Università degli Studi di Modena e Reggio Emilia

Facoltà di Ingegneria di Modena

Corso di Laurea in Ingegneria Informatica

ANNOTAZIONE LESSICALE AUTOMATICA DI SCHEMI IN SISTEMI DI INTEGRAZIONE DEI DATI: ANALISI E SVILUPPO DI TECNICHE PER NOMI COMPOSTI

Relatore:
Chiar.mo Prof. Sonia Bergamaschi

Correlatore:
Dott. Ing. Serena Sorrentino

Candidato:
Elena Parmiggiani

Anno Accademico 2007/2008

PAROLE CHIAVE

Annotazione

Disambiguazione

NomiComposti

WordNet

JWNL

Indice

Indice delle figure

Indice delle tabelle

Introduzione

L'analisi dei termini composti (o, più in generale, delle sequenze di termini) è una sfida con cui i ricercatori si cimentano da molto tempo, da prima che Tim Berners Lee introducesse, nel 2001, il concetto di Web Semantico. Essa si colloca all'interno del più vasto problema del *Natual Language Processing* (NLP) e dell'*Information Retrieval* (IR). A causa del rapido sviluppo del web, si è via via reso sempre più necessario interrogare sorgenti dati estremamente eterogenee: per questo motivo si sono sviluppati sistemi in grado di integrare tali risorse. Tra questi sistemi, si colloca MOMIS (Mediator EnvirOment for Multiple Information Sources). MOMIS [1,2,3] è un sistema di I_3 (Integrazione Intelligente delle Informazioni) che estrae automaticamente i dati provenienti da documenti strutturati e semi-strutturati e ne realizza una fusione intelligente.

All'interno di MOMIS, il meccanismo di annotazione semantica associa a ciascun termine della sorgente uno o più significati. Tali significati sono selezionati tra quelli associati al termine in esame, rispetto all'ontologia lessicale WordNet. Questo meccanismo, tuttavia, non permette di annotare i termini composti eventualmente presenti nella sorgente, se non presenti all'interno del database lessicale. L'ostacolo deriva dalla natura semantica del problema: l'interpretazione delle parole composte è, nel caso migliore, solo parzialmente riconducibile alla sua analisi sintattica e morfologica: è necessario reperire informazioni riguardanti il ruolo di ogni singola parola all'interno del composto. Appare subito evidente come la semantica sia il fulcro della questione: la *semantica* è il ramo della linguistica che si occupa dello studio dei significati delle parole e dei loro cambiamenti. È chiaro che ad ogni parola corrispondono più significati, e che, per ciascuno di questi significati, tale parola andrà a relazionarsi in modo diverso rispetto alle parole vicine (sia dentro la frase che nel composto di cui è parte). Come diretta conseguenza, vi è la costituzione di un grafo di relazioni che lega tra loro i diversi significati di ogni termine.

Una parola composta è formata da un termine principale, l'*head noun* (o, semplicemente, *head*), e da uno o più modificatori, il/i *modifier(s)*. La risoluzione di un composto, quindi, consiste nell'identificazione del ruolo assunto da ciascun membro all'interno della sequenza di parole. Si tratta di un'operazione necessaria e preliminare rispetto alla disambiguazione vera e propria, che è, invece, la ricerca di un significato o di un singolo lemma da sostituire al composto.

La risoluzione e la disambiguazione delle parole composte, pertanto, costituiscono un problema che comporta l'analisi dell'informazione implicitamente contenuta in ogni parola e di quella che

essa apporta in quanto membro di un composto più ampio. Non si può quindi prescindere né dalla semantica, né dalla sintassi. Una seconda problematica è causata dalla lingua: ogni lingua (si pensi alle differenze tra la grammatica e la logica italiane e inglesi) ha un insieme di regole differenti e in alcuni casi variabili, ragion per cui è necessario prescindere da qualsiasi supposizione riguardo all'ordine delle parole o al loro ruolo. Come è vero che in una parola composta inglese (ad esempio *apple pie*, *student identity*, *university member*) l'*head* sta a destra rispetto al *modifier*, come da sintassi, è altrettanto vero che in italiano e in altre lingue neolatine l'ordine non è assolutamente scontato e l'analisi deve essere effettuata caso per caso. Inoltre, quelle che sono parole composte in inglese nella forma nome + nome, in lingue come l'italiano potrebbero, ad esempio, presentarsi nella forma nome + aggettivo o aggettivo + nome, in modo da renderne la disambiguazione più facile.

È ora necessario specificare che, con "termini composti", si intendono sequenze di due o più parole, separate quindi da spazio bianco, che abbiano uno o più significati se considerate insieme. Non sono considerati termini composti, pertanto, parole come "portachiavi" o "taglialegna", i quali, non presentando separatori, sono da considerarsi termini singoli. Sono invece termini composti, ad esempio, *last name* e *member identifier*. Da qui in avanti, ci si riferirà a "termini composti" (a volte sintetizzato con "composti") come sinonimo di "parole composte". "Termini composti", "parole composte" e "composti" saranno usati indifferentemente.

Questo lavoro di tesi si occupa, in particolare, della disambiguazione di parole composte presenti in documenti strutturati e semi-strutturati (database, documenti XML, ma anche tabelle, fogli di calcolo, etc...). Lo scopo è realizzare un'estensione per il sistema MOMIS, che sia in grado di estrarre in maniera automatica dei pattern linguistici che costituiscono delle parole composte, di trovare le relazioni che legano i termini componenti e di annotarne il significato in modo che se ne tenga traccia.

Il punto di partenza di questo lavoro è stato lo studio delle ricerche e degli esperimenti già effettuati dai ricercatori: si tratta, nella quasi totalità dei casi, di algoritmi che quindi non affrontano il problema della posizione di *head* e *modifier(s)*, e che hanno seguito diversi approcci, a seconda del tipo di applicazione. Inoltre, questi algoritmi sono quasi sempre rivolti alla disambiguazione di parole (composte e non) per mezzo di ampi corpora, mentre solo di recente i ricercatori hanno iniziato a far riferimento all'informazione contenuta nel solo composto e alla sua posizione

all'interno di un *thesaurus*.

Successivamente, si è rivolta l'attenzione ai quattro grandi sotto-problemi in cui possiamo dividere la disambiguazione di parole composte:

1. L'estrazione di sequenze di termini che possano essere considerati parole composte e il riconoscimento del ruolo dei componenti (nome, aggettivo,...);
2. La distinzione fra *head* e *modifier* (o *modifier*s);
3. Lo sviluppo di un algoritmo per la ricerca delle relazioni che intercorrono tra i termini e deduzione di un significato da attribuire al nome composto;
4. L'annotazione del risultato così ottenuto all'interno del sistema (MOMIS, nel caso di questa tesi).

La struttura della tesi è quindi la seguente:

- **Capitolo 1** → Questo capitolo ha lo scopo di introdurre gli strumenti di base per l'analisi delle parole composte: la struttura e il funzionamento del dizionario concettuale WordNet e le principali misure della similarità e della relazioni tra termini che sono state presentate nella storia dell'analisi delle parole composte.
- **Capitolo 2**: → In questo capitolo si trova l'esposizione dello stato dell'arte nell'analisi di parole parole composte. I metodi proposti vengono così suddivisi:
  - ✓ metodi basati su statistiche effettuate su grandi volumi di testi, o corpora;
  - ✓ metodi basati sull'analisi del contenuto semantico dei termini, che non facciano affidamento su informazioni precedentemente codificate.
- **Capitolo 3** → Qui si trova la presentazione dell'algoritmo deputato alla disambiguazione di termini composti (e, più nello specifico, di nomi composti), oggetto del presente lavoro di tesi, degli strumenti utilizzati per realizzarlo e dell'analisi dei risultati con esso ottenuti.
- **Capitolo 4** → Nel quarto capitolo vengono presentati, discussi e analizzati i risultati ottenuti attraverso l'applicazione dell'algoritmo alle sorgenti fornite.

Capitolo 1

1 La misura della relazione tra concetti e la similarità

Le prime ricerche di un algoritmo che permetta di disambiguare le parole composte risalgono quantomeno agli anni '70-'80. Tra gli algoritmi proposti, alcuni si occupano di disambiguazione di nomi composti, mentre altri si rivolgono in generale alle parole composte, e anche alle cosiddette *Multiword Expressions* (MWE), cioè a qualsiasi combinazione o sequenza di parole che può andare da idiomi, nomi propri, composti, fino a frasi fatte.

Mark Lauer [5] definisce i nomi composti (in inglese *compound nouns*) come "*una costruzione di occorrenza comune in una lingua che consiste in una sequenza di nomi, e che agisce sintatticamente come un nome*". Egli li analizza sintatticamente per mezzo dell'utilizzo ricorsivo della regola

$$N \rightarrow N\,N,$$

dove N è una sequenza di uno o più nomi. Nel caso dell'inglese, sequenze di più nomi formanti un nome composto sono frequentissime. Non è invece il caso dell'italiano, dove quelli che in inglese sono composti nella forma nome + nome, in italiano diventano aggettivo + nome, o nome + aggettivo, oppure introducono una preposizione (nome + preposizione + nome). Pertanto, in questa tesi ci si rivolgerà alla risoluzione ed alla disambiguazione di generiche parole composte, allo scopo di realizzare un algoritmo il più possibile indipendente dalla lingua della sorgente dei dati.

Inoltre, i vari algoritmi che sono stati proposti nel tempo differiscono tra loro per il tipo di approccio seguito: alcuni si basano su metodi statistici, altri sulla semantica.

- ➔ I metodi statistici sono concentrati sul contesto nel quale il nome composto si trova, e cercano le informazioni riguardanti gli elementi in esame all'interno di corpora, ad esempio la probabilità di un termine di apparire vicino ad un altro o in un determinato contesto. Si tratta di approcci utili qualora l'analisi si svolga su grandi raccolte di documenti di tipo non strutturato.
- ➔ I metodi basati sull'informazione semantica, invece, si occupano dell'aspetto lessicale e cercano le possibili relazioni che intercorrono tra le due parole appartenenti ad un nome composto andando a ricercare la reciproca posizione dei termini nel grafo realizzato a partire

da qualche database lessicale (come WordNet) oppure facendo riferimento a nomi composti già precedentemente disambiguati. Questi algoritmi sono più indipendenti dal dominio di applicazione rispetto ai primi.

Storicamente l'attenzione è stata rivolta in misura maggiore al primo tipo di approccio. Questo, come si diceva, a causa dell'assenza di ampie risorse lessicali (dizionari online ad utilizzo libero) cui fare riferimento. Già Lauer, nel 1994 [5], in riferimento a due precedenti algoritmi proposti per la disambiguazione di parole composte, dice "*both these works were aimed solely at syntactic disambiguation. The goal of semantic interpretation remains to be investigated*".

La quantificazione della relazione lessico-semantica ha trovato grande applicazione nel NLP, e molte e varie misure sono state proposte. La necessità di determinare la relazione semantica (o il suo inverso, la distanza semantica) tra due concetti è dovuta al suo utilizzo in applicazioni come la disambiguazione di parole, la determinazione della struttura di testi, il riassunto e l'annotazione di testi, l'estrazione e il recupero di informazioni (IR), l'indicizzazione automatica, la selezione lessicale e la correzione automatica di errori all'interno di un testo. È importante notare che la relazione semantica è un concetto più generale rispetto alla similarità, in quanto esprime quanto due concetti sono simili, basandosi su una gerarchia del tipo *is-a*. Ad esempio, un'*automobile* può essere considerata più simile a una *barca* che ad un *albero*, poiché *barca* e *automobile* condividono *veicolo* come progenitore comune. D'altro canto, la relazione dice quanto due concetti "hanno a che fare", senza che sia implicata una similarità. Ad esempio, se *automobile* e *barca* sono simili, *automobile* e *ruota* sono in relazione tra loro. Un *coltello* è usato per tagliare il *pane*, *caldo* è il contrario di *freddo,* la *neve* è formata a partire dall'*acqua*... etc.

1.1 WordNet

WordNet è un network lessicale di grandi dimensioni per la lingua inglese sviluppato presso la Princeton University da George A. Miller e dai suoi collaboratori. È disponibile gratuitamente per il download o l'utilizzo online sul sito http://www.cogsci.princeton.edu/wn. Il copyright di WordNet è in possesso dell'università di Princeton (1997). La licenza d'uso ne prevede un utilizzo gratuito anche al di fuori della ricerca, con il solo vincolo che siano espressi gli autori e l'indirizzo web di riferimento. Attualmente (ottobre 2008), le versioni più aggiornate sono la 2.1 per Windows e la 3.0 per i sistemi Unix-based.

A differenza di quanto si potrebbe pensare, WordNet non è un canonico dizionario online. Si tratta

anzi di un articolato sistema di riferimento la cui struttura è ispirata alle contemporanee teorie psicolinguistiche riguardanti la memoria lessicale umana [4]. La sua struttura richiama, infatti, il funzionamento del pensiero: i termini non sono elencati in ordine alfabetico, ma sono raggruppati in synset (gruppi di sinonimi, cioè di vocaboli aventi un comune significato, nella terminologia di WordNet), tra loro collegati mediante relazioni, sia di tipo semantico sia di tipo lessicale. Ogni termine può possedere vari significati, a seconda dei quali viene raggruppato in uno o più synset. Il synset a sua volta può essere collegato mediante relazioni semantiche ad altri synset, così come il singolo termine può avere legami lessicali con altri termini. In WordNet sono presenti le quattro categorie sintattiche principali: nomi, aggettivi, verbi e avverbi. Alcune relazioni riguardano i soli nomi, altre solo i verbi, e così via.

Originariamente il progetto di Miller e dei suoi colleghi, era quello di elaborare un supporto per la consultazione concettuale dei dizionari comuni, piuttosto che il classico scorrimento in senso alfabetico, che doveva essere utilizzato insieme ad un dizionario online di tipo convenzionale. A mano a mano che il lavoro proseguiva, tuttavia, richiedeva una sempre più ambiziosa formulazione. WordNet è il risultato.

La terminologia di WordNet

Segue ora un riepilogo dei principali termini di WordNet che verranno ampiamente utilizzati nei capitoli a seguire.

- ***Synset***: gruppo di termini appartenenti alla stessa categoria sintattica che condividono il medesimo significato. Ad ogni synset è associato l'insieme dei lemmi che lo compongono e il significato corrispondente.
- ***Lemma***: parola/termine appartenente ad uno o più synset. In caso il lemma sia composto da più termini (in WordNet si trovano infatti alcuni composti di uso molto comune, come ad esempio a*pple_pie*), questi vengono uniti tramite underscore (_). Con “lemma” si fa riferimento alla forma scritta o al suono di una parola.
- ***Significato***: concetto o idea di base associato ad un synset.
- ***Categoria sintattica***: funzione grammaticale ricoperta da una parola. In WordNet sono presenti: nomi, aggettivi, verbi e avverbi. I file in cui sono contenuti i termini sono divisi in base alla categoria sintattica.
- ***Glossa***: definizione associata ad un synset. In genere si tratta di una breve descrizione del

significato.

- ***Relazione semantica***: relazione che lega due diversi synset appartenenti alla stessa categoria sintattica. Si veda dopo per la descrizione delle relazioni semantiche presenti in WordNet.
- ***Relazione sintattica***: relazione che lega due termini (lemmi) appartenenti alla stessa categoria sintattica ma a synset diversi. Si veda dopo per la descrizione delle relazioni lessicali presenti in WordNet.
- ***Forma della parola:*** rappresentazione ortografica di una singola parola o di una stringa composta da parole singole unite da underscore (_) che rappresenta un unico concetto.

La matrice lessicale

Come si è detto, all'interno di WordNet ogni termine può comparire in più di un synset, e in ogni synset si trovano uno o più termini. Inoltre, ad ogni synset è possibile associare vari significati. Questa corrispondenza molti-a-molti dà luogo ad una mappatura tra i significati e le forme delle parole. Assunzione di partenza è che differenti categorie sintattiche di termini hanno differenti tipi di mappatura.

Possiamo pensare alla struttura di un WordNet come ad una matrice avente i termini come colonne e i loro significati come righe. Ogni elemento implica la possibilità di rappresentare il termine di quella colonna con il significato di quella riga. Quindi, l'elemento $E_{1,1}$ implica che il termine F_1 può essere utilizzato per esprimere il significato M_1. Se vi sono più elementi nella stessa colonna, allora la parola è polisemica. Se invece vi sono più elementi nella stessa riga, i termini sono tra loro sinonimi (relativamente a un determinato contesto). Vengono perciò generate le proprietà di:

Sinonimia: proprietà di un concetto/significato di avere due o più termini in grado di esprimerlo. Un gruppo di sinonimi, qui si ribadisce, è detto synset.

Polisemia: caratteristica di uno stesso termine di avere due o più significati.

	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: La matrice lessicale di WordNet

Poiché la nascita di WordNet è da contestualizzarsi nell'apparizione di nuove teorie psico-linguistiche, è interessante notare come gli psicolinguisti usino rappresentare le loro teorie attraverso diagrammi composti da frecce e scatole. In questa notazione, una matrice lessicale può essere vista come due scatole con frecce che vanno da una all'altra in entrambi i versi. Una scatola potrebbe essere etichettata come “Significati” e l'altra come “Forme”. Questa rappresentazione chiarisce la differenza tra i significati (relazioni di significato, nella scatola dei significati) e i termini (relazioni tra i termini, nella scatola delle forme). Nella sua concezione iniziale, in WordNet erano presenti sono relazioni di tipo semantico tra lemmi. Era una rappresentazione della scatola dei significati. È però diventato man mano chiaro come le relazioni lessicali, quelle della scatola delle forme, non potessero essere ignorate. Ad oggi, WordNet distingue tra relazioni semantiche e relazioni lessicali, anche se l'enfasi rimane sulle prime.

Le relazioni semantiche

Poiché una relazione semantica è una relazione tra significati, ovvero, nel caso di WordNet, tra synset, è naturale pensare ad essa come ad un puntatore tra synset. Caratteristica delle relazioni semantiche è che sono reciproche: se esiste una relazione semantica R tra i synset $\{x_1, x_2,... x_n\}$ e $\{y_1, y_2,... y_n\}$, allora vi sarà anche una relazione R' tra $\{y_1, y_2,... y_n\}$ e $\{x_1, x_2,... x_n\}$.

Segue ora una descrizione delle relazioni di tipo semantico che esistono in WordNet.

Iponimia e ipernimia

Un concetto rappresentato dal synset $\{x_1, x_2, \ldots x_n\}$è iponimo del concetto rappresentato dal synset $\{y_1, y_2, \ldots y_n\}$ se una persona di madrelingua inglese accetta come vera l'affermazione che *x è un (is a, kind of) y.*

Per quanto riguarda l'iponimia tra verbi, si parla di *troponimia* (*manner_of*): il verbo X è un troponimo di Y se X esprime un'azione Y fatta in una qualche maniera.

La relazione inversa dell'iponimia è l'ipernimia: si tratta del legame tra un concetto sovraordinato ad un altro, più specifico, che ne eredita le caratteristiche principali ma differenziandosi per qualcosa.

Le relazioni di iponimia/ipernimia sono le più diffuse in WordNet e generano una struttura *is-a,* come ad esempio *house* (casa) *is-a (kind of) dwelling* (abitazione), dove *house* è iponimo di *dwelling*,cioè *house* è uno dei vari tipi di *dwelling*, viceversa *dwelling* è ipernimo di *house*, cioè ne contiene il concetto. Si tratta della relazione gerarchica maggiormente sfruttata negli algoritmi per la disambiguazione di parole composte e per il calcolo delle similarità. Questa tesi si occupa di disambiguazione di parole composte all'interno di documenti strutturati e semi-strutturati: in questo tipo di documenti le relazioni di tipo *is-a* sono spesso rintracciabili e quindi riutilizzabili.

(n) **apple** (fruit with red or yellow or green skin and sweet to tart crisp whitish flesh)

- direct hyponym / full hyponym
  - S: (n) crab apple, crabapple (small sour apple; suitable for preserving) *"crabapples make a tangy jelly"*
  - S: (n) eating apple, dessert apple (an apple used primarily for eating raw without cooking)
  - S: (n) cooking apple (an apple used primarily in cooking for pies and applesauce etc)
- direct hypernym / inherited hypernym / sister term
  - S: (n) edible fruit (edible reproductive body of a seed plant especially one having sweet flesh)
  - S: (n) pome, false fruit (a fleshy fruit (apple or pear or related fruits) having seed chambers and an outer fleshy part)

Figura 2: Estratto da WordNet

Meronimia e olonima

Un concetto rappresentato dal synset $\{x_1, x_2, \ldots x_n\}$ è meronimo di un altro concetto, rappresentato dal synset $\{y_1, y_2, \ldots y_n\}$, se per un madrelingua inglese è vera la frase: y ha un (*has a*) x come parte/

membro/sostanza (*as a part/member/substance*), o anche semplicemente, x è una parte di (*part of*) y. Anche la relazione di meronimia/olonimia, come quella di iponimia/ipernimia, risulta essere particolarmente utilizzata nella realizzazione di gerarchie a partire dai synset di WordNet, per stabilire un tipo di collegamento aggiuntivo che permette di uscire dalla sola gerarchia *is-a* e indagare su legami differenti.

Implicazione (*entailment*)

Si dice che 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*. Si tratta di un tipo di relazione che lega soltanto i verbi, ed è anche conosciuta con il nome di troponimia. Esprime una relazione simile alla meronimia per quanto concerne i nomi, ma con la caratteristica di essere anche relazione di tipo lessicale (tra i singoli verbi), e non esclusivamente semantica (tra synset).

Relazione causale (*cause to*)

La relazione causale riguarda i verbi ed è simile a quella di implicazione, ma non richiede alcun vincolo temporale.

Coordinazione

Non si tratta di un tipo di relazione base, ma derivato. Due synset sono coordinati se condividono uno stesso ipernimo, ovvero sono due differenti specializzazioni dello stesso concetto. Ad esempio, *dog* e *wolf* sono coordinati in quanto condividono l'ipernimo *canine*.

Similarità

È il tipo di relazione alla base dell'organizzazione reciproca degli aggettivi. Sappiamo che tra loro i synset composti da aggettivi possono essere legati da relazioni di antinomia (*hot* e *cold*, *heavy* e *light*): per questo vengono denominati *head synset*, o synset principali. Ad essi sono collegati attraverso la relazione di similarità altri synset, che condividono in modo indiretto le relazioni di antinomia con i synset principali.

Le relazioni lessicali

Sinonimia

La definizione data da Leibniz [4] di sinonimia è la seguente:

"Due espressioni sono sinonime se la sostituzione di una con l'altra non cambia il valore di verità della frase all'interno della quale è avvenuta la sostituzione."

Tuttavia, per quanto la sinonimia rientri a pieno titolo tra le relazioni lessicali, essa non è sempre espressa formalmente in WordNet, per un semplice motivo: ne è a fondamento. Come detto, i termini sono raggruppati in gruppi di sinonimi (synset), per cui la relazione di sinonimia tra le coppie appartenenti ad un synset si evince implicitamente.

Alla luce della definizione precedente, risulta praticamente impossibile trovare due sinonimi veri. Proviamo ad enunciare un'altra definizione, che esprima la sinonima come indipendente dal contesto:

"Due espressioni all'intero di un contesto sono sinonime se la sostituzione di una con l'altra, all'interno di quel contesto, non cambi il valore di verità della frase."

Se ne deduce, ovviamente, che non vi possano essere sinonimi tra categorie sintattiche diverse.

Antinomia

L'antinomia è una relazione lessicale che ad una parola associa il suo contrario. A livello di logica booleana, l'antinomo di x è -x. Questo non è sempre valido nel campo della linguistica. Consideriamo l'esempio di *hot*: *hot* e *cold* sono antinomi, ma dire che un determinato oggetto non sia *hot* [-*hot*], non implica automaticamente che questo sia *cold*. Molte cose presentano una temperatura intermedia che non si può definire né calda né fredda.

Attributo

La relazione di attributo è quella che lega un nome all'aggettivo (di tipo descrittivo) che ne descrive un valore. In WordNet i synset di nomi sono collegati tramite puntatori agli aggettivi che ne conferiscono un valore.

Vedi anche (*see also*)

La relazione nota come *see also* è una relazione lessicale che lega singoli lemmi di synset differenti. I motivi di tale relazione possono essere molto differenti fra loro.

Pertinenza

Riguarda gli aggettivi relazionali. Un aggettivo relazionale è un aggettivo che deriva la sua forma direttamente da un nome, anche se può variare leggermente nell'ortografia e ha uno scopo che può essere riassunto in una espressione del tipo: *associato con*, *pertinente a* o *di* in riferimento al nome da cui discende.

1.2 Calcolo della similarità

Nel contesto dell'interpretazione dei composti, una relazione semantica è la relazione che intercorre tra l'*head* e il *modifier*. Tipicamente, le applicazioni computazionali, come l'interpretazione automatica di parole composte, di cui questa tesi si occupa, richiedono l'esistenza di una relazione semantica, più che di una semplice similarità. Pertanto, non è possibile prescindere dallo studio delle relazioni che intercorrono tra i termini (e, più in generale, tra i concetti) all'interno di una tassonomia.

Il database lessicale WordNet si dimostra estremamente adatto per effettuare le misure sulla similarità, poiché organizza i nomi e i verbi in gerarchie basate su relazioni *is-a*, che però non superano i confini delle parti del discorso, cosicché le misure basate su WordNet sono limitate a giudizi tra coppie di nomi e coppie di verbi. Gli aggettivi e gli avverbi invece non vengono considerati in quanto non organizzati in una gerarchia del tipo *is-a*. Comunque, WordNet fornisce molte informazioni aggiuntive non gerarchiche, e, in più, ogni concetto (o senso di una parola) è descritto da una piccola glossa o definizione. Le misure di relazione sono basate su questo genere di informazioni aggiuntive, così da poter essere applicate ad un più ampio range di coppie di concetti.

Un modo semplice di calcolare similarità e, più in generale, di calcolare relazioni semantiche in una tassonomia qual è WordNet è di considerarlo come un grafo e identificare la relazione con la lunghezza del percorso tra i concetti: "*The shorter the path from one node to another, the more similar they are*" [7]. Più due nodi sono vicini, più sono simili, e hanno qualcosa in comune. Questo approccio è stato preso, per esempio, da Rada e dai suoi colleghi [6] non applicato a WordNet ma a

MeSH (Medical Subject headings), una gerarchia semantica di termini usata per indicizzare gli articoli nel sistema di recupero bibliografico Medline. I 15000 termini della rete MeSH formano una gerarchia su nove livelli basati sulla relazione BROADER-THAN. Al di là della semplicità della loro intuizione, gli autori sono stati in grado di ottenere risultati sorprendentemente buoni. Ovviamente, il loro successo si spiega in parte con l'osservazione di Lee, Kim and Lee [8], i quali osservano che, nel contesto delle reti semantiche, i percorsi più brevi tra due concetti non sono sufficienti a rappresentare la distanza semantica tra quei concetti, ma quando i percorsi sono ristretti ad una gerarchia del tipo *is-a*, allora la minore lunghezza del percorso misura la distanza concettuale. Altro motivo del loro risultato è certamente la specificità del dominio di applicazione, che assicura l'omogeneità del grafo.

1.2.1 Le misure di similarità

WordNet::Similarity (http://wn-similarity.sourceforge.net/) è un pacchetto software sviluppato nell'università del Minnesota e gratuitamente disponibile che rende possibile la misura delle similarità e delle relazioni tra coppie di concetti e, per estensione, tra coppie di parole [9, 10]. Esso fornisce sei misure di similarità e tre misure di relazione, tutte ovviamente basate sulle gerarchie *is-a* del database WordNet. Questo software è implementato attraverso moduli Perl che prendono in input due concetti, e che ritornano un valore numerico che rappresenta il grado in cui questi concetti sono simili o in relazione. Le misure della similarità presenti in *WordNet::Similarity* sono in tutto sei e sono così suddivise in base all'impostazione seguita: tre sono basate sulla lunghezza del percorso tra i concetti (*path-based measures*) e tre sul contenuto informativo (*information content-based measures*), che è una misura della specificità di un concetto basata su corpus.

Le misure basate sul percorso

- *lch* (Leacock & Chodorow 1998): trova il percorso più breve tra due concetti *len(c1,c2)* e scala (accorgimento dovuto al fatto che in una tassonomia i vari collegamenti non rappresentano distanze uniformi) tale valore della lunghezza di percorso massima nella gerarchia *is-a* (cioè la profondità D della tassonomia) in cui i concetti compaiono. La forza della relazione è data da:

$$similarity_{LC} = -\log[\frac{len(c_{1,}c_2)}{2D}]$$

- *wup* (Wu & Palmer 1994): trova la lunghezza del percorso al nodo d'origine (root) a partire dal concetto più specifico che i termini in input c_1 e c_2 hanno in comune (ossia il minimo comune sovraordinato);
- *path*: la misura ottenuta è uguale all'inverso della più breve lunghezza di percorso tra due concetti.

Le misure basate sul contenuto informativo

- *res* (Resnik 1995): l'approccio di Resnik [11] è stato il primo tra quelli riconosciuti a considerare insieme l'ontologia e il corpus. Guidato dall'intuizione che la similarità tra una coppia di concetti può essere giudicata come "il livello al quale essi condividono informazioni", Resnik definisce la similarità tra due concetti i cui lemmi siano presenti in WordNet come il contenuto informativo del loro minimo comune sovraordinato (least common subsumer, lso) *lso(c1,c2)*:

$$similarity_R(c_{1,}c_2) = -\log p(lso(c_1, c_2))$$

dove *p(c)* è la probabilità di incontrare un'istanza di un synset (un concetto) *c* in un determinato corpus. Il contenuto informativo *IC(c)* di c è dunque definito come -log*p(c)*. Va notato che *p* è monotonica man mano che si sale lungo la tassonomia: se c1 is-a c2 allora $p(c2) \geq p(c1)$. Ad esempio, qualora scendendo nella tassonomia incontrassimo *nickel*, di certo abbiamo incontrato *coin*, quindi $p(coin) \geq p(nickel)$.

- *lin* (Lin 1998): aumenta della somma dei contenuti informativi individuali il contenuto informativo del minimo comune sovraordinato dei due concetti. Questa misura segue direttamente dalla teoria sulla similarità di Lin [13] tra oggetti arbitrari:

$$similarity_L(c_1, c_2) = \frac{[2\log p(lso(c_1, c_2))]}{[\log p(c_1) + \log p(c_2)]}$$

- *jcn* (Jiang & Conrath 1997): è simile alla misura *lin* con la differenza che sottrae il contenuto informativo del minimo comune sovraordinato alla somma dei contenuti informativi individuali, e quindi ne prende l'inverso per convertirlo in una misura di similarità partendo da una distanza. In altre parole, rappresenta la probabilità condizionata di incontrare un'istanza di un synset figlio data un'istanza del synset padre:

$$dist_{JC}(c_1, c_2) = 2\log(p(lso(c_1, c_2))) - (\log(p(c_1)) + \log(p(c_2)))$$

1.2.2 Le misure di relazione

Le misure della relazione supportate da *WordNet::Similarity* sono:

- *hso* (Hirst & St.Onge 1998): è una misura basata sul percorso e classifica le relazioni di WordNet come dotate di direzione [12]. Ad esempio, le relazioni *is-a* sono dirette verso l'alto (cioè verso la radice), mentre le relazioni *has-part* sono orizzontali. Stabilisce la relazione tra due concetti cercando di trovare un percorso tra essi che non sia né troppo lungo né che cambi direzione troppo spesso:

$$rel_{HS}(c_{1,} c_2) = -\log[\frac{len(c_{1,} c_2)}{2\mathrm{D}}]$$

- *lesk* (Banerjee & Pedersen 2003): usa il testo della glossa (la corta definizione fornita da WordNet) come unica rappresentazione per il concetto che sta dietro alla parola. La misura è assegnata trovando le intersezioni tra le glosse dei concetti considerati così come di quelli che sono direttamente connessi ad essi ed assegnando dei punteggi a tali intersezioni,
- *vector* (Patwardhan 2003): come *Lesk* usa il testo della glossa, ma crea una matrice di co-occorrenza a partire da un corpus con le glosse di WordNet. Ogni parola dotata di contenuto presente nella glossa è associata ad un vettore di contesto, un *context vector*. Ogni glossa è a sua volta rappresentata da un *gloss vector*, che è la media di tutti i *context vector* delle parole trovate in quella glossa. La misura della relazione tra i concetti è misurata attraverso il coseno dell'angolo formato da una coppa di *gloss vector*.

Nella letteratura, si sono dimostrati prevalenti tre tipi di approcci alla valutazione delle misure di similarità:

- il primo tipo [13] è un esame teorico di una data misura per le proprietà ricercate. La misura non è definita direttamente da una formula, ma è derivata da un set di assunzioni riguardo la similarità. In altre parole, se l'assunzione è ritenuta ragionevole, la misura di similarità segue necessariamente. Tuttavia, questo genere di analisi può andare bene per valutare l'efficacia di una misura o per comparare un gruppo di misure, ma non può essere nulla più che un filtro “grezzo”;
- il secondo approccio consiste nel confronto con il giudizio umano. Poiché questo viene considerato giusto per definizione, è il miglior test per la bontà di una determinata misurazione. Tuttavia, anche questo approccio presenta problemi, in quanto si cerca di andare incontro a qualcosa che sia completamente oggettivo e il più possibile automatico;
- la terza impostazione, invece, consta nella valutazione delle misure rispetto alla loro prestazione all'interno di una particolare applicazione NLP.

Capitolo 2

2 Metodi di risoluzione per termini composti: stato dell'arte

La disambiguazione di parole composte è un campo di studio della linguistica che risale agli anni '70 e '80, quindi a ben prima che fossero disponibili i mezzi informatici moderni. I primi a cimentarsi in questo ambito furono degli studiosi di linguistica e di matematica, e non, come si sarebbe portati a pensare, di informatica. Sono infatti le formule matematiche ottenute da questi studiosi ad essere alla base del calcolo della similarità e della relazione tra termini, i due cardini su cui si fondano la classificazione e la disambiguazione delle parole composte.

Quelli che vengono presentati di seguito sono soltanto alcuni degli innumerevoli algoritmi e metodi proposti. Si è focalizzata l'attenzione in particolare su quelli che sono di interesse per la ricerca cui questa tesi è rivolta.

Innanzitutto, come già precedentemente detto, si ritiene fondamentale suddividere gli algoritmi proposti in due grandi gruppi: quelli che si fondano su calcoli statistici effettuati su corpora (raccolte di testi, articoli, etc.) e quelli che analizzano soltanto il contenuto semantico e grammaticale dei termini, facendo affidamento su database lessicali.

2.1 Metodi statistici basati su corpora

I metodi statistici per la disambiguazione di parole composte che fanno affidamento sull'esistenza di grandi corpora sono storicamente sempre stati i più gettonati. I termini tecnici all'interno di testi appaiono spesso nella forma di parole composte, soprattutto per quanto riguarda la lingua inglese. Si tratta di una costruzione tanto frequente quanto ambigua la cui interpretazione fa affidamento su informazioni extra-sintattiche. Numerosi metodi statistici per disambiguare parole composte sono riportati in letteratura, spesso con ottimi risultati. Tuttavia, una significativa caratteristica di tutti questi approcci è che essi utilizzano composti precedentemente analizzati e disambiguati, risultando perciò esposti al problema dei dati sparsi avulsi da un contesto. Questa difficoltà è stata in qualche modo affrontata attraverso l'uso di risorse di conoscenza aggiunte a mano per raccogliere statistiche sui concetti, più che sui termini, ma l'indipendenza dal dominio è di conseguenza risultata sacrificata.

2.1.1 Il modello dell'adiacenza e il modello della dipendenza

Il primo problema da affrontare nella disambiguazione di termini composti è quello del parsing o dell'analisi sintattica, ossia l'analisi di un flusso continuo in input (letto per esempio da un file o una tastiera) in modo da determinare la sua struttura grammaticale grazie ad una data grammatica formale. Un *parser* è un programma che esegue questo compito. Una volta estratta una sequenza di termini distinti che possono formare una parola composta, si può procedere con il passo successivo, cioè il cosiddetto *bracketing*, che consiste nell'individuare i sotto-composti all'interno della sequenza in modo ricorsivo, fino a dedurre quali parole costituiscano l'*head* e quali il *modifier*. Appare evidente come, una volta trovato un metodo per analizzare le sequenze di N termini, è possibile analizzare anche i semplici e diffusi composti di soli due termini.

Nel 1980, Marcus [16] propone una procedura analitica che Lauer [14, 15] chiama *adjacency model*, o modello dell'adiacenza. La procedura si basa sulla determinazione dell'accettabilità di un nome composto.

Dati tre nomi consecutivi n_1, n_2 e n_3:

- se né [n_1 n_2] né [n_2 n_3] è semanticamente accettabile, allora vai alla struttura alternativa;
- se [n_2 n_3] è semanticamente più accettabile di [n_1 n_2], allora considera [n_2 n_3];
- altrimenti, considera [n_1 n_2].

Rimane tuttavia il problema di definire “l'accettabilità” di un composto piuttosto che di un altro. Solo con il tempo ci si è resi conto che ciò che poteva determinare l'accettabilità erano le statistiche applicate ad un corpus, e questa idea è stata alla base degli algoritmi a seguire.

Uno dei più semplici è stato proposto da Pustejovsky et al. [17]: dato un composto di tre parole, si conduce una ricerca all'interno del corpus per ognuno dei due possibili sotto-componenti. Quello tra i due che viene trovato è quindi scelto come coppia da porre tra parentesi. Per chiarire, si pensi all'esempio di *backup compiler disk*. La sua analisi sarà, seguendo il modello dell'adiacenza:

- [$backup_N$ [$compiler_N$ $disk_N$]], quando *compiler disk* è apparso altrove nel corpus;
- [[$backup_N$ $compiler_N$] $disk_N$], se *backup compiler* è apparso altrove.

Tuttavia, non vi è mai stata una valutazione di questo algoritmo con esperimenti pratici.

La proposta di Liberman e Sproat [18] è invece più sofisticata e considera già il concetto di

frequenza delle parole nel composto. La loro idea consta nel comparare la mutua informazione tra le due coppie di parole adiacenti e nell'unire tra parentesi la coppia che registra il valore più alto. Tuttavia, anche in questo caso non vi è stata applicazione pratica a parte la dimostrazione che il metodo ha funzionato correttamente su quattro esempi.

La terza proposta basata sul modello dell'adiacenza arriva da Resnik [19], ed è ancora più complessa. Resnik fa riferimento alla *Selectional Association* (SA), o Associazione Selezionale (AS), tra un predicato e una parola, e la definisce come il contributo della parola alla probabilità condizionata del predicato. In statistica, per probabilità condizionata si intende l'incertezza residua di una variabile casuale *Y*, posto che il valore di una seconda variabile casuale *X* sia noto, e si indica con *P(Y|X)*. Allo stesso modo, quindi, la probabilità condizionata del predicato è data dal grado di incertezza in esso rimasto, alla luce del contributo apportato dal valore assegnato alla parola. L'associazione tra ogni coppia di parole nel composto è, pertanto, calcolata prendendo il massimo valore di SA tra tutti i possibili modi di guardare la coppia come predicato e argomento. Comunque, sono due le cose veramente originali nel lavoro di Resnik:

1) Insieme a Hearst, ha introdotto la nozione di *Conceptual Association* (CA), o Associazione Concettuale (AC), nella quale le associazioni sono misurate tra gruppi di parole che si considera rappresentare dei concetti, in contrasto con le singole parole. Tale approccio, basato su classi, è utilizzato in quanto permette di generalizzare ogni osservazione, riducendo in questo modo la quantità di dati richiesti alla disambiguazione;
2) viene utilizzato un thesaurus online, in questo caso WordNet (Lauer farà invece uso del Roget's Thesaurus), che permette di utilizzare come concetti i sensi (synset) in cui divide i vari termini.

L'applicazione di tali novità introdotte appare in Lauer [5]. Lauer definisce la $CA(t_1,t_2)$ come la mutua informazione tra ogni coppia ordinata (t_1,t_2) di categorie nel thesaurus, pesata per evitare ambiguità. Essa misura il grado al quale la categoria modificante predice la categoria modificata, e viceversa. Siano quindi:

AMBIG(w) = il numero di categorie del thesaurus in cui la parola w compare (cioè l'ambiguità di w);

$COUNT(w_1,w_2)$ = il numero di istanze di w1 modificante w2 nel training set (cioè tra i composti già disambiguati);

$$FREQ(t_1, t_2) = \sum_{w_1 \in t_1} \sum_{w_2 \in t_2} \frac{COUNT(w_1, w_2)}{AMBIG(w_1) * AMBIG(w_2)}$$

Pertanto si ottiene:

$$CA(t_1, t_2) = \frac{FREQ(t_1, t_2)}{\sum_i FREQ(t_1, i) * \sum_i FREQ(i, t_2)}$$

con i che spazia tra tutte le categorie del thesaurus.

È da notare che la misura di $CA(t_1,t_2)$ è asimmetrica: rappresenta la tendenza di t_1 a modificare t_2 all'interno di un nome composto, ed è diverso da $CA(t_2,t_1)$, che è la misura della tendenza di t_2 a modificare t_1.

Lauer utilizza la CA per effettuare il bracketing dei termini all'interno di un composto, e quindi come discriminante per stabilire il grado di accettabilità di una struttura $[n_1\ n_2]$, quando n_1 è un nome che appare in t_1 e n_2 in t_2. Possiamo formalmente rappresentare questo parametro come $Pr(t_1 \rightarrow t_2)$, dove l'evento $t_1 \rightarrow t_2$ denota la modifica di un nome in t_2 da parte di un nome in t_1.

È ora necessario introdurre la variante al modello dell'adiacenza di Marcus, che è quello che Lauer ha definito *dependency model*, o modello delle dipendenza.

Data una sequenza di tre nomi n_1, n_2 e n_3:

- Determinazione dell'accettabilità delle strutture $[n_1\ n_2]$ e $[n_1\ n_3]$;
- se la seconda è più accettabile, allora costruisci prima la $_{\text{coppia}}$ $[n_2\ n_3]$;
- altrimenti, costruisci prima $[n_1\ n_2]$.

Andando a riprendere l'esempio fatto per il modello dell'adiacenza, si ottiene:

- $[backup_N\ [compiler_N\ disk_N]$, quando backup *disk* è più accettabile;
- $[[backup_N\ compiler_N]\ disk_N]$, se *backup compiler* è più accettabile.

Il grado di accettabilità è ancora una volta basato su misure statistiche applicate ad un corpus. L'idea è che l'albero ottenuto a partire dal parsing catturi la struttura delle relazioni semantiche all'interno di un composto.

Su questa base, si consideri l'esempio *pottery coffee mug*. Quando

CA(*pottery,mug*)>>CA(*pottery,coffee*), allora il bracketing (*pottery*(*coffee mug*)) è preferibile. Per ogni ni (i=2 o i=3), si devono scegliere le categorie S_i (con n_1 in S_i) e Ti (con w_i in T_i) in modo che CA(Si,Ti) sia maggiore. Queste categorie rappresentano i più significativi tra i possibili significati delle parole per ogni possibile coppia. È stata così scelta la coppia avente il valore più alto di CA in termini di mutua informazione tra le categorie del thesaurus a cui le parole appartengono.

In composti aventi più di tre parole, questa procedura può essere generalizzata scegliendo, tra tutti i possibili bracketing, quello per il quale il prodotto delle maggiori CA è massimizzato.

2.1.2 Il Latent Semantic Indexing

Buckeridge e Sutcliffe [20] muovono una critica ai metodi di Resnik e Lauer: i loro approcci fanno affidamento su variazioni dei metodi che cercano sotto-componenti dei composti in altri punti dei corpora, e li usano per decidere come sono poi strutturati i composti più lunghi e ambigui. Tuttavia, c'è sempre la possibilità che tali sistemi incontrino coppie di *modifier-head* nella fase di testing (cioè quando si disambiguano parole composte nuove) che non sono mai occorse nella fase di training, essendo così forzati a ricorrere a qualche strategia di default. Questo problema è in qualche modo alleviato, nei lavori di Resnik e Lauer, dal fatto che le statistiche vengono raccolte su concetti piuttosto che su termini singoli. Il difetto rimane però che queste tecniche sono basate su fonti aggiunte a mano, e l'applicabilità dei loro approcci è pertanto limitata dalla copertura di tali risorse. Per questo avrebbero prestazioni certamente inferiori quando applicati a domini tecnici nei quali buona parte del vocabolario utilizzato non è disponibile né in WordNet né nel Roget's Thesaurus. Fonti di conoscenza come queste andrebbero aggiornate manualmente ogni volta che il sistema è applicato ad un nuovo dominio. Sarebbe quindi preferibile avere un metodo per misurare la CA che sia meno dominio-dipendente e che non faccia alcun affidamento sulla presenza di sotto-componenti già disambiguati nel training.

L'applicazione del *Latent Semantic Indexing* (LSI, a volte chiamato anche *Latent Semantic Analysis*, LSA) proposta da Buckeridge e Sutcliffe costituisce un modo per misurare l'accettabilità del bracketing.

Ma cos'è il LSI? Si tratta di un moderno metodo di retrieval, brevettato nel 1988 da Deerwester et al. [32], che cerca di “estrarre i concetti” presenti nei documenti, e quindi di superare il limite proprio di ogni tecnica che si basa sulla presenza o meno di termini, per stabilire la rilevanza di un

documento rispetto ad una query.

Partendo da una rappresentazione vettoriale dei documenti, in cui ogni coordinata corrisponde a un termine, il LSI cerca di "proiettare" i vettori dei documenti in un "sotto-spazio semantico latente" a dimensionalità ridotta, in cui le coordinate sono i "concetti". Intuitivamente, un concetto può essere visto come un insieme di termini che occorrono (frequentemente) insieme negli stessi documenti. Di fatto, il LSI opera un *clustering* (il *clustering* o analisi dei *cluster* o analisi di raggruppamento è un insieme di tecniche di analisi dei dati volte alla selezione e raggruppamento di elementi omogenei in un insieme di dati. Tutte le tecniche di *clustering* si basano sul concetto di distanza tra due elementi) dei termini (e dei documenti).

Ma come si ottiene la rappresentazione vettoriale dei documenti? La tecnica del LSI prende in input una raccolta di documenti a partire dai quali si costruisce una matrice m x n A, dove ogni colonna rappresenta un documento e ogni riga una parola. La casella a_{ij} della matrice denota la frequenza con la quale il termine i occorre all'interno del documento j.

Alla base del LSI vi è una tecnica di algebra lineare, il *Singular Value Decomposition* (SVD), che funziona nel seguente modo:

1. Si consideri la matrice A dei pesi con m righe ed n colonne, dove m = numero dei termini, n = numero dei documenti e si supponga che A abbia rango r = min{m,n};
2. La matrice A può essere decomposta in modo univoco nel prodotto di tre matrici:

$$A = U \times \Sigma \times V^T$$

   dove:

   T indica la trasposta;

   Σ è la matrice diagonale dei concetti r x r, contenente i valori singolari di A in ordine discendente;

   U è la matrice m x r di similarità termini-concetti. È una matrice le cui r colonne sono orto-normali ($U^T \times U = I$);

   V è la matrice n x r di similarità documenti-concetti. Anche le sue r colonne sono orto- normali;
3. Conservando soltanto i k maggiori valori singolari e settando a zero i rimanenti, una nuova matrice diagonale Σ_k è ottenuta;
4. A questo punto il prodotto di $U \times \Sigma_k \times V^T$ è la matrice m x n A_k che è solo approssimativamente uguale ad A. Questo SVD troncato re-rappresenta le relazioni termine-

documento in A utilizzando solo gli assi di maggiore variazione, comprimendo e rendendo omogenei i dati in A. Questa fase di compressione cattura le regolarità fondamentali nel modello della co-occorrenza delle parole, ignorando le variazioni meno significative che potrebbero essere dovute a idiosincrasie nell'uso dei termini in singoli documenti.

Il risultato, ottenuto attraverso la riduzione della matrice così ottenuta, è che le parole che ricorrono in documenti simili sono rappresentate da vettori simili, anche se queste parole non occorrono mai insieme nello stesso documento. Ogni vettore può così essere interpretato come un "sunto" dell'uso contestuale di un termine. Le parole sono simili al livello in cui occorrono in documenti simili.
La cosa significativa è che può in questo modo essere calcolata geometricamente una misura della similarità o associazione tra coppie di parole (cioè l'accettabilità), tipicamente calcolando il coseno dell'angolo tra i vettori che rappresentano le parole. Questo libera dalla costrizione di avere sotto-componenti precedentemente disambiguati nella fase di training.

Un'altra applicazione molto interessante del LSI è stata fatta da Baldwin *et al.* [21]. Viene utilizzato il LSI per determinare la similarità tra le MWE e i loro sotto-componenti, con l'idea che un'alta similarità indica una scomponibilità maggiore. La scomponibilità viene qui definita come il grado al quale è possibile "*ascrivere la semantica di una MWE a quella dei suoi componenti*", ovvero dedurre il significato della MWE a partire da quello dei suoi componenti. Uno dei punti focali di questa applicazione si trova nel oggetto cui è rivolta: le MWE, e quindi non solo le parole composte più semplici o addirittura i soli nomi composti. Vengono distinti tre tipi di MWE (in base alla sotto-classificazione degli idiomi di Nunberg et al. [22]), in base al grado di scomponibilità:

- non scomponibili: non è possibile alcuna analisi riguardo la scomponibilità, la MWE è semanticamente "impenetrabile". Esempi: *shoot the breeze*, *hot dog*, etc. Le uniche variazioni sintattiche che possono subire sono l'inflessione verbale (*shoots the breeze*, ...) e la riflessività pronominale (*wet oneself*, *wet themselves*, ...);
- idiosincraticamente scomponibili: sono scomponibili ma i loro componenti, presi singolarmente, non hanno lo stesso significato. Ad esempio: *spill the beans*, *radar footprint*, etc. Possono subire limitate variazioni sintattiche;
- semplicemente scomponibili: sono le MWE "istituzionali", cioè le parole composte di cui ci si occupa in tutta questa tesi. Ad esempio *traffic light*, *kindle excitement*, etc. Presentano

un'alta variabilità sintattica e i loro sotto-componenti hanno significati semplici anche considerati singolarmente.

Altro tipo di classificazione delle MWE largamente utilizzato in letteratura è il seguente:

- MWE endocentriche: sono un iponimo dell'*head*. Corrispondono alle MWE semplicemente scomponibili;
- MWE esocentriche: non costituiscono un iponimo dell'*head*. Corrispondono alle MWE idiosincraticamente scomponibili e non scomponibili.

Anche in questo caso, il LSI è utilizzato per rappresentare le parole all'interno di uno spazio vettoriale e ottenere un vettore a partire da ogni termine incontrato, distinguendo il caso in cui si tratta di un verbo, di un nome o di un aggettivo (si ricorda che in inglese l'omografia è frequentissima).

Vengono poi estratte le parole composte costituite da:

- nome + nome;
- verbo + particella (verbi fraseologici).

Nella fase successiva, viene utilizzato WordNet per valutare:

1) l'iponimia;
2) la distanza semantica.

1) Si tratta del metodo più rapido per valutare la scomponibilità dei termini. Nel caso delle MWE semplicemente scomponibili i sotto-componenti saranno presumibilmente iponimi o sinonimi della MWE (salvo alcune eccezioni). Non esiste invece relazione di iponimia per gli altri due tipi di MWE presentati.
2) Viene derivata una distanza semantica basata sull'analisi delle relative posizioni dei sensi o dei concetti cui appartengono i termini all'interno di WordNet. I metodi per valutare la distanza semantica o similarità sono quelli esposti in precedenza. Nel metodo qui descritto, viene utilizzata l'implementazione di Patwardhan (2003), ma anche le altre misure potrebbero essere applicate.

Per confrontare le similarità basate sul LSI con quelle basate su WordNet, vengono fatte una regressione lineare e un'analisi di correlazione tra le coppie di dati.

Tuttavia, come risultato si è registrato che il LSI non è in grado di riprodurre i valori delle

similarità derivati da WordNet, sia nel caso di composti nome+nome che di verbi fraseologici.

2.1.3 L'algoritmo di Su, Wu, Chang per l'estrazione di composti

Su, Wu e Chang [23] hanno proposto un metodo di estrazione automatica dei composti basato su corpora.

L'algoritmo è così sviluppato:

- Normalizzazione di ogni parola del corpus nella sua forma base attraverso un analizzatore morfologico;
- estrazione dei composti candidati (sequenze di 2 o 3 parole);
- scansione del corpus da destra e da sinistra utilizzando finestre di dimensione 2 e 3;
- divisione dei composti candidati in:
  - composti;
  - non composti;

Utilizzando una frazione di probabilità:

$$\lambda = \frac{P(\vec{x}|M_c) * P(M_c)}{P(\vec{x}|M_{nc}) * P(M_{nc})}$$

dove:

M_c: evento che la sequenza sia un composto;

M_{nc}: evento alternativo che la sequenza non sia un composto;

x: osservazione associata alla sequenza effettuata in base a tre fattori, in ordine:

1) Mutua informazione: misura dell'associazione tra termini. È la probabilità di un gruppo di parole di trovarsi insieme comparata a quella di trovarsi separate:

$$I(x;y) \equiv \log_2 \frac{P(x,y)}{P(x) * P(y)}$$

$$I(x;y;z) \equiv \log_2 \frac{P_D(x,y,z)}{P_I(x,y,z)}$$

2) Frequenza relativa della i-esima sequenza:

$$r_i = \frac{f_i}{k}$$

dove:

f_i: numero totale di occorrenze della i-esima sequenza nel corpus;

K: numero medio di occorrenze dei singoli termini.

3) POS (*part-of-speech*): vengono eliminati i composti che non sono strutturati con nome + nome, aggettivo + nome, oppure, nel caso di sequenze di tre termini, nome + nome + nome, nome + preposizione + nome, aggettivo + nome + nome.

2.2 Metodi basati sull'informazione semantica

Con il passare del tempo e lo sviluppo delle risorse a disposizione, la ricerca nella disambiguazione di parole composte ha iniziato sempre più a rivolgersi verso approcci basati sulla semantica. Sostanzialmente, si tratta di tecniche che esplorano il tipo di relazione che intercorre tra i costituenti di una parola composta. La novità è che si è provato a fare questo senza il supporto di conoscenza precedentemente ottenuta, né usando informazioni pre-codificate riguardanti la semantica di nomi o aggettivi. Inoltre, e anche questo costituisce un'importante novità, la maggior parte dei metodi impiega ricche ontologie e ignora il contesto di utilizzo, rendendo possibile così l'applicazione di questi algoritmi a qualsiasi campo. In questo senso, è stato anche possibile ridurre la quantità di conoscenza necessaria per l'interpretazione dei composti: Fan et al. (2003) sostengono infatti che sia sufficiente acquisire a priori solo poche informazioni di base riguardo i singoli componenti, in particolare bastano i concetti ad essi direttamente associati (le distinzioni ontologiche) e gli assiomi riferiti ad un concetto, cioè asserzioni riguardo le relazioni tra quel determinato concetto e gli altri concetti vicini.

2.2.1 L'algoritmo di Vanderwende

Già nel 1993 Lucy Vanderwende [24], ricercatrice Microsoft, realizza un algoritmo per la disambiguazione di nomi composti basato sulla manipolazione di un set di regole generali, ad

ognuna delle quali è stato assegnato un peso, e di una procedura per trovare la relazione che intercorre tra i membri del nome composto. Il risultato è un elenco ordinato di interpretazioni e disambiguazioni parziali di senso dei nomi, tenendo nota di quali significati sono più rilevanti in ciascuna delle possibili interpretazioni. Lo studio è rivolto alle sequenze di due nomi, ma nel caso di più elementi è sufficiente interpretare le coppie a turno finché la migliore interpretazione non va a formare il *modifier* o l'*head*. Il solo tipo di informazioni acquisite a priori provengono dall'analisi delle definizioni estratte da un dizionario online.

La tabella 2 mostra una classificazione schematica che ricapitola la maggior parte delle classi cui appartengono i nomi composti che sono stati studiati nelle precedenti ricerche di linguistica. Alla relazione che intercorre tra *head* e *modifier* viene generalmente dato un nome convenzionale, come *Purpose* (scopo) o *Location* (luogo). Lo schema secondo cui sono stati classificati i composti in questo sistema è stato formulato attraverso un set di *wh-questions*, partendo dall'ipotesi che una sequenza di nomi possa essere correttamente interpretata e classificata in base alla *wh-question* a cui il *modifier* (che per l'inglese è il primo nome) meglio risponde.

L'algoritmo è cosi strutturato: vengono dapprima applicate delle regole generali, che possono essere considerate come una descrizione della configurazione degli attributi semantici e sintattici che portano ad una particolare interpretazione (e quindi classificazione) dei composti. Tipicamente accade che più di una combinazione di attributi semantici identifichi un composto come appartenente ad una determinata classe. Per risolvere le ambiguità, innanzitutto le regole sono state

divise tra quelle che forniscono attributi per l'*head* e quelle che ne forniscono al *modifier*. In secondo luogo, ad ogni regola è stato associato un peso.

Viene eseguita una procedura generale di accoppiamento che ritorna un peso che riflette quanto vicine sono le due parole del composto, in questo caso quanto il valore di un attributo è relazionato a quello di un dato termine. Il peso così ottenuto è aggiunto a quello della regola per arrivare infine al punteggio associato all'applicazione della regola nel suo insieme.

2.2.2 Il riconoscimento di relazioni di Barker e Szpakowicz

L'algoritmo appena visto appare essere molto interessante nell'ottica dell'analisi dello stato dell'arte. Tuttavia, in fase applicativa, risulta abbastanza macchinoso.

Un approccio di tipo diverso viene proposto da Barker e Szpakowicz [25]. La loro idea è quella di trovare delle costruzioni che siano simili a un dato composto da disambiguare, per le quali le relazioni siano già state trovare. Questo è reso possibile per mezzo di un sistema semi-automatico che identifica le relazioni senza utilizzare alcun tipo di conoscenza precedente riguardo la semantica di nomi e aggettivi. Viene anzi effettuata una procedura di accoppiamento parziale su sequenze precedentemente disambiguate che porta al tentativo di interpretazione di un nuovo input.

Tuttavia, quello che vale la pena evidenziare della ricerca di Barker e Szpakowicz non è tanto l'algoritmo in sé, che parte da un'idea di base che in futuro troverà altre applicazioni, quanto piuttosto l'insieme di relazioni *modifier-head* che essi hanno redatto e che sarà preso a riferimento (tal quale oppure con piccole modifiche) da moltissimi ricerche successive. Ecco la lista delle venti relazioni utilizzate:

Agent (agt)	Material (matr)
Beneficiary (benef)	Object (obj)
Cause (caus)	Possessor (poss)
Container (cnt)	Product (prod)
Content (cont)	Property (prop)
Destination (dest)	Purpose (purp)
Equative (equa)	Result (resu)
Instrument (inst)	Source (src)
Located (led)	Time (time)
Location (loc)	Topic (top)

Tabella 1: Le 20 relazioni proposte da Barker e Szpakowicz

Prima che vengano assegnate delle relazioni *noun-modifier* il sistema deve riconoscere quali parole compongono effettivamente l'*head* e quali il *modifier*, riducendo la sequenza a coppie di termini da analizzare ricorsivamente. Si consideri l'esempio: *dynamic high impedance microphone*. Attraverso la fase di bracketing, si ottiene: *(dynamic ((high impedance) microphone))*. Le sottofrasi risultanti consistono di un *head* (che può essere un composto a sua volta) e di un *modifier* (che pure può essere un composto) e sono: *high impedance*, *high_impedance microphone* e *dynamic high_impedance_microphone*.

Una volta che è stata assegnata la relazione, il sistema deve annotare la risoluzione per poterne usufruire in futuro. Invece di memorizzare le intere sequenze e le relative analisi, i *modifier* e gli *head* composti vengono ridotti ai loro *head* locali (ossia viene memorizzato solo l'*head* del sotto-composto (nell'esempio, la coppia ridotta a partire da *dynamic high impedance microphone* è *dynamic microphone.* Se *dynamic high impedance microphone* è già stato analizzato, allora questo aiuterà a risolvere, per esempio, il nuovo composto *(dynamic (cardioid (vocal microphone))).*

2.2.3 L'utilizzo di una gerarchia: l'esempio di MeSH

Alcuni ricercatori di Berkeley hanno esplorato la possibilità di utilizzare una gerarchia lessicale allo scopo di categorizzare i termini partendo da parole composte, e quindi usare questa appartenenza ad una categoria per determinare la relazione che intercorre tra le parole del composto. Sono stati presi in considerazioni nomi composti provenienti dal dominio biomedico prelevati dal corpus Medline

con il supporto della gerarchia lessicale MeSH (*Medical Subject headings*), in lingua inglese [26]. Sorprendentemente, è stato trovato che è sufficiente ricorrere alla giustapposizione delle appartenenze a categorie entro la gerarchia lessicale per determinare la relazione tra *head* e *modifier*.

La supposizione di partenza è semplice: un modo per comprendere la relazione tra due parole in un composto di due nomi è quello di porre i componenti in un rapporto di *head-modifier*, e assumere che l'*head* abbia una struttura tale per cui dal suo significato si possa capire che genere di cose possono essere fatte ad esso, di cosa è fatto, di cosa è parte, e così via. Si pensi ad esempio alla parola *kinfe* (*coltello*). I coltelli sono creati per particolari attività, possono essere fatti di vari materiali, possono essere usati per tagliare o manipolare vari tipi di cose, etc. Un set di relazioni e di possibili composti contenenti il termine *coltello* sono:

(Used-in): kitchen knife, hurting knife;

(Made-of): steel knife, plastic knife;

(Instrument-for): carving knife;

(Used-on): mead knife, putty knife;

Alcune relazioni sono applicabili a certe classi di nomi soltanto. La struttura semantica dell'*head* dà il range di possibilità. Se è possibile predire il comportamento dei componenti, allora lo è anche predire la relazione che li lega.

La gerarchia lessicale MeSH è suddivisa al suo interno in quindici sotto-gerarchie (alberi), ognuno corrispondente a una delle maggiori branche della medicina. Ad esempio, l'albero A rappresenta l'Anatomia, il B l'Organismo, il C le Patologie, ... e così via. Ogni albero ha dei sotto-alberi. L'anatomia, ad esempio, è composta dalle Regioni del Corpo (ID: A01), dal Sistema Muscolo-scheletrico (A02) etc. Queste sono le categorie di "livello 0".

Questi nodi hanno dei nodi figli, come ad esempio Addome (A01.047), Schiena (A01.176), che sono i figli di "livello 1" delle Regioni del Corpo. Più lunga è l'ID del termine MeSH, più lungo è il percorso dalla radice e più precisa è la descrizione. Per esempio, l'Emicrania ha ID C10.228.140.546.800.525, che significa: C (Patlogia), C10 (Patologia del Sistema Nervoso), C10.228 (Patologia del Sistema Nervoso Centrale), e così a seguire. Al 2001, in MeSH sono presenti oltre 35000 ID univoche. A molte parole sono assegnati varie MeSH ID, così che occorrono in più di un punto entro la gerarchia. La struttura di MeSH può essere interpretata come una rete.

Il procedimento utilizzato per trovare la corretta relazione semantica tra l'*head* e il *modifier* del

composto è il seguente:

1. Vengono estratti composti di due nomi cercando coppie di parole adiacenti che siano classificate entrambe come nomi da un tagger e compaiano nella gerarchia MeSH, e nessuna delle parole precedenti o seguenti sia presente in MeSH;
2. Viene usato MeSH per caratterizzare sia l'*head* sia il *modifier* secondo le rispettive categorie semantiche di appartenenza. Se un nome possiede più di una MeSH ID, vengono riportate tutte le possibili categorizzazioni;
3. I composti vengono riportati in un piano cartesiano, con la categoria MeSH per il primo nome sull'asse delle X e quella del secondo nome sull'asse delle Y. Ogni intersezione indica il numero di composti classificati sotto le due corrispondenti categorie MeSH. Viene presa in esame la distribuzione delle intersezioni, per vedere se sia uniforme o meno. Se vale l'ipotesi di partenza che ai composti che si trovano entro la stessa coppia di categorie venga assegnata la stessa relazione, allora se la maggior parte di essi cade all'interno di sole poche coppie di categorie, è necessario determinare soltanto quale relazione intercorra tra un sottoinsieme delle possibili coppie;
4. Viene dapprima selezionato un sottogruppo di categorie da esaminare nel dettaglio. Sono esaminate a mano le relazioni (parafrasate) tra i nomi del 20% dei composti presenti nel sottogruppo, per poi vedere se la parafrasi trovata è la stessa per tutti i composti nel gruppo. Se così è, allora il livello corrente della coppia di categorie è da considerarsi come il livello corretto di descrizione. Se, altrimenti, vengono trovate molte parafrasi diverse, allora l'analisi deve scendere di un livello nella gerarchia. Questo procedimento deve essere ripetuto finché tutti i nomi composti del gruppo ottenuto non presentano al loro interno relazioni di tipo uniforme. Ad esempio, i seguenti composti sono stati mappati con la stessa coppia di categorie, A01 (Regioni del Corpo) e A07 (Sistema Cardiovascolare): *shoulder artery* (arteria della spalla), *ankle artery* (arteria della caviglia), *leg veins* (vene della gamba), *limb vein* (vena degli arti), etc... Tutti questi composti sono stati giudicati come simili nel senso che le relazioni tra le due parole componenti sono simili, perciò non è necessario scendere ulteriormente lungo la gerarchia. La coppia (A01, A07) è pertanto chiamata “regola”, dove una regola è una coppia di categorie sotto la quale tutti i nomi composti hanno la sessa relazione. Per concludere, tranne in tre casi, la discesa lungo la gerarchia è stata fatta solo per il secondo nome del composto. Questo può essere dovuto a

due fattori: 1) generalmente in inglese il secondo nome ha il ruolo del *modifier*, per cui richiede più specificità; 2) negli esempi esaminati, i termini più eterogenei dominavano il secondo nome.

L'interesse di questo lavoro sta nell'aver apportato l'evidenza che i livelli più alti di una gerarchia lessicale possono essere usati per classificare accuratamente le relazioni che intercorrono tra parole composte da due nomi all'interno di un dominio tecnico, che è il passo che precede l'effettiva risoluzione del significato del composto in uno dei possibili modi di procedere nella disambiguazione.

2.3 Metodi basati sulla similarità e loro ibridi

Segue ora la descrizione di tre approcci basati sul calcolo delle similarità semantica, tutti testati su un dataset comune di pubblico accesso, secondo il lavoro di Kim e Baldwin [27, 29, 30]: SENSE COLLOCATION, CONSTITUENT SIMILARITY e Co-training utlizzando sia SENSE COLLOCATION che CONSITUENT SIMILARITY.

1. Il metodo SENSE COLLOCATION è basato sulla coppia di sensi dei nomi di un composto. L'idea di base è che i nomi composti che hanno la stessa o una simile collocazione di senso (ossia, il significato) tendono ad avere la stessa relazione semantica. Ad esempio, *car factory* e *automobile factory* condividono l'interpretazione convenzionale di MAKE, che è predetta da *car* e *automobile*, che hanno lo stesso senso all'interno dei composti, e da *factory* che è utilizzata con lo stesso significato in entrambi. La probabilità $P(r|f_i f_j)$ (semplificata in $P(r|f_{ij})$ di una relazione semantica r per i sensi f_i e f_j è calcolata in base alla massima stima della probabilità:

$$P(r|f_{ij}) = \frac{n(r, f_{ij})}{n(f_{ij})}$$

La relazione semantica preferita per una determinata combinazione di sensi è quella che massimizza la probabilità:

$$r_{max} = argmax_{r \in R} P(r|f_{ij}) = argmax_{r \in R} P(f_{ij}|r) P(r)$$

2. L'intuizione dietro al metodo CONSTITUENT SIMILARITY è simile a quella del metodo SENSE COLLOCATION, nell'idea che nomi composti da parole simili tendo a condividere la stessa relazione semantica. La differenza fondamentale è che questo metodo non presuppone che noi conosciamo il senso di ogni termine (cioè la similarità è calcolata a livello della parola, piuttosto che a livello del senso). Ad esempio, possiamo trovare che l'istanza di test *chocolate milk* si avvicina molto a *apple juice* e quindi predice che la relazione semantica sarà *material*. L'idea è formulata nell'equazione seguente. Sia S_A la similarità tra i composti $(N_{i,1}, N_{i,2})$ e $(B_{j,1}, B_{j,2})$:

$$S_A((N_{i,1}, N_{i,2}), (B_{j,1}, B_{j,2})) = \frac{((\alpha S1 + S1) * ((1-\alpha) S2 + S2))}{2}$$

dove *S*1 è la similarità tra i due *modifier* (cioè $S(N_{i,1}, B_{j,1})$) e S2 quella tra i due *head* (cioè $S(N_{i,2}, B_{j,2})$), α è un fattore di peso compreso tra [0,1] . I punteggi di similarità sono calcolati tra i sensi di WordNet utilizzando il metodo di Wu e Palmer implementato in *WordNet::Similarity* (vedi paragrafo sulle misure di similarità).

3. Il co-training attraverso il sense collocation (SCOLL CO-TRAINING) è basato sul metodo SENSE COLLOCATION e sulla sostituzione lessicale (Kim e Baldwin, 2007). Espande il training set di nomi composti a partire da un numero relativamente piccolo di istanze disambiguate a mano. Fa cioè uso di istanze modellate con un processo di bootstrap. Per esempio, se assumiamo che *automobile factory* abbia relazione semantica MAKE, allora tutti i nomi composti generati da sinonimi, ipernimi, e parole sorelle dei suoi costituenti sarebbero aggiunti tra le istanze di training, come ad esempio: *car factory* (sinonimo), *vehicle factory* (ipernimo) e *truck factory* (sister word). Si noti che la sostituzione coinvolge solo un costituente alla volta per evitare variazioni estreme.
4. Il co-training attraverso la Constituent Similarity (CS CO-TRAINING) è pure un metodo di co-training, ma basato sulla CONSTITUENT SIMILARITY piuttosto che sul SENSE COLLOCATION. L'idea di partenza è che quando i nomi composti sono interpretati utilizzando il metodo della CONSTITUENT SIMILARITY, le previsioni sono più affidabili quando la similarità semantica è maggiore. Quindi, la soglia della similarità viene progressivamente ridotta, e vengono

incorporate nel training set in fase di bootstrap istanze che possiedono un'alta similarità. Questo significa che il metodo della CONSTITUENT SIMILARITY è applicato acquisendo composti con similarità maggiore o uguale a una data soglia. Successivamente, nell'iterazione successiva, vengono aggiunti i composti acquisiti nel training set, all'interno del quale il numero di istanze aumenta in modo monotono ad ogni passo. Ad ogni iterazione, inoltre, il valore della soglia viene diminuito e le istanze sono ordinate in senso decrescente in base a tale valore.

2.4 L'idea di Fan, Barker, Porter: le informazioni necessarie per interpretare i nomi composti

James Fan, Ken Barker e Bruce Porter [31], del dipartimento di ingegneria informatica dell'università del Texas, ad Austin, propongono un'investigazione su base empirica riguardo alla conoscenza preliminare richiesta al fine di interpretare un composto (nello specifico della loro ricerca si tratta di un composto di due nomi, anche se il risultato è valido anche per sequenze più lunghe attraverso l'applicazione di tecniche di bracketing).

I precedenti studi di linguistica si sono concentrati sullo studio dell'interpretazione dei nomi composti con l'obiettivo di scegliere una singola categoria semantica per ogni coppia a partire da una lista di categorie stilata a mano. A livello computazionale, il processo di disambiguazione è stato affrontato con corpora di esempi precedentemente risolti, ma quasi nessuna informazione sui nomi che formavano il composto. La soluzione tipica proposta, infatti, era basata su pattern statistici scoperti all'interno di tali corpora.

Lo scopo di Fan, Barker e Porter è più generale: vengono cercate sequenze di relazioni semantiche (invece di una singola categoria) che connettono due nomi in un composto. Le relazioni semantiche da loro utilizzate appartengono ad un elenco di circa cinquanta ruoli tematici come *agent*, *object*, *has-part, location*, etc. Ad esempio, dato il composto *animal_virus*, l'interpretazione classica sarebbe: “animal virus *is-a* virus *in* (an) animal”, mentre quella composta da una combinazione di relazioni semantiche potrebbe essere: “an animal_virus *is-a* virus that is the *agent of* an invade, such that the *object of* the invade is the cell *part-of* an animal”.

2.4.1 L'algoritmo

Questo sistema di acquisizione delle informazioni, insomma, interpreta con successo un composto se trova una sequenza sensibile di relazioni semantiche tra il nome *head* e il suo *modifier* e ne costruisce una rappresentazione formale corretta.

```
Given a noun compound of the form <C1, C2>:

  0. RESULT = nil
  1. breadth-first search starting from C1:
    if the current level is deeper than maximum depth, then
       step 2
    if the current level has C2 or concepts that are superclasses/subclasses of C2,
       then
       append RESULT with the paths from C1 to these concepts and go to step 2
    else
       breadth-first search the next level along all the semantic relations
  2. breadth-first search starting from C2:
    if the current level is deeper than maximum depth, then
      step 3
    if the current level has C1 or concepts that are superclasses/subclasses of C1,
       then
       append RESULT with the paths from C2 to these concepts and go to step 3
    else
       breadth-first search the next level along all the semantic relations
  3. sort RESULT in ascending order based on the lengths of the paths in RESULT
Return RESULT
```

Figura 3: L'algoritmo di Fan, Barker e Porter

L'algoritmo (vedi Testo 2) prende in ingresso un composto nella forma $<C_1,C_2>$, dove C_1 e C_2 sono i concetti mappati a partire dai membri del composto. Ciascuno dei primi due step conduce una ricerca attraverso tutti gli archi che rappresentano le relazioni semantiche che collegano i concetti C_1 e C_2 ad altri. La prima ricerca parte da C_1 e cerca C_2 qualsiasi sottoclasse o superclasse di C_2. La seconda ricerca compie l'operazione inversa. Lo step 3 combina i risultati ordinandoli in base alla lunghezza del percorso.

Vediamo un esempio:

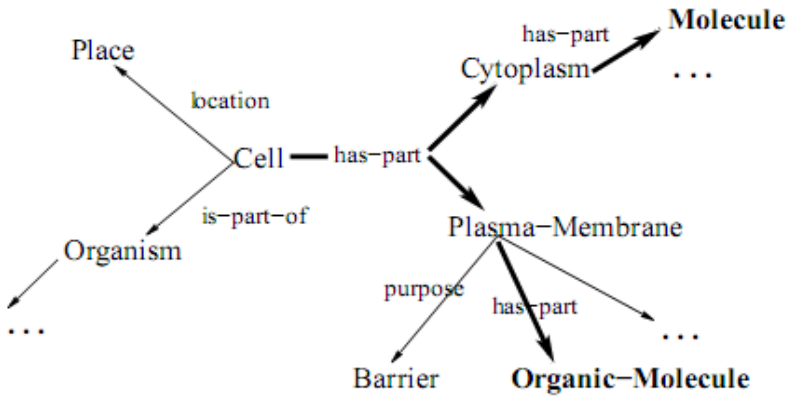

Figura 4: Un esempio del funzionamento dell'algoritmo: dato il composto cell_lipid, le linee in grassetto mostrano il percorso trovato dalla ricerca che parte da Cell e termina a Molecule e Organic-Molecule, entrambi superclassi di Lipid.

Nell'esempio riportato in figura, dato il composto *cell_lipid* (un lipide è una molecola di grasso), la ricerca comincia a *Cell* per attraversare tutto le relazioni semantiche che da *Cell* si dipartono, come *has-part* e *location*. La ricerca si ferma a due nodi, *Organic-Molecule* e Molecule, entrambi superclassi di *Lipid*, e la definizione del composto è da scegliere tra questi due percorsi lungo il grafo concettuale:

a) “The organic-molecule part-of the plasma-membrane part-of the cell”;
b) “The molecule part-of the cytoplasm part-of the cell”.

Vi sono due ragioni per le quali l'algoritmo si ferma quando una sottoclasse o una superclasse dell'obiettivo viene trovata:

1) quando viene trovata una sottoclasse, per inferenze ne deduciamo che c'è un percorso anche tra il concetto di partenza e il concetto di arrivo, infatti ogni istanza di una sottoclasse è anche istanza della classe obiettivo;
2) quando viene trovata una superclasse dell'obiettivo, per inferenza deduciamo che potrebbe

esserci anche un percorso tra il concetto di partenza e quello di arrivo: un'istanza della superclasse può essere istanza della classe obiettivo.

L'interpretazione viene considerata corretta qualora venisse restituito un percorso sensibile, cioè giudicato sensato da una persona. Se vengono restituiti più percorsi della stessa lunghezza, allora ne viene scelto uno come principale.

2.4.2 La base informativa e la sensibilità dell'algoritmo

La conoscenza di base necessaria per realizzare questo algoritmo è costituita da:

1. un set di concetti;
2. un set di assiomi associati a ciascun concetto. Gli assiomi di un concetto sono asserzioni di relazioni semantiche tra quel concetto ed altri ad esso collegati. Ad esempio, tra gli assiomi relativi ad *Action* troviamo "*every Action has an object, which is an Entity*".

La base informativa è stata costruita a mano a partire da WordNet, cui sono stati aggiunti i livelli ontologici superiori provenienti da altre librerie.

Il grado di importanza di ogni livello all'interno dell'ontologia è stato misurato attraverso una serie di eliminazioni: quando un livello viene eliminato, i concetti presenti a quel livello e tutti i relativi assiomi vengono cancellati dall'ontologia. In assenza di eliminazioni, i valori di *precision* (numero di risposte corrette diviso numero di risposte date) e *recall* (numero di risposte corrette diviso numero di risposte possibili) si aggirano entrambe intorno all'80%. Eliminando il primo livello causa un crollo di questi indici. A mano a mano che il livello eliminato di volta in volta è più basso, l'impatto è minore e le prestazioni dell'algoritmo migliorano avvicinandosi ai risultati ottenuti senza eliminazioni.
È stato pertanto dedotto che a rendere tanto importanti, per non dire fondamentali, i livelli superiori sono due motivi:

1. I livelli più alti includono concetti che contengono importanti distinzioni ontologiche.

Eliminarli porta alla creazione di molte più interpretazioni e, dato che viene considerata solo la prima, la probabilità che sia corretta è considerevolmente ridotta.

2. Sebbene contengano relativamente pochi assiomi, gli assiomi nei livelli in cima all'ontologia sono importanti per lo scopo: nei livelli superiori troviamo gli assiomi più usati, come il fatto che "*every Action involves an object that is acted upon*".

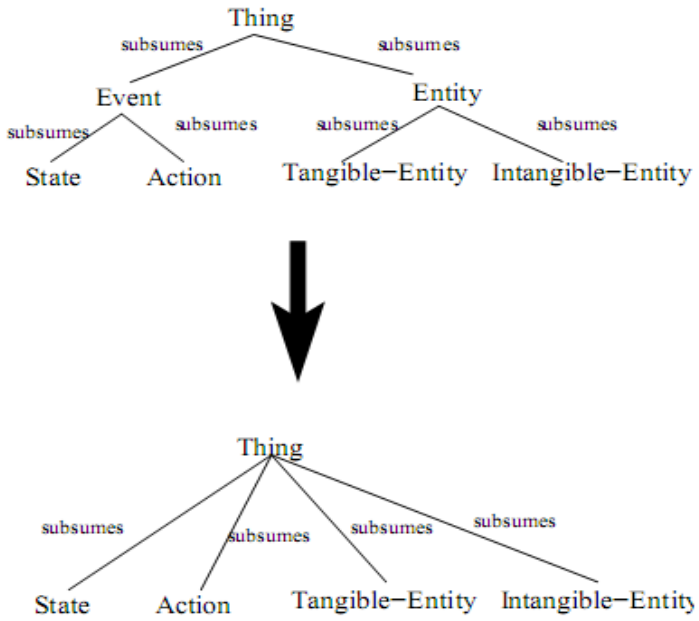

Figura 5: L'eliminazione del livello 1 in una tassonomia campione. Il livello 0 è dato da Thing.

Capitolo 3

3 Algoritmo di risoluzione dei termini composti

L'idea su cui si basa l'algoritmo qui presentato non è totalmente nuova, piuttosto prende spunto dalle ricerche passate per dedurre le informazioni necessarie ad acquisire ed interpretare i termini composti in modo corretto ed efficiente.

Quello che invece viene presentato è un algoritmo per risolvere i termini composti che richieda il minimo delle risorse possibili e che sia il più possibile portabile ed indipendente dal contesto di applicazione. Verrà pertanto dimostrato come con strumenti relativamente semplici sia possibile ottenere buoni risultati.

Sono due i possibili modi per ottenere la risoluzione dei composti:

- dato un composto di due termini, distinguere *head* e *modifier* (nel caso di più di due termini, si può applicare un procedimento ricorsivo) rende già disponibile la conoscenza strettamente necessaria per avere il significato del composto. Si pensi a *Certificate_Name*: aver identificato che *Name* è l'*head*, permette di capire che *Certificate_Name* è un nome (*Name*) con la caratteristica di aver a che fare (la relazione che intercorre tra i due termini è solo un passo successivo) in qualche modo con un certificato (*Certificate*), e non il contrario;
- solo in secondo luogo (ad esempio nel caso in cui non sia stato possibile distinguere in modo certo *head* e *modifier* senza dare semplicemente per scontato che l'*head* sia a destra e il *modifier* a sinistra) si può pensare di reperire una catena, o più precisamente un albero, di termini e relazioni che li collegano a un parente (o un figlio) in comune da cui i due rami dell'albero divergano. I tipi di relazioni all'interno dell'albero possono essere tra loro diversi, oppure sempre dello stesso tipo. Nel caso di questa tesi, si è ricercato un albero di soli ipernimi in quanto verificato sperimentalmente che altri tipi di relazioni (soli iponimi, soli olonomi e così via), per il tipo di composti disambiguati, non davano alcun risultato.

Come accennato nei paragrafi precedenti, l'unico strumento strettamente necessario per fare questo è il grafo concettuale all'interno del quale i termini occupano una determinata posizione. Nel

contesto di questa tesi, WordNet.

L'idea di partenza per ottenere la soluzione è la seguente: data la rappresentazione concettuale di un database lessicale e un composto di due parole, il significato di quest'ultimo è ottenuto a partire dal suo contenuto informativo, sia di tipo grammaticale, sia di tipo semantico, indipendentemente dagli strumenti utilizzati. Si analizza dapprima il ruolo grammaticale dei due termini; nel caso in cui questo non fornisse alcuna informazione che non possa essere già dedotta dalla posizione di tali termini all'interno del composto, si procede con la ricerca della più breve sequenza di relazioni semantiche che lega i due termini. In altre parole, è comunque possibile ottenere la definizione di un composto a partire dai synset che dividono all'interno della rappresentazione a grafo i due componenti e dalle relazioni che intercorrono tra tali synset presi a due a due.

Segue ora una descrizione del funzionamento dell'algoritmo.

3.1 Il funzionamento dell'algoritmo

La prima operazione da compiere, per poter interagire con il dizionario WordNet, è quella di stabilire con esso una connessione, passandogli come parametro il file *file_properties.xml*, contenente tutti i dettagli relativi alla versione del dizionario utilizzata, il path di installazione e così via.

In seguito, al fine di annotare i risultati in modo da tenerne traccia per analisi future, è necessario connettersi anche con la locale installazione di un database. Per la realizzazione di questo progetto, come già anticipato, è stato scelto MySQL.

Fatte queste premesse, se tutto è andato a buon fine, è possibile iniziare la fase di parsing di un file (in questo caso con estensione .xml), al fine di reperire il maggior numero possibile di composti da risolvere.

Ecco la procedura seguita dall'algoritmo realizzato:

1. Si controlla il valore del tag in ingresso. Vengono presi in considerazione tre tipi:
   a) *Source*: rappresenta la sorgente di provenienza degli attributi a seguire; se il tag ha questo valore, allora viene memorizzato così da essere inserito nella tabella del database al momento dell'inserimento dei valori degli attributi contenuti nella sorgente che costituiranno termini composti. La coppia (nome_sorgente, valore_composto) andrà a costituire la chiave primaria nella tabella dei risultati in modo di tenere traccia della

provenienza di ciascun composto e di non perdere il valore di alcuni composti qualora questi comparissero in più di una sorgente nello stesso file;

b) *Interface*: questo tag può essere identificato come il nome di una tabella originaria da cui sono stati estrapolati i dati e l'inizio dell'elenco dei vari attributi in essa presenti. Mano a mano che viene effettuato il parsing di un campo *Interface*, viene tenuta memoria del valore corrente del campo "name";

c) *Attribute*: tag che rappresenta il valore degli attributi contenuti in *Interface*.

Si entra nella fase di risoluzione solo qualora i valori dei tag in ingresso siano di tipo *Interface* oppure *Attribute*. Sono esaminati i valori del campo "*name*".

2. Si entra nella fase di "pulitura" del valore. Sono eliminati eventuali trattini o underscore, sostituendoli con spazi bianchi, e spezzati i camel case, quando presenti. Si ricorda che WordNet non riconosce underscore o trattini, che pertanto devono essere sostituiti con spazi bianchi. In questo modo, nel caso in cui il valore "pulito" ritornato sia effettivamente un composto (cioè semplicemente se si ottiene una stringa di lunghezza maggiore di 1), se ne inizia l'analisi.
3. Il primo passo per risolvere il composto è effettuare il controllo che esso non sia già presente in WordNet. È il caso di composti come *apple pie*, *first name*, etc. Se viene trovata una glossa, allora la disambiguazione è terminata e si può passare al composto successivo, altrimenti si continua con il passo 4.
4. Il secondo controllo da effettuare consiste nel verificare se uno dei membri del composto ha un valore uguale a quello del campo *Interface*, oppure è in esso contenuto. Se così è, allora questo membro all'interno del composto costituisce certamente il *modifier* (nel caso di composto a due membri, l'altro sarà per esclusione l'*head*), la risoluzione è terminata e si può passare al composto successivo. Ad esempio, se il valore del campo "*name*" della *Interface* corrente è *Certificates*, e il valore dell'attributo in esame è *CertificateName*, allora certamente *Certificate* sarà il *modifier* e *Name* l'*head*, poiché *Certificate* è contenuto in *Certificates*. Se questo invece non accade, si va al passo 5.
5. Se i passi precedenti non hanno dato alcun esito, allora si procede con l'analisi grammaticale dei membri del composto, ora presi due a due. È stata fatta la scelta, in questa fase, di considerare i membri del composto in coppia per rendere i valori annotati il più possibile riutilizzabili. Ad esempio, per il valore *Product_Customer_Type*, è stata analizzata dapprima

la coppia (*Customer*, *Type*), poi la coppia (*Product, Customer*), così da avere già a disposizione la soluzione corretta nel caso una di queste due coppie ricomparisse in altro contesto. Se uno dei due membri non esiste in WordNet, oppure se può assumere soltanto la funzione di verbo, l'analisi del composto corrente è terminata e si può passare al seguente. Altrimenti, si distinguono due casi:

a) se uno dei due membri del composto è un aggettivo e l'altro è un nome, allora certamente l'aggettivo è il *modifier* e il nome è *l'head*, in quanto il nome è modificato dall'aggettivo. In caso affermativo, il composto è risolto e si può passare al successivo;
b) se invece entrambi i membri sono nomi, poiché le precedenti fasi non hanno dato alcun esito, si assume che il membro più a destra sia l'*head* e quello a sinistra il *modifier*. È un'assunzione sensata soltanto per la lingua inglese, ma si è tenuto conto del fatto che la grande maggioranza di sorgenti esaminate sono in inglese. Si passa quindi al punto 6.

6. Nel caso della coppia nome + nome, si procede con la ricerca dell'albero di ipernimi che collega i due elementi. Viene trovato un ipernimo comune, che altro non è che il *least common subsumer*, a partire dal quale i due rami dell'albero divergeranno, in modo asimmetrico, fino a giungere a ciascuno dei due membri. Purtroppo questo metodo ha dato spesso come risultato quello di ritornare, come *least common subsumer*, synset di WordNet con scarso valore informativo, come *Abstaction* o *Entity*. Tuttavia, vi sono casi in cui questa tecnica ha dato risultati interessanti: ad esempio, l'ipernimo comune trovato del composto *Expiration_Date* è *Measure*, che sicuramente può andare a esprimere il significato di *Expiration_Date* con buona approssimazione. Si tratta della vera e propria fase di disambiguazione in quanto si cerca un synset da sostituire al composto e che ne comprenda il significato. Nel caso in cui nemmeno questa fase abbia dato risultati, la risoluzione termina, i valori di *head*, *modifier* e della soluzione sono da considerarsi NULL e si passa al composto successivo.

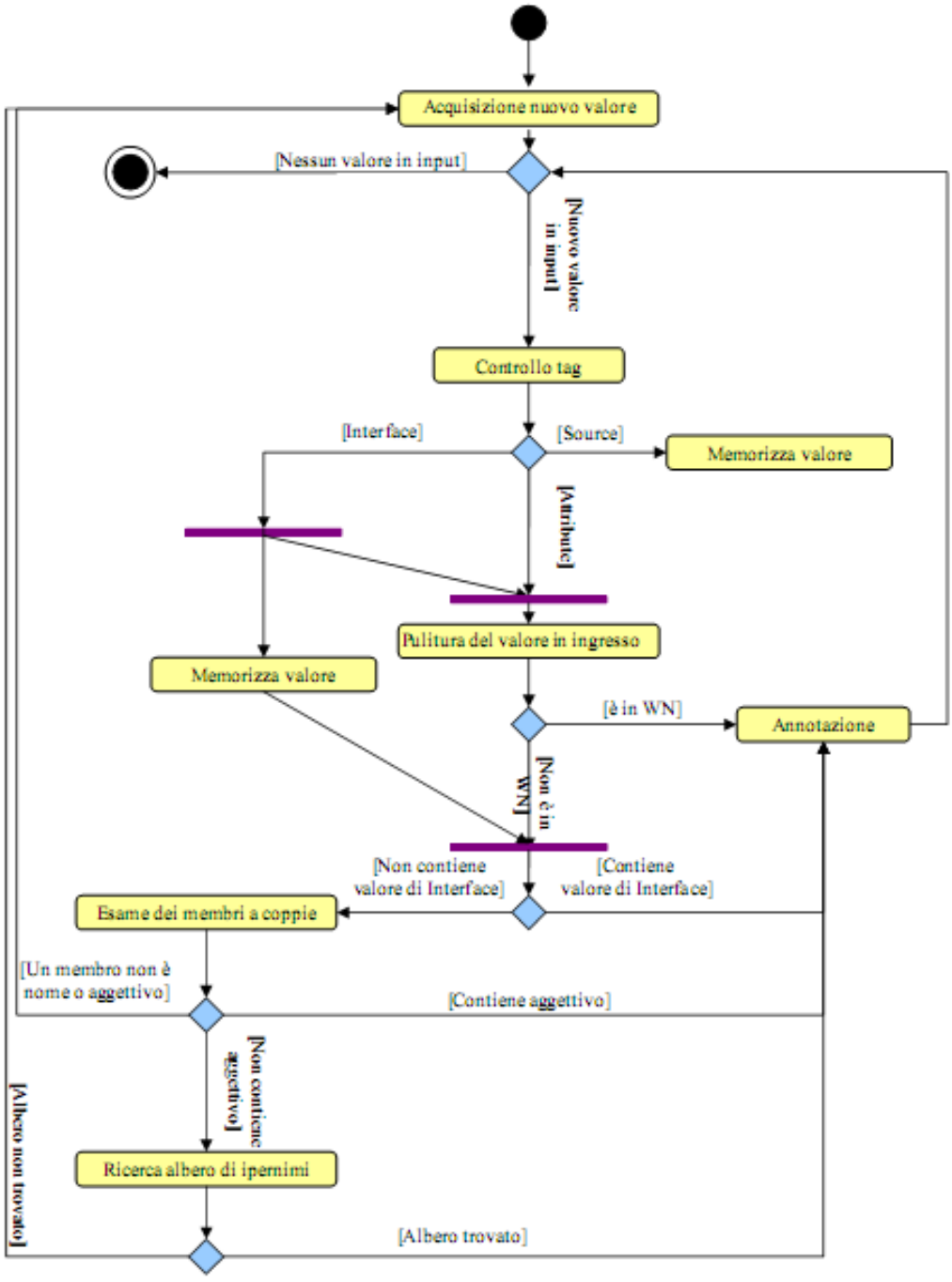

Figura 6: Activity diagram del funzionamento dell'algoritmo

Segue ora un breve riepilogo del procedimento seguito dall'algoritmo per cercare il significato di un composto:

I. Pulitura del composto: rimozione di eventuali trattini, underscore o camel case;

II. Controllo della presenza del composto in WordNet. Se è presente, si riporta la glossa e si passa al composto successivo; Altrimenti, si va al passo III;
III. Se il valore di uno dei membri del composto eguaglia oppure è contenuto in quello dell'interfaccia, allora questo sarà il *modifier*. Nel caso di composti a due membri, l'altro sarà per esclusione l'*head.* Se questo controllo ha dato risposta affermativa, allora si passa al composto successivo; altrimenti si procede con l'algoritmo spezzando il composto in coppie di termini;
IV. Se uno dei due membri non esiste in WordNet oppure può essere soltanto verbo, allora l'analisi termina; altrimenti, si passa al punto V;
V. Se uno dei due termini è aggettivo e l'altro è nome, allora l'aggettivo è il *modifer* e il nome l'*head*. In questo caso, si passa al composto seguente. Altrimenti, si va al passo VI;
VI. Si ricerca l'albero di ipernimi che collega i due membri. Se viene ritornato un parente comune, la disambiguazione è andata a buon fine. In caso contrario è fallita. In entrambi i casi, si passa al composto successivo.

3.2 Strumenti per la risoluzione dei termini composti

Alla luce dei risultati ottenuti nel corso degli ultimi anni nel campo della ricerca semantica rivolta alla disambiguazione dei termini composti, sono due le considerazioni che si possono trarre:

- ➔ L'interpretazione dei termini composti è un problema la cui soluzione è da ricercare attraverso l'analisi delle relazioni che intercorrono tra i costituenti. Questa è l'unica strada che consenta di generalizzare abbastanza l'algoritmo di risoluzione per renderlo il più possibile indipendente da qualsiasi contesto applicativo. Ad esempio, il composto *concrete_floor* significa "*floor MADE-OF concrete*". È necessario pertanto trovare gli strumenti più adatti ad ottenere la relazione che intercorre tra due (o più di due, ricorsivamente) termini.
- ➔ Non è detto che sia necessario utilizzare strumenti estremamente sofisticati: i costi in termini computazionali e temporali potrebbero essere maggior dei benefici potenziali e degli effettivi risultati. Nell'algoritmo presentato più avanti in questa tesi, si parte dal presupposto che per interpretare con successo una parola composta servano solamente le informazioni "di scheletro" dei componenti: cioè, la loro classificazione tassonomica (attraverso le

relazioni *is-a*, cioè ipernimia e iponimia) e/o di *parte-insieme* (olonomia e meronimia) all'interno di un grafo.

Quella che segue è una descrizione degli strumenti, fondamentali e accessori, utilizzati per realizzare l'algoritmo.

3.2.1 L'utilizzo di WordNet

La risorsa migliore per ottenere queste informazioni è, senza ombra di dubbio, proprio WordNet. Esso costituisce la base informativa fondamentale utilizzata per estrarre informazioni riguardanti i membri di un composto. L'idea fondamentale è quella di vedere WordNet come un grafo, nel quale i synset siano i nodi, e le relazioni che intercorrono tra i synset gli archi di collegamento. Ogni synset è in relazione con uno o più synset. Le tipologie di relazione tra un synset sorgente s_1 e un synset obiettivo s_2 che prenderemo in considerazione sono due:

- Relazione simmetrica (*symetric relationship*) → è una relazione nella quale s_2 sarà sempre tra gli antenati di s_1, e viceversa. Tipico esempio è la sinonimia.
- Relazione asimmetrica (*asymmetric relationship*) → è un tipo di relazione nel quale l'albero di ipernimi (come se si trattasse di un albero genealogico, o, meglio ancora, di una tassonomia) di s_1 e s_2 hanno un punto di divergenza definito, ossia il parente comune (*common parent*), altrimenti noto come *least common subsumer* nella teoria trattata nel capitolo sullo stato dell'arte. Non necessariamente s_1 e s_2 hanno la stessa distanza dal parente comune.

Un composto viene quindi scisso nei suoi due componenti a ciascuno dei quali corrisponderà un synset e quindi un nodo di WordNet.

L'idea che questa tesi andrà a dimostrare è la seguente: dopo aver analizzato il contenuto semantico dei due membri del composto partendo dalle informazioni ottenute da WordNet, in ultima analisi è possibile effettuare la disambiguazione trovando il percorso più breve che lega i due synset all'interno di WordNet e le relazioni che intercorrono tra i vari nodi attraversati (cioè un percorso di relazioni semantiche).

3.2.2 La Java WordNet Library

La Java WordNet Library (JWNL) è una libreria scritta da John Didion presso la Princeton University in puro codice Java che permette di avere accesso ai dati all'interno di WordNet a livello di API. Il codice sorgente è disponibile all'indirizzo http://sourceforge.net/projects/jwordnet. Si tratta di un progetto riconosciuto anche dagli autori di WordNet, infatti informazioni al riguardo sono disponibili alla sezione "Related Projects" dell'homepage di WordNet http://wordnet.princeton.edu/. Attualmente (ottobre 2008) la versione più aggiornata della JWNL è la 1.4, che permette l'interfacciamento con le versioni più recenti di WordNet, la 2.1 per Windows e la 3.0 per i sistemi Unix-based.

I motivi per cui la JWNL è stata scelta per la realizzazione del programma Java che permette la disambiguazione di termini composti sono i seguenti:

- Trattandosi di puro codice Java e non contenendo codice nativo, è assolutamente portabile e indipendente dalla piattaforma di utilizzo;
- Essendo un progetto direttamente collegato a WordNet e sviluppato allo stesso modo all'interno di Princeton, garantisce un'ottima interazione con il database lessicale;
- Rende l'accesso a WordNet estremamente semplice: è infatti sufficiente creare un collegamento (attraverso l'oggetto *Dictionary*) con l'installazione locale del programma WordNet;
- Mette a disposizione metodi estremamente semplici ed intuitivi per navigare WordNet.

L'utilizzo della libreria

Ad un primo impatto, la libreria colpisce per il gran numero di classi da cui è composta. Tuttavia, come si evince anche dalla JavaDoc http://nlp.stanford.edu/nlp/javadoc/jwnl-docs/allclasses-noframe.html, molte classi sono tra loro collegate e quelle veramente necessarie per realizzare le funzioni del programma oggetto di questa tesi sono poche. Segue una descrizione delle classi e dei metodi fondamentali e delle operazioni preliminari da svolgere.

Il progetto di Eclipse e l'inclusione della libreria

Il programma Java per la disambiguazione dei termini composti è stato realizzato attraverso l'ausilio dell'ambiente di sviluppo integrato (*Integrated Development Environment* – IDE) Eclipse, gratuitamente disponibile al sito: http://www.eclipse.org/. La JWNL è stata aggiunta attraverso la configurazione del Build Path del progetto creato con Eclipse, inserendo il file *jar* scaricato insieme ai sorgenti tra le *Referneced Libraries*, senza dimenticare di aggiungere al progetto anche il file xml *file_properties.xml*, contenente tutte le impostazioni iniziali e necessario per l'interazione con il dizionario.

La configurazione iniziale

Una volta scaricati i sorgenti della JWNL all'indirizzo precedentemente citato, è necessario apportare una modifica al file di configurazione *file_properties.xml*, andando a modificare la linea che include la versione di WordNet utilizzata e quella che contiene il path di installazione del dizionario, inserendo il percorso in cui esso è installato nella macchina locale, ad esempio:

```
<param name=”dictionary_path” value=”C:\Programmi\WordNet\2.1\dict”/>
```

Risolto questo primo passo, è necessario inizializzare la libreria all'interno del programma che si va a scrivere:

```
JWNL.initialize(new FileInputStream(propsFile))
```

Dove la stringa “propsFile” altro non è che il percorso assoluto in cui si trova il file *file_properties.xml*.

Ultimo passo per quanto riguarda la fase preliminare è quello di creare un istanza dell'oggetto *Dictionary*, ad esempio chiamandola, per semplicità, wordnet:

```
wordnet = Dictionary.getInstance()
```

Esauriti questi tre semplici passi, è possibile ora utilizzare i metodi messi a disposizione dalla libreria.

***L'oggetto* IndexWord**

Per utilizzare la JWNL, è necessario familiarizzare con il modo con cui essa tratta i termini. Come detto, in WordNet i termini sono raggruppati in synset, cioè in gruppi di sinonimi. Ogni synset è

caratterizzato univocamente da un offset. Ogni termine può essere presente all'interno di più synset in base al significato e alla funzione grammaticale, che nella JWNL viene denominata *part of speech* (da qui in avanti *pos*), cioè nome, aggettivo, verbo o avverbio. L'oggetto *IndexWord* viene creato per rappresentare un termine singolo attraverso una *pos* univoca e i significati che assume in WordNet. Ad esempio:

```
IndexWord word = wordnet.getIndexWord(POS.VERB,"eat")
```

Questo ci rende possibile navigare tra i vari significati della parola "*eat*", intesa come verbo:

```
Synset[] senses = word.getSenses();
        for (int i = 0; i < senses.length; i++) {
                System.out.println(word + ": " + senses[i].getGloss());
        }
```

oppure di prendere in considerazione solo il primo significato del verbo "eat", ossia il più frequente:

```
Synset sense = word.getSense(1)
```

Le relazioni tra i synset

A questo punto, è possibile, per ogni dato synset, cercare tutti i synset ad esso collegati, direttamente o indirettamente, attraverso i vari tipi di relazioni presenti su WordNet. Il presupposto è questo: in WordNet, un synset che è in relazione con un altro, punta ad esso, e viceversa.

Si possono presentare due casi:

1) Avendo a disposizione un solo synset, la classe *PointerUtils* fornisce una selezione di metodi che ritornano una lista di synset cui il synset dato punta. Si consideri ad esempio la seguente riga di codice:

```
PointerTargetNodeList relatedList = PointerUtils.getInstance().getHyponymTree(sense,3)
```

che fornisce una lista (l'albero) di tutti gli iponimi (i figli) del synset *sense*. Il numero intero è opzionale e dice fino a che profondità devono essere presi i synset figli.

2) Se invece si hanno a disposizione due synset (o due *IndexWord*, poiché è possibile ottenere una *IndexWord* partendo da un synset e viceversa), si può cercare la relazione o la catena di relazioni che intercorre tra essi utilizzando i metodi della classe *RelationshipFinder*. Questa

classe permette di trovare solamente una catena di relazioni dello stesso tipo, per cui è necessario fornire come parametro anche il tipo di relazione desiderata, che sarà un oggetto *PointerType*. Vi è un *PointerType* per ogni tipo di relazione presente in WordNet:

PointerType.SYNONYM,

PointerType.HYPERNYM,

etc...

Si consideri l'esempio:

```
RelationshipList list = RelationshipFinder.getInstance().findRelationships(synset1, synset2,
PointerType.HYPENYM);
        if (!list.isEmpty()) {
                Relationship rel = (Relationship) list.get(0);
                System.out.println(rel);
        }
```

Viene creata un istanza (*list*) della classe *RelationshipList* la quale riporta la catena di synset che collegano il sysnet1 e il synset2 attraverso una serie di oggetti *Relationship* (cioè di relazioni) di ipernimia. Nel caso la lista ritornata non sia vuota, allora viene creato e stampato a video un oggetto *rel* che riporta, a titolo di esempio, solo la prima relazione trovata. Una volta che si ha a disposizione un oggetto *Relationship*, possono essere utilizzati vari metodi per ottenere informazioni riguardo alla relazione trovata. È ad esempio possibile conoscerne la profondità, cioè i gradi di separazione, attraverso *getDepth()*, oppure navigare attraverso i nodi della relazione trovata con *getNodeList()*: recuperata la lista dei synset da attraversare, è possibile scorrerla utilizzando un'istanza dell'interfaccia *Iterator*. Ad esempio:

```
System.out.println("The depth of this relationship is: " +  rel.getDepth());
```

```
        PointerTargetNodeList nodelist = rel.getNodeList();
                Iterator i = nodelist.iterator();
                        while (i.hasNext()) {
                                PointerTargetNode related = (PointerTargetNode) i.next();
                                System.out.println(related.getSynset());
                }
```

***Il metodo* getBestPos()**

Come noto, in inglese la stessa forma ortografica di una parola può assumere svariate funzioni grammaticali, cioè differenti *parts of speech*. Si pensi a *cut*, che può presentarsi come nome, come verbo e come aggettivo. In un contesto di disambiguazione dei termini che prescinde da qualsiasi informazione precedentemente reperita riguardo il ruolo logico e grammaticale delle parole, è necessario implementare un metodo che sia in grado di recuperarne la funzione grammaticale più frequente. Si sa che in WordNet il primo gruppo di synset riportati sono quelli aventi la funzione grammaticale più comune. Essendo questo lavoro di tesi incentrato sulla disambiguazione di termini composti, il problema è, nella maggior parte dei casi, circoscritto agli aggettivi e ai nomi. I composti reperibili all'interno di file xml o database difficilmente saranno formati da gruppi di parole che non siano nomi e/o aggettivi. Tuttavia, non sempre la *pos* più frequente è quella di interesse per la disambiguazione in atto. Il problema si è presentato, ad esempio, con il lemma *press*, analizzando il composto *press_report*: è evidente che nel contesto di questo composto, sia *press* sia *report* sono da intendersi come nomi. Purtroppo, utilizzando il metodo *getAllPos()* della classe *POS* della JWNL**,** che ritorna un array contenente tutte le possibili *pos* per una data stringa (si ricorda che senza avere a disposizione la *pos* non è possibile creare una *IndexWord*), si ottiene che queste non sono ordinate allo stesso modo in cui le ordina WordNet. Risulta pertanto che la *pos* più frequente per *press*, e quindi da considerare in fase di disambiguazione (ad esempio creando un oggetto *IndexWord*), sia verbo. Questo ovviamente porta a una risoluzione sbagliata del composto, poiché il synset che viene considerato cercando una relazione tra *press* e *report* è quello avente funzione di verbo e significato di “esercitare pressione o forza a o contro”, invece che il synset corretto con funzione di nome.

Una soluzione al problema è stata trovata adottando il seguente accorgimento: è stata realizzato un metodo, denominato *getBestPos()* ed ispirato ad un altro avente lo stesso nome presente in RiTa (http://www.rednoise.org/rita/wordnet/documentation/index.htm) una libreria basata sulla JWNL ma per l'utilizzo dell'ambiente di programmazione Processing (http://processing.org/). Tale metodo prende in input una parola in forma di stringa e ne considera la polisemia per ognuna delle possibili *pos* presenti in WordNet (al solito: nome, aggettivo, verbo e avverbio), per poi ritornare la *pos* con la quale la parola assume i significati (sensi) più differenti tra loro. In questo modo, è stato possibile effettivamente ritrovare il synset più opportuno per la disambiguazione. Permane comunque la

caratteristica della funzione, comune anche a quelle fornite dalla libreria JWNL, di individuare alcuni termini indicanti colori, come *red* (che dalla mente umana, nella maggioranza dei casi, viene interpretato subito come aggettivo, come nell'esempio *red_cat*), che vengono considerati nomi. Fortunatamente, viene preso il synset che presenta come ipernimo diretto il synset *colour*, per cui è semplice risalire al corretto significato di *red*.

3.2.3 Il parsing di file XML

I file sorgenti esaminati per ricercare parole e nomi composti sono di tipo XML. Questo semplifica la fase di parsing poiché Java mette a disposizione librerie per analizzare i file con questa estensione, che sono state importate nella classe TestXMLHandler.

```
import org.xml.sax.*;
import org.xml.sax.helpers.*;
```

Nello specifico, sono stati considerati i valori del campo “name” dei tag *Source*, *Interface* e *Attribute*.

3.2.4 L'interfacciamento con il database

MySQL è disponibile per il download all'indirizzo http://www-it.mysql.com/. Si tratta di un database management system (*DBMS*) relazionale, composto da un client con interfaccia a caratteri e un server, entrambi disponibili sia per sistemi Unix sia per Windows, anche se prevale un suo utilizzo in ambito Unix. Possiede delle interfacce per diversi linguaggi, compreso un driver ODBC, due driver Java e un driver per Mono e .NET. Il codice di MySQL è di proprietà della omonima società, viene però distribuito con la licenza GNU GPL oltre che con una licenza commerciale.

Poiché Java mette a disposizione le librerie e i driver per potersi interfacciare con MySQL all'interno di un programma, questo è stato scelto per effettuare il salvataggio dei dati che via via è stato necessario memorizzare. I vari composti che sono stati individuati in fase di parsing sono stati inseriti nella tabella *Disambiguazione* (colonna *compound*), la quale inoltre riporta le seguenti colonne: *source* (il nome della sorgente di provenienza del composto), *head* (dove compare l'*head*,

se identificato), *modifier* (l'eventuale valore del *modifier*, se identificato) e *solution* (l'eventuale definizione o l'albero di iperinimi reperiti in fase di disambiguazione).

All'interno della tabella, è stata definita come primary key la coppia (*compound*, *source*): in questo modo, per ogni sorgente sono riportati i composti effettivamente presenti, anche se questi sono già stati inseriti per un'altra sorgente. In tabella 4 è possibile visualizzare un estratto di quanto salvato su database: per il composto *ability type*, ad esempio, viene messo a NULL il valore del campo *solution*, poiché l'algoritmo è riuscito a distinguere *head* e *modifier* seguendo criteri che saranno esposti a seguire. Per il composto *activity type*, si può vedere che viene invece riportato l'albero di ipernimia.

COMPOUND	SOURCE	HEAD	MODIFIER	SOLUTION
ability type	*Archivist*	type	ability	NULL
ability value	*Archivist*	value	ability	NULL
ability-xrd fk	*Archivist*	NULL	NULL	NULL
activity type	*Archivist*	type	activity	activity -> act -> event -> psychological_feature <- cognition <- content <- idea <- concept <- category <- kind <- type
additional street	*Archivist*	street	additional	NULL
certificate date	*Archivist*	date	certificate	NULL

Tabella 2: Estratto della tabella Disambiguazione

3.3 Il codice e le classi realizzate

L'algoritmo implementato è stato scritto completamente in codice Java, con il supporto dell'ambiente di sviluppo Eclipse. È stato realizzato un progetto di nome *Disambiguazione*, all'interno del quale vi è il package *wordnet*, dove si trovano le quattro classi Java in cui è suddiviso l'algoritmo:

- **TestXMLHandler** → classe deputata al parsing dei file sorgenti con estensione *.xml*. Vengono considerati i valori degli attributi per i tag *Source* (file sorgente iniziale), *Interface* (ottenuto, ad esempio, a partire dal nome di una tabella all'interno di un database) e *Attribute*

(ad esempio, i nomi delle colonne della tabella). Sono inoltre compiute le operazioni preliminari di "pulitura" dei composti, al fine di eliminare eventuali trattini, underscore o camel case che WordNet non riconoscerebbe. Con il composto così adattato, si controlla che non sia già presente in WordNet. Se non lo è, il controllo viene passato alla funzione *Disambiguation* della classe WordNetdemo.

- **DBInterface** → è la classe che si occupa dell'interfacciamento con il database e che implementa alcune funzioni che contengono le query più utili da eseguire per inserire i valori ottenuti in tabella. Nel caso di questo progetto, è stato utilizzato il database MySQL (si veda più avanti per i dettagli).
- **WordNetdemo** → in questa classe si trovano:
  - la funzione *void main(String)*: si occupa della creazione del collegamento con WordNet e della sequenza di operazioni preliminari che il programma deve compiere dopo aver preso in input una parola composte. Se queste operazioni vanno a buon fine, il controllo è passato al metodo *getInput* della classe TestXMLHandler;
  - la funzione *void Disambiguation(String, String, String, String)*: le viene passato il controllo dalla funzione che si occupa del parsing dei file XML e prende in input quattro stringhe: il valore del composto così come è preso dal file sorgente, il valore del composto adeguatamente "pulito" da underscore, trattini o camel case e i due membri del composto. Questa funzione viene richiamata solo nel caso in cui il composto non sia già presente in WordNet. All'interno di questo metodo si hanno l'analisi grammaticale dei termini e, se necessario, la ricerca della più breve sequenza di relazioni e synset che dividono i due membri del composto.
- **WordNetHelper** → come dice il nome, si tratta di una classe di supporto. In essa sono stati implementati tutti i metodi di supporto alla funzione *Disambiguation* della classe WordNetdemo per effettuare:
  - la ricerca della *pos* migliore per ogni termine, attraverso il metodo *getBestPos(String)*;
  - la creazione di un oggetto *IndexWord*, con il metodo *getIndexWords(Char Sequence)*;
  - la ricerca del percorso più breve che intercorre tra due *IndexWord* dato il tipo di relazione e la stampa a video della sequenza di termini da attraversare, tramite la funzione *findRelationshipDemo(IndexWord, IndexWord, PointerType)*;
  - la ricerca del *least common subsumer* date due *IndexWord* appartenenti a synset di

WordNet, con la funzione *getCommonParent(IndexWord, IndexWord)*;

- la ricerca dell'eventuale figlio comune, o *common child*, ai due membri di un composto. Nel caso due synset avessero un terzo synset in comune tra i loro meronimi e/o iponimi comuni, allora questo va a realizzare la disambiguazione del composto.

Oltre a questi metodi fondamentali, se ne trovano numerosi altri che hanno lo scopo di incapsulare macrosequenze di operazioni consecutive che vengono spesso richieste dai metodi principali nella loro implementazione.

- **ComparableIndexWord** → si tratta di una piccola classe Java di supporto alla funzione *getBestPos(String)* e importata dalla già citata libreria RiTa, per semplice comodità.
- **CompoundCase** → questa classe contiene i metodi per riconoscere i composti anche in caso essi siano scritti in camel case o con una qualsiasi altra ortografia che non corrisponde a quella riconosciuta da WordNet: ad esempio se i due termini sono separati da un trattino o da un underscore. Ricordiamo che i termini composti di cui si occupa questa tesi sono quelli reperibili in documenti strutturati e semi-strutturati, all'interno dei quali i composti difficilmente sono scritti come stringhe consecutive e facilmente individuabili.

  Vi sono quattro metodi all'interno della classe *CompoundCase*:

  - *boolean isCompound(String)*: presa in input una data stringa, conta il numero di trattini, underscore oppure spazi eventualmente presenti al suo interno:nel caso il numero ritornato sia maggiore di zero, si tratta di un composto e la funzione ritorna true;
  - *String deCompound (String)*: nel caso in cui la funzione *isCompound* sia risultata *true*, allora *deCompound* viene utilizzata per scomporre i composti che presentano trattini, spazi o underscore e che siano stati dati in input come un unica stringa; viene ritornata una stringa contenente uno spazio bianco laddove vi erano trattini o undescore, così da rendere poi questa stringa facilmente scindibile in più stringhe, ad esempio attraverso vari metodi messi a disposizione dalla classe String;
  - *boolean isCamel (String)* → prende in input una stringa e controlla se è scritta in camel case (cioè se al suo interno vi sono più termini distinti solo dall'iniziale maiuscola ma senza spazi), facendo attenzione anche al caso in cui una sequenza di lettere consecutive fosse in maiuscolo; ad esempio, anche il termine *StudentID* viene riconosciuto come scritto in camel case;
  - *String unCamelize(String)* → nel caso in cui la funzione *isCamel* abbia riconosciuto

come scritto in camel case un dato composto, *unCamelize* si occupa di scinderlo in una stringa con uno spazio a dividere i termini distinti. Anche in questo caso, è stato preso l'accorgimento di scindere correttamente stringhe come *StudentID* in "*Student ID*" e non, erroneamente, "*Student I D*"**.**

Capitolo 4

4 Analisi dei risultati

L'algoritmo, la cui realizzazione è stato lo scopo di questa tesi, è stato testato su sette differenti sorgenti contenute in due file in formato XML.

Le sorgenti utilizzate nella fase di valutazione dell'algoritmo rappresentano uno degli scenari di applicazione previsti all'interno del progetto *MIUR FIRB Network Peer for Business* (NeP4B). Maggiori informazioni riguardanti il progetto sono reperibili all'indirizzo http://www.dbgroup.unimo.it/nep4b. In particolare, tali sorgenti sono state estratte da siti web di compagnie di software, automazione etc., contenenti informazioni relative al profilo aziendale ed ai prodotti offerti da ciascuna compagnia. Tali sorgenti, inoltre, sono risultate particolarmente indicate per la valutazione dell'algoritmo proposto in questa tesi, essendo ricche di termini composti.

Il programma Java realizzato è stato fatto partire due volte, una per ciascun file, e ogni composto presente nelle sorgenti è stato inserito nella tabella *Disambiguazione*. In complesso, per ogni sorgente sono stati individuati dai 57 agli 85 composti, con l'eccezione di una sola sorgente la quale ne presentava appena 16. L'efficacia dell'algoritmo realizzato è stata verificata attraverso il calcolo dei valori di *precision* e *recall*:

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

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

Figura 7: Le forumule di precision e recall

Nome Sorgente	Numero di composti identificati	Numero di risposte date	Numero di risposte corrette	Precision (%)	Recall (%)
Archivist	57	44	40	93.18	71.93
CastGroup	60	44	40	95.55	70
Delin	60	44	40	95.55	70
GeneralTeleinformatica	57	43	40	93.02	70.18
GrafficheAlice	16	11	11	100	68.75
1_CopiaModena	85	67	60	89.55	70.59
1_Elefondati	74	62	51	82.26	68.92

Tabella 3: La tabella Disambiguazione: i risultati per le sorgenti considerate

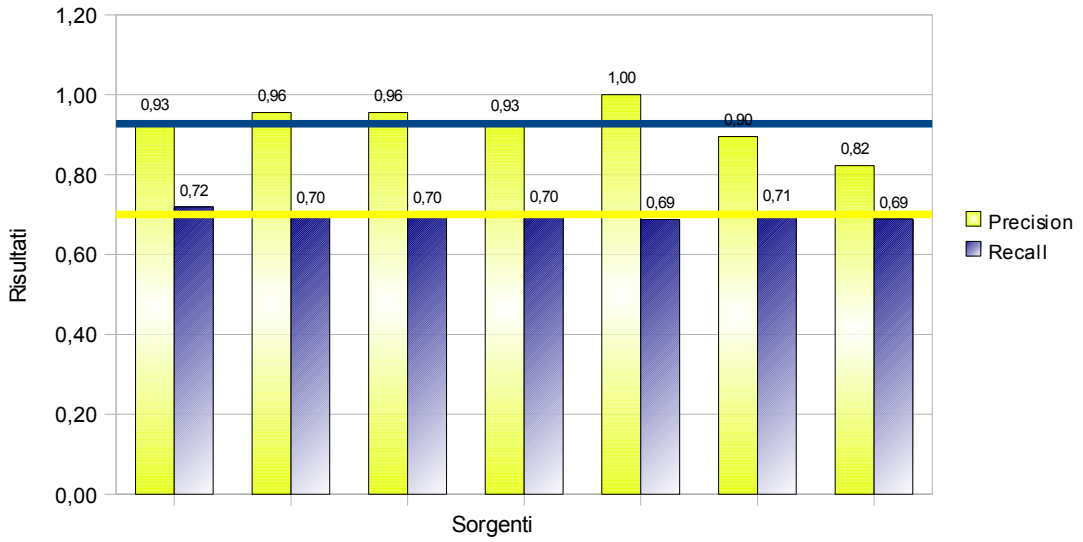

Figura 8: Precision e recall per le sorgenti esaminate con valore medio

Dalla tabella 5 e dal grafico 4 si evince subito che i risultati ottenuti sono positivi. In particolare, sono stati ottenuti i seguenti valori medi:

	Numero di composti identificati	Numero di risposte date	Numero di risposte corrette	Precision (%)	Recall (%)
VALORE MEDIO	**58**	**45**	**40**	**92,73**	**70,05**

Tabella 4: Valori medi dei risultati ottenuti

Si deduce che, per 58 composti in media analizzati in ogni sorgente, il sistema risponde in 45 casi. Di questi, 40 sono da considerarsi come risposte positive. Pertanto, l'algoritmo realizzato ottiene un livello di *precision* che si attesta a quasi il 93%, e una *recall* di poco più del 70%.

Questi valori sono giustificati dalle definizioni stesse di *precision* e di *recall*: i 13 composti in media in cui il sistema non ha dato risposta vanno a pesare sul solo valore della *recall*, che, poiché tiene conto di tutte le risposte possibili, presenta una percentuale di quasi venti punti inferiore rispetto alla *precision*. Questo risultato rende necessario approfondire i motivi per i quali il sistema in alcuni casi non ha dato risposta.

I seguenti paragrafi presentano nel dettaglio i criteri seguiti per la verifica dei risultati ed esaminano le cause dei valori ottenuti.

4.1 I criteri dell'analisi

L'analisi dei risultati è stata effettuata completamente a mano, e sono seguiti i seguenti criteri:

- ✓ all'interno del file XML esaminati, sono stati individuati tutti i presunti composti: a seguito di una fase di “pulitura”, in cui sono stati eliminati eventuali trattini, underscore o camel case, sono stati inseriti in tabella soltanto i composti effettivi, ovvero le stringhe che contenessero più di un termine. Non sono stati considerati i termini singoli;
- ✓ una risposta è stata considerata come data dal sistema qualora questo non abbia ritornato valore NULL per almeno un campo: ad esempio, la risposta al composto *ability xrd fk* presente in tabella 4 e avente i valori di *head*, *modifier* e *solution* tutti NULL, è stata contata tra le risposte non date;
- ✓ una risposta è stata considerata come corretta qualora essa presentasse, in alternativa:

1. una corretta distinzione tra *head* e *modifier* nel caso di composto contenente un aggettivo oppure il valore corrente di *Interface* (caso valido ai soli composti che ricorressero all'interno del tag *Attribute*);
2. una corretta definizione qualora il composto sia presente in WordNet. In questo caso non è necessaria la distinzione tra *head* e *modifier*, i cui campi sono settati a NULL;
3. un albero di ipernimi che connetta il corretto synset di appartenenza dell'*head* con il corretto synset di appartenenza del *modifier*. La correttezza dei synset individuati è stata valutata in base agli ipernimi diretti: nel caso del composto *Certificate_Type*, ad esempio, il synset di appartenenza di *Type* è stato giudicato corretto in quanto il suo diretto ipernimo identificato nell'albero è *Kind*, che è stato considerato come il significato corretto. In questo caso, il termine più a destra è stato considerato di default come *head*, e quello più a sinistra come *modifier*, e così sono stati riportati in tabella;
4. nel caso in cui il composto presentasse al suo interno un verbo o un avverbio, al fine di giudicare come corretta la risposta è bastato che il sistema distinguesse correttamente il termine destro dal termine sinistro, anche se la percentuale di casi in cui si sono riscontrati verbi è molto bassa e non sono invece mai stati trovati avverbi.

4.2 Studio delle risposte non date e degli errori

In tabella 7 sono stati riportati i dati riguardanti le risposte non date dal sistema. Lo scopo è capire quali errori sono dovuti a lacune dell'algoritmo e quali sono invece dovuti a cause non prevedibili o a termini che non sono presenti all'interno di WordNet.

Nome Sorgente	Numero di composti identificati	Numero di risposte date	Numero di risposte non date perché non in WordNet	Risposte date ma sbagliate (per qualsiasi motivo)
Archivist	57	44	6	4
CastGroup	60	44	10	4
Delin	60	44	10	4
GeneralTeleinformatica	57	43	6	3
GrafficheAlice	16	11	3	0
1_CopiaModena	85	67	13	7
1_Elefondati	74	62	7	9
VALORE MEDIO	**58**	**45**	**8**	**4**

Tabella 5: Risultati dettagliati per le sorgenti

Attraverso un'analisi dettagliata di tutti i composti reperiti all'interno delle sorgenti in fase di annotazione dei risultati, è possibile risalire alla fonte delle risposte che il sistema non è stato in grado di dare o che sono state date sbagliate.

Le cause principali per una mancata risposta sono le seguenti:

- i membri di alcuni composti non sono presenti all'interno di WordNet; questo accade per varie ragioni:
  - si tratta di preposizioni, articoli oppure di altre *pos* che non sono all'interno di WordNet;
  - i termini sono stati scritti in origine con abbreviazioni o sigle dalle quali non è possibile risalire al termine originario. Ad esempio, sono ricorrenti abbreviazioni, come *comm* o *ref*, e sigle come *xrd* o *fk*;
- a causa della quantità di operazioni simultanee gestite dal programma, si sono verificati mancati inserimenti in tabella apparentemente ingiustificati: si è visto che sono dovuti principalmente a problemi imprevedibili in fase di parsing o momentanei fallimenti della connessione con il database o con WordNet.

D'altro canto, si sono registrati casi in cui il programma ha dato risposta sbagliata. Questi sono principalmente dovuti a:

- è stata reperita la definizione sbagliata qualora il composto fosse presente in WordNet. Ad

esempio, per il composto *street name* è stata ritornata la glossa: *"an alternative name that a person chooses or is given (especially in inner city neighborhoods); "her street name is Bonbon""*, che non è corretta rispetto al significato di *street name* nel contesto, inteso come "*the name of a street"*;

- è stato considerato il synset sbagliato nel caso in cui dovesse essere riportato l'albero di ipernimia, nel quale il parente comune ottenuto non aveva senso se sostituito al composto orginario;
- il sistema ha a volte provato a distinguere *head* e *modifier* anche in caso in cui uno dei due membri fosse inesistente all'interno di WordNet, nonostante a livello di codice siano stati inseriti tutti i controlli opportuni affinché questo non si verificasse.

4.3 Considerazioni

I risultati ottenuti sono da considerarsi pienamente soddisfacenti. Infatti la percentuale media di risposte date dal sistema è del 77%. Focalizzando l'attenzione sul restante 23% medio di risposte non date, si evince che il 59% delle volte una risposta non è stata data poiché un membro non era presente all'interno di WordNet. Si tratta di un eventualità che ha a che fare solamente con il modo in cui i dati sono stati inseriti nelle sorgenti originarie, ma nulla a che vedere con l'efficacia dell'algoritmo. In altre parole, le cause di una mancata risposta sono da imputarsi ad errori di codice oppure ad errori computazionali soltanto nel 41% degli errori registrati.

La tabella successiva riporta i risultati dettagliati:

Nome Sorgente	Risposte non date	Risposte non date/composti totali (%)	Risposte non in WN	Risposte non in WN/ risposte non date (%)
Archivist	13	23	6	46
CastGroup	16	27	10	63
Delin	16	27	10	63
GeneralTeleinformatica	14	25	6	43
GrafricheAlice	5	31	3	60
1_CopiaModena	18	21	13	72
1_Elefondati	12	16	7	58
VALORE MEDIO	**13**	**23**	**8**	**59**

Tabella 6: Analisi dettagliata delle risposte non date dal sistema

Conclusioni e sviluppi futuri

Il lavoro svolto per la presente tesi ha consentito di realizzare un programma Java per la risoluzione dei termini composti all'interno di documenti strutturati e semi-strutturati. L'indagine preliminare alla realizzazione dell'algoritmo si è svolta partendo dagli strumenti necessari per l'analisi dei termini e giungendo alla discussione dei vari metodi di risoluzione e di disambiguazione dei termini composti presenti in letteratura. Attraverso lo studio di questi, è stato possibile dedurre quali fossero le informazioni, le tecniche e i mezzi basilari per implementare un algoritmo nuovo, che fosse il più possibile portabile ed indipendente dal dominio di applicazione.

Al termine dell'analisi dei risultati ottenuti, sono stati registrati livelli di precisione (numero di risposte corrette date dall'algoritmo in rapporto al numero di risposte date) estremamente positivi in relazione a quanto effettivamente richiesto dall'algoritmo implementato. La richiesta primaria è stata quella di distinguere il termine principale dal modificatore tra i due membri di un composto. In alternativa, a meno che il composto non fosse già presente in WordNet, è stato riportato l'albero di ipernimia. L'ipernimia è quella relazione che intercorre tra due termini t_1 e t_2 per la quale, se t_1 è una sottoclasse di t_2 (cioè t_1 *is a* t_2), allora si dice che t_2 è ipernimo di t_1, oppure che t_1 è iponimo (relazione di iponimia) di t_2. Utilizzando solo questo tipo di relazione, è evidente che viene perso il contenuto informativo che si avrebbe utilizzando anche relazioni di tipo *part of* (meronimia) e *has part* (olonimia). Tuttavia, allo stato attuale di WordNet, l'ipernimia è il tipo di relazione che garantisce sempre che venga reperito un albero di relazioni, indipendentemente dal dominio di provenienza dei termini composti reperiti. A volte, l'albero delle relazioni di ipernimia ha dato buoni risultati: risalendo lungo le catene di ipernimi di due membri di un composto, si è trovato un termine che potesse, con buona approssimazione, essere sostituito al composto originario.
Obiettivo per il futuro è quello di reperire all'interno di un dizionario concettuale, che sia possibilmente più ricco di WordNet, anche il più grande termine “figlio” in comune ai due membri (navigando le relazioni di iponimia e meronimia) del composto nella gerarchia, così da ottenere un termine che riassuma l'informazione riguardante entrambi.

Attraverso un'analisi dettagliata, è stato successivamente evidenziato come gli eventuali errori in fase di risoluzione o le mancate risposte da parte del sistema fossero in larga parte dovute a

mancanze di WordNet. Il database lessicale, purtroppo, presenta alcuni difetti:

- le relazioni di tipo lessicale e semantico presenti sono poche e troppo poco specifiche; per questo motivo l'ipernimia è stata la sola relazione sempre utilizzabile, anche se purtroppo il parente comune ai due membri del composto trovato era spesso di livello così alto all'interno della gerarchia di WordNet da far perdere tutto il contenuto informativo del composto;
- il grado di specificità con cui vengono specializzati i synset (ovvero dei set di sinonimi), soprattutto per quelli di basso livello, è eccessivo; questo ha portato a volte al reperimento del synset sbagliato. È evidente come in sorgenti contenenti termini composti formati da lemmi generici i risultati saranno migliori che nel caso in cui fossero presenti lemmi estremamente specifici;
- d'altro canto, sono assenti i termini tecnici. Questo ha effetto qualora l'algoritmo venga applicato a sorgenti in cui i termini composti siano in larga parte di dominio tecnico.

Questi difetti hanno pesato soprattutto sulle risposte non date dal sistema, risultando in una *recall* mai superiore al 72%. La *recall* è, infatti, calcolata come il numero di risposte corrette date dal sistema in rapporto a tutte le possibili risposte.

L'attività svolta all'interno di questa tesi merita di essere continuata, approfondita e perfezionata. In particolare, sono questi i punti sui quali sarebbe interessante focalizzarsi al fine di realizzare un metodo per la risoluzione e la disambiguazione di termini composti sempre migliore:

- possibilità di sostituire al composto un termine singolo che ne contenga tutta l'informazione, possibilmente il termine “figlio” più grande in comune ai due membri, in modo da realizzare appieno la fase di disambiguazione (ovvero l'assegnazione di un significato univoco) dei termini composti;
- identificazione e suddivisione dei modificatori per composti di più di due termini e riconoscimento delle mutue relazioni;
- ricerca e utilizzo di risorse migliori e più approfondite del solo database lessicale WordNet, così da poter ricercare relazioni più specifiche.

Bibliografia

Articoli consultati

[1] *The MOMIS Methodology for Integrating Heterogeneous Data Sources*, D. Beneventano, S. Bergamaschi, IFIP World Computer Congress. Toulouse France, 22-27 August 2004.

[2] *Extending A Lexicon Ontology For Intelligent Information Integration*, R. Benassi, S. Bergamaschi, A. Fergnani, D. Miselli, European Conference on Artificial Intelligence (ECAI2004). Valencia, Spain, 22-27 August 2004.

[3] *Automatic Annotation For Mapping Discovery In Data Integration Systems*, S. Bergamaschi, L. Po, S.Sorrentino, Proceedings of the Sixteenth Italian Symposium on Advanced Database Systems, SEBD 2008, Mondello, PA, Italy, 22-25 June 2008.

[4] *Introduction to WordNet: An On-line Lexical Database*, George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross and Katherine Miller, 1993.

[5] *Conceptual Association For Compound Nouns Analysis*, Mark Lauer, 1994.

[6] *Development And Application Of A Metric On Semantic Nets*, Rada et al., 1989.

[7] *Evaluating WordNet-based Measures Of Lexical Semantic Relatedness*, Alexander Budanitsky, Greame Hirst, 2006.

[8] *Information Retrieval Based On Conceptual Distance In IS-A Hierarchies*, Lee et al., 1993.

[9] *WordNet::Similarity – Measuring The Relatedness Of Concepts*, Ted Pedersen, Siddharth Patwardhan, Jason Michelizzi, 2004.

[10] *Semantic Distance In WordNet: An Experimental, Application-oriented Evaluation Of Five Measures*, Alexander Budanitsky, Graeme Hirst, 2001.

[11] *Using Information Content To Evaluate Semantic Similarity In A Taxonomy*, Philip Resnik, 1995.

[12] *Lexical Chains As Representations Of Context For The Detection And Correction Of Malapropism*, Graeme Hirst, David St-Onge, 1998.

[13] *An Information-Theoretic Definition Of Similarity*, Dekang Lin, 1998.

[14] *Designing Statistical Language Learners: Experiments On Noun Compounds*, Mark Lauer, PhD Thesis, 1995.

[15] *Corpus Statistics Meet The Noun Compound: Some Empirical Results*, Mark Lauer, 1995.

[16] *A Theory Of Syntactic Recognition For Natural Language*, Mitchell Marcus, 1980.

[17] *Lexical Semantic Techniques For Corpus Analysis*, J. Pustejovsky, S. Bergler, P. Anick, 1993.

[18] *The Stress And Structure Of Modified Noun Phrases In English*, M. Liberman, R. Sproat, 1992.

[19] *Selection And Information: A Class-based Approach To Lexical Relationships*, Philip Resnik, PhD Thesis, 1993.

[20] *Disambiguating Noun Compounds With Latent Semantic Indexing*, Alan M. Buckeridge, Richard F.E. Sutcliffe, 2002.

[21] *An Empirical Model Of Multiword Expression Decomposability*, Timothy Baldwin, Colin Bannard, Takaaki Tanaka, Dominic Widdows, 2003.

[22] *Idioms*, Geoffrey Nunberg, Ivan A. Sag, Tom Wasow, 1994.

[23] *A Corpus-based Approach To Automatic Compound Extraction*, Keh-Yih Su, Ming-Wen Wu, Jing-Shin Chang, 1994.

[24] *Algorithm For Automatic Interpretation Of Noun Sequences*, Lucy Vanderwende, 1994.

[25] *Semi-Automatic Recognition Of Noun-Modifier Relationships*, Ken Barker, Stan Szpakowitcz, 1998.

[26] *The Descent Of Hierarchy, And Selection In Relational Semantics*, Barbara Rosario, Marti A. Hearst, Charles Fillmore, 2002.

[27] *Benchmarking Noun Compounds Interpretation*, Su Nam Kim and Timothy Baldwin, 2008.

[28] *Models For The Semantic Classification Of Noun Phrases, Dan Moldovan*, Marta Tatu, Daniel Antohe and Roxana Girju, 2004.

[29] *Automatic Interpretation Of Noun Compounds Using WordNet::Similarity*, Su Nam Kim, Timothy Baldwin, 2005.

[30] *Interpreting Noun Compounds Using Bootstrapping And Sense Collocation*, Su Nam Kim, Timothy Baldwin, 2007.

[31] *The Knowledge Required To Interpret Noun Compounds*, James Fan, Ken Barker, Bruce W. Porter, 2003.

[32] *Indexing by Latent Semantic Analysis,* Scott Deerwester, Susan T. Dumais, Richard Harshman, 1990.

Siti internet consultati

http://www.dbgroup.unimo.it

http://www.dbgroup.unimo.it/Momis/

http://www.dbgroup.unimo.it/pubs.html#III

http://wordnet.princeton.edu/

http://wn-similarity.sourceforge.net/

http://portal.acm.org/portal.cfm

http://citeseerx.ist.psu.edu/

http://www.nlm.nih.gov/mesh/

http://nlp.stanford.edu/nlp/javadoc/jwnl-docs/allclasses-noframe.html

http://www.rednoise.org/rita/wordnet/documentation/index.htm

http://processing.org/

http://www.eclipse.org/

http://lsa.colorado.edu/

http://www.dbgroup.unimo.it/nep4b

Ringraziamenti

Un sentito ringraziamento al relatore, Professoressa Sonia Bergamaschi, per avermi dato la possibilità di svolgere questa tesi e per la cortesia e la professionalità con cui mi ha assistito.

Rivolgo anche un grande ringraziamento all'Ing. Serena Sorrentino, per l'interesse e la disponibilità con cui mi ha sempre seguita anche nei momenti di maggiore difficoltà nella realizzazione di questo lavoro.

Infine, desidero ringraziare tutte le persone a me vicine, con particolare attenzione al mio ragazzo Stefano per il grande sostegno e l'incoraggiamento datimi.