Allo scopo di migliorare l'accuratezza del *parsing*, le glosse vengono estese secondo le seguenti regole:

-le glosse di avverbi sono estese con l'avverbio stesso seguito da "*is*", posti all'inizio della glossa stessa.

Es: *entirely is* without any others being included or involved.

-le glosse di aggettivi sono estese con l'aggettivo stesso seguito da "is something", posti all'inizio della glossa stessa;

-le glosse dei verbi sono estese con "to"+ il verbo +"is to", posti all'inizio della glossa stessa;

-le glosse dei nomi sono estese con il nome stesso seguito da "is", posti all'inizio della glossa stessa.

XWN, per effettuare il *tagger* delle glosse utilizza il Brill's tagger.

Allo scopo di ottenere un *parsing* il più efficiente ed affidabile possibile, si utilizzano due *parser* differenti: il *parser Chraniak* ed un *parser in-house* molto simile a quello di Collins.

Il *parsing* delle glosse è stato poi classificato in tre categorie distinte, a seconda del livello di accuratezza ottenuto:

-è attribuita qualità di tipo GOLD quando il *parser* è stato controllato manualmente;

-è attribuita qualità di tipo SILVER quando entrambi i *parser* utilizzati concordano sull'esito ottenuto;

-è attribuita qualità di tipo NORMAL in tutti gli altri casi, ovvero quando i due *parser* danno esiti differenti e non vi è alcun intervento umano. In questo caso viene scelto l'esito del *parser in–house.*

POS	# TOTAL	# GOLD	# SILVER	# NORMAL
Noun	94,633	0	40,064	54,569
Verb	14,596	14,596	0	0
Adjective	20,279	14,568	5,711	0
Adverb	4,005	1,154	2,851	0

Tabella 4.8-Parse trees per ciascuna categoria sintattica

Trasformazione in Forma Logica (LFT)

Una forma logica è uno stadio intermediario tra il *parse* sintattico e la forma semantica.

Il processo di trasformazione in forma logica riceve come input il *parse-tree* della glossa, e successivamente assegna i predicati e gli argomenti alle parole. Anche in questo caso si definiscono LTF GOLD, SILVER o NORMAL. Di seguito si riporta la tabella con la distribuzione delle LTF per ogni categoria sintattica.

POS	# TOTAL	# GOLD	# SILVER	# NORMAL
Noun	94,868	32,844	7,228	54,796
Verb	14,441	14,441	0	0
Adjective	20,380	16,059	4,321	0
Adverb	3,994	3,994	0	0

Tabella 4.9-Trasformazioni in forma logica per ciascuna categoria sintattica

Annotazione Semantica delle Glosse di WN

Il progetto XWN aiuta a trasformare le glosse di WN in un formato che consenta la derivazione di nuove informazioni semantiche e di nuove relazioni logiche.

In XWN, ad ogni annotazione, è stata attribuita un'etichetta che da indicazioni rispetto al livello di precisione dell'annotazione stessa:

- *Gold* quando la correttezza dell'annotazione è stata controllata manualmente;

- *Silver* per i termini annotati in maniera automatica e con esito concorde di entrambi i sistemi di disambiguazione;
- *Normal* per il resto dei termini annotati automaticamente facendo riferimento al sistema *xwnDisambiguation.*

Per esempio, termini come i verbi "*to have"* o "*to be*" sono stati sottoposti ad annotazione automatica.Di seguito è riportata una tabella, la quale mostra la suddivisione delle glosse e dei termini, rispetto ad ogni parte del parlato con riferimento alla versione 2.0-1 di XWN.

Set of glosses	Number of glosses	Open class words	Monosemous words	"Gold" words	"Silver" words	"Normal" words
Noun glosses	79,689	505,946	138,274	10,142	45,015	296,045
Verb glosses	13,508	48,200	6,903	2,212	5,193	30,813
Adjective glosses	18,563	74,108	14,142	263	6,599	50,359
Adverb glosses	3,664	8,998	1,605	1,829	385	4,920

Tabella 4.10-Termini disambiguati per ogni categoria

Formato del file

Le informazioni descritte in precedenza sono raccolte all'interno di un file xml. Per rendere più esplicita la struttura del contenuto di tali file si rende opportuno, riportare di seguito una parte di definizione dello schema xml, riguardante la disambiguazione semantica.

```
<xsd:simpleType name="puncType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="([^a-zA-Z0-9])+"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="wfType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="pos" type="wPosType" use="required"/>
      <xsd:attribute name="lemma" type="xsd:string" use="optional"/>
      <xsd:attribute name="quality" type="qualityType" use="optional" default="normal"/>
      <xsd:attribute name="wnsn" type="senseType" use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="wsdType">
  <xsd:all>
     <xsd:element name="punc" type="puncType" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element name="wf" type="wfType" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:all>
</xsd:complexType>
```

```
<xsd:element name="xwn">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="gloss" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="synonymSet" type="xsd:string"/>
            <xsd:element name="text" type="xsd:string"/>
            <xsd:element name="wsd" type="wsdType"/>
            <xsd:element name="parse" type="parseType" minOccurs="1" maxOccurs="unbounded"/>
            <xsd:element name="lft" type="lftType" minOccurs="1" maxOccurs="unbounded"/>
          </xsd:sequence>
          <xsd:attribute name="synsetID" type="synsetIDType" use="required"/>
          <xsd:attribute name="pos" type="glossPosType" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="ver" type="xsd:string"/>
    <xsd:attribute name="wnver" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

Ogni file e limitato dal tag <*xwn*>. Tale tag contiene l'attributo *"ver"* il quale indica la corrente versione di riferimento (2.0-1), e l'attributo "*wnver"* il quale, invece, indica la versione di riferimento di WN (2.0). Le glosse sono rappresentate all'interno del tag <*gloss*>, il quale include l'insieme di sinonimi, il testo della glossa, il parse tree, la forma logica, e la disambiguazione semantica dei termini della glossa. In particolare, la parte riguardante la disambiguazione del testo include termini rappresentati dal tag <*wf*>. Il tag <*wf*> possiede i seguenti attributi:

- *pos*-attributo necessario, indica la parte del parlato al quale il termine appartiene secondo il *tagger* di Brill;
- *quality*-tale attributo può assumere tre valori distinti *"Gold", "Silver", "Normal"*, e indica, come specificato in precedenza la qualità dell'annotazione semantica;
- *lemma*- rappresenta lo *stem* del termine;
- *wnsn*-rappresenta il senso scelto per l'annotazione del termine.

Per ogni synset viene generato un datagramma <*gloss*>...</*gloss*>.

Il tag "*gloss"* possiede:

- un attributo "*pos"* il quale esprime la parte del discorso rappresentata dal termine in questione, nel nostro caso un avverbio;
- un attributo "*synset ID"* il quale identifica in maniera univoca il synset in considerazione.

All'interno del tag <*gloss*>...</*gloss*> sono inseriti altri tag:

- *synonymSet* :l'insieme dei sinonimi del synset;

- *text* :la glossa del synset (definitione + esempi);
- *disambiguazione del testo* : il senso disambiguato della definizione della glossa;
- *parse* : il parse dell'albero della definizione della glossa ;
- *lft* : la traduzione in forma logica della definizione della glossa.

Di seguito si mostrerà un esempio del contenuto del file xml relativo agli avverbi:

```
<gloss pos="ADV" synsetID="00002223">
  <synonymSet>barely, hardly, just, scarcely, scarce</synonymSet>
 <text>
  by a small margin; "they could barely hear the speaker"; "we hardly knew them"; "just missed being hit"; "had scarcely rung the bell when the door flew open"; "would have scarce arrived before she would have found some excuse to leave"- W.B.Yeats
</text>
  <disambiguazione del testo>
     <wf pos="IN" >by</wf>
     <wf pos="DT" >a</wf>
     <wf pos="JJ" lemma="small" quality="gold" wnsn="1" >small</wf>
     <wf pos="NN" lemma="margin" quality="gold" wnsn="2" >margin</wf>
  </disambiguazione del testo>
<parse quality="SILVER">
(TOP (S (ADVP (RB barely) )
      (VP (VBZ is)
          (PP (IN by)
              (NP (DT a) (JJ small) (NN margin) ) ) )
       (. .) ) )
</parse>
 <lft quality="GOLD">
 barely:RB(e1) -> by:IN(e1, x1) small:JJ(x1) margin:NN(x1)
 </lft>
</gloss>
<---END EXAMPLE--->
```

Gli attribuiti in *"wnsn"* sono ricavati e ottenuti utilizzando diversi metodi di disambiguazione del testo alcuni dei quali sono stati presentati nel paragrafo precedente. Nel paragrafo successivo si descriverà brevemente il processo di disambiguazione del testo delle glosse di WN.

Partecipazione al senseval-3

Come descritto in precedenza Senseval è un'organizzazione internazionale il cui scopo è quello di valutare i sistemi di disambiguazione del testo, riferiti e rivolti a differenti linguaggi e a diffenti aspetti del linguaggio stesso. XWN è stato presentato durante il Senseval-3, ottenendo nel complesso risultati significativi (82% di precision, 79% di recall), relativamente alla disambiguazione delle glosse.

4.2.3 Algoritmo di Disambiguazione basato su XWN

A dimostrazione dell'utilità di XWN, descriviamo un algoritmo di disambiguazione del testo proposto in [**60**], ispirato alle catene lessicali che utilizza come risorsa di conoscenza non solo WordNet ma anche la sua estensione XWN. In particolare, come vedremo, combinando diversi metodi di disambiguazione, sfrutta le relazioni che possono essere dedotte dai termini delle glosse disambiguate.

In WordNet come sappiamo, ogni concetto è circondato da un *micro* contesto rappresentato dalla sua glossa. Tale algoritmo estrae da questo *micro* contesto, i concetti più rappresentativi e realizza un'unica lista di relazioni topiche. Successivamente, per ogni coppia di synset *Si* ed *Sj*, fornisce un meccanismo per determinare i percorsi che collegano tali due concetti tramite realzioni topiche, ovviamente se tali percorsi esistono. In tal modo si possono calcolare le relazioni topiche di un concetto, dal punto di vista di un altro concetto. Naturalmente da principio può essere esteso a più di due concetti.

Le relazioni topiche possono essere estratte da vari sorgenti. Per primo, questo algoritmo considera le glosse dei vari synset. Poiché i concetti all'interno delle glosse sono utilizzati per definire il synset, essi sanno chiaramente in relazione con quest'ultimo.

Ogni concetto *Ci,j* all'interno estratto dalla glossa del synset *Si*, punta ad un synset *Sj* che ha all'interno della propria glossa. Il concetto *Cj,k* può anche essere rilevante per il synset originario *Si.* Sebbene, tale meccanismo possa essere espanso ulteriormente, Moldovan e al. si fermano a questo primo livello.

Altre sorgenti sono rappresentate dall' ipernomo S_{Hi} e dalla sua glossa associata al concetto $C_{Hi,l}$, e dalle glosse nelle quali è usato il synset *Si*. Questi synset, indicati in figura 4.8, con *Sm* possono essere messi in relazione con *Si*, in quanto fanno parte della sua definizione.

In tal modo, è possibile relazionare fra loro, synset associati a categorie sintattiche differenti, incrementando anche la connessione all'interno delle gerarchie di WordNet.

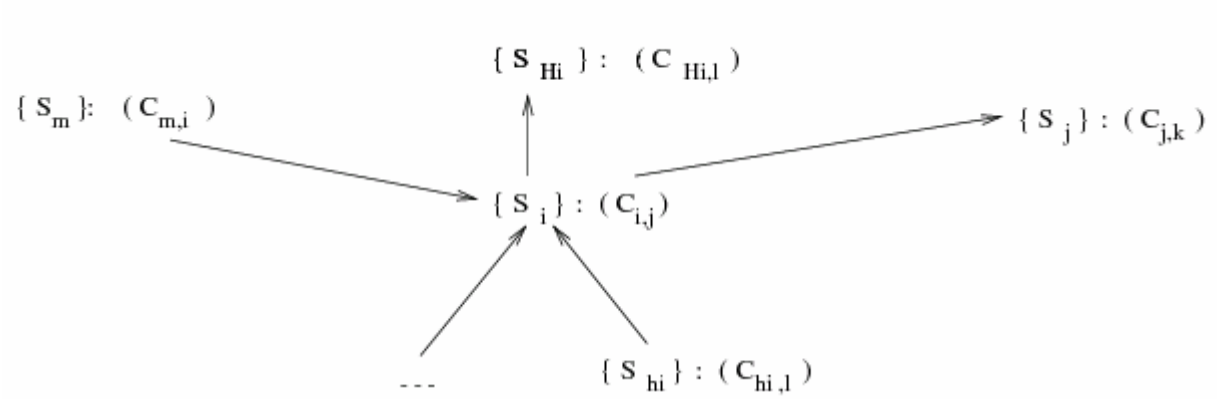


Figura 4.8- Esempio di relazioni ricavate da sorgenti differenti

La figura 4.8,mostra cinque synset differenti, ognuno con una propria lista di relazioni, rappresentata attraverso una linea verticale. All'interno della lista di *Si* esisteranno relazioni indicate con r_{ij} che puntano al synset S_j. Moldovan e al. hanno sviluppato un software che consente di determinare in maniera automatica i percorsi di connessione tra due synset posti ad una certa distanza come mostrato in figura 4.9.

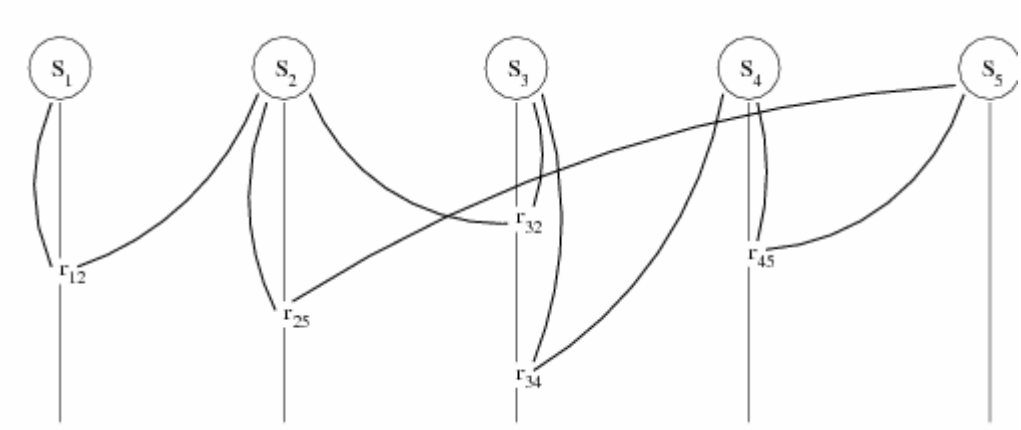


Figura 4.9 Percorso stabilito fra synset attraverso le relazioni che li legano

Il percorso *V* in figura 4.10, collega direttamente *Sj* con *Si*, attraverso le relazioni $r_{i,j}$ o $r_{j,i}$. Il percorso *W* consente un synset intermediario fra *Si* ed *Sj*. In maniera del tutto simile il percorso *VW* consente due synset intermediari e *WW* tre. In questo algoritmo si considereranno solo percorsi con lunghezza massima pari a cinque synset. Per ogni tipo di percorso, esistono differenti tipi di connessioni a seconda del verso considerato.

Se, per esempio, si suppone di voler trovare se esiste o meno una relazione tra due termini, il sistema proverà a determinare tutti i possibili percorsi tra i sensi delle due parole.

Name	**Path**
V - path	$S_i - S_j$
W - path	$S_i - S_k - S_j$
VW - path	$S_i - S_k - S_l - S_j$
WW - path	$S_i - S_k - S_l - S_m - S_j$

Figura 4.10-Tipi di percorsi possibili

Il metodo di disambiguazione combina tre approcci:

1. euristiche;
2. densità concettuali;
3. statistiche su corpus annotati.

I metodi sono elencati per livello di accuratezza decrescente, ma per livello di generalità di applicazione crescente.

Le catene lessicali, in molti approcci, sono stabilite utilizzando solo le relazioni di WordNet. In tale algoritmo, invece, si sfruttano anche le relazioni derivanti dai legami fra le glosse. Tuttavia, non tutte le relazioni hanno lo stesso peso nel relazionare due synset. L'algoritmo assegna un peso ad ogni relazione in base ad una valutazione empirica che porta ai risultati riportati in tabella 4.11.

Le prime tredici relazioni sono estratte da WordNet e il loro peso è definito in letteratura. La relazione GLOSS, sussiste tra un synset e le parole della sua glossa, mentre la relazione RGLOSS sussiste fra le parole della glossa e i rispettivi synset.

In questo algoritmo vengono, poi, applicate una serie di *euristiche* allo scopo di determinare il percorso migliore e quindi il senso corretto; tali euristiche sono:

1) I percorsi brevi sono generalmente preferibili a percorsi lunghi.
2) Le relazioni tra i synset non sono tutte eguali ma vano pesate in base ai valori in tabella 4.11.
3) L'ordine delle relazioni che compongono un singolo percorso è rilevante.
4) Il tipo di nodi incontrati lungo un percorso è rilevante.

Relation	Weight
HYPERNYM	08
HYPONYM	0.7
ENTAIL	0.7
SIMILAR	0.9
IS-MEMBER-OF	0.5
IS-STUFF-OF	0.5
IS-PART-OF	0.5
HAS-MEMBER	0.5
HAS-STUFF	0.5
HAS-PART	0.5
CAUSE-TO	0.5
SEE-ALSO	0.5
PERTAIN	0.5
GLOSS	0.6
RGLOSS	0.2

Tabella 4.11-Peso attribuito a ciascuna relazione

La motivazione alla base dell'euristiche 1 e 2 è ovvia. Per quanto concerne l'euristica numero tre, si considerino i seguenti percorsi:

(C1) → R-GLOSS → (C2) → GLOSS (C3)

(C4) → GLOSS → (C5) → R-GLOSS (C6)

C1 e C3 appartengono alla stessa glossa, mentre C4 e C6 sono in relazione solo attraverso C5. Il primo percorso è più "forte" del secondo, e ciò è dovuto all'importanza dell'ordine in cui si susseguono le relazioni. L'euristica numero 4, deriva dal fatto che percorsi che passano attraverso concetti comuni come, per esempio, *have:v#1*, sono in genere più "deboli" di percorsi che incontrano termini specifici.

Il numero di percorsi fra due concetti, può essere un indice di quanto siano in relazione quest'ultimi: più percorsi esistono e più forte sarà il legame li unisce.

Moldovan e al. realizzano un algoritmo che consente di individuare in maniera automatica se due concetti sono in relazione fra loro. Dato un insieme di concetti, si assegna un peso ad ciascun concetto, mentre a tutti gli altri concetti contenuti in WordNet è associato peso zero. Ogni concetto vicino, riceverà un valore ottenuto dal prodotto tra il peso originale del

concetto e il peso della relazione. Se si indica *Wg* il peso originale del concetto e con *Wr* il peso della relazione, allora il valore del concetto a distanza uno è *Wd1*:

$$Wd1 = Wg * Wr$$

Per esempio consideriamo il concetto *mother:n#1*. Noi assegniamo a tale concetto peso 10. Il concetto *parent:n#1* il quale è *hypernym* del concetto *mother:n#1*, riceve valore 8. I concetti *mommy:n#1*, *mother_in_law:n#1*, *surrogate_mother:n#1*,*Mary:n#1* i quali rappresentano *hyponym* di *mother* ricevono punteggio 7. I concetti *woman:n#1*, *give_birth:v#1*, *child:n#1* ricevono valore 6. Il concetto *grandma:n#1* il quale possiede il concetto *mother:n#1* all'interno della sua glossa riceve valore 2.

Successivamente, i concetti con distanza 2 dal concetto originale, ricevono un valore risultante dal prodotto del peso originale, moltiplicato per il peso della prima relazione e per il peso della seconda relazione; tale valore, è successivamente aggiustato in base ad un parametro che considera l'ordine delle due relazioni. Se si indica con *Wd2* il valore del concetto a distanza 2 dal concetto originale, e con *Wr1* e *Wr2* i pesi di ciascuna relazione, e con *A1,2* il parametro di aggiustamento che considera l'ordine delle relazioni *R1* e poi *R2a,l* formula che si ottiene è la seguente:

$$Wd2 = Wg * Wr1 * Wr2 * A1,2$$

I valori dei vari parametri sono stati determinati sperimentalmente. In particolare ad $(A_{r\text{-}gloss,\ gloss})$ è assegnato il valore 3, mentre ad (*A gloss, r-gloss*) il valore 0.1, ad $(A_{iperonimia,\ iponimia})$ è associato il valore 2, mentre per $(A_{\ iponimia,\ ipernimia})$ è associato il valore 0.8. Per tuuti gli altri percorsi d lunghezza due, al parametro A si associa il valore 1.

In generale, dato un concetto *C'* a distanza *N* dal concetto originale *C*, e tale che tra il concetto *C* e il concetto *C*' esiste un percorso che contiene le relazioni *r1*, *r2*, …, *rn*, allora il concetto *C*' riceverà come valore:

$$Wdn = Wg * Wr1* Wr2*A1,2*Wr3* A2,3 *…* Wrn*An-1,n$$

Consideriamo ora la misura di genericità di un concetto indicata con *MGc*. Tale misura dipende dal numero di glosse all'interno delle quali il concetto compare detto *Nr –gloss.* In particolare la relazione è la seguente:

$$MGc = CONST / (CONST + Nr\text{-}gloss)$$

dove per questo algoritmo si è scelto *CONST= 500.*
Se il percorso fra *C* e *C'* è:

$$C \rightarrow R1 \rightarrow C1 \rightarrow R2 \rightarrow C2 \rightarrow \ldots \rightarrow Cn\text{-}1 \rightarrow Rn \rightarrow Cn = C'$$

la formula per il calcolo del valore del concetto destinazione *C'* diventa:

$$Wdn = Wg * Wr1 * MGc1 * Wr2 * A1,2 * MGc2 * \ldots * Wrn * An\text{-}1,n * MG\ cn$$

Poiché ad un concetto si può giungere attraverso più percorsi, il valore totale è dato dalla soma dei singoli valori dei percorsi. Per esempio il concetto *parent:n#1* è ipernomo del concetto *mother:n#1,* ed esso riceve un valore 8; ma il concetto *mother:n#1* compare anche nella sua glossa, così che tra *mother:n#1* e *parent:n#*1 sussista una relazione di R-GLOSS. Da qui il concetto *parent:n#1* riceve anche il valore 2. Di conseguenza i valori 8 e 2 vengono sommati e al concetto alla fine verrà associato il valore 10. Tale comportamento esprime il principio espresso precedentemente, che affermava che più percorsi esistono fra due concetti e più questi saranno in relazione fra loro.
Di seguito si riporterà il codice dell'algoritmo di diffusione dei pesi:

Inputs:
the weights for all WordNet concepts;
the weight of source concept (WG);
the current path from the source concept (path);
exception list of concepts;
left distance (d).

Output:
updated weights for all WordNet concepts.

1. if ($d < 0$)
then return;
2. current_concept = last concept in the path
3. compute the current value for the current path following the formula:
$W_{DN} = W_G * W_{R1} * MG_{C1} * W_{R2} * A_{1,2} * MG_{C2} * ... * W_{RN} * A_{N-1,N} * MG_{CN}$;
4. add this value to the current concept.
$weight[current\_concept] += W_{DN}$
5. For all neighbour concepts i of current_concept that are not already in the path or in the exception_list do:
$new\_path = path \cup current\_node$;
6. call SpreadWeights(weight[], WG, new_path, exception_list, $d - 1$);

Capitolo 5

5 Intergrazione di WordNet Domains in MOMIS

Il problema di origine dal quale nasce questa tesi, è il tentativo di identificare tecniche o strumenti che consentano all'utilizzatore di MOMIS di poter annotare in maniere automatica o almeno semi-automatica, i termini contenuti all'interno di una determinata risorsa . Come abbiamo già detto in precedenza, attualmente, il processo di annotazione è reso possibile solo attraverso un procedimento di annotazione completamente manuale, assistito da WordNet Editor, che vede come risorsa di conoscenza lessicale e semantica l'ontologia di WordNet (come descritto nel capitolo 2). Il processo di annotazione, presuppone l'esigenza di disambiguare i termini, ovvero di attribuirgli il senso corretto a seconda di quello che è il contesto all'interno del quale tali termini vengono utilizzati .

Tale tesi mira, appunto, ad individuare le possibili tecniche e i possibili algoritmi che potrebbero trovare applicazione nella nostra casistica.

Tuttavia, come ampiamente ripetuto in precedenza, l'analisi di tali tecniche ha individuato principalmente, alcune lacune all'interno di WordNet come esclusiva risorsa, in problematiche di disambiguazione del testoDi conseguenza, sono state analizzate alcune possibili risorse lessicali di conoscenza che potessero ampliare WordNet. In particolare, nel precedente capitolo, si sono descritti XWN e WND. Successivamente si è scelto, in base alle motivazioni descritte di seguito, di integrare all'interno di MOMIS, e in particolare del suo database semantico "momiswn", WND allo scopo di realizzare e testare un processo di disambiguazione semi-atomatico e non supervisionato.

Questo capitolo è così organizzato: nel primo paragrafo si illustreranno le motivazioni alla base della scelta di WND come risorsa da integrare a WordNet; il secondo paragrafo, descriverà in maniera dettagliata il database WND, analizzando la struttura dei dati e della sua gerarchia di dominio; il terzo paragrafo, descriverà brevemente il processo d'integrazione di WND in MOMIS e le conseguenti modifiche al suo database semantico; infine, l'ultimo paragrafo, riporterà i risultati dei test effettuati allo scopo di verificare l'effettiva utilità di WND in MOMIS.

5.1 Motivazioni

L'obiettivo principale di questa tesi, è quello di individuare metodologie ed algoritmi di disambiguazione del testo, allo scopo di consentire un processo di annotazione, automatica o semi-automatica dei termini provenienti da differenti sorgenti d'informazione.
La nostra analisi, ha delineato l'esistenza di molteplici algoritmi ed approcci al problema della disambiguazione, molti dei quali non sono neppure stati citati, rappresentando solo un tentativo isolato

Ciò che emerge chiaramente, può essere essenzialmente riassunto nei seguenti punti:

- *Impossibilità di disambiguare correttamente tutti i termini*. Un processo che consenta di disambiguare in maniera corretta tutti i termini estratti da una sorgente di dati, ad oggi, è praticamente irrealizzabile. Ciò è da attribuirsi, sia all'alto livello di complessità della linguistica, sia a alle limitazioni intrinseche delle sorgenti di conoscenza oggi disponibili;
- *Evidenza delle limitazioni nell'uso di WordNet*.Nonostante WordNet abbia riscosso un ampio successo nell'ambito della disambiguazione del testo (e in generale nell'analisi lessicale), dal suo utilizzo sono emerse alcune lacune che possono essere così riassunte:
  - Mancanza di un lessico specifico per determinati settori di applicazione.
  - L'assenza di relazioni esplicitamente rappresentate fra synset relativi allo stesso dominio ma appartenenti a categorie sintattiche differenti.
  - Il numero insufficiente di interconnessioni fra termini utilizzati nello stesso dominio.

- L’insieme limitato di relazioni lessico-semantiche rappresentate.
- Il livello di granularità spesso eccessivo per le applicazioni comuni, di distinzione fra synset (es. alcuni verbi posseggono più di 40 sensi possibili).
- La mancanza di completezza per quanto riguarda i termini composti, i quali risultano difficili da annotare, con gli strumenti forniti da WordNet

A causa delle limitazioni precedentemente delineate, negli ultimi anni sono stati sviluppati sistemi che tentano di sopperire a tali mancanze, estendendo WordNet. Tra queste le più utilizzate sono XWN e WND

- *Necessità di utilizzo di metodi composti di disambiguazione*. Dall’analisi dei singoli approcci al problema della disambiguazione, è emerso come in realtà quest’ultimi se applicati singolarmente, consentano di ottenere prestazioni limitate. Negli ultimi anni, la ricerca nell’ambito della disambiguazione, si sta orientando sempre più verso la realizzazione di algoritmi che combinino due o più singoli approcci allo scopo di colmare le lacune ed i limiti che l’utilizzo dei singoli ha . Ciò che si viene a delineare, è l’idea che la soluzione al problema della disambiguazione del testo debba essere inteso come un processo incrementale e selettivo, che consenta di restringere gradualmente il campo dei sensi possibili da attribuire ai termini.

- *Analisi della tipologia di sorgente di dati ed del livello di precisione richiesto dall’applicazione.* Un altro elemento importante emerso dalla nostra analisi, a sostegno degli approcci composti, è che in realtà la metodologia da applicare dipende strettamente dalla tipologia dei dati con cui si ha a che fare. Di conseguenza e seconda che si debba annotare un testo piuttosto che una gerarchia di directory, l’approccio dovrà essere differente, in quanto la tipologia di termini e di dati che li caratterizza è molto differente. Inoltre, in un contesto che prevede l’integrazione di multiple sorgenti eterogenee, l’accuratezza dei risultati diviene un elemento fondamentale. Di conseguenza l’obiettivo è quello di ottenere un livello alto di accuratezza delle annotazioni anche a scapito della copertura. In generale un’applicazione che vuole implementare un algoritmo di disambiguazione, deve stabilire inizialmente l’ordine di priorità tra la correttezza e la completezza dei risultati

L’analisi di tutti questi elementi ha portato a concentrare la nostra attenzione su una delle estensioni di WordNet, WND.

La scelta sull’estensione da adottare è ricaduta su WordNet Domains, in quanto, esso si presenta come un’efficace soluzione a molte delle limitazioni precedentemente delineate. In particolare, consente di relazionare in maniera esplicita synset relativi allo stesso dominio ma appartenenti a categorie sintattiche differenti (2); consente di incrementare, anche se parzialmente il numero di interconnessioni fra termini utilizzati nello stesso dominio (3); aumenta il livello di relazioni lessico-semantiche (4); consente di abbassare il livello di granularità di distinzione fra i synset eccessivo in WordNet (5).

Un’altra motivazione è legata al fatto che, come WordNet, si presta a supporto di meccanismismi di disambiguazione sia visionati che non.

In realtà all’interno di WordNet esiste già un’informazione di dominio, ma è limitata a pochi lemmi e non può perciò essere utilizzata come criterio di disambiguazione. Nonostante, tutt’oggi esistano algoritmi anche molto complessi basati su WordNet Domains, che ne colgono e ne sfruttano tutte le varie potenzialità, in questa nostra prima analisi, ci si limiterà al tentativo di sfruttare la capacità di WND di fornire informazioni riguardanti il dominio o i domini di pertinenza di una determinata sorgente di dati

Il nostro studio, parte da un contesto come MOMIS, all’interno del quale nessun algoritmo di selezione è ancora implementato, e dove l’utente deve manualmente selezionare il senso corretto tra la moltitudine di quelli proposti da WordNet (si pensi che per alcuni verbi esistono fino a 40 sensi possibili), WordNet Domains rappresenta un primo passo significativo di semplificazione del processo, in quanto consente di delineare l’ambito di pertinenza dei dati e conseguentemente restringe il campo dei sensi possibili da attribuire ai termini contenuti nella sorgente stessa.

5.2 Integrazione di WordNet Domains

5.2.1 Struttura di WordNet Domains

Il package di WordNet Domains può essere ottenuto liberamente, al sito http://wndomains.itc.it. La prima versione di WND uscì nel maggio del 2001, e si basava sulla gerarchia iniziale e sulla versione di WordNet 1.6. Nel 2003 uscì WND versione 1.1; nel

maggio 2003 uscì un'ulteriore release WND 1.1.1, così come quella uscita nel maggio 2004 v. 1.2. Finalmente nel gennaio del 2005 uscì la versione principale 2.0 la quale include oltre a WND 2.0 anche WN-affect 1.0. Pur mantenendosi rispetto a WordNet 1.6 questa nuova versione presentò modifiche significative alla gerarchia dei domini. Per agevolare il confronto con le gerarchie della precedente edizione il pacchetto venne arricchito delle opportune documentazioni. Subito dopo, nel febbraio 2005, uscì la versione 3.0 aggiornata rispetto alla versione di WordNet 2.0. Questa, all'interno del suo package, includeva tutto il pacchetto della versione 2.0 più la versione beta di WND 3.0 .L'ultima release corrisponde al settembre 2005, la quale apporta alcune modifiche a causa di associazioni di dominio errate nel passaggio alla nuova gerarchia.

L'informazione di dominio che associa a ciascun synset una determinata etichetta, è fornita tramite un semplice file di testo. Le informazioni sono memorizzate in base alla seguente struttura :

byte_offset-categoria_sintattica dominio_1 dominio_2…dominio_ n

dove *byte_offset-categoria_sintattica* identificano i synset così come quest'ultimi sono identificati all'interno di WordNet, mentre dominio_1 dominio_2…dominio_n indica i nomi dei domini associati al relativo synset. Infatti ad ogni synset possono essere associati uno o più domini separati all'interno del file da uno spazio.

Esempio:

00179486-n play sport

Come si nota dall'esempio, il byte offset, inoltre, è composto da esattamente 8 cifre, e nel caso in cui richieda un numero inferiore di cifre queste vengono semplicemente *shiftate* a destra e gli spazi liberi sono riempiti con degli zeri.

Le categorie sintattiche sono rappresentate sempre attraverso la stessa simbologia utilizzata in WordNet:

-n → indica un nome

-v → indica un verbo

-a → indica un aggettivo

-s → indica un aggettivo satellite.

-r → indica un avverbio

Il file risulta ordinato per categoria sintattica e per ciascuna di esse per *byte offset* crescente. All'interno del package è presente inoltre un file contenente *WordNet Affect* il quale etichetta ciascun synset attraverso un dominio "affettivo". Tale file non è stato considerato nella nostra analisi in quanto fornisce un genere di informazioni specifiche non richieste per i nostri propositi di disambiguazione.

5.2.2 La Gerarchia di Domini

Le etichette di dominio dei WordNet Domains (WND), come abbiamo già affermato nel capitolo precedente, sono estratte dalle sistema di classificazione *Dewey Decimal Classification* (DDC) . Sebbene tale WordNet Domains Hierarchy (WDH), sia stata applicata con successo nell'ambito della disambiguazione del testo, tale versione ha evidenziato alcuni problemi legati principalmente ad una mancanza di chiara semantica per alcune etichette, e al bilanciamento della copertura dei vari synset.

Una gerarchia di dominio, rappresenta una risorsa indipendente dal linguaggio, utilizzabile in diverse applicazioni. La nozione di dominio può essere paragonata a quelle di campo semantico, categoria ecc…Nel caso di WND, con il termine dominio si intende un'area di conoscenza la quale può essere riconosciuta ed identificata, in base ad un determinato criterio, come unitaria. Un dominio può essere caratterizzato dal nome di una disciplina all'interno della quale è sviluppata, da una certa area di conoscenza (es. chimica) o da un oggetto specifico dell'area di conoscenza (es. food). Sebbene gli oggetti della conoscenza, e le discipline che li studiano, siano chiaramente in relazione fra loro, tale relazione in realtà nelle applicazioni di disambiguazione, non così semplice da determinare.

Un'altro dualismo legato ai domini della conoscenza, è rappresentato dalla loro manifestazione, sia attraverso i singoli termini, che attraverso l'intero testo. Quindi la nozione di dominio può essere utilizzata, sia nello studio dei singoli termini, dove descrive l'area di conoscenza al quale appartiene il concetto, sia nello studio dei testi, dove descrive argomento principale in esso contenuto. Tuttavia, anche questi due ambiti di applicazione in realtà sono in stretta relazione fra loro. Per loro natura, i domini possono essere organizzati in gerarchie basate su una relazione di specificità. Per esempio, si può affermare che il dominio TENNIS

sia più specifico del dominio SPORT . Le gerarchie di dominio possono essere intergrate facilmente all'interno di altre risorse. WND rappresenta un esempio di tale integrazione.
Un requisito fondamentale di una gerarchia di dominio, è di possedere una semantica ed una struttura ben definita.
La prima versione della WDH era composta da 164 etichette di dominio selezionate a partire da dizionari a da alcuni campi della DCC. Le etichette di dominio erano organizzate in cinque alberi principali, ciascuno dei quali raggiungeva una profondità massima pari a quattro. La figura 5.1 mostra un frammento di uno dei principali alberi della gerarchia originale di WND.
Le etichette di dominio furono inizialmente concepite per essere orientate all'applicazione, di conseguenza sono state intergrate in WordNet con il principale proposito di associare ad ogni synset un dominio, in modo che, tale informazione, potesse essere utilizzata durante il processo di disambiguazione. Il secondo livello della gerarchia rappresenta i domini di base come ART, SPORT, RELIGION e HISTORY, mentre il terzo livello esprime un livello maggiore di specializzazione, e include domini come per esempio DRAWING, PAINTING, TENNIS ecc…
Sebbene la prima versione sia stata utilizzata in molte applicazioni con successo, Bentivogli e al. in [**63**] essa ha presentato alcuni problemi. Primo le etichette di dominio non posseggono una semantica definita; il significato delle etichette può essere suggerito dal loro nome ma non vi è alcuna precisa indicazione sul loro contenuto semantico.
Secondo, non è chiaro se i domini di base posseggano effettivamente i requisiti di copertura e bilanciamento fra i synset, necessari per essere definiti tali. Infatti, viene solo supposto che tali domini posseggano un grado di specificità tale da poter essere posti sullo stesso livello, e che nel contempo rappresentino tutta la conoscenza umana.
Tuttavia, si fa notare come queste caratteristiche non siano sempre ottenibili. Un esempio è rappresentato da il dominio VETERINARY il quale è posto allo stesso livello gerarchico di ECONOMY, ma in realtà possegga un livello di specificità differente. Inoltre, non vengono rappresentate tutte le branche della conoscenza umana (es. il dominio HOME è assente).
A tutte queste problematiche, gli ideatori di WND, hanno provato a porre soluzione con una nuova versione della gerarchia, basata completamente su DDC (edizione 21), che è stata utilizzata come punto di riferimento per definire una chiara semantica dei domini e per evitare sovrapposizioni fra gli stessi. Ricordiamo che tali assunzioni sono possibili in quanto

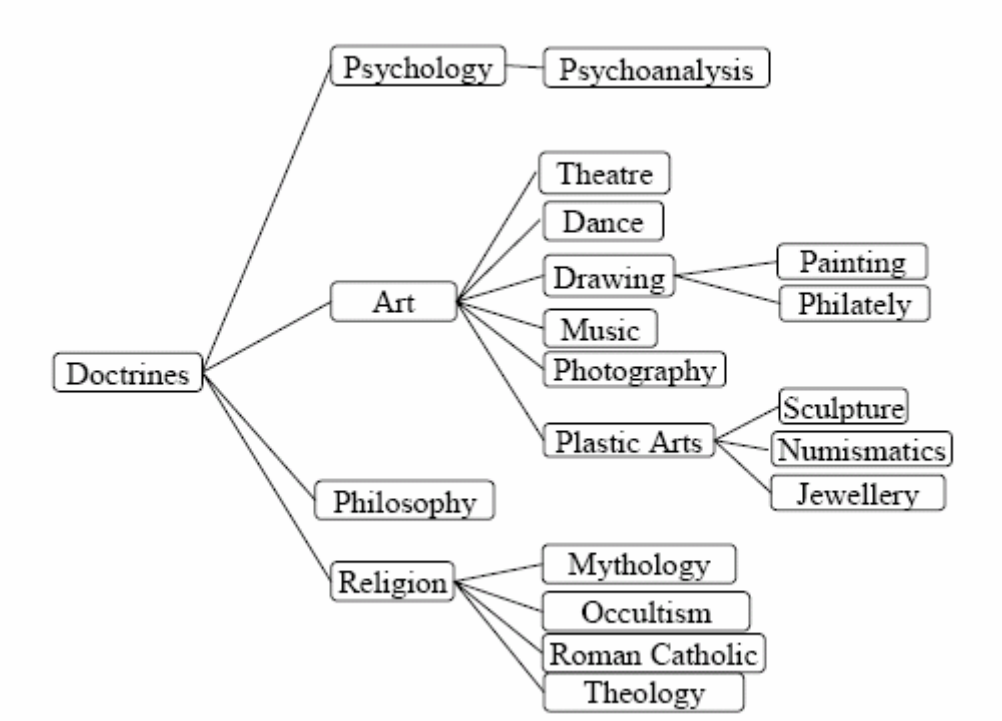


Figura 5.1- Frammento della gerarchia originale di WordNet Domains

il sistema DDC sistema, è la tassonomia più ampiamente utilizzata e riconosciuta per la classificazione di librerie, e fornisce un sistema logico per classificare ogni elemento della conoscenza, attraverso codici soggetto ben definiti e organizzati in maniera gerarchica. Ogni categoria DDC è rappresentata da un codice numerico con nell'esempio in figura 5.2. Il primo numero rappresenta la classe principale, il secondo la divisione, e l'ultimo indica la sezione.

700 *Art*
730 *Plastic Arts*
736 *Carving*
736.2 *Precious Stones*
736.23 *Diamonds*
736.25 *Sapphires*
736.4 *Wood*
738 *Ceramic Arts*
739 *Art Metalwork*
740 *Drawing*
750 *Painting*

Figura 5.2-Esempio di classificazione nella DDC

Le revisione alla prima versione aiutano a soddisfare i seguenti requisiti:

1) semantica: ogni etichetta della gerarchia deve possedere una precisa semantica in modo da poter facilmente essere disambiguata;
2) disgiunzione: le interpretazioni delle singole etichette non devono sovrapporsi;
3) copertura di base: la conoscenza umana deve essere rappresentata nella sua totalità dai domini di base;
4) bilanciamento: la maggior parte dei domini di base deve possedere un livello comparabile di granularità.

Il primo requisito viene soddisfatto associando ad ogni etichetta di dominio una o più etichette dei codici DDC come mostrato nella tabella in figura 5.3.

WDH Domains	DDC Codes
Art	[700-(790-(791.43,792,793.3), 710,720,745.5)]
Plastic arts	730
Sculpture	[731:735]
Numismatics	737
Jewellery	739.27
Drawing	[740-745.5]
Painting	750
Graphic arts	760
Philately	769.56
Photography	770
Music	780
Cinema	791.43
Theatre	[792-792.8]
Dance	[792.8,793.3]

Figura 5.3-Frammento della nuova WDH con i rispettivi codici DDC

Inoltre per rafforzare il parallelismo gerarchico fra la nuova WDH e la DDC, molti domini sono stati ricollocati all'interno della gerarchia in modo tale da rispettare le relazioni gerarchiche della DDC.

Il secondo requisito viene soddisfatto semplicemente assicurandosi che nessun codice della DDC sia associato a più di un dominio della WDH.
Il terzo e il quarto requisito sono sempre assicurati dall'utilizzo della DDC.
In appendice sono riportate in dettaglio le modifiche alla WDH originale.

5.2.3 Modifiche al DataBase di MOMIS

WordNet Domains è stato intergrato all'interno del progetto MOMIS. Il processo d'integrazione, ha richiesto una serie di modifiche al database "*momiswn"* di MOMIS descritto nel capitolo 2.
Riportiamo brevemente la struttura del database originale, il quale contiene una serie di tabelle che consentono di rappresentare tramite un database relazionale, il contenuto informativo di WordNer. Tali tabelle sono:

- ***wn_synset***: contiene essenzialmente le glosse dei vari synset; tali glosse sono associate ai rispettivi synset attraverso i campi *byte_offset* e *syntactic_category* che consentono di identificare un determinato synset in base alla notazione utilizzata in WordNet.
- ***wn_relationship_type****:* contiene i tipi di relazioni previsti nella versione di iniziale (1.6) di WN.
- ***wn_relationship_type_new:*** contiene li nuovi tipi di relazioni inserite all'interno della versione 2.0 di WN comprensive di quelle della versione precedente.
- ***wn_relationship****:* contiene tutte le relazioni fra i synset di WordNet.
- ***wn_lemma_synset:*** associa ciscun lemma, ai relativi possibili synset.
- ***wn_lemma:*** contiene tutti i lemma presenti in WordNet e per ciascuno indica la categoria sintattica di appartenenza.
- ***wn_extender:*** contiene i codici assegnati a ciascuna estensione del database.

Prima di iniziare le nostre modifiche al database, per consentire a chiunque di riconoscere velocemente (e decidere se utilizzare o meno) le nostre modifiche, si è inserito all'interno della tabella *wn_extender*, una nuova estensione nominata "*WordNet Domains*" a cui è stato associato il codice identificativo "3". Di conseguenza, per riconoscere velocemente all'interno delle tabelle le modifiche da noi apportate, basterà controllare il valore del campo *"wn_extended_id"*.

Per introdurre WND, sono state apportate una serie di modifiche, senza tuttavia andare a compromettere l'integrità delle informazioni già contenute nel database.

L'idea di base è stata quella di fornire l'informazione di dominio attraverso una associazione tra il synset ed i domini ad essi associati.

Il processo d'integrazione può essere suddiviso in tre parti principali:

1) modifiche alle tabelle esistenti;
2) creazione di nuove tabelle;
3) inserimento dei dati.

Durante la prima fase abbiamo innanzitutto verificato se effettivamente tutte i lemmi corrispondenti ai nomi dei domini di WND fossero presenti all'interno di WordNet. Come sappiamo, infatti, WordNet non è completo dal punto di vista lessicale, esso non include svariati termini di uso comune, inoltre, presenta lacune per quanto concerne i termini composti. Molti nomi di domini, sono in forma plurale, e, di conseguenza, non sono presenti in WordNet pur esistendo la loro forma singolare, in quest'ultimo.Per continuità e coerenza con la risorsa utilizzata si è scelto di considerare assenti, anche questi termini, invece di forzare, tali domini, alla forma singolare. Dall'analisi delle etichette di WND, si evince che ben quindici etichette di dominio non possiedono il corrispondente lemma in WordNet. Alcuni di questi sono nomi composti, altri plurali, altri ancora (es. *paranormal),* non sono semplicemente presenti. Tali lemmi sono:

animals, artisanship, atomic_physic, betting, body_care, book_keeping, buildings, electrotecnology, graphic_art, nautical, paranormal, plants, psycological_feature, pure_science, vehicles.

Per poter associare a ciascun synset, i propri domini di appartenenza, di conseguenza si è deciso di inserire manualmente, ciascun lemma mancante, stando attenti di rispettare tutti i vincoli del database relazionale. Si è osservato inoltre che tutti i lemmi di dominio appartengono alla categoria sintattica dei nomi.

Successivamente, si sono inserite all'interno della tabella *wn_relationship_new* due nuove relazioni: "***Member of this domain", "Domain of synset".***

La prima relazione sussiste fra un synset e il suo dominio di appartenenza, mentre la seconda rappresenta la relazione inversa che lega ciascun dominio ai propri synset.

Come vedremo, in realtà, durante la nostra sperimentazione utilizzeremo solo la relazione diretta ovvero *Member of this domain*, ma per futuri propositi potrebbe essere richiesta anche la relazione inversa.

Il secondo passo, è stata la creazione della tabella ***wn_domain*** la quale contiene al suo interno tutti i lemmi dei domini ed i synset identificativi di quest'ultimi.
Tale creazione ha richiesto il processo di disambiguazione dei lemmi dei domini. Ciò è stato realizzato manualmente, ed in base alla documentazione presente all'interno dei package di WND riguardo alla semantica dei domini.
L'associazione tra il nome del dominio ed il synset corrispondente, è necessaria dal momento in cui, ciò che andiamo a realizzare, è, essenzialmente, un'associazione tra due synset. In realtà WND non fa questo, ma bensì associa semplicemente un'etichetta di dominio ai vari sensi, non considerando il dominio come lemma. Tuttavia, il nostro approccio consente di garantire continuità rispetto alla struttura di WordNet. WordNet non viene modificato, si aggiungono semplicemente nuove relazioni.
Il terzo passo consiste nell'integrazione vera e propria di WND. Una volta forniti tutti gli strumenti necessari per inserire l'informazione di dominio come una semplice relazione, si possono finalmente inserire le relazioni di dominio.
Essendo semplicemente delle relazioni, potrebbero essere inserite all'interno della tabella *wn_relationship*, tuttavia per distinguere le relazioni di WND dalle altre proprie di WordNet, si è preferito creare una nuova tabella formata dagli stessi campi, e dagli stessi vincoli d'integrità *di wn_relationship*. Tale tabella è stata indicata con ***wn_relationship_domain***.
Per inserire i dati in tabella, è stato realizzato un semplice programma in Java che ha interagito con il database, attraverso la tecnologia JDBC.
In figura 5.4 vengono mostrati alcuni record della tabella risultante *wn_relationship_wnd*.
Il campo *wn_relationship_id*, identifica semplicemente la relazione, *wn_source_synset_id* e *wn_target_synset_id* rappresentano nel caso di relazione "*Member of domain*" *(wn_relationship_type_id* uguale a 26) rispettivamente, il synset associato ad un termine e il synset associato ad un dominio. Nel caso di relazione inversa, si inverto anche i ruoli dei due synset. *wn_source_lemma_number e wn_target_lemma_number*, rappresentano rispettivamente i lemmi associati ai precedenti al synset souce e al synset target. Per

associazione di dominio ad un synset, vengono, quindi, create tante relazioni quanti sono i lemmi associati al synset stesso.

Infine, *wn_extender_id* valendo sempre "3", indica che tutte le relazioni rappresentate in figura fanno parte dell'estensione di WND.

	WN_RELATIONSHIP_ID	WN_SOURCE_SYNSET_ID	WN_TARGET_SYNSET_ID	WN_SOURCE_LEMMA_NUMBER	WN_TARGET_LEMMA_NUMBER	WN_RELATIONSHIP_TYPE_ID	WN_EXTENDER_ID
1	411316	32312	45970	1	3	25	3
2	411315	45970	32312	3	1	26	3
3	411314	32312	45970	1	2	25	3
4	411313	45970	32312	2	1	26	3
5	411312	32312	45970	1	1	25	3
6	411311	45970	32312	1	1	26	3
7	411310	32312	45969	1	2	25	3
8	411309	45969	32312	2	1	26	3
9	411308	32312	45969	1	1	25	3
10	411307	45969	32312	1	1	26	3
11	411306	110009	45968	4	3	25	3
12	411305	45968	110009	3	4	26	3
13	411304	110009	45968	4	2	25	3
14	411303	45968	110009	2	4	26	3
15	411302	110009	45968	4	1	25	3
16	411301	45968	110009	1	4	26	3
17	411300	32312	45968	1	3	25	3
18	411299	45968	32312	3	1	26	3
19	411298	32312	45968	1	2	25	3
20	411297	45968	32312	2	1	26	3
21	411296	32312	45968	1	1	25	3
22	411295	45968	32312	1	1	26	3
23	411294	32312	45967	1	1	25	3
24	411293	45967	32312	1	1	26	3
25	411292	110009	45966	4	1	25	3

Figura 5.4- Frammento dei record della tabella contenente l'informazione di dominio

5.3 Test sull'applicabilità di WordNet Domains

Questo paragrafo ha lo scopo di verificare l'effettiva applicabilità del sistema WordNet Domains all'interno del processo di disambiguazione dei termini.

L'obiettivo è quello di individuare un meccanismo di disambiguazione semi-automatico e completamente non supervisionato. Sono stati eseguiti una serie di test sulla sorgente di relazioni di dominio rappresentata dalla tabella *wn_relationship_wnd,* integrata nel database "momiswn" a partire dalla sorgente originale di WordNet-Domain(WND).

Tali test si basano sull'analisi e il confronto dei sensi e delle annotazioni ricavate attraverso l'utilizzo delle relazioni di WND rispetto all'annotazione manuale.

Si suppone, a tale scopo, che le annotazioni manuali siano da considerarsi corrette.

In particolare, si è deciso di effettuare tali test su due tipologie di dati differenti:

1. Dati provenienti dal progetto WISDOM: tali dati si compongono essenzialmente di tre cluster, ognuno contenente lemmi ricavati dalle pagine web di attività commerciali, come hotel, ristoranti, campeggi ecc…La tipologia di lemmi estratti da tali cluster risulta perciò molto simile, ciò che li differenza principalmente è la dimensione dei dati in essi contenuti, via via crescente dal cluster1 al cluster3.

2. Dati provenienti dalle directory dei motori di ricerca GOOGLE e YAHOO: in questo caso si considera un solo unico insieme di lemmi estratti da tali sorgenti. Ciò che differenzia principalmente tale tipologia di dati dai precedenti, è la maggior varietà dei lemmi e le dimensioni nettamente superiori (si passa dai 71 lemmi estratti dal cluster_3 a ben 776 lemmi ricavati da queste directory.

Tali tipologie di sorgenti di dati, si differenziano dai corpus di testi sui quali sono stati testati nella maggiorparte dei casi, i vari algoritmi analizzati nel capitolo 3. Tuttavia, queste rappresentano tipologie di sorgenti su cui MOMIS si trova ad operare nella maggiorparte dei casi.

5.3.1 Tipologie di test effettuati

Lo scopo principale dei test effettuati, è quello di verificare quanti e quali lemmi delle sorgenti di dati sia possibile annotare in maniera automatica, utilizzando come unico criterio di scelta dei synset, i domini ad essi associati da WND.
Il criterio iniziale con cui si è deciso di annotare i lemmi tramite WND, è quello di selezionare tra i vari domini associati ai vari synset di ciascun lemma, il primo dominio, se esiste, tra i primi *n* domini con frequenza maggiore. Quest'ultimi sono il risultato di un semplice conteggio sulle loro occorrenze all'interno della tabella contenente, per ogni lemma, i vari synset possibili e i domini ad essi associati (tramite WND).
Si è deciso di effettuare le seguenti tipologie di test:

- **Test 1**: analizzare e determinare l'eventuale numero *n* dei domini a frequenza maggiore da utilizzare nel processo di disambiguazione dei termini. Per ogni valore di n, determinare il numero di annotazioni estratte, e il numero di annotazioni corrette;

- **Test 2**: Studio dell'influenza e dell'importanza dei termini monosemici all'interno del processo di disambiguazione basato su WND. Tale studio si basa sulla considerazione che, tali termini, essendo associati ad un solo synset, ci potrebbero fornire un'indicazione sui domini presenti all'interno della sorgente dati.

- **Test 3**: Analisi e studio delle annotazioni manuali (supposte corrette), allo scopo di individuare la presenza di una polarizzazione di dominio, ovvero se esiste effettivamente all'interno dei dati uno o più domini polarizzati.

- **Test 4:** Si ripeterà il test 1, con lo scopo di annotare questa volta solo i termini non monosemici;

- **Test 5:** si analizzerà la sovrapposizione dei primi tre domini ad occorrenza maggiore, all'interno dei singoli lemmi, cercando di individuare eventuali annotazioni date da più domini contemporaneamente.

Durante lo svolgimento dei test, si eseguiranno i calcoli sui dati secondo due diversi approcci:

- considerando anche i lemmi la cui annotazione manuale li porta ad essere etichettati come factotum; i risultati ottenuti utilizzando tale approccio verranno di seguito indicati con la sigla CF (Con Factotum);
- viceversa considerando solo i lemmi la cui annotazione manuale gli consente (non essendo factotum) di essere annotati tramite il contenuto informativo portato dai domini; in tal caso si indicheranno i risultati con la sigla SF (Senza Factotum).

Tale doppio approccio, è motivato dal fatto che i termini per i quali l'annotazione manuale li fa corrispondere al dominio factotum, non potranno essere mai annotati tramite questo metodo, non appartenendo a nessun specifico dominio. Tuttavia nelle applicazione reali, le annotazioni corrette non si conoscono e di conseguenza, non è possibile escludere a priopri i lemmi associati a factotum. Per questi motivi durante la nostra analisi, sarà interessante valutare le prestazioni ottenute tramite WND in entrambe le situazioni.

La valutazione dei risultati sarà effettuata in termini di precision e recall:

$$\text{Recall} = \frac{\text{number of correct annotations}}{\text{total number of annotations}}$$

$$\text{Precision} = \frac{\text{number of correct annotations retrieved}}{\text{total number of annotations retrieved}}$$

I seguenti paragrafi sono così organizzati: il primo si occuperà di descrivere in maniera più dettagliata l'esito dei singoli test applicati a ciascuna sorgente; nel secondo si analizzeranno i dati in maniera sommaria dal punto di vista dei risultati, e in maniera specifica dal punto di vista delle prestazioni nell'uso di WND.

5.3.2 Risultati

Dati provenienti dal progetto WISDOM

CLUSTER 1

Il cluster_1 è composto da 25 lemmi di cui 6 risultano annotati manualmente con un synset associato al dominio factotum. Di conseguenza nell'analisi che esclude i factotum si considereranno solo i rimanenti 19 lemmi.

Test 1

La tabella successiva mostra per ogni dominio individuato le relative occorrenze:

Dominio	Occorrenze
Factotum	**20**
Town_planning	**7**
Computer_science	**6**
Geography	**5**
Administration	**4**

Linguistics	**4**
Sociology	**3**
Tourism	**3**
Post	**3**
Telephony	**2**
Art	**2**
Buidings	**2**
Quality	**2**
Person	**2**
Publishing	**1**
Free_time	**1**
Time_period	**1**
Economy	**1**
Military	**1**
Commerce	**1**
Nautical	**1**
Number	**1**
Money	**1**

Tabella 5.1 Occorrenze dei domini all'interno del cluste-1 delle sorgenti di dati di WISDOM

Numero di domini selezionati	Numero totale di termini valutati	Numero di termini disambiguati	Numero di disambiguazioni corrette	Precision	Recall
1	**25**	**7**	**5**	**0,2**	**0,71**
2	**25**	**11**	**9**	**0,36**	**0,82**
3	**25**	**13**	**10**	**0,4**	**0,77**
4	**25**	**13**	**10**	**0,4**	**0,77**
5	**25**	**15**	**12**	**0,48**	**0,8**

Tabella 5.2- Risultati CLUSTER-1 nel caso CF

Numero di domini selezionati	Numero totale di termini valutati	Numero di termini disambiguati	Numero di disambiguazioni corrette	Precision	Recall
1	19	5	5	0,26	1,00
2	19	9	9	0,47	1,00
3	19	10	10	0,53	1,00
4	19	10	10	0,53	1,00
5	19	12	12	0,63	1,00

Tabella 5.3- Risultati CLUSTER-1 nel caso SF

Da una prima analisi dei risultati ottenuti si evidenzia chiaramente come la disambiguazione attraverso i domini, in generale, ottenga risultati migliori escludendo dai dati i lemmi etichettati come factotum. Tale risultato era ovviamente più che attendibile viste le precedenti osservazioni sui lemmi etichettati con factotum.Tuttavia non si evidenzia una vera e propria polarizzazione di un singolo dominio: il dominio con più occorrenze ricopre infatti solo il 26% dei lemmi SF. Considerando i primi due domini i risultati migliorano considerevolmente arrivando al 48% SF. Tali risultati devono però tener in considerazione l'influenza notevole dei lemmi monosemici. Si noti inoltre come in questo caso il 100% delle annotazioni effettuate SF risultino corrette. In ultimo, il quarto dominio lasca inalterati i valori in tabella, il che sta a significare che nessun lemma è stato disambiguato grazie a tale dominio.

Test2

Il cluster_1 si compone di 14 lemmi monosemici su 25 nel caso CF (includendo il dominio factotum) pari al 56% del totale dei lemmi, e di 12 lemmi monosemici su 19 nel caso SF (escludendo il dominio factotum) pari al 63% del totale dei lemmi.
Consideriamo le occorrenze dei domini associati ai lemmi monosemici CF (includendo il dominio factotum):

Dominio	Occorrenze
Town_planning	3
Computer_science	3
Tourism	2
Factotum	2
Buidings	2
Art	2
Buidings	2

Telephony	**2**
Post	**1**
Commerce	**1**
Economy	**1**
Publishing	**1**
Money	**1**

Tabella 5.4–Occorrenze dei domini dei lemmi monosemici del CLUSTER-1

Dai risultati ottenuti si nota come in media più della metà dei lemmi sia monosemico, il che implica che tali lemmi sono praticamente già annotati.
Inoltre si evidenzia che, in maniera concorde con quanto evidenziato durante il test 1 , esiste una parziale polarizzazione sui due domini *town_planning* e *computer_science* .

Test 2

Consideriamo le annotazioni manuali:

Dominio	Occorrenze
Factotum	**6**
town_planning	**5**
computer_science	**4**
geography	**3**
Tourism	**3**
telephony	**2**
buildings	**2**
linguistics	**2**
administration	**2**
Publishing	**1**
Economy	**1**
commerce	**1**
Money	**1**
Post	**1**
Free_time	**1**

Tabella 5.5- Occorrenze dei domini nel CLUSTER-1 in base alle annotazioni manuali

Dalla tabella si nota una coincidenza con i domini a maggiore occorrenza ottenuti nel test 1. Ciò significa che la parziale polarizzazione evidenziata nel test 1 sui domini t*own_planning* e *computer_science* si ritrova all'interno delle annotazioni corrette.

Test 4

Numero di domini selezionati	Numero totale di termini valutati	Numero di termini disambiguati	Numero di disambiguazioni corrette	Precision	Recall
1	11	4	2	0,18	0,50
2	11	4	2	0,18	0,50
3	11	6	3	0,27	0,50
4	11	6	3	0,27	0,5
5	11	8	5	0,45	0,63

Tabella 5.6 –Risultati per i soli termini polisemici del CLUSTER-1 CF

Numero di domini selezionati	Numero totale di termini valutati	Numero di termini disambiguati	Numero di disambiguazioni corrette	Precision	Recall
1	7	2	2	0,29	1,00
2	7	2	2	0,29	1,00
3	7	3	3	0,43	1,00
4	7	3	3	0,53	1,00
5	7	5	5	0,63	1,00

Tabella 5.7-Risultati per i soli termini polisemici del CLUSTER-1 SF

Test 5

Di seguito si riporteranno i risultati ottenuti considerando tutte le possibili combinazioni tra i primi tre domini, e specificando se i domini associati ad un determinato lemma indicano come annotazione lo stesso synset o synset diversi. Appare ragionevole pensare che, nel caso in cui più domini indichino il medesimo synset come annotazione per un determinato lemma, tale annotazione sarà ancora più probabilmente quella più corretta.

Consideriamo tutte le possibili combinazioni:

town_planning/computer_science: 2 annotazioni con synset non coincidenti;

town_planning/geography: 2 annotazioni con synset coincidenti;
computer_science/geography: nessuna annotazione;
town_planning/computer_science/geography: nessuna annotazione.

Osservazioni:

Per quanto concerne il cluster_1 si evidenzia principalmente una 'parziale' polarizzazione dei lemmi verso due domini in particolare, i quali ricoprono circa il 50% delle annotazioni. Visto il peso considerevole rappresentato dai domini monosemici, bisognerebbe verificare quanti dei lemmi annotati attraverso i primi due domini, rientrano in quest'ultima categoria.
Inoltre tali risultati, seppur discreti, derivano da una sorgente di lemmi limitata, ulteriori e più attendibili considerazioni , potranno essere effettuate per i test successivi i quali vengono sottoposti a quantitativi di dati più numerosi.

CLUSTER 2

Il cluster_2 si compone di 58 lemmi di cui solo 38 risultano essere annotabili tramite i domini, ovvero i rimanenti 20 risultano etichettati con factotum in base all'annotazione manuale.

Test 1

La tabella successiva mostra per ogni dominio individuato le relative occorrenze:

Dominio	Occorrenze
Factotum	**72**
Computer_science	**11**
Town_planning	**9**
Sociology	**8**
Person	**8**
Administration	**8**
Linguistics	**8**
Geography	**7**
Economy	**6**
Military	**4**
Mathematics	**3**
Money	**3**

Post	**3**
Publishing	**2**
Music	**2**
Quality	**2**
Art	**2**
Number	**2**
Telephony	**2**
Buildings	**2**
Tourism	**2**
Commerce	**2**
Mechanics	**1**
Aviation	**1**
Geology	**1**
Anatomy	**1**
Industry	**1**

Tabella 5.8- Occorrenze dei domini nel CLUSTER-2

Numero di domini selezionati	Numero totale di termini valutati	Numero di termini disambiguati	Numero di disambiguazioni corrette	Precision	Recall
1	58	11	9	0,16	0,82
2	58	18	16	0,28	0,89
3	58	26	18	0,31	0,69
4	58	27	19	0,33	0,70
5	58	27	19	0,33	0,70
6	58	28	19	0,33	0,68

Tabella 5.9- Risultati CLUSTER-2 nel caso CF

Numero di domini selezionati	Numero totale di termini valutati	Numero di termini disambiguati	Numero di disambiguazioni corrette	Precision	Recall
1	38	9	9	0,24	1,00
2	38	16	16	0,42	1,00
3	38	24	18	0,47	0,75
4	38	25	19	0,50	0,77
5	38	25	19	0,50	0,76
6	38	25	19	0,50	0,76

Tabella 5.10- Risultati CLUSTER-2 nel caso SF

Dai dati ottenuti si evidenzia una parziale polarizzazione dei lemmi su due domini. Disambiguando con n=2 si riescono ad annotare infatti fino al 42% dei lemmi SF.
In media circa l'80% delle annotazioni SF risultano essere corrette, come si nota dai valori di Recall riportati in tabella.
Si noti inoltre la coincidenza dei primi due domini individuati nei test del cluster_1.
Ciò probabilmente è da attribuirsi ad una elevata similitudine della tipologia di dati utilizzati.

Test2

I lemmi monosemici contenuti dal cluster_2 risultano essere in numero 26 su 58 CF, e 16 su 38 SF. In entrambi i casi quindi rappresentano circa la metà dei lemmi considerati.

Dominio	Occorrenze
Factotum	**10**
Computer_science	**8**
Town_planning	**2**
Publishing	**2**
Tourism	**2**
Buidings	**2**
Telephony	**2**
Post	**1**

Tabella 5.11- Occorrenze dei domini per il lemmi monosemici nel CLUSTER-2

Dalla tabella si evidenzia come i termini monosemici diano un netto contributo al dominio *computer_science*. In questo caso la polarizzazione di dominio è concentrata sul solo dominio *computer_science.*

Test 3

Consideriamo le annotazioni manuali:

Dominio	Occorrenze
Factotum	**20**
Computer_science	**9**

Town_planning	**7**
Linguistics	**5**
Geography	**5**
Administration	**4**
Economy	**2**
Buildings	**2**
Number	**2**
Sociology	**2**
Publishing	**2**
Tourism	**2**
Telephony	**2**
Post	**1**
Commerce	**1**
Person	**1**
Mathematics	**1**

Tabella 5.12- Occorrenze dei domini nelle annotazioni manuali del CLUSTER-2

Anche in questo caso, come si era verificato durante l'analisi del cluster_1, i primi domini coincidono con quelli ottenuti per il test1.

Anche qui si evidenzia una parziale polarizzazione sui domini *computer_science* e *town_planning.*

Test 4

Numero di domini selezionati	Numero totale di termini valutati	Numero di termini disambiguati	Numero di disambiguazioni corrette	Precision	Recall
1	**32**	**2**	**0**	**0,00**	**0,00**
2	**32**	**6**	**4**	**0,13**	**0,67**
3	**32**	**14**	**6**	**0,19**	**0,43**
4	**32**	**15**	**7**	**0,22**	**0,47**
5	**32**	**16**	**7**	**0,22**	**0,44**

Tabella 5.13 –Risultati per i soli lemmi polisemici del CLUSTER-2 CF

Numero di domini selezionati	Numero totale di termini valutati	Numero di termini disambiguati	Numero di disambiguazioni corrette	Precision	Recall
1	22	0	0	0,00	0,00
2	22	4	4	0,18	1,00
3	22	12	6	0,27	0,50
4	22	13	7	0,32	0,54
5	22	13	7	0,32	0,54

Tabella 5.14 –Risultati per i soli lemmi polisemici del CLUSTER-2 SF

Test 5

Consideriamo tutte le possibili combinazioni:

computer_science/town_planning: 2 annotazioni con synset non coincidenti;

computer_science/sociology: nessuna annotazione;

town_planning/sociology: nessuna annotazione;

computer_science/town_planning/sociology: nessuna annotazione.

CLUSTER 3

Il cluster_3 è composto da 71 lemmi di cui 49 annotabili tramite i domini, in quanto i restanti 22 risultano essere etichettati con factotum in base all'annotazione manuale.

Test 1

La tabella successiva mostra per ogni dominio individuato le relative occorrenze:

Dominio	Occorrenze
Factotum	87
Commerce	19
Buildings	13
Economy	11
Computer_science	11
Mathematics	10
Town_planning	9
Person	9

Sociology	**6**
Geography	**6**
Baseball	**6**
Linguistics	**5**
Number	**5**
Music	**5**
Administration	**5**
Art	**4**
Post	**4**
Anatomy	**3**
Money	**3**
Tourism	**3**
Law	**3**
Literature	**2**
telecomunication	**2**
Medicine	**2**
Health	**2**
Publishing	**2**
physiology	**2**
Quality	**2**
Enterprise	**2**
Card	**2**
psychology	**1**
School	**1**
Racing	**1**
Telephony	**1**
Psychological_feature	**1**
Physics	**1**
Free_time	**1**
Military	**1**

Tabella 5.15- Occorrenze dei domini nel CLUSTER-3

Numero di domini selezionati	Numero totale di termini valutati	Numero di termini disambiguati	Numero di disambiguazioni corrette	Precision	Recall
1	71	7	6	0,08	0,86
2	71	16	14	0,20	0,88
3	71	22	17	0,24	0,77
4	71	33	26	0,37	0,79
5	71	38	30	0,42	0,79
6	71	0	0	0,00	#DIV/0!

Tabella 5.16- Risultati CLUSTER-3 nel caso con factotum

Numero di domini selezionati	Numero totale di termini valutati	Numero di termini disambiguati	Numero di disambiguazioni corrette	Precision	Recall
1	49	6	6	0,12	1,00
2	49	15	14	0,29	0,93
3	49	19	17	0,35	0,89
4	49	28	26	0,53	0,93
5	49	32	30	0,61	0,94
6	49	0	0	0,00	#DIV/0!

Tabella 5.17- Risultati CLUSTER-1 nel caso con factotum

Dai risultati ottenuti si verifica che l'ipotetica polarizzazione evidenziata dalla tabella precedente sul dominio *commerce* in realtà non è verificata. Quest'ultimo infatti ci consente di annotare solo il 12% circa dei lemmi SF. Per ottenere risultati discreti di annotazione bisogna considerare almeno i primi 4 domini. Dal confronto con i risultati dei predenti cluster si evidenzia come al crescere del numero dei lemmi cresca anche il numero di domini da considerare. Sembrerebbe quindi esistere all'interno dei cluster una polarizzazione legata ad un numero di domini proporzionale alle dimensioni dei dati. Si noti infatti come il numero dei domini presenti passi da 22 del cluster_1 a 27 del cluster_2, fino a 37 nel caso di quest'ultimo cluster. Aumentando la dispersione dei lemmi all'interno dei domini aumenta di conseguenza anche il numero dei domini da considerare per evidenziare un minimo di polarizzazione.

Si nota inoltre come, in generale quasi tutte le annotazioni SF risultino corrette: in media circa il 94% delle annotazioni SF risultano corrette.

Una nuova problematica che si è evidenziata durante questo test, è la possibilità che ad un lemma siano associati più synset, ciascuno associato al medesimo dominio. Ciò

probabilmente sta a significare, come già ipotizzato in precedenza, la possibilità di associare più synset, e quindi più annotazioni, ad un singolo lemma.

Test2

Consideriamo ora i lemmi monosemici:

Dominio	Occorrenze
Factotum	**13**
Computer_science	**9**
Town_planning	**4**
Buildings	**2**
Tourism	**2**
Commerce	**1**
Publishing	**1**
Telephony	**1**
Post	**1**
Administration	**1**

Tabella 5.18- Occorrenze dei domini nei lemmi monosemici del CLUSTER-3

RI lemmi monosemici risultano essere in numero di 28 su 71 CF, e 15 su 49 SF.
Rispetto ai precedenti cluster, in questo caso i termini monosemici risultano in numero minore rispetto al totale, ma comunque presenti in quantità significativa. In questo caso i domini con più occorrenze non rispecchiano quanto ottenuto nel test1.
In questo caso, quindi, l'influenza sui risultati del test1 sarà minore rispetto a quella avuta nei cluster precedenti.

Test 3

Consideriamo i domini associati alle annotazioni manuali:

Dominio	Occorrenze
Factotum	**22**
Computer_science	**9**
Buildings	**9**
Town_planning	**7**

Commerce	**6**
Geography	**5**
Economy	**5**
Mathematics	**4**
Administration	**3**
Linguistics	**3**
Tourism	**3**
Baseball	**2**
Post	**2**
Law	**2**
Publishing	**1**
Telephony	**1**
Medicine	**1**
Enterprise	**1**

Tabella 5.19- Occorrenze dei domini nelle annotazioni manuali del CLUSTER-3

Dai dati riportati in tabella, si evidenzia come i domini con occorrenza maggiore non rispecchiano quelli ottenuti durante il test1. In particolare si noti come il dominio *commerce*, nel test 1 al primo posto, risulti essere solo la quarta scelta tra i domini in questo caso. Questo forse giustifica il limitato numero di annotazioni ottenute attraverso quest'ultimo dominio durante il test1.

Test 4

Numero di domini selezionati	Numero totale di termini valutati	Numero di termini disambiguati	Numero di disambiguazioni corrette	Precision	Recall
1	**43**	**6**	**5**	**0,12**	**0,83**
2	**43**	**13**	**11**	**0,26**	**0,85**
3	**43**	**19**	**14**	**0,33**	**0,74**
4	**43**	**21**	**14**	**0,33**	**0,67**
5	**43**	**26**	**18**	**0,42**	**0,69**

Tabella 5.20 –Risultati per i soli termini polisemici del CLUSTER-3 CF

Numero di domini selezionati	Numero totale di termini valutati	Numero di termini disambiguati	Numero di disambiguazioni corrette	Precision	Recall
1	34	5	5	0,15	1,00
2	34	12	11	0,32	0,92
3	34	16	14	0,41	0,88
4	34	18	14	0,41	0,78
5	34	22	18	0,53	0,82

Tabella 5.21-Risultati per i soli termini polisemici del CLUSTER-3 SF

Test 5

Consideriamo tutte le possibili combinazioni:

commerce/buidings: 1 annotazione con synset coincidente;

commerce/economy: 2 annotazioni con synset coincidenti;

buildings/economy: nessuna anotazione;

commerce/buidings/economy: nessuna annotazione.

Osservazioni:

Per quanto concerne i dati del progetto WISDOM, contenuti all'interno dei tre cluster analizzati,

Dati provenienti dalle directory di YAHOO e GLOOGLE

I cluster_yahoo_google contiene 776 lemmi annotati di cui 511 annotabili attraverso i domini mentre i rimanenti 265 sono etichettati come factotum in base all'annotazione manuale.

Test 1

Consideriamo le occorrenze dei domini tra i lemmi (per brevità si riporteranno solo i domini con occorrenza maggiore di 20)

Dominio	Occorrenze
Factotum	975
Time_period	140
Politics	110
Person	92
Sociology	90
Religion	83
Publishing	78
Administration	71
Economy	70
Law	69
Pedagogy	65
Literature	51
History	49
Biology	48
Anatomy	44
Enterprise	41
Computer_science	38
Military	33
Buidings	29
Chemistry	29
Commerce	29
Anthropology	29
School	29
Metrology	29
Sport	26
Telecomunication	25

Tourism	**25**
Physiology	**25**
Linguistics	**24**
Music	**24**
Geography	**23**
Animals	**22**
Psychological_features	**22**
Industry	**21**

Tabella 5.22- Occorrenze dei domini nelle directory di YAHOO_GOOGLE

Numero di domini selezionati	Numero totale di termini valutati	Numero di termini disambiguati	Numero di disambiguazioni corrette	Precision	Recall
1	776	64	40	0,05	0,63
2	776	153	70	0,09	0,46
3	776	208	102	0,13	0,49
4	776	262	139	0,18	0,53
5	776	305	164	0,21	0,54
6	776	354	180	0,23	0,51
7	776	367	183	0,24	0,50
8	776	396	197	0,25	0,50
9	776	415	208	0,27	0,50
10	776	424	212	0,27	0,50

Tabella 5.23- Risultati nelle directory di YAHOO_GOOGLE caso CF

Numero di domini selezionati	Numero totale di termini valutati	Numero di termini disambiguati	Numero di disambiguazioni corrette	Precision	Recall
1	511	49	40	0,08	0,82
2	511	109	70	0,14	0,64
3	511	152	102	0,20	0,67
4	511	187	139	0,27	0,73
5	511	225	164	0,32	0,73
6	511	267	180	0,35	0,67
7	511	280	183	0,36	0,65
8	511	307	197	0,39	0,64
9	511	323	208	0,41	0,64
10	511	328	212	0,41	0,65

Tabella 5.24- Risultati nelle directory di YAHOO_GOOGLE caso SF

Analisi dei risultati:

Dai risultati ottenuti si evidenzia come per ottenere il 41% delle annotazioni corrette bisogna considerare ben i primi 10 domini. Questo concorda con quanto supposto in precedenza: all'aumentare dei lemmi aumenta la dispersione di quest'ultimi su più domini, di conseguenza per ottenere un numero significativo di annotazioni bisogna considerare più domini.In questo si ottengono ben 116 domini differenti, quindi per ottenere circa il 50% delle annotazioni corrette si dovrà utilizzare circa il 12% dei domini ottenuti. Per il cluster_3 si sono considerati i primi 4 domini per ottenere un'annotazione corretta del 53% dei lemmi pari circa al 13% dei domini, per il cluster_2 sempre i primi 4 domini pari al 14% (si ricorda che le dimensioni del cluster_2 e del cluster_3 sono molto simili, rispettivamente 58 CF e 38 SF per il cluster_2 e 71 CF e 49 SF per il cluster_3), mentre per il cluster_1 circa il 53% delle annotazioni corrette è stato ottenuto considerando i primi 3 domini pari 13% del totale dei domini ottenuti.
Ciò che di diverso, rispetto ai casi precedenti, accade in questo cluster è il fatto che solo in media il 65% delle annotazioni ottenute risulta coincidente con l'annotazione manuale, come si evince dai valori di recall riportati in tabella. Sarebbe interessante in questo caso andare a verificare attraverso il controllo delle glosse, se effettivamente questi synset sono completamente sbagliati o se viceversa possono essere comunque utilizzati per annotare il lemma.

Test 2

Consideriamo i lemmi monosemici:

Dominio	Occorrenze
Factotum	**166**
Sociology	**88**
Politics	**75**
Economy	**64**
Person	**46**
Religion	**40**
Administration	**36**
Publishing	**30**

Tabella 5.25- Occorrenze dei domini nei lemmi monosemici delle directory di YAHOO_GOOGLE

In totale i lemmi monosemici risultano essere 258, pari a circa il 33% del totale dei lemmi CF. Se non consideriamo i factotum risultano essere 201 pari a circa il 40% del totale.
Dalla tabella si evidenzia come i domini ad occorrenza maggiore a parte '*time_period*' siano più o meno gli stessi della tabella relativa al test 1. In particolare i domini *economy* e *sociology* in pratica trovano la loro totale occorrenza nei lemmi monosemici.

Test3

Consideriamo le annotazioni manuali:

Dominio	Occorrenze
Factotum	**252**
Religion	**46**
Person	**41**
Time_period	**40**
Sociology	**34**
Law	**29**
Politics	**29**
Computer_science	**20**
Anthropology	**19**
Tourism	**18**
Economy	**17**
Military	**16**
Publishing	**16**
Pedagogy	**15**
Enterprise	**14**
History	**13**
Sexuality	**13**
Geography	**11**
Free_time	**11**
Linguistics	**11**
Town_planning	**10**
Literature	**10**

Tabella 5.26- Occorrenze dei domini delle annotazioni manuali delle directory di YAHOO_GOOGLE

Per brevità si sono riportati solo I domini con occorrenze maggiori o uguali a 10.
Dalla tabella si può notare come i domini ad occorrenza maggiore seppur in ordine differente, rispecchiano quelli individuati nel test 1.

Test 4

Numero di domini selezionati	Numero totale di termini valutati	Numero di termini disambiguati	Numero di disambiguazioni corrette	Precision	Recall
1	518	64	40	0,08	0,63
2	518	153	70	0,14	0,46
3	518	208	102	0,20	0,49
4	518	209	103	0,20	0,51
5	518	224	117	0,23	0,52
6	518	241	124	0,24	0,51
7	518	241	124	0,24	0,51
8	518	241	124	0,24	0,51
9	518	244	126	0,24	0,52
10	518	253	130	0,25	0,51

Tabella 5.27- Risultati dei soli lemmi polisemici directory di YAHOO_GOOGLE caso CF

Numero di domini selezionati	Numero totale di termini valutati	Numero di termini disambiguati	Numero di disambiguazioni corrette	Precision	Recall
1	310	49	40	0,13	0,82
2	310	109	70	0,23	0,64
3	310	152	102	0,33	0,67
4	310	153	103	0,33	0,67
5	310	168	117	0,38	0,70
6	310	178	124	0,40	0,70
7	310	178	124	0,40	0,70
8	310	178	124	0,40	0,70
9	310	180	126	0,41	0,70
10	310	185	130	0,42	0,70

Tabella 5.28- Risultati dei soli lemmi polisemici directory di YAHOO_GOOGLE caso CF

Test 5

Per brevità in questo caso non si considereranno tutte le possibili combinazioni, ma si riporteranno solo quelle che portano ad annotazioni:
time_period/person: 2 annotazioni da synset diversi;
person/politics: 9 annotazioni da synset diversi;

Considerazioni:
Considerando solo i lemmi non monosemici i valori di recall e precision, come era facile immaginarsi peggiorano.

5.3.3 Analisi dei risultati

Dai dati ottenuti nell'analisi di tutte le tipologie di dati, si evidenzia la mancanza di una e vera e propria polarizzazione di dominio. Ciò che si nota è piuttosto una polarizzazione su un insieme di domini, il cui numero cresce al crescere dei dati. In particolare si è osservato che per ottenere un'annotazione corretta di circa il 50% dei lemmi, sia necessario considerare circa il 13-15% dei domini più frequenti. Questo fa supporre che piuttosto che una polarizzazione di dominio, esista una polarizzazione su una percentuale di domini, corrispondenti a quelli con occorrenza maggiore.
A differenza dei dati del progetto WISDOM, inoltre, si ottengono valori di Recall più bassi, il che probabilmente è da attribuire alla maggiore varietà e numero dei dati. Quindi, ad una prima analisi, il metodo di annotazione basato su WND sembrerebbe più efficiente quando applicato a sorgenti di dati di dimensioni limitate.
I test effettuati fin'ora si sono limitati a considerare solo i primi domini, allo scopo, principalmente, di evidenziare un'eventuale polarizzazione dei lemmi verso uno o due domini.
Un ulteriore test potrebbe essere quello di tentare di annotare il numero maggiore di lemmi possibile, considerando valori di n per il test 1, via via crescenti, allo scopo di verificare quanti lemmi risultino effettivamente annotati correttamente utilizzando come unica fonte di informazione i domini con cui tali lemmi risultano etichettati.

Un eventuale futuro algoritmo di disambiguazione dovrà inoltre tener conto della problematica relativa a quale dominio scegliere ogni qual volta si presentino domini a pari occorrenza. In questo e nei futuri test non è stato applicato alcun criterio di scelta, il dominio è stato selezionato nella maggior parte dei casi in ordine alfabetico come riportato in tabella.
Tali iniziali risultati andrebbero ulteriormente approfonditi verificando un'eventuale compatibilità di più differenti annotazioni. Per ora abbiamo definito a priori, come scorrette, le annotazioni discordanti con quelle manuali, ma ciò potrebbe essere un'assunzione parzialmente errata: un lemma infatti potrebbe accettare più annotazioni, senza che l'una escluda necessariamente l'altra.

Di seguito si analizzeranno i dati, cercando di annotare i termini con la massima accuratezza consentita.

Dati di WISDOM

Nel grafico in tabella 5.5, i risultati dei dati provenienti dal progetto WISDOM nel caso "con factotum". Dalla figura si evidenzia come convenga considerare i primi tre domini, per disambiguare attraverso il nostro meccanismo basato sui domini, ottenendo un 90% di accuratezza sul circa 30% dei termini. Si ricorda, inoltre, che in questo caso il numero medio di termini considerati, per ciascun cluster, sono pari a circa 51.

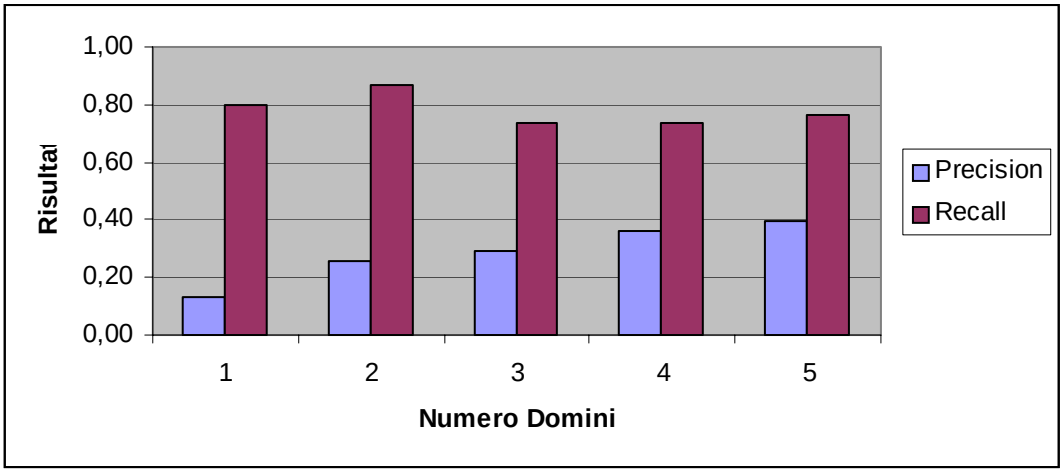


Figura 5.5-Recall e Precision dei dati di WISDOM nel caso con Factotum

Il grafico in figura 5.6, mostra i risultati nel caso “senza factotum”. In tal modo è possibile vedere chiaramente le prestazioni dell’algoritmo senza il rumore introdotto dai termini factotum. Come ci si aspettava, in questo caso, si ottengono prestazioni migliori, in quanto, basta considerare i primi due domini, per disambiguare circa il 40% dei lemmi con un’accuratezza quasi del 100%. In questo caso si sono analizzati in media, per ogni cluster, circa 41 lemmi.

In figura 5.7, si mostrano i risultati del test, analizzando i soli termini polisemici nel caso con factotum. In questo caso la media dei termini considerati è di circa pari a 29, e conviene considerare solo i primi quattro domini, per ottenere le annotazioni accurate di circa il 30% dei termini.

In figura 5.8, si mostrano i dati sui lemmi polisemici di WISDOM, nel caso senza factotum, e converrebbe considerare nella disambiguazione, solo i primi due domini, raggiungendo comunque una copertura intorno al 30%.

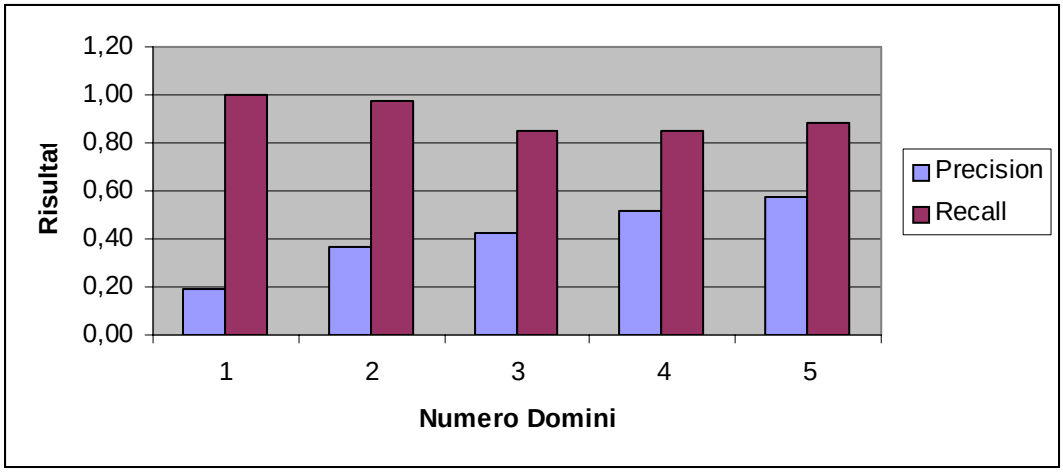


Figura 5.6-Recall e Precision dei dati di WISDOM nel caso senza factotum

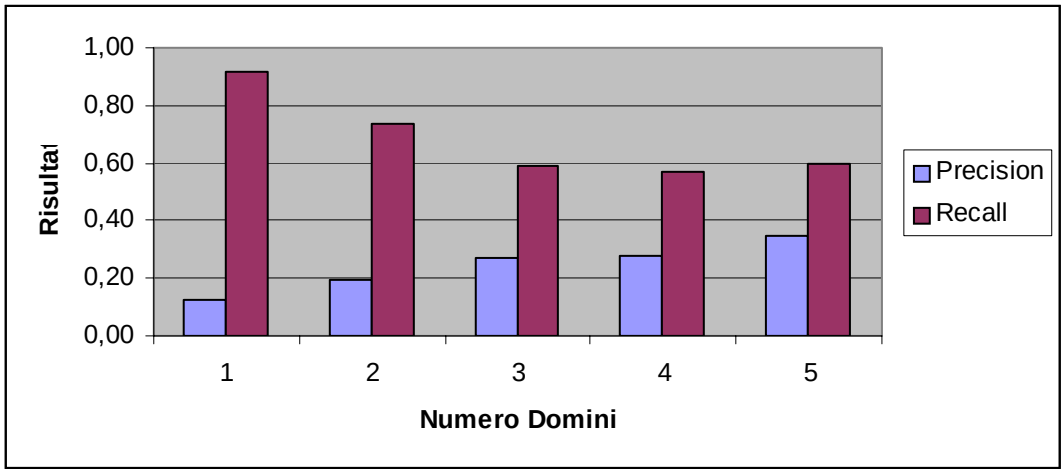


Figura 5.7-Recall e Precision dei dati di WISDOM dei lemmi polisemici caso con factotum

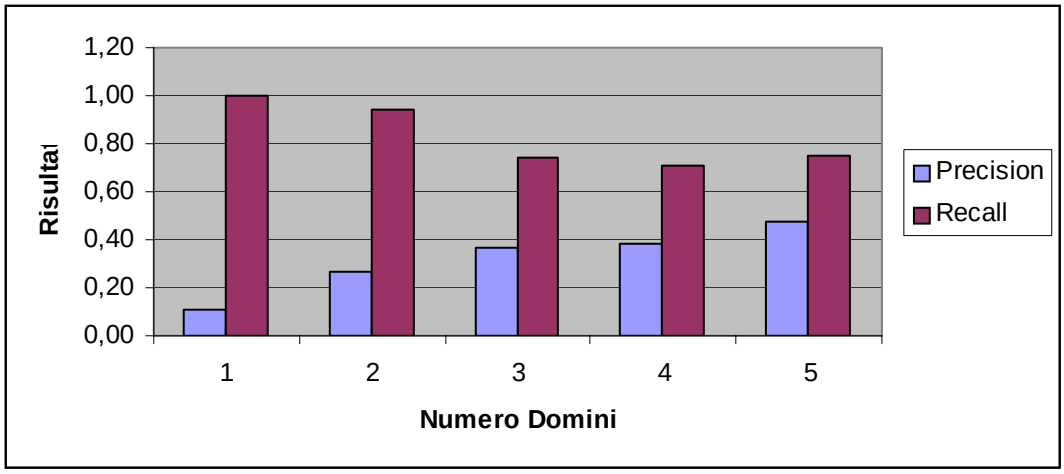


Figura 5.8-Recall e Precision dei dati di WISDOM dei lemmi polisemici caso senza factotum yahoo CF

Dati provenienti dalle directory di GOOGLE e YAHOO

In figura 5.9 si mostrano i risultati dei dati provenienti dalle directory di YAHOO e GOOGLE, nel caso con factotum. Si sono considerati circa 800 lemmi e, considerando i primi quattro domini, si disambigua circa il 17% dei termini con quasi l'80% di accuratezza.

Nel caso senza factotum (figura 5.10), invece considerando circa 500 termini si disambigua circa il 25% dei termini con quasi l'80% dell'accuratezza.

Nel caso dei lemmi polisemici (figura 5.11, 5.12), considerando i primi quattro domini , si disambigua circa il 30% dei lemmi con un'accuratezza pari a circa l'80%.

La motivazione alla base, del numero minore di termini disambiguati, nel caso GOOGLE e YAHOO, rispetto ai dati provenienti dal progetto WISDOM, è da ricercarsi probabilmente nell'ampio contesto considerato. In questo caso, forse, sarebbe stato più opportuno testare il nostro approccio, con una finestra di contesto più limitata. Ulteriori test, sull'uso di WordNet Domains, potrebbero per esempio, valutare come variano le prestazioni al variare della dimensione del contesto considerato.

In generale, possiamo comunque affermare, che tramite il solo utilizzo di WordNet Domains, sia possibile annotare, in media, circa il 30% dei lemmi di una risorsa con un livello di accuratezza intorno al 90%.

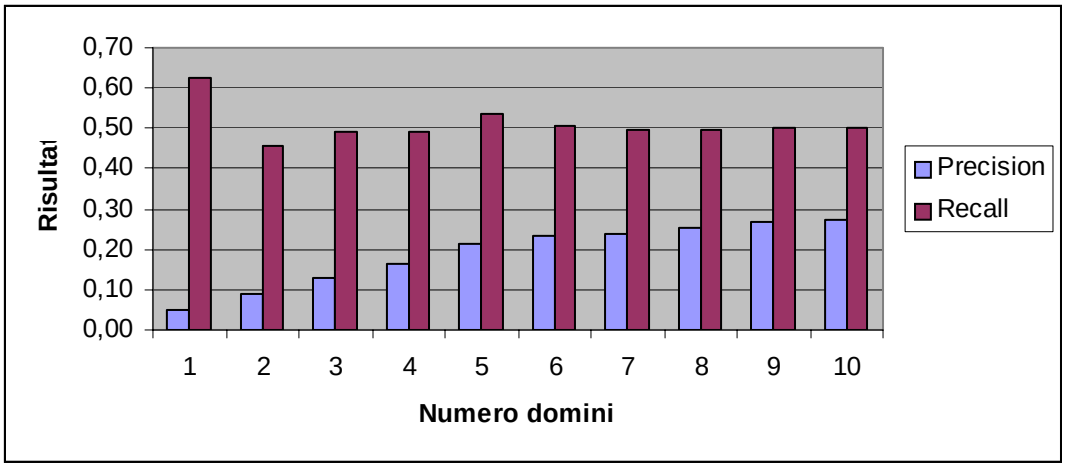


Figura 5.9- Recall e Precision dei dati di YAHOO e GOOGLE nel caso con factotum

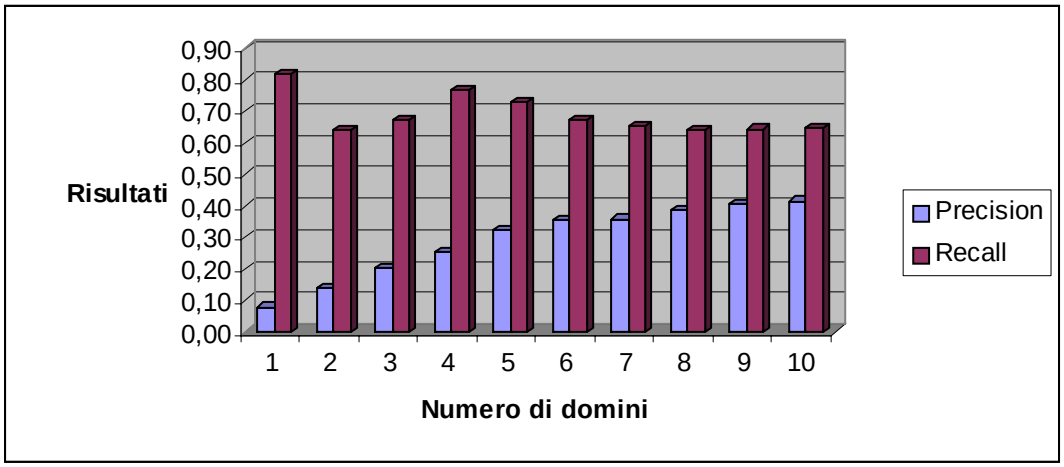


Figura 5.10-Recall ePrecision dei dati di YAHOO eGOOGLE nel caso senza factotum

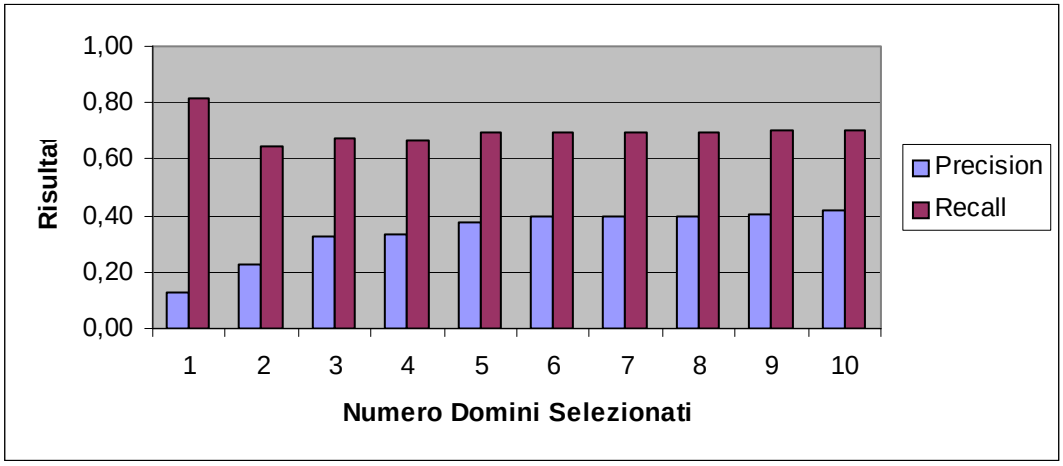


Figura 5.11-Recall e Precision dei lemmi polisemici di YAHOO e GOOGLE nel caso senza factotum

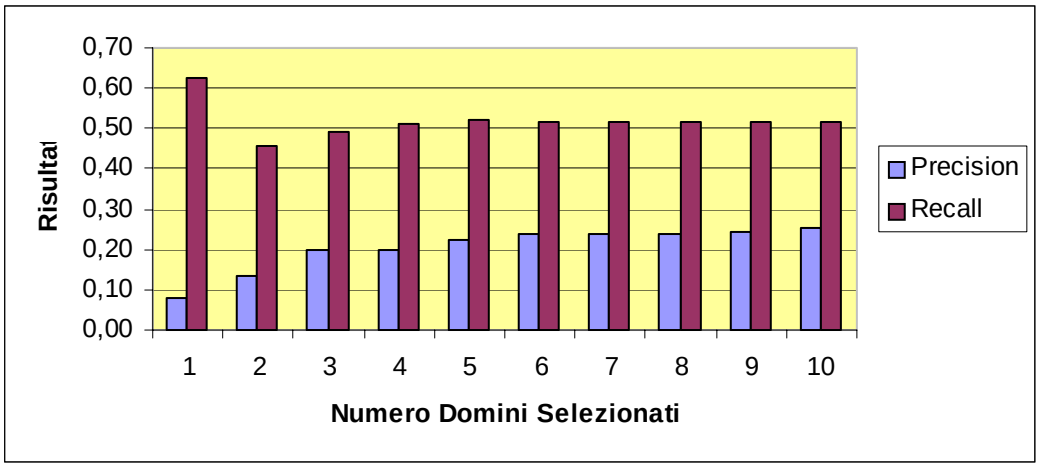


Figura 5.12-Recall e Precision dei lemmi polisemici di YAHOO e GOOGLE caso senza factotum

Conclusioni e sviluppi futuri

Attraverso il lavoro di ricerca che ha caratterizzato questa tesi, è stato possibile delineare e classificare, i vari metodi ed algoritmi di disambiguazione del testo, proposti in letteratura, e basati sull'utilizzo del database lessicale WordNet.
Ciò che è emerso, è che attualmente, tali algoritmi, consentono di raggiungere prestazioni diverse a seconda delle caratteristiche della sorgente di dati utilizzata (dimensione, tipologia di dati…), e che, in generale, queste non superano in termini di precisione, il 70%-80%. Ciò, è dovuto a diversi fattori tra cui alcune significative lacune presenti in WordNet.
Di conseguenza, la nostra analisi, è progredita verso lo studio di alcune estensioni di WordNet, proposte in letteratura, al fine di poter fornire una base di conoscenza completa al processo di disambiguazione. In particolare, si sono determinate due estensioni rilevanti:

- **eXtended WorNet** la quale arricchisce il database lessicale realizzando la disambiguazione dei termini delle glosse, e quindi, inoltre, fornisce una via per incrementare il livello di relazioni lessico-semantiche all'interno di WordNet.
- **WordNet Domains** il quale rappresenta un'ontologia di dominio, in grado di associare ad ogni synset di WordNet, il proprio dominio o domini di competenza.

In questa tesi, si è scelto di concentrare la nostra attenzione, principalmente, su quest'ultima estensione, poiché essa propone una via per porre soluzione a molte delle lacune individuate in WordNet. WordNet Domains, è stato quindi integrato all'interno del database di MOMIS, ed è stato testato al fine di valutare la sua effettiva utilità.
I risultati dei test effettuati, hanno mostrato come effettivamente, l'informazione di dominio, sia rilevante all'interno del processo di disambiguazione, e come, con il suo solo utilizzo, sia possibile, disambiguare parzialmente, ma con buona precisione, i termini all'interno di una sorgente di dati. Inoltre, la metodologia utilizzata per disambiguare, all'interno dei nostri test, è completamente automatica e non richiede l'intervento dell'utente o di corpus di dati pre-annotati.
Per quanto concerne, il valore parziale di annotazioni ottenute, ciò è motivato dal fatto che, nel nostro caso, si è data priorità alla correttezza delle annotazioni, rispetto al quantitativo

totale di termini disambiguati. Ovviamente, in applicazioni che richiedono un livello di accuratezza minore, sarà possibile disambiguare un numero maggiore di termini.
Dalla nostra analisi, si è ,inoltre, evidenziato, come l'utilizzo di metodologie composte (ovvero combinanti due o più approcci) consenta di ottenere, in generale, prestazioni migliori rispetto, all'utilizzo dei singoli metodi di disambiguazione. Di conseguenza, il numero parziale di annotazioni ottenute utilizzando come risorsa solo WordNet Domains, potrebbe essere incrementato, estendendo ulteriormente il database lessicale di MOMIS, per esempio anche attraverso eXtended WordNet. Ciò che si viene a delineare, è quindi, l'idea che la soluzione al problema della disambiguazione del testo debba essere intesa come un processo incrementale e composto, all'interno del quale l'estensione di WordNet con WordNet Domains rappresenta solo un primo, ma significativo, passo. Inoltre, in futuro, considereremo la realizzazione di un algoritmo parametrizzabile, che permetta, in base alle caratteristiche della sorgente di dati (es: tipologia di dati, dimensione ecc…) , di selezionare il metodo di disambiguazione più opportuno.

Riferimenti

[1] Gorge A. Miller: *WordNet: a lexical database for english.* Communications for the ACM,38(11): 39-41, 1995.

[2] R. Hull and R. King et al. *Arpa I3 reference architecture.* 1995. Reperibile presso: http://www.isse.gmu.edu/I3_Arch/index.html/.

[3] Gio Wiederhold et al. *Intergrating artificial intelligence and database technology*. Journal of Intelligent Integration System, 2/3 Giugno 1996.

[4] F.Saltor and E. Rodriguez .*On intelligent access to heterogeneous information*. In Proceeding of the 4th KRDB Workshop, Atene, Grecia, Agosto 1997.

[5] D. Beneventano, S. Bergamaschi, S.Lodi, e C. Sartori. *Consistency checking in complex object database schemata with integrity constraints*. Technical Report 103, CIOC , Bologna, Italia , 1994.

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

[7] D.Beneventano, S.Bergamaschi, C.Sartori e M.Vincini. *ODB-tools: a description logics based tool for schema validation and semantic query ottimization in object oriented databases*. In Proc. Of Int. Conference of the Italian Association for Artificial Intelligence (AI*IA, 97), Roma, 1997.

[8] D.Beneventano, S.Bergamaschi, C.Sartori, e M.Vincini. *Odb-qoptimizer: a tool for semantic query optimization in oodb*. In Proc. Of Int. Conf. On Data Engineering ICDE'97, Birningham, UK, April 1997.

[9] D.Beneventano, S.Bergamaschi, C.Sartori, e M.Vincini. *A description logics based tools for schema validation and semantic query optimization in oodb*. In Proc. Of Int. Conf. On Data Engineering ICDE'97, Birningham, UK, April 1997.

[10] S.Castano e V.De Antonellis. *Deriving global conceptual views from multiplie information sources*. In preProc. Of ER'97 Preconference Symposium on Conceptual Modeling, Historical Perspective and future Directions, 1997.

[11] T. Catarci e M. Lenzerini. *Rapresenting and using interschema knowledge in cooperative information systems.* Journal of Intelligent and Cooperative Information Systems, 2(4)375-398, 1993.

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

[13] R.G.G. Cattel, editor. *The object Database Standard: ODMG93*. Morgan Kaufman Publisher, San Francisco, CA, 1997.

[14] R Benassi, S.Bergamaschi, A.Fergnani e D.Miselli. *Extending a Lexicon Ontology for Intelligent Information Integration.*

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

[16] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval, Addison-Wesley, 1999.

[17] J.Morris e G.Hirst, 1991. *Lexical cohesion computed by thesaural relations as an indicator of the structure of text*. Computational Linguistics, 18:21-45.

[18] K.Halliday e R.Hasan. *An Introduction to Functional Grammar,* Edward Arnold, Londra, 1985.

[19] G.Hirst e D.StOnge. *Lexical chains as representations of context for the detection and correction of malatropisms WordNet: An electronical lexical database,* C.Fellbaum (editor), Cambrige, MA: The MIT Press, 1998.

[20] R. Barzilay, M.Elhadad. *Using Lexical Chains for Text Summarization,* in ACL/Eacl-97 summarization workshop, pages 10-18, Madrid 1997.

[21] R. Barzilay. *Lexical Chains for Text Summarization*, M.Sc degree of Ben-Gurion University of the Negev, 1997.

[22] H.G.Silber, K.F.McCoy. *Efficiently Computed Lexical Chains as an Intermediate Representation for Automatic Text Summarization*. Computational Linguistics, vol. 28, 2002.

[23] M. Galley e K.McKeown. *Improving Word Sense Disambiguation in Lexical Chaining.*

[24] W.Gale, K.Church, e D.Yarowsky. *One sense per discourse*. In Proc. Of the DARPA Speech and Natural Language Workshop, 1992.

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

[26] M.Lesk. *Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from ice cream cone.* In Proceeding of SIGDOC ’86, 1986.

[27] S. Banerjee and T.Pedersen. *An adapted Lesk algorithm for word sense disambiguation using WordNet*. In Proceeding of the Third International Conference on Intelligent Text Processing and Computational Linguistics, Città del Messico, Febbraio 2002.

[28] G.Zipf. *Human Behavior and the Principle of Least Efford.* Addison-Wesley, Cambridge, MA, 1949.

[29] G.Hirst, A.Budanitsky. *Evaluating WordNet-based measured of lexical Semantic Relatedness.* Association for Computational Linguistics, 2005.

[30] J.Michelizzi. *Semantic Relatedness Applied to all word sense disambiguation.* Tesi per il master in scienze, Università del Minnesota, Luglio 2005.

[31] Claudia Leacock e M. Chodorow. *Combining local context and WordNet similarity for word sense identification.* In Christian Fellbaum, editor, WordNet: An electronical lexical database, capitolo 11, pagine 265-283, MIT Press, 1998.

[32] Philip Resnik. *Semantic similarity in a taxonomy: an information-based measure and its applications to problems of ambiguity in natural language*. Journal of Artificial Intelligence Research, 11:95-130, 1999.

[33] S.Banerjee e T.Pedersen. *Extended gloss overlap as a measure of semantic relatedness.* In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, pagine 805-810, Acapulco, Messico, Agosto 2003.

[34] J.Jiang e D.Conrath. *Semantic similarity based on corpus statistics and lexical taxonomy.* In Proceeding of International Conference Research on Computational Linguistica (ROCLING X), Taiwan, 1997.

[35] S. Patwardhan. *Using WordNet-based Context Vectors to Estimate the Semantic Relatedness of Concepts*. 2006.

[36] D.Lin. *An information-theoretic definition of similarity*. In Proceeding of the Fifteenth International Conference on Machine Learning (ICML-98), pagine 296-304, Madison , Wisconsin, 1998.

[37] T. Pedersen, S.Banerjee, S. Patwardhan. *Maximizing semantic relatedness to perfom word sense disambiguation.* Technical report UMSI 2005/25, University of Minnesota Supercomputing Institute, Marzo 2005.

[38] S.Patward, S.Banerjee e T.Pedersen. *Using measure a of semantic relatedness for word sense disambiguation.* In Proceeding of Fourth International Conference on Intelligent Text Processing and Computational Linguistica (CICLING 03), città del Messico, Messico, Febbruaio 2003.

[39] Z. Harris. *Distributional structure.* In J.J. Katz, editor, The Philosophy of Linguistics, pagine 26-47. Oxoford University Press, New York.

[40] K. Laudauer e S.T. Dumais. *A solution to plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge.* Psychological Rewiew, 104:211-240.

[41] H. Schulze. *Automatic word sense discrimination.* Computational Linguistics, 24(1):97-123.

[42] R. Mihalcea. *Unsupervised Large Vocabolary Word Sense Disambiguation with Graph-based Algorithms for Sequence Data Labeling 2002.*

[43] D.Yarowsky. *Unsupervised word sense disambiguation rivaling supervised methods*. In Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, pp. 189-196. Cambridge, MA, 1995.

[44] C.Strapparava, A. Ghiozzo e C.Giuliano. *Pattern Abstration and Term Similarità for Sense Disambiguation.* In Proceeding of the 3rd ACL workshop on the Evaluation of system for the Semantic Analys is of Text. (SENSEVAL-3). Barcellona, Spagna 2004.

[45] R. Mihalcea e E.Faruque. *SenseLearner: minimally supervised word sense disambiguation for all word in open text.* In Proceeding of the 3rd ACL workshop on the

Evaluation of system for the Semantic Analys is of Text. (SENSEVAL-3). Barcellona, Spagna, Luglio 2004.

[46] W.Daelemans , J.Zavrel, K van der Sloot. *Timbl: Tilburg memory based learner, version 4.0, refence guide.* Technical report, University of Antwerp, 2001.

[47] R.Mihalcea. *Instance based learning with automatic feature selection applied to Word Sense Disambiguation.* In Proceeding of the 19th International Conference on Computational Linguistics (COLING 2002), Taipei, Tawan, Agosto 2002.

[48] D.Lin. *Using syntactic dependency as local context to resolve word sense ambiguity.* In Proceeding of the Association Computational Linguistics,Madrid, Spagna 1997.

[48] R.Navigli e P.Velardi. *Structural Semantic Interconnection: A knowledge-based approach to word sense disambiguation.* IEEE Transaction on pattern analysis and machine intellige, vol. 27 num. 7, luglio 2005.

[49] A.Novischi. *Combining methods for word sense disambiguation of WordNet glosses.* American Association for Artificial Intelligence, 2004.

[50] S.Brody, R.Navigli, M.Lapata. *Esemble Methods for Unsupervised WSD.* Proceedings of the 21th International Conference Linguistics and 44th Annual Meeting of the ACL, pagine 97-104, Sydney, Luglio 2006.

[51] D.McCarthy et. al..*Finding Predominant Word Senses in Untagged Text.* 2005

[52] F.Mandreoli, R.Martoglia, E.Ronchetti. *Versatile Structural Disambiguation for Semantic-aware Applications.*

[53] S.Johansson. *The tagged LOB Corpus*. Norwegian Computing Centre of the Humanities, 1986.

[54] B.Magnini, S.Strapparava et. al.. *The role of domain Information in Word Sense Disambiguation.* Natural Language Engineering, 25 luglio 2002.

[55] D.Yarowsky e R.Florian. *Evaluating sense disambiguation across diverse parameter space.* Natural Language Engineering 8 (4), 293-310. 2002.

[56]A.Gliozzo, C.Strapparava, I. Dagan. *Unsupervised e Supervised Exploitation of Semantic Doamins in Lexical Disambiguation.* 2002.

[57] R.Krovetz. *More than one sense per discourse*. Technical report, Princeton, 1998. NEC Research Institute.

[58] R.Mihalcea, Dan I Moldovan. *EXtended WordNet: progress report*. In Proceeding NAACL Workshop on WordNet and other Lexical Reources, Pittsburg, PA, 2001.

[59] S.Harabagiu e al.. *WordNet 2- a morphologically and sematically enhance resource*. In Proceeding of SIGLEX –'99.

[60] Dan Moldovan, A.Novischi. *Lexical Chains for Question Aswering*. 1999

[61] M.Castillo, F.Real, G.Rigau. *Disambiguating WordNet Glosses*. SENSEVAL-3, 2004

[62] B.Magnini, C.Strapparava, G.Pezzullo e A.Ghiozzo. *Comparing Ontology-based and Corpus-Based Domain Annotation in WordNet.* In Proceeding of first International WordNet Conference, 2002.

[63] L. Bentivogli e al.. *Revising the WordNet Domains Hierarchy: semantics, covarage e balanncing.* Corpus Linguistics 2003, Conference, Lancaster United Kingdom.

[64] A.Diekema. Dewey Decimal Classification, 1998.

Ringraziamenti

Un sentito ringraziamento alla Professoressa Sonia Bergamaschi, e all'Ing. Laura Po per l'aiuto fornito durante la realizzazione di questa tesi.

Ringrazio, la mia famiglia, che mi ha sempre sostenuto durante tutti questi anni di studio.

Inoltre, desidero ringraziare tutte le persone a me vicine che mi hanno "sopportato" e in coraggiato nei momenti meno facili.