UNIVERSITÀ DEGLI STUDI
DI MODENA E REGGIO EMILIA

Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Tecniche Efficaci di Schema Disambiguation Indipendenti dalle Lingue

Sandra Mantovani

Tesi di Laurea

Relatore:
Prof. Riccardo Martoglia

Anno Accademico 2007/2008

RINGRAZIAMENTI

Ringrazio l'Ing. Riccardo Martoglia per l'incessante disponibilità e assistenza. Rivolgo un ringraziamento particolare ai miei genitori, a mia sorella e tutti i miei amici che in questi anni mi hanno sempre sostenuto.

PAROLE CHIAVE

Word Sense Disambiguation

Multilinguismo

Schema

Dizionario

POS Tagger

Indice

Introduzione **1**

Parte I – Il caso di studio **4**

1. Word Sense Disambiguation **5**

1.1 Letteratura 7

1.1.1 Free text WSD: Identificazione del contesto e quantificazione della correlazione tra termini 7

1.1.2 Tecniche di *structural disambiguation* e annotazione 9

1.2 STRIDER 11

1.3 Pre-elaborazione degli schemi 14

1.4 Espansione del contesto 15

1.4.1 Estrazione del contesto del grafo 16

1.4.2 Estrazione del contesto nascosto 19

1.5 Algoritmi di disambiguazione 20

1.5.1 Correlazione dei termini 21

1.5.2 Contributo del contesto del grafo 23

1.5.3 Contributo del contesto nascosto 25

1.5.4 Contributo della frequenza d'uso 28

1.6 Feedback 29

1.7 Le fonti di conoscenza esterne 30

2. Strumenti linguistici **33**

2.1 Dizionari 33

2.2 Interrogazione dei dizionari on-line 34
2.3 Dizionari con struttura ad ipernimi 35
2.3.1 ItalWordNet.....36
2.3.2 MultiWordNet 37
2.4 Dizionari non strutturati 38
2.4.1 Dizionari classici 38
2.4.2 Dizionari collaborativi.....42
2.5 Dizionari machine-generated 44
2.6 Conclusioni.....45
2.7 Stemming 46
2.8 Part-of-speech tagger.....46
2.8.1 TreeTagger 48
2.8.2 TextPro 50
2.8.3 Altri POS tagger 51
2.9 Parser 52
2.10 Morph-it! 53
2.11 Conclusioni.....55

Parte II – Estensione di STRIDER 56
3. Progetto 57
3.1 Requisiti funzionali 57
3.2 Scenario: Estrazione dei sostantivi relativi al termine dato 59
3.3 Use Case Diagram 60
3.3.1 Use case 01: Inserimento parametri 62
3.3.2 Use case 02: Richiesta pagina WEB 62
3.4 Activity Diagram 63
3.5 Componenet diagram e classi Java del progetto 66
3.6 Morph-it! come POS tagger 69
3.7 Integrazione, parametri e modifiche all’interfaccia grafica 71
4. Prove sperimentali 77
4.1 Experimental setting.....77

4.2 Prove sperimentali 80
4.2.1 Confronto tra il dizionario 1 e il dizionario 2 80
4.2.2 Valutazione del contributo degli esempi nei dizionari 1 e 2 81
4.2.3 Confronto tra il contesto standard e il contesto completo 83
4.2.4 Confronto tra *ebay* e *miniEbay* 86
4.2.5 Confronto tra dizionario 1, dizionario 2 e dizionario 3 87
5. Conclusioni e sviluppi futuri 90

Parte III – Appendici 93
A. Il *crossing set* di default usato nella Parte I 94
B. Elenco delle possibili "features" di Morph-it! 95
C. Codice SQL per la creazione dello strumento linguistico che utilizza Morph-it! e il nuovo tag set 99
D. Schemi SQL e contesto standard 102

Indice delle immagini

Figure

Figura 1. Architettura del sistema STRIDER. 12
Figura 2. Esempio ordine d'acquisto in lingua inglese, *Purchase order*. 13
Figura 3. Porzione del grafo OIM-RDF dello schema d'esempio. 15
Figura 4. Componenti della fase di estrazione del contesto. 16
Figura 5. Esempio di selezione dei nomi e del calcolo dei pesi per la parola *line*.18
Figura 6. Overview dei differenti contributi per la disambiguazione dei termini. 20
Figura 7. Esemplificazione grafica dell'algoritmo per l'estrazione del contesto del grafo, per la disambiguazione del generico termine t. 24
Figura 8. Esemplificazione grafica dell'algoritmo per l'estrazione del contesto nascosto del generico termine t. 26
Figura 9. Screenshot di una ricerca effettuata con la versione on-line di IWN. ... 36
Figura 10. Screenshot di una ricerca effettuata con la versione on-line di MWN. 37
Figura 11. Screenshot di una ricerca effettuata con la versione on-line del dizionario "De Mauro Paravia". 39
Figura 12. Screenshot di una ricerca effettuata con la versione on-line del dizionario "il Sabatini Coletti". 40

Figura 13. Screenshot di una ricerca effettuata con la versione on-line del dizionario “Garzanti”. 41
Figura 14. Screenshot di una ricerca effettuata con la versione on-line del “Dizionario Italiano”. 43
Figura 15. Screenshot di una ricerca effettuata con la versione on-line del “Wikitionary”. 44
Figura 16. Screenshot di una delle versioni on-line del tool “TreeTagger”..... 49
Figura 17. Screenshot della versione on-line del tool “TextPro”..... 51
Figura 18. Screenshot della versione on-line del database “Morph-it!”. 53
Figura 19. Diagramma degli attori. 60
Figura 20. Prima versione di UCD relativo allo scenario “Creazione della struttura dati contenente i sostantivi relativi alle definizioni del termine dato”. 61
Figura 21. Seconda versione di UCD relativo allo scenario “Creazione della struttura dati contenente i sostantivi relativi alle definizioni del termine dato”. 61
Figura 22. Activity diagram relativo alla sequenza di operazioni per estrarre i nomi. 64
Figura 23. Activity diagram relativo all’estrazione dei nomi per i termini polisemici. 65
Figura 24. Component diagram relativo alle classi Java del progetto. 69
Figura 25. Screenshot della parte del pannello di configurazione con i nuovi parametri “dizionarioIta” e “language”. 71
Figura 26. GUI originale di STRIDER. Particolare sulla terzultima colonna di destra, contenente le gerarchie di ipernimi. 72
Figura 27. GUI per la lingua italiana, differisce dall’originale per la colonna contenente le gerarchie di ipernimi, sostituita con gli esempi delle definizioni. 73
Figura 28. Contribuiti di disambiguazione per gli schemi in lingua italiana. 74
Figura 29. Parametri del pannello di configurazione per l’esecuzione con schemi in italiano. 76
Figura 30. Schema SQL *cinema*. 103

Figura 31. Schema SQL *ebay*. 103
Figura 32. Schema SQL *miniEbayAbbigliamento*. 104
Figura 33. Schema SQL *miniEbayFilm*. 104
Figura 34. Schema SQL *miniEbayMusica*. 105
Figura 35. Schema SQL *ordine*. 105
Figura 36. Schema SQL *università*. 106
Figura 37. Schema SQL *vacanza*. 106

Tabelle

Tabella 1. Rappresentazione del contesto nascosto del termine (line, 5)..... 19
Tabella 2. Esempio di output dell'applicazione TreeTagger, per la frase inglese "The TreeTagger is easy to use."..... 50
Tabella 3. Esempio di categorizzazione delle parole presenti nel database Morph-it!. 54
Tabella 4. Caratteristiche degli schemi utilizzati. Per la colonna dizionario, la legenda è la seguente: 1 "De Mauro Paravia", 2 "il Sabatini Coletti", 3 di Google™. 78
Tabella 5. *Crossing set* di default per ogni formato accettato da STRIDER. 94

Grafici

Grafico A. Confronto tra il dizionario 1 e 2 sullo schema *ordine*, utilizzando gli esempi nelle definizioni. 81
Grafico B. Confronto tra i valori di *precision* dei dizionari 1 e 2 valutati sullo schema *cinema*, con e senza esempi. 82

Grafico C. Confronto tra i valori di *precision* dei dizionari 1 e 2 valutati sullo schema *vacanza*, con e senza esempi. 82
Grafico D. Confronto dei risultati per lo schema *ebay* valutato sul dizionario 1, variando il contesto..... 84
Grafico E. Confronto dei risultati per lo schema *vacanza* valutato sul dizionario 1, variando il contesto..... 85
Grafico F. Confronto dei risultati ottenuti con il dizionario 2 per lo schema *ebay*, miniEbayAbbigliamento, miniEbayFilm e miniEbayMusica utilizzando il contesto standard. La seconda parte del grafico rappresenta una media di tutti i valori ottenuti con la serie miniEbay. 86
Grafico G. Confronto dei risultati ottenuti per lo schema *miniEbayAbbigliamento*, valutato sul dizionario 1, 2 e 3 con esempi, nel contesto standard..... 88
Grafico H. Confronto dei risultati ottenuti per lo schema *miniEbayFilm*, valutato sul dizionario 1, 2 e 3 con esempi, nel contesto standard..... 89

Algoritmi

Algoritmo 1. *GraphContextContr*(*t*,*N*) 23
Algoritmo 2. *HiddenContextContr*(*t*,*N*) (versione indipendente dalle gerarchie). 27

Introduzione

La questione del *Word Sense Disambiguation* (WSD), ha suscitato notevole interesse nel campo dell'informatica già dagli anni '50. Essa ha l'obiettivo di assegnare il significato corretto ad un termine ambiguo, a seconda del contesto in cui esso si trova, un processo che la mente umana effettua spontaneamente. Il problema nasce dal fatto che le parole utilizzate nel linguaggio naturale possono assumere diversi significati, che variano a seconda dell'ambito in cui sono inserite. Per questo motivo, la questione del WSD risulta essere di notevole interesse negli svariati campi dell'informatica che trattano il linguaggio naturale, come la traduzione automatica, l'information retrieval, i sistemi di parsing, di question answering, di dialogo e più in generale in tutti quei sistemi in cui è richiesto un livello di analisi semantica. L'ambito del WSD che viene maggiormente studiato è quello che rende il processo di disambiguazione totalmente automatizzato, senza nessun intervento dell'utente.

In particolare, lo *Schema Disambiguation* è quel processo che consente di eliminare le ambiguità semantiche dei termini che costituiscono uno schema, applicando le tecniche algoritmiche fornite dal WSD. Con il termine schema intendiamo un insieme di informazioni rappresentate in una struttura, che può essere in diversi formati, come SQL o XML; esso può contenere ad esempio le categorie di ebay, le tabelle che costituiscono un database, e così via. Una volta che è stata effettuata la disambiguazione di tutti i termini, lo schema viene annotato, cioè viene appuntato il significato corretto accanto ad ogni parola. Nello

Schema Disambiguation, il contesto di una data parola sarà determinato dagli altri termini che costituiscono lo schema, proporzionalmente alla loro vicinanza con la parola stessa; di conseguenza verrà utilizzata la loro influenza per individuarne il senso corretto. Nel processo di disambiguazione, è necessario utilizzare una fonte di conoscenza esterna per ricercare i significati dei termini e per valutare la correlazione tra parole; a questo proposito vengono impiegati dizionari, corpora testuali e motori di ricerca on-line.
In letteratura esistono diversi sistemi che si occupano di Schema Disambiguation totalmente automatizzato, un esempio è STRIDER; questa è un'applicazione Java creata dall'*Information Systems Group* (ISGroup), un gruppo di ricerca con sede presso il Dipartimento di Ingegneria dell'Informazione (DII). Il progetto in questione consente, applicando le tecniche di Word Sense Disambiguation, di disambiguare i termini contenuti in schemi in formati differenti. L'applicazione è molto legata a WordNet, un dizionario strutturato ad ipernimi per la lingua inglese; esso è il più utilizzato nell'ambito del WSD, in quanto fornisce sia le definizioni testuali tipiche di un vocabolario, ma anche informazioni aggiuntive relative agli ipernimi e alla frequenza dei lemmi nella lingua inglese.

Lo studio condotto nell'ambito di questa tesi, ha avuto lo scopo di valutare e implementare la possibilità di rendere STRIDER indipendente dalla lingua; infatti l'applicazione è sempre stata utilizzata per la lingua inglese con l'ausilio del dizionario WordNet. Proprio per questo è stata condotta un'analisi di nuove fonti di conoscenza esterne e strumenti linguistici da cui attingere nella fase di annotazione, pre-elaborazione del testo e valutazione della correlazione tra termini. In particolare sono state effettuate delle ricerche relative alle risorse disponibili per la lingua italiana, valutando le fonti di conoscenza esterne che potevano essere applicabili. Per supportare l'annotazione, è stato necessario analizzare i dizionari disponibili, sia quelli strutturati ad ipernimi, sullo stile di WordNet, che quelli tradizionali contenenti semplici definizioni ed esempi, come i dizionari classici e i collaborativi reperibili da internet. Nella fase di pre-elaborazione del testo, sono state effettuate ricerche riguardanti alcuni strumenti

linguistici, come i POS tagger e i database di parole categorizzare, per la lingua italiana. In più, per valutare la correlazione tra termini, sono stati analizzati i dati relativi alle frequenze di occorrenza delle parole forniti da alcuni motori di ricerca; quest'ultimo studio era già stato effettuato per la versione in lingua inglese di STRIDER.
A seguito della ricerca e della valutazione delle fonti di conoscenza esterne e degli strumenti linguistici impiegati nella fase di annotazione, è stata implementata la nuova funzionalità per la lingua italiana, dando molta importanza alla modularità del codice, in modo da rendere agevole l'estensione ad un'altra lingua, perorando l'obiettivo del multilinguismo. A questo proposito è da notare che il percorso seguito nella fase di analisi e implementazione è ripercorribile per una qualsiasi altra lingua e le nuove estensioni possono agevolmente coesistere nella stessa versione, rendendo STRIDER un sistema multilingue.
Per ultimo è stata effettuata una valutazione sperimentale, basata sulla stima dell'efficacia; sono stati testati alcuni dizionari on-line, con peculiarità diverse, su schemi SQL caratterizzati da molteplici argomenti e dimensioni, variando il contesto dei termini.

La trattazione della tesi è stata suddivisa in cinque capitoli, che ora verranno brevemente illustrati. Il capitolo 1 esporrà un approfondimento relativo alla letteratura presente sul Word Sense Disambiguation e descriverà la struttura di STRIDER assieme agli algoritmi che implementa. Il capitolo 2 presenterà le fonti di conoscenza esterna e gli strumenti linguistici per la lingua italiana, che sono stati studiati per l'integrazione nell'applicazione. Il capitolo 3 presenterà, tramite schemi UML, la struttura del progetto e le nuovi classi Java che sono state implementate. Il capitolo 4 esporrà le prove sperimentali effettuate per testare il funzionamento della nuova versione di STRIDER su schemi in lingua italiana, utilizzando tre dizionari on-line differenti. Nel capitolo 5 verranno trattate le conclusioni e alcune idee per nuove estensioni. Infine, nelle appendici A, B, C e D, sono riportate informazioni utili per la comprensione degli esempi della Parte I, i codici SQL, i tag set e gli schemi utilizzati nelle prove sperimentali.

Parte I:

Il caso di studio

Capitolo 1

Word Sense Disambiguation

Word Sense Disambiguation (WSD) è il problema di assegnare il senso corretto di un termine a seconda del contesto in cui esso si trova, un processo automatico e ben conosciuto dalla mente umana. Questo problema nasce dal fatto che i termini utilizzati nel linguaggio e più in generale nelle informazioni sono ambigui. Per esempio, ricercando la parola *batteria* sul sito di Ebay®, è possibile notare che è assegnato a categorie come “Telefonia e cellulari”, “Informatica”, “Fotografia”, “Elettronica di consumo”, “Strumenti musicali”, “Case, arredamento e bricolage” e così via. WSD è quindi un tipico problema di classificazione: data una parola e i suoi possibili significati, definiti da una fonte di dati esterna, come ad esempio un dizionario, classifica la sua occorrenza in un determinato contesto, assegnandola a uno o più dei suoi sensi.

Ad esempio, determinare il senso del termine *batteria* inserito nel seguente contesto:

“Nuova batteria acustica completa di piatti, sgabello e bacchette ”

utilizzando la definizione fornita dal dizionario di lingua italiana on-line “Il Sabatini Coletti”:

batteria [bat-te-rì-a] s.f

1 Gruppo di fuoco dell'artiglieria costituito da quattro o sei pezzi di uguale calibro: *b. da campagna, contraerea* || figg. scoprire le proprie b., mostrare i propri progetti, le proprie capacità operative | dare fuoco alle b., attaccare risolutamente

2 Insieme di elementi, uguali o affini, cooperanti in una funzione comune: *una b. di cani da caccia* || b. da cucina, attrezzatura completa di pentole | allevamento in b., quello intensivo, in cui gli animali sono tenuti sempre al chiuso, a temperatura costante | b. solare, apparecchiatura che trasforma la luce in energia elettrica

3 Ogni oggetto, apparecchiatura che ha la funzione di accumulare e trasferire energia elettrica: *la b. è scarica*; *caricare la b.* || fig. avere le b. scariche, essere stanco, spossato, depresso

4 mus. Insieme degli strumenti a percussione in un'orchestra o in un complesso musicale: *suonare la b.*

5 sport. Prova eliminatoria nelle competizioni sportive: *vincere una b.*

Esistono tre tipi di tecniche di WSD, ognuna basata su un approccio differente:

- Metodi basati sull’intelligenza artificiale (*AI-based methods*);
- Metodi che utilizzano la conoscenza (*Knowledge-driven methods*);
- Metodi basati su corpus o dati (*Corpus-based* o *Data-driven methods*).

L’approccio utilizzato nel sistema STRIDER[1], è di tipo *Knowledge-driven*, e consiste quindi nell’utilizzo di un dizionario o del WWW come fonte di conoscenza esterna. In alcuni casi, verrà anche introdotto il metodo *Corpus-based*, utilizzando motori di ricerca via web, che gestiscono grandi quantità di informazioni, estratte senza elaborazione da parte dell’uomo.

[1] STRucture-based Information Disambiguation ExpeRt

Lo Schema Disambiguation, è quel processo che consente di eliminare l'ambiguità dei termini contenuti in uno schema in un particolare formato, utilizzando le tecniche fornite dal WSD; usufruendo delle informazioni fornite degli altri termini che costituiscono lo schema, è possibile disambiguare efficacemente i termini che lo compongono. Nell'applicazione STRIDER, vengono proposti diversi algoritmi applicabili a schemi di vario genere, quali ad esempio SQL, RDF, XML.
Nelle prossime sezioni verranno riportate informazioni riguardanti la letteratura relativa al problema del WSD, e la struttura del sistema STRIDER; le nozioni presenti sono state tratte dagli articoli [Mandreoli et al. 2005], [Mandreoli et al. 2006] e [Mandreoli et al. submitted].

1.1 Letteratura

L'approccio utilizzato in STRIDER prende spunto da alcune idee e articoli riguardanti il problema di disambiguazione applicata al *free text*. Con "Free Text WSD", si intende il problema del Word Sense Disambiguation applicato a testi non strutturati, come ad esempio paragrafi e frasi, che differenziano notevolmente da quelle strutturati, quali schemi SQL, RDF, XML e così via.
La prossima sezione tratterà nella prima parte di questa letteratura, e nelle successive delle tecniche disponibili nel campo di annotazione e disambiguazione strutturale.

1.1.1 Free text WSD: Identificazione del contesto e quantificazione della correlazione tra termini

La free text WSD è sicuramente il genere di Word Sense Disambiguation più frequentemente studiata, e per la quale sono presenti più tecniche risolutive. Molti

di questi metodi sono adattabili al problema dello Schema WSD, e per questo motivo è fondamentale comprendere il loro funzionamento. Il meccanismo di eliminazione dell'ambiguità che viene applicato è duplice: in primo luogo è necessario identificare e rappresentare correttamente il contesto del termine da disambiguare, in secondo luogo è necessario trovare un metodo per correlare semanticamente il termine al contesto.
E' comunemente noto che è fondamentale la ricerca del contesto di una parola per poterla disambiguare, ed è possibile tramite due differenti approcci:

- *The bag of words approach*, nel quale il contesto è costituito solo dall'insieme delle parole vicine al termine in questione;
- *The relational information approach*, che estende il precedente aggiungendo alcune informazioni riguardanti la loro distanza o la relazione che le coinvolge.

L'approccio utilizzato in STRIDER è il secondo, che è il più completo e generalmente fornisce un risultato migliore [Ide and Veronis 1998].
Nello stesso modo sono stati proposti vari metodi per quantificare la correlazione e quindi assegnare un senso ad ogni termine. Come spiegato precedentemente, quello adottato è di tipo *Knowledge-driven*; questo approccio estrae le informazioni per la correlazione dei termini da una fonte di conoscenza esterna, come ad esempio un dizionario elettronico; per la lingua inglese, il dizionario WordNet è sicuramente la fonte di conoscenza esterna più utilizzata [Budanitsky and Hirst 2001; Li uet al. 2005; Mihalcea et al. 2004], in quanto fornisce tramite le gerarchie di ipernimi un efficiente supporto per la misura della similarità; la forma più comune è basata sui percorsi, *path*, dei termini [Leacock anh Chodorow 1998].
Nuove ricerche scientifiche hanno proposto un ulteriore metodo per quantificare la correlazione dei termini, basato sulle co-occorrenze delle parole. Visto che questo approccio richiede un insieme di corpora molto amplio per poter garantire una buona efficacia, ha avuto una rapida diffusione solo da quando è stato

possibile utilizzare le informazioni presenti nel web [Cimiano et al. 2004]. Ad esempio, in [Terra and Clarke 2003] gli autori propongono varie formule per il calcolo delle co-occorrenze applicabili al WSD, dove le frequenze statistiche sono derivate da enormi corpora estratti con tecniche di ricerca nel web. In particolare, la formula del *Pointwise Mutual Information* (PMI), derivata da altri studi [Turney 2004] è una delle più efficaci [Cilibrasi and Vitanyi 2007], ed è utilizzata per definire una nuova misura di similarità, la "Google Distance"; questa si basa unicamente sulle frequenze del motore di ricerca Google™, e i test condotti dagli autori mostrano risultati paragonabili a quelli ottenibili con il dizionario WordNet. Altri approcci più recenti utilizzano le informazioni contenute in Wikipedia [Strube and Ponzetto 2006], l'enciclopedia libera del web.
Anche STRIDER, utilizza in alcuni casi, il metodo *Corpus-based*, che si avvale appunto della conoscenza posseduta da un motore di ricerca; Google™ e Yahoo!® sono due dei più utilizzati.

1.1.2 Tecniche di *structural disambiguation* e annotazione

La disambiguazione di strutturle, è conosciuta come uno dei problemi più frequenti nelle applicazioni di ambito semantico. Fino ad oggi però, è stato studiato soltanto in ambiti molto ristretti, come nel caso di *schema matching* o di dati XML, e sono pochi i metodi generali che sono stati presentati. In alcune applicazioni di *schema matching*, la vicinanza semantica tra nodi si basa su approcci sintattici, come lo *string matching* che considera anche i sinonimi (es. [Melnik et al. 2002; Madhavan et al. 2001]). In più, una buona quantità di approcci statistici del WSD sono stati proposti in ambito di *matching* (es. [Madhavan et al. 2005]). Però, come già detto, hanno un'applicabilità limitata e si basano su lunghe fasi di addestramento e necessitano di dati che non sempre sono disponibili. Come per l'approccio strutturale, in [Theobald et al. 2003] gli autori propongono una tecnica per i dati XML, dove la disambiguazione è applicata ai nomi dei tag presenti nel documento. Il contesto di un tag è costituito da un

insieme di parole contenenti il nome del tag stesso, il contenuto testuale dell'elemento e il testo degli elementi ad esso subordinati, e successivamente viene espanso includendo i relativi termini recuperati con WordNet. Questo contesto viene poi confrontato con gli altri associati ai sensi di WordNet relativi al termine, per essere disambiguato in base al significato, tramite tecniche vettoriali standard. Un simile scenario viene proposto in [Tagarelli and Greco 2004], nel quale si disambigua, applicando l'algoritmo del percorso più breve (*shortest path algorithm*) in un grafo pesato, che è costruito rispecchiando il percorso dal termine alla radice e utilizzando i collegamenti tra termini forniti da WordNet. Per la costruzione del grafo, le relazioni di WordNet sono navigate per un solo passo. [Bouquet et al. 2006] presenta una tecnica di disambiguazione sfruttando la struttura gerarchica dello schema ad albero, assieme alle gerarchie di WordNet. Però, per far sì che questo tipo di approccio sia totalmente efficiente, le relazioni dello schema devono coincidere almeno in parte con quelle di WordNet. In [Mandreoli et al. 2005; 2006] gli autori presentano un approccio di disambiguazione strutturale che è applicabile a differenti contesti semantici. Il metodo mette le prime basi in quest'ambito estraendo e sfruttando informazioni sul contesto strutturale, includendo informazioni relazionali tra i nodi. Però questo approccio è strettamente dipendente dalla struttura di WordNet, in quanto utilizza metodi computazionali basati soltanto sui i collegamenti tra i termini di WordNet, ed è ottimizzato e testato solo su schemi XML ad albero, non prevedendo soluzioni per strutture di formato differente.

Per ultimo, si può considerare l'approccio di annotazione automatica [Uren et al. 2006]. I metodi attuali per l'annotazione semantica del Web suppongono la disponibilità di un'ontologia che descriva il dominio d'interesse. Più alta è l'accuratezza dell'ontologia, più alta è la precisione dell'annotazione. Sono attualmente disponibili molti frameworks e tools che forniscono gli strumenti di annotazione, come ad esempio Mangrove [McDowell et al. 2003]. Sono supportati i documenti HTML, essendo questo il formato più frequente in Internet; in ogni caso, la creazione di supporti per altri formati per l'annotazione semantica (come schemi relazionali, schemi XML, ontologie) è un problema attualmente

poco studiato. Inoltre, in molti casi, il processo di annotazione è manuale, dal momento che la maggior parte dei tools si limita a richiedere l'intervento dell'utente nello scegliere e inserire le annotazioni. D'altra parte, mentre gli strumenti totalmente automatizzati per l'annotazione di documenti Web scarseggiano, esistono alcuni sistemi semi-automatici, che aiutano l'utente nella scelta, suggerendo sia il soggetto di annotazione che la sua annotazione prelevata da una particolare fonte di conoscenza. Per esempio, [Ciravegna et al. 2002] e [Svatek et al. 2004] sono basati su approcci d'apprendimento statistici e supervisionati, mentre Armadillo [Ciravegna et al. 2004] e KowItAll [Etzioni et al. 2004] hanno automatizzato le funzioni di annotazione basate su quelle che non prevedono supervisione. I primi sistemi richiedono nella fase di addestramento, frequenti interventi dell'utente, mentre i secondi necessitano di una selezione manuale di una grande quantità di documenti già annotati. D'altronde, in tutti questi casi, le tecniche impiegate annotano documenti interi e non permettono la suddivisione in annotazione più piccole, come viene invece richiesto nei processi di disambiguazione strutturale. Infine, essendo questi approcci statistici, condividono gli stessi svantaggi che abbiamo discusso per i relativi metodi statistici applicati alla *free text* WSD.

1.2 STRIDER

In questa sezione verrà presentata la struttura di STRIDER e il flusso di informazioni coinvolte nel processo di disambiguazione.

STRIDER è un'applicazione progettata per eseguire efficacemente la disambiguazione di diversi formati di dati, tra cui alberi, schemi relazionali e grafi, e disambigua i termini contenuti nelle etichette di ciascun nodo, utilizzando diverse fonti di conoscenza esterne; la Figura 1 descrive l'intera architettura del sistema. Il risultato finale del processo di disambiguazione è costituito da una classifica dei sensi plausibili per ogni termine.

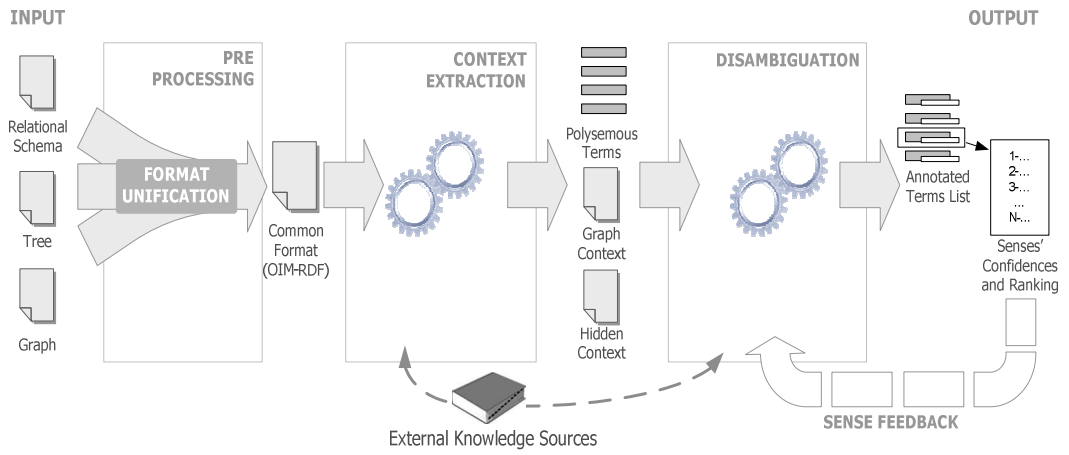


Figura 1. Architettura del sistema STRIDER.

L'esecuzione dell'applicazione si compone di diverse fasi, ognuna eseguita da un componente specifico (nei prossimi paragrafi verranno dettagliate singolarmente):

- Per prima cosa, avendo lo scopo di gestire dati in un formato uniforme, STRIDER converte il formato di input in un grafo RDF (fase di **Pre-elaborazione**, *Pre-processing phase*, paragrafo 1.3);
- Successivamente, nella fase di **Estrazione del contesto** (*Context extraction phase*, paragrafo 1.4), viene estratta dal grafo una lista di termini polisemici e ad ognuno di essi è assegnato un contesto, che contiene delle informazioni utili per individuare il senso corretto nella frase attuale. Esso include sia le informazioni estratte dal grafo (*graph context*), sia le informazioni estratte dalle fonti di conoscenza esterna (*hiddn context*);
- Per ultima, la fase di **Disambiguazione** (*Disambiguation*, paragrafo 1.5), che è eseguita utilizzando informazioni relazionali e algoritmi *knowledge-driven*, fornisce in output una classifica dei sensi plausibili per ogni termine.

In più, per poter raffinare i risultati iniziali, sono disponibili diverse tecniche di **Feedback**, sia manuali, che automatizzate.

Nei prossimi paragrafi verranno illustrate in dettaglio le tecniche utilizzate, accompagnate da esempi; a questo scopo, lo schema SQL DDL rappresentato in Figura 2 viene utilizzato come esempio guida. Esso raffigura in lingua inglese un ordine d'acquisto (*Purchase order*), e contiene vari termini con significati ambigui. Solo per citarne uno, il termine *line* può avere molti sensi, e nel dizionario WordNet se ne contano ben 29, ad esempio (*s*1) "*a formation of people or things one beside another*", come "*line of soldiers*", oppure (*s*5) "text *consisting to a row of words*", oppure (*s*9) "*cable, transmission line*" o ancora (*s*22) "*line of products, line of merchandise*", e così via.

È da sottolineare che saranno descritti per completezza, tutti gli algoritmi principali su cui si basa l'applicazione, anche se alcuni di questi non sono stati utilizzati direttamente nello sviluppo del progetto.

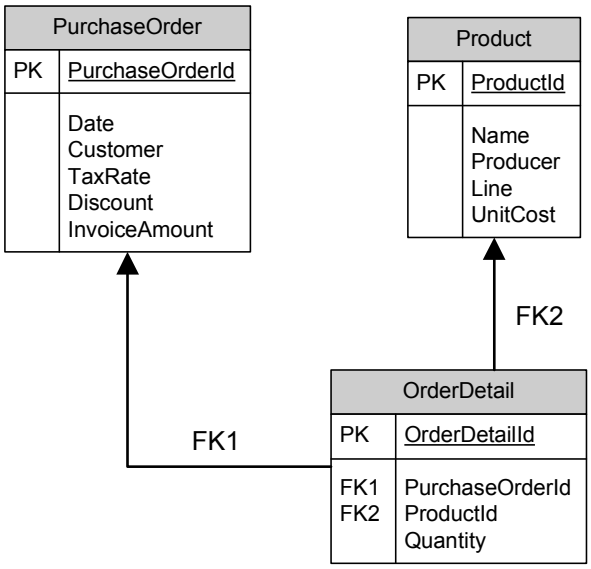


Figura 2. Esempio ordine d'acquisto in lingua inglese, *Purchase order*.

1.3 Pre-elaborazione degli schemi

Per preparare lo schema alle fasi successive, è necessaria un'elaborazione iniziale, che ha lo scopo di unificare tutti i formati in input, in grafo RDF. Le strutture accettate da STRIDER sono di tre tipi:

- **Schemi relazionali**: tabelle e viste descritte con costrutti SQL DDL;
- **Alberi**: codificati in schemi XML, che possono descrivere documenti XML, directory web e tassonomie;
- **Grafi**: codificati in RDF.

Essendo supportati molti formati, l'applicazione offre la possibilità di adattasi a diversi tipi di scenari, come database industriali, applicazioni semantiche nel web e così via.
E' stato scelto RDF come formato comune, in quanto garantisce sia il mantenimento della ricchezza semantica, che la libertà d'espressione. In particolare, per l'unificazione del formato, è stato adottato un procedimento di traduzione universale [Melnik et al. 2003; Melnik et al. 2002]: per prima cosa viene convertito il modello in input, in un meta-modello secondo l'*Open Information Model* (OIM) [Shutt et al. 1999], codificato in RDF, e successivamente si esegue una mappatura tra lo schema iniziale e i modelli convertiti. E' da notare che l'approccio seguito facilita molto l'aggiunta di nuovi formati di input, in quanto richiede solo di implementare le regole di conversione tra il nuovo formato e il comune. Il risultato di questo processo viene chiamato *OIM-RDF graph*; esso è costituito da una serie di nodi e di archi orientati, correlati di etichetta, che collegano i nodi. Il processo di disambiguazione si concentrerà su tutte le etichette dei nodi, dato che essi catturano tutto il contenuto informativo degli schemi a cui siamo interessati. La Figura 3 mostra una porzione del grafo OIM-RDF riferito all'esempio guida. Per la tabella *Product*, le etichette che verranno disambiguate saranno *Product*, *Line* e *ProductId*, che sono i nomi dei nodi &1, &2 e &4.

Considerando che i grafi OIM-RDF sono molto grandi, essi vengono salvati in memoria secondaria, in particolare adottando il meccanismo di memorizzazione proposto in [Wilkinson et al. 2003] per il framework di persistenza della struttura Jena. Il compromesso dello spazio per il tempo, introducendo politiche di ridondanza nella memorizzazione, migliora le prestazioni nel recupero dei dati attraverso sistemi di localizzazione e cache.

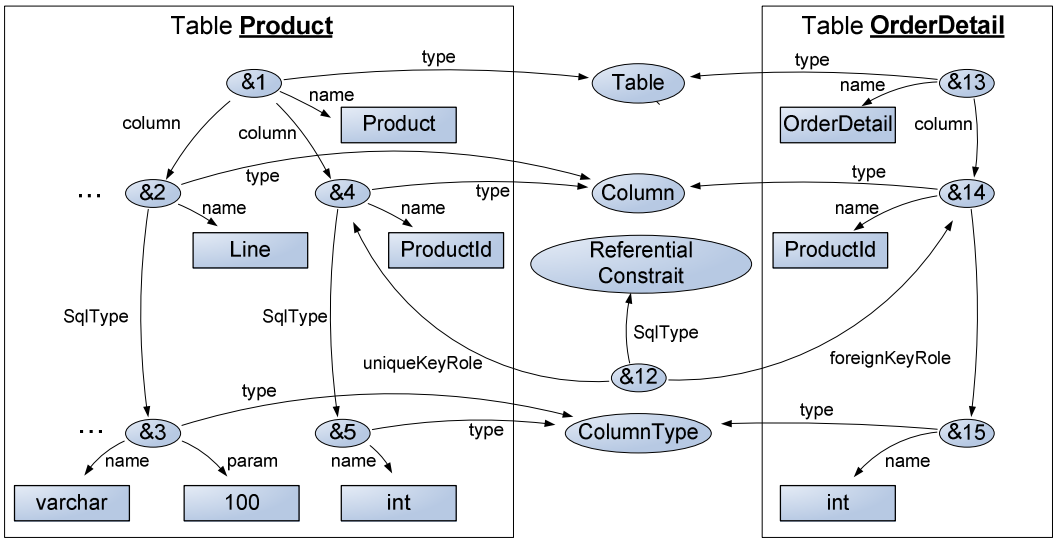


Figura 3. Porzione del grafo OIM-RDF dello schema d'esempio.

1.4 Espansione del contesto

Dopo che lo schema di input è stato convertito in un grafo OIM-RDF, è necessario estrarre la lista di termini polisemici dai nodi del grafo. Per primo, l'estrazione di termini e sensi (*Terms & Senses Extraction*), Figura 4, ricava i termini polisemici dalle etichette degli N nodi e associa ogni termine (t, N)[2] a una lista di sensi Senses(t, N) = [s1, s2, s3, …, sk]. In linea di massima, tale elenco è l'insieme completo dei sensi forniti da un dizionario, ma può anche essere una

[2] Notare che lo stesso termine può essere incluso più di una volta e che la disambiguazione è strettamente dipendente dal nodo a cui l'istanza appartiene

versione suggerita da un essere umano o da macchine esperte. E' da notare che l'estrazione dei termini può generarne più di uno, ne è un esempio *productId* che contiene i termini *product* e *id*. Molte volte termini insignificanti o molto frequenti, come congiunzioni e articoli, possono essere fonte di disturbo nella fase di disambiguazione, quindi vengono scartate. Per ultimi vengono estratti rispettivamente il contesto del grafo e il contesto nascosto (*Graph Context* e *Hidden Context*).

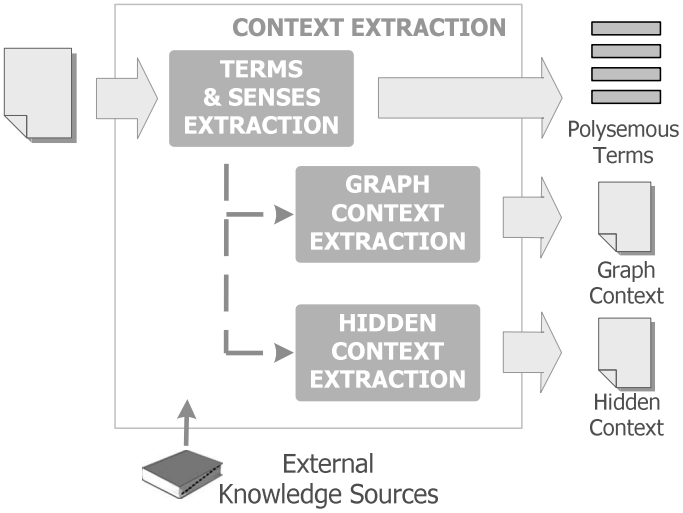


Figura 4. Componenti della fase di estrazione del contesto.

1.4.1 Estrazione del contesto del grafo

Individuare il corretto senso dell'etichetta di ogni nodo può essere possibile solo analizzando il contesto costituito dalle etichette dei nodi che lo circondano. L'estrazione del contesto del grafo, è la fase in cui si contestualizza ogni termine (t, N) attraverso l'estrazione del suo contesto Gcont(t, N) dal grafo OIM-RDF. Nella forma più semplice, il grafo del contesto contiene le etichette dei termini di tutti i nodi del grafo che sono raggiungibili da N; tuttavia a seconda del grafo, una scelta diversa può essere migliore. Infatti, se vengono inclusi termini distanti, il loro contributo può diventare fuorviante. Per esempio, la disambiguazione del

termine *line* appartenente alla tabella *Product*, riceve un contributo positivo dai termini vicini, come *product* e *cost*, ma può invece ricevere effetti negativi da altri più lontani, che sono ugualmente raggiungibili, come *customer*.

Per questo motivo si introduce la nozione di *crossing setting*, che associa ad ogni termine un'astrazione del grafo. Un *crossing setting* σ è costituito dalla soglia di raggiungibilità τ e un set di tipi di archi, dove ogni tipo è una vista del grafo OIM-RDF e che è abbinata a due costi di attraversamento in (0, 1], uno per ogni direzione. I costi vengono utilizzati per valutare la distanza tra due nodi nel grafo; minore è il costo, maggiore è la vicinanza dei due nodi connessi dell'arco in questione. Per ogni formato, i valori di default di *crossing setting* sono riportati nell'appendice A, e l'utente ha la possibilità di modificarne il valore. Nel caso di schemi relazionali, esistono due tipi di archi: SR, che rappresenta la relazione strutturale tra una tabella e ogni sua colonna, FK, che identifica l'arco di chiave esterna o *foreign key*.

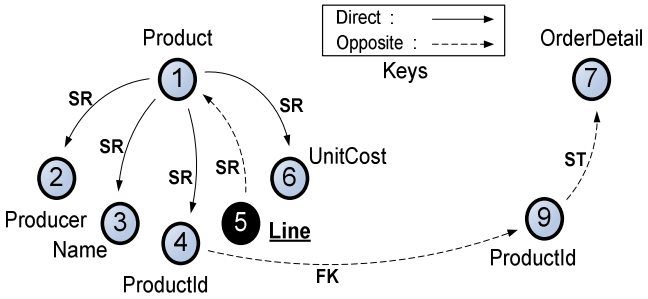
Applicando le definizioni degli archi, definite dal *crossing setting* σ, otteniamo una vista del grafo OIM-RDF che diventerà l'input della fase di estrazione del contesto. Quest'ultimo segue un approccio relazionale, associando ogni termine polisemico (t, N) al suo grafo di contesto Gcont(t, N), che è costituito da un insieme di tuple ((t^g, N^g), Senses(t^g, N^g), weight(N^g)), una per ogni termine t^g estratto da ogni nodo N^g, raggiungibile da N nella vista del grafo OIM-RDF. Ogni nodo N' è detto raggiungibile da N se e solo se esiste un percorso da N a N' il cui costo totale è minore o uguale alla soglia di raggiungibilità τ. L'applicazione utilizza l'algoritmo di Dijkstra; quindi, il costo totale SPC(N, N') del percorso minimo dal nodo N ad ogni nodo N' è calcolato sommando i singoli costi associati ad ogni relazione di crossing setting, a seconda del verso di percorrenza e così è definito l'insieme di nodi raggiungibili N^g (SPC(N, N^g) $< \tau$).

Non tutti i nodi influiscono allo stesso modo nel disambiguare un termine, infatti il nodo che è più vicino a quello relativo al termine e che è connesso con un arco di peso minore, influenza maggiormente. Per questo motivo viene associato ad ogni nodo raggiungibile N^g un peso weight(N^g) calcolato applicando una funzione gaussiana decadente per la distanza:

$$weight(N^g) = 2 \cdot \frac{e^{-\frac{(SPC(N,N^g))^2}{8}}}{\sqrt{2\pi}} + 1 - \frac{2}{\sqrt{2\pi}}$$

La Figura 5 mostra un esempio di contesto del grafo relativo al termine *line*, comprendendo tutti i nodi raggiungibili dal nodo in questione. È possibile ottenerlo applicando il *crossing set* di default al grafo OIM-RDF. Ad ogni nodo è associato un identificatore univoco, che è utilizzato nella tabella per mostrare i costi totali e i pesi corrispondenti.

Di conseguenza, si nota che il contesto del grafo Gcont(*line*, 5), è costituito dai termini (*product*, 1), (*producer*, 2), (*name*, 3), (*product*, 4), (*id*, 4), (*unit cost*, 6), (*product*, 9), (*id*, 9), (*order*, 7) e (*detail*, 7).



Node	SPC (Total cost)	Weight
1	0.8	0.939
2	0.8+1.0=1.8	0.734
3	0.8+1.0=1.8	0.734
4	0.8+1.0=1.8	0.734
6	0.8+1.0=1.8	0.734
9	0.8+1.0+0.1=1.9	0.710
7	0.8+1.0+0.1+0.8=2.7	0.523

Total Costs and Weights for reachable nodes for node ***Line***

Figura 5. Esempio di selezione dei nomi e del calcolo dei pesi per il termine *line*.

È da notare che tutti i termini appartenenti allo stesso nodo hanno lo stesso peso e che questi termini possono essere formati da più di una parola se il lemma corrispondente è contenuto all'interno della fonte di conoscenza; ne è un esempio il nodo 6, dove *unit cost* è costituito da due parole. Di conseguenza si ottiene che la tupla di Gcont(*line*, 5) relativa al nodo (*product*, 1) utilizzando WordNet come fonte di conoscenza esterna [Miller 1995] è :

((*product*, 1),[s1=commodities offered for sale ("good business depends on having good merchandise"; "that store offers a variety of products"),…,s5=a quantity obtained by multiplication ("the product of 2 and 3 is 6"),…],0.939).

1.4.2 Estrazione del contesto nascosto

Poiché la maggior parte della semantica è contenuta all'interno dei nomi [Ide and Veronis 1998], abbiamo notato che una nuova e significativa fonte di informazione è nascosta nei sostantivi che vengono utilizzati per spiegare i significati delle parole; per questo motivo è stato creato il "contesto nascosto" ("*hidden context*").

La fase di estrazione del contesto, arricchisce il contesto di ogni termine (t, N) con il nuovo contesto Scont(s) di ogni senso s in Senses(t,N), comprendendo l'insieme dei sostantivi estratti dalle definizioni, dagli esempi e dalle altre spiegazioni fornite dal dizionario, utilizzando un *Part-of-speech tagger* (questo strumento linguistico verrà descritto nel paragrafo 2.8). Questo contributo è molto utile quando il contesto del grafo fornisce poche informazioni, ad esempio nel caso in cui il grafo OIM-RDF è molto piccolo.

s_k	**Explanation**	**Examples**	*Scont(s_k)*
s_1	a formation of people or things one beside another	the *line* of soldiers advanced with their bayonets fixed; they were arrayed in *line* of battle; the cast stood in *line* for the curtain call	battle, bayonet, cast, curtain, formation, …
…	…	…	…
s_{22}	a particular kind of product or merchandise	a nice *line* of shoes	Kind, line, merchandise, product, shoes, …
…	…	…	…

Tabella 1. Rappresentazione del contesto nascosto del termine (line, 5).

La Tabella 1 mostra alcuni dei 29 sensi del temine *line*, assieme alle sue definizioni (*explanation*s), esempi (*examples*) e Scount(s) (i termini estratti sono riportati in ordine alfabetico). Il dizionario WordNet facilita questo sistema di tabulazione, in quanto fornisce per ogni senso s, la definizione e la lista di esempi. Molti dizionari, anche di lingue diverse dall'inglese, permettono questo genere di suddivisione; ciò è molto importante, in quanto rende possibile l'estendibilità ad

altre lingua o l'arricchimento delle stesse definizioni con un'altra fonte di conoscenza.

Nel prossimo paragrafo verranno descritti gli algoritmi principali su cui si basa il funzionamento di STRIDER.

1.5 Algoritmi di disambiguazione

La maggioranza degli algoritmi utilizzati in STRIDER, seguono un approccio *knowledge-driven*: lo scopo del loro lavoro consta nel correlare semanticamente ogni termine da disambiguare con i termini contenuti nel suo contesto, *graph context* e *hidden context*, mediante l'utilizzo di fonti di conoscenza.

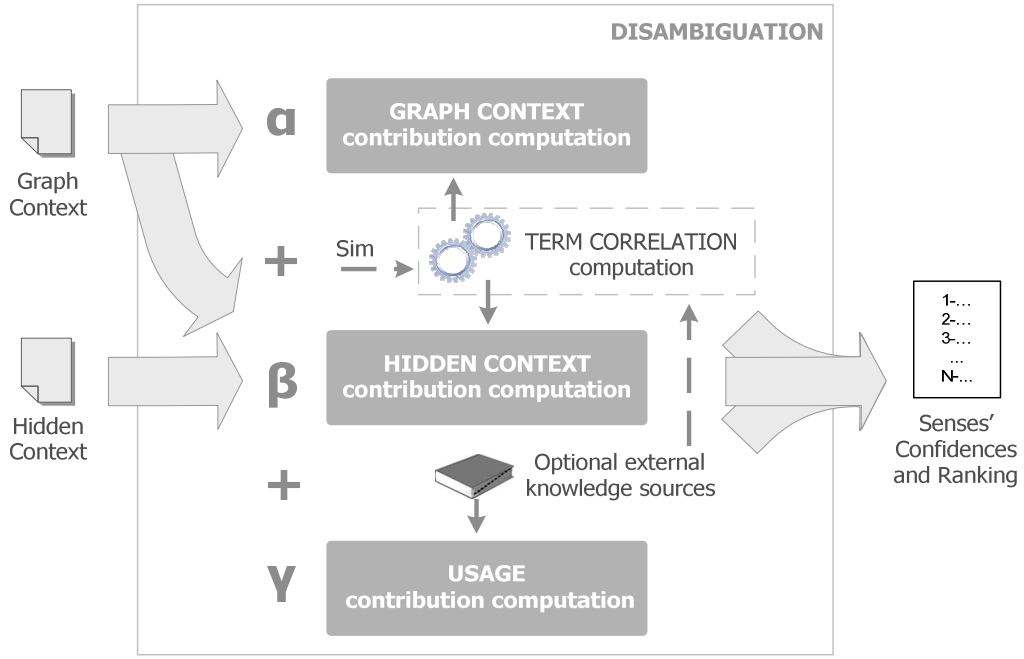


Figura 6. Overview dei differenti contributi per la disambiguazione dei termini.

Il risultato finale di tutto il processo, è una classifica dei sensi per ogni termine, come viene mostrato nella Figura 6. La graduatoria è stata concepita in modo da permettere sia la disambiguazione automatica, che quella assistita dall'utente. Per

questo motivo, viene associato un vettore di confidenze φ per ogni senso $s_k \in$ Senses(t,N) con un punteggio $\varphi(k)$ compreso tra 0 e 1 che esprime il valore di confidenza nel sceglierlo come esatto per il termine (t, N).
Come mostrato in Figura 6, la confidenza è composta da tre diversi contributi: il contributo del grafo, quello del contesto nascosto e quello di frequenza d'uso. I primi due dipendono dal contesto e producono due vettori di confidenza chiamati rispettivamente φ_G e φ_H. L'ultimo utilizza la frequenza d'uso dei sensi fornita da WordNet e/o da un'altra fonte di conoscenza, producendo un vettore φ_U. Il vettore finale delle confidenze φ si ottiene dalla loro combinazione lineare:

$$\varphi = \alpha * \varphi_G + \beta * \varphi_H + \gamma * \varphi_U, \qquad \text{con } \alpha + \beta + \gamma = 1$$

dove α, β e γ sono parametri che possono essere modificati liberamente a seconda del peso che si vuole dare a ciascun contributo.
Verrà ora ripreso l'esempio del termine *line*, ricordando che il senso corretto è s_{22} "*line of products, line of merchandise*". Utilizzando la tecnica automatica, l'approccio proposto consente di disambiguare correttamente il temine in questione assieme alla maggior parte degli altri termini presenti nello schema. In particolare, utilizzando i valori di default $\alpha = \beta = 0.4$ e $\gamma = 0.2$, il valore massimo del vettore finale di confidenze è $\varphi(22) = 0.61$, molto maggiore rispetto ad esempio a $\varphi(1) = 0.24$; ciò significa che STRIDER associa la confidenza massima al senso corretto.

1.5.1 Correlazione dei termini

Dato un termine, la confidenza che si associa ai suoi sensi, dipende dalla correlazione tra il termine stesso e ciascun termine che appartiene al suo contesto. A questo proposito, il calcolo della correlazione tra parole, diventa fondamentale nell'uso degli algoritmi di estrazione del contesto del grafo e del contesto nascosto.

Il calcolo della correlazione tra termini può utilizzare sia un approccio *knowledge-driven*, che un approccio *corpus-based*, prelevando le informazioni necessarie da una fonte di conoscenza esterna. Per quantificare la correlazione tra due termini t_x e t_y, è possibile seguire due tecniche differenti, rispettivamente associate agli approcci sopra citati: il primo utilizza un dizionario, mentre il secondo un grande corpora testuale.

Nel primo caso, vengono utilizzate le gerarchie di ipernimi-iponimi di WordNet, attraverso la misura della similarità denominata Leacock-Chodorow [Leacock and Chodorow 1998]:

$$sim(t_x, t_y) = \begin{cases} -ln\frac{len(t_x,t_y)}{2 \cdot H} & \text{if } \exists \text{ a common ancestor} \\ 0 & \text{otherwise} \end{cases}$$

dove len(t_x, t_y) è il minimo tra il numero di link che connettono ogni senso in Senses(t_x) e ogni senso in Senses(t_y), e H è l'altezza della gerarchia degli ipernimi (H = 16 in WordNet). L'insieme più piccolo degli antenati di t_x e t_y nella gerarchia di ipernimi viene indicato con mch(t_x, t_y) (*minimum common hypernymy*, minimo comune ipernimo). Questo tipo di misura è utile quando non sono a disposizione altre fonti di conoscenza aggiuntive; d'altro canto, quando è fruibile una vasta quantità di documenti, o semplicemente una connessione ad internet che permette l'accesso ad un comune motore di ricerca, le informazioni sulle co-occorrenze di termini possono essere facilmente trasformate in correlazione tra termini, utilizzando delle particolari funzioni, come quella di Jaccard (utilizzata recentemente in [Strube and Ponzetto 2006]), PMI [Turney 2004] o NGD [Cilibrasi and Vitanyi 2006]. In STRIDER viene utilizzata la *term co-occurrence similarity* basata su una versione esponenziale del PMI, che produce valori positivi di similarità:

$$sim(t_x, t_y) = e^{PMI(t_x,t_y)} = \frac{M \cdot f(t_x, t_y)}{f(t_x) \cdot f(t_y)}$$

dove M è una costante di normalizzazione, f(t) è la frequenza del termine t e $f(t_x,t_y)$ è la frequenza della co-occorrenza dei termini t_x e t_y. Attraverso determinati test e controllando i risultati, si è concluso che per STRIDER questa versione del PMI è più efficace delle funzioni di Jaccard e NGD, in particolare se f(t) è calcolata interrogando grandi motori di ricerca, come Yahoo!® o Google™. Preferendo quest'ultima tecnica per il calcolo della similarità, STRIDER diventa un sistema meno legato alla struttura di WordNet, prevedendo la possibilità di essere esteso ad altri dizionari inglesi e più in generale ad altre lingue.

1.5.2 Contributo del contesto del grafo

L'individuazione del senso corretto per ogni etichetta, è possibile analizzando il contesto del grafo di ogni termine coinvolto, e più in particolare, quantificando la correlazione tra il termine stesso e quelli nel suo contesto.
L'Algoritmo 1 rappresenta il contributo del contesto del grafo per un termine (t,N).

Algoritmo 1. ***GraphContextContr*(*t,N*)**

1: **for each** s_k in *Senses*(*t, N*) **do** *//cycle in t senses*
2: $\varphi_G[k] = 0$
3: *norm = 0*
4: **for each** t^g_i in *Gcont*(*t, N*) **do** *//cycle in graph context terms*
5: $mch(t, t^g_i)$ = the minimum common hypernyms of *t* and t^g_i
6: $numHyp(mch(t, t^g_i), s_k)$ = number of hypernyms in $mch(t, t^g_i)$ ancestors of s_k
7: $\varphi_G[k] = \varphi_G[k] + GCC[k][i]$
8: $norm = norm + GCN[i]$
9: $\varphi_G[k] = \varphi_G[k] / norm$
10: **return** φ_G

La versione presentata, per semplicità, prende in input un solo termine alla volta; in realtà, per motivi si efficienza, l'attuale implementazione prende in input un intero schema.

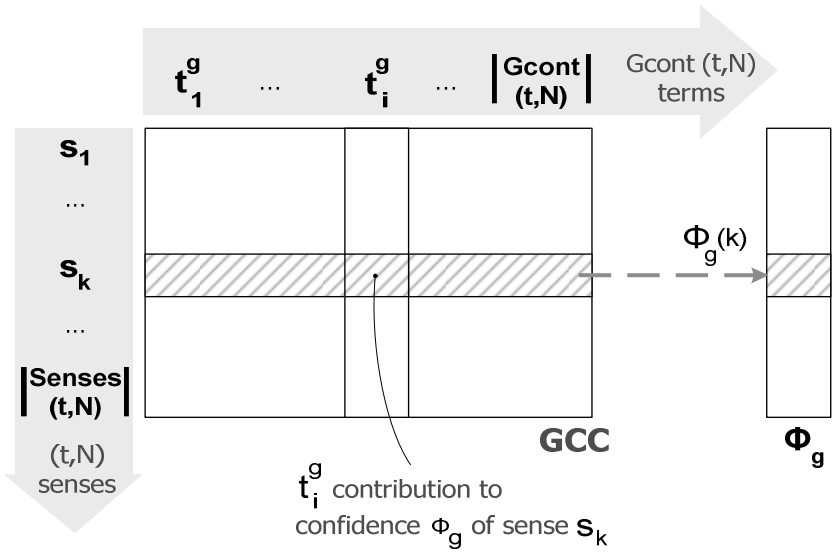


Figura 7. Esemplificazione grafica dell'algoritmo per l'estrazione del contesto del grafo, per la disambiguazione del generico termine t.

L'algoritmo prende in input un termine (t, N) da disambiguare e produce un vettore φ_G contenente i valori di confidenza per ogni senso in Senses(t,N).

L'idea su cui si basa l'algoritmo, che deriva originariamente da [Resnik 1995], è che i termini nel Gcont(t, N) forniscono un buon quadro informativo e dei buoni indizi su quale senso scegliere per t. Inoltre, il contributo di ogni termine t^g nel Gcont(t, N) è proporzionale al peso weight(N^g_i) associato al suo nodo N^g, come calcolato nella fase di estrazione del contesto.

Come esemplificato graficamente nella Figura 7, l'algoritmo è composto da due cicli (dimensioni): uno sui sensi di t (ciclo esterno alla linea 1, dimensione "k") e uno sui termini nel Gcont(t, N) (ciclo interno alla linea 4, dimensione "i"). In particolare, ogni termine t^g_i contribuisce nella confidenza del senso s_k con un valore calcolato mediante la matrice del *Graph Context Computation* (GCC):

$$GCC[k][i] = sim(t, t^g_i) * numHyp(mch(t, t^g_i), s_k) * weight(N^g_i),$$

dove sim(t, t^g_i) è la similarità tra t e t^g_i (calcolata utilizzando una delle due funzioni proposte nel paragrafo 1.5.1), numHyp(mch(t, t^g_i), s_k) è il numero dei minimi comuni ipernimi mch(t, t^g_i) che è antenato di s_k e weight(N^g_i) è il peso

associato al nodo t^g_i nel contesto del grafo. La confidenza finale $\varphi_G(k)$ per ogni senso s_k si ottiene sommando (linea 7) tutti i contributi disponibili per ogni senso (riga k della matrice in figura) e normalizzando (linea 9) il risultato finale. La normalizzazione riconduce tutti i punteggi nell'intervallo 0…1 ed è eseguita sulla base del numero totale dei minimi comuni ipernimi tra t e t^g_i. In particolare, è calcolata tramite la matrice di *Graph Context Normalization* (GCN):

$$GCN[i] = sim(t, t^g_i) * |mch(t, t^g_i)| * weight(N^g_i),$$

dove GCN[i] è la normalizzazione per il termine t^g_i.

1.5.3 Contributo del contesto nascosto

Lo scopo principale del contributo del contesto nascosto è di quantificare la correlazione semantica tra il contesto del grafo di un termine polisemico e il contesto nascosto di ciascuno dei suoi sensi. L'algoritmo è stato creato in modo da seguire le stesse linee guida di quello presentato nel paragrafo 1.5.2. La differenza principale sta nel fatto che ogni senso s_k del termine t è rappresentato da un insieme di termini in Scont(s_k). Quindi esso confronta semanticamente il Gcont(t, N) dato con i contesti del senso fornito, e il Scont(s_k) che conterrà i termini più simili al Gcont(t, N) sarà quello con probabilità maggiore. Come già detto, il seguente algoritmo è molto simile al precedente, e per questo motivo non verrà riscritto. Il calcolo avviene in tre dimensioni (cicli) totali: l'esterno sui sensi s_k di t (dimensione "k"), il secondo sui termini t^g_i nel Gcont(t, N) (dimensione "i") e l'interno sui termini t^s_j in ogni Scont(s_k) (dimensione "j"). Le confidenze vengono sommate alle linee 7-8 dell'Algoritmo 1, sostituendo GCC e GCN rispettivamente con la matrice tridimensionale *Hidden Context Computation* (HCC) e la matrice bidimensionale *Hidden Context Normalization* (HCN), calcolate nel seguente modo:

$$HCC[k][i][j] = sim(t^g_i, t^s_j) * numHyp(mch(t, t^g_i), s_k) * weight(N^g_i)$$

$$HCN[i][j] = sim(t^g_i, t^s_j) * |mch(t, t^g_i)| * weight(N^g_i)$$

È da notare che, invece di utilizzare il termine t per disambiguare, $sim(t^g_i, t^s_j)$ confronta il termine t^g_i del Gcont(t, N) con il termine t^s_j rappresentando il senso s_k. Come raffigurato nella Figura 8, quest'operazione viene fatta per tutti i termini nel contesto confrontando tutti i termini che rappresentano tutti i sensi: in questa rappresentazione grafica, ogni cella HCC[k][i] contiene un vettore di similarità, uno per ogni t^s_j; la somma di questi valori rappresenta il contributo di t^g_i per s_k.

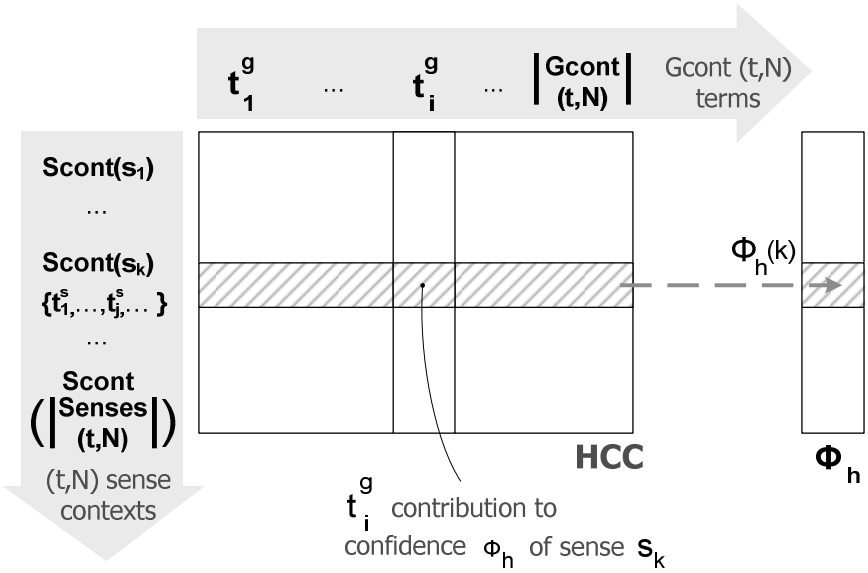


Figura 8. Esemplificazione grafica dell'algoritmo per l'estrazione del contesto nascosto del generico termine t.

Questo tipo di algoritmo è basato sul calcolo degli ipernimi comuni condivisi da coppie di termini; analizzando le performance, si è notato che ha un'ottima efficacia. In alcuni casi, però, la struttura ad ipernimi di WordNet potrebbe non essere adeguata o sufficiente per descrivere un concetto oppure per disambiguarlo. D'altra parte, è anche importante rendere il sistema estendibile, prevedendo l'utilizzo di altri tipi di dizionari, anche per lingue differenti, che non possiedono

una struttura gerarchica di questo tipo. Per tutti questi motivi, STRIDER può utilizzare un algoritmo alternativo, l'Algoritmo 2, che pesa i diversi contributi di similarità in base ai pesi del Gcont(t, N) e in modo totalmente indipendente dalle gerarchie del dizionario WordNet. Dato un senso s_k (linea 1) e un termine t^g_i del Gcont(t, N) (linea 4), la similarità $sim(t^g_i, t^s_j)$ tra t^g_i e ogni termine t^s_j in $Scont(s_k)$ è calcolata (linea 6-7), sommando i singoli valori per mezzo di una funzione parametrica f() e infine aggiungendo il risultato alla confidenza del senso $\varphi_H[k]$ (linea 8), pesato sulla base del peso del grafo, $weight(N^g_i)$. È possibile utilizzare diverse forme della funzione f(), come ad esempio max() o mean(); è stato provato che in questo caso, mean() è la più efficace. Infine, la normalizzazione viene eseguita in due passi: per ogni senso s_k, i valori di $\varphi_H[k]$ vengono divisi per la somma di tutti i pesi, in modo da ottenere una media pesata della similarità dei termini (linea 10), poi i valori in φ_H vengono ricondotti nell'intervallo 0…1 (linea 11), in modo da essere compatibili con gli altri contributi.

Questo algoritmo sarà alla base dell'estensione di STRIDER, che lo renderà indipendente dalla lingua; a questo proposito verrà utilizzato anche per l'italiano.

Algoritmo 2. *HiddenContextContr*(*t,N*) (versione indipendente dalle gerarchie)

1: **for** k = 1 to $|Senses(t, N)|$ **do** *//cycle in senses s_k of term t*
2: $\varphi_H[k] = 0$
3: $sumWeights = 0$
4: **for each** t^g_i in $Gcont(t, N)$ **do** *//cycle in graph context terms*
5: $vsim_{ti} = [0, \ldots, 0]$
6: **for each** t^s_j in $Scont(s_k)$ **do** *//Cycle in sense context terms*
7: $vsim_{ti}[j] = sim(t^g_i, t^s_j)$
8: $\varphi_H[k] = \varphi_H[k] + f(vsim_{ti}) * weight(N^g_i)$
9: $sumWeights = sumWeghts + weight(N^g_i)$
10: $\varphi_H[k] = \varphi_H[k] / sumWeights$
11: $\varphi_H = \varphi_H / max(\varphi_H)$
12: **return** φ_G

1.5.4 Contributo della frequenza d'uso

L'ultimo contributo è basato sulla frequenza d'uso del senso nella lingua inglese. A differenza dei precedenti, questo contributo è indipendente dal contesto e fa affidamento soltanto sulla fonte di conoscenza esterna. L'idea di base è di tentare di emulare il senso comune in un essere umano nella scelta del significato giusto di un termine, quando il contesto dà solo un piccolo aiuto. In STRIDER sono implementati due metodi per il calcolo della confidenza φ_U: entrambi si basano solo sull'utilizzo di un dizionario, come WordNet. Il primo metodo sfrutta il fatto che WordNet ordina i sensi di ogni termine t in base allo loro frequenza d'uso. In questo modo, come già proposto in [Mandreoli et al. 2005], è possibile calcolare il valore $\varphi_U(k)$ per ogni senso s_k tramite una funzione di decadimento lineare, la quale ha come valore iniziale $\varphi_U(1) = 1$, e che decrementa il valore di fiducia nella scelta di s_k, proporzionalmente alla sua posizione, $pos(s_k)$:

$$\phi_U[k] = linearDecay(s_k) = \begin{cases} 1 - \rho \frac{pos(s_k)-1}{|Senses(t)|-1} & \text{if } |Senses(t)| > 1 \\ 1 & \text{otherwise} \end{cases}$$

dove $0 < \rho < 1$ è un parametro che viene generalmente impostato a 0.8 e |Senses(t)| è la cardinalità di Senses(t). In questo modo, è possibile quantificare in modo grossolano la frequenza dei sensi, dando piena confidenza al primo e all'ultimo un decadimento di 1:5.

La versione estesa di WordNet, fornisce delle informazioni più dettagliate riguardo la frequenza dei sensi, nella forma dei *frequency counts* $f_c(s_k)$, che sono valori interi associati ad ogni s. Più è grande $f_c(s_k)$, maggiore è la probabilità data al senso s_k di un termine; questi valori sono stati dedotti sulla base di informazioni estratte da grandi corpus di documenti in lingua inglese.

La seconda funzione per il calcolo della frequenza d'uso proposta in STRIDER è ideata per l'utilizzo della *frequency counts*, ed è generalmente più precisa ed

efficace della prima, in quanto la funzione di decadimento non è lineare, ma è creata in modo differente per ogni termine:

$$\phi_U[k] = freqCountDecay(s_k) = \begin{cases} 1 - \rho \frac{fc(s_1) - fc(s_k)}{fc(s_1)} & \text{if } fc(s_1) > 0 \\ 1 & \text{otherwise} \end{cases}$$

Naturalmente questo secondo approccio è strettamente legato alle informazioni aggiuntive fornite de WordNet; il primo, invece, è applicabile ad un qualsiasi dizionario, di qualunque lingua, che abbia la caratteristica di ordinare le definizioni in base alla frequenza d'uso, è sarà il metodo utilizzato per l'integrazione dell'italiano in STRIDER.

1.6 Feedback

Gli algoritmi di disambiguazione presentati sono capaci di fornire buoni risultati anche con una sola esecuzione. In ogni caso, per ottenere risultati migliori, sono state implementate diverse tecniche di *feedback*, che raffinano i risultati iniziali lanciando altri cicli di esecuzione. Ad ogni ciclo i che segue il primo (i = 2,…, n), l'insieme dei sensi $Senses(t, N)_i$ per i termini (t, N) è una ridotta versione dei sensi $Senses(t, N)_{i-1}$.

In ogni caso è sempre possibile il *feedback* manuale, nel quale l'utente ha la possibilità di disattivare/attivare i sensi, in modo da aiutare il sistema nella scelta del significato corretto. In STRIDER sono implementate anche tre tecniche automatiche di *feedback*, che nella maggior parte dei casi, migliorano in modo significativo i risultati, senza l'intervento dell'utente. I metodi proposti, sono gradualmente più complessi, e sono tutti basti sull'idea che il rimuovere i sensi di posizione più bassa al primo ciclo di esecuzione, fornisce riscontri positivi nella disambiguazione, finche il disturbo prodotto da questi significati che si considerano errati, viene eliminato. I tre metodi sono i seguenti:

- Metodo 1 – *Simple auto-feedback*: Il risultato della prima esecuzione viene raffinato automaticamente, disabilitando i contributi di tutti i sensi che seguono la posizione X, nel seguente ciclo di esecuzione. In STRIDER è possibile scegliere un valore molto basso di X, come 2 o 3, in quanto gli algoritmi implementati assicurano, nella maggioranza dei casi, che il senso corretto occupi una delle posizioni più alte;
- Metodo 2 – *"Knockout" auto-feedback*: Alla fine di ogni ciclo di esecuzione, il senso di ogni termine che occupa la posizione più bassa viene disabilitato. Il processo termina quando per ogni termine restano X sensi. Rispetto al precedente, questo metodo richiede più di un ciclo di esecuzione, ma risulta più efficace, in quanto elimina i sensi in modo più graduale. Il numero massimo di cicli d'esecuzione dipende dal massimo numero di sensi tra i termini e dal valore di X:
- Metodo 3 – *Stabilizing "knockout" auto-feedback*: È un metodo che deriva dal precedente: in questo caso viene memorizzato un vettore Δ per ogni termine, in cui $\Delta(k)$ rappresenta il modulo della variazione di confidenza di s_k tra il passo attuale (i) e il precedente (i-1): $\Delta(k) = |\varphi_i(k) - \varphi_{i-1}(k)|$. Quando la confidenza di un vettore diventa stabile, cioè quando la norma euclidea del vettore Δ è minore di un certo valore ε, rimangono solo i primi X sensi. L'efficacia di questo metodo è paragonabile a quella del metodo 2, ma ha il pregio di richiedere meno cicli di esecuzione.

È da sottolineare che STRIDER non implementa le tecniche di feedback nel caso venga utilizzato l'Algoritmo 2 per il calcolo del contesto nascosto.

1.7 Le fonti di conoscenza esterne

Come già accennato nei paragrafi precedenti, STRIDER è molto legato alle caratteristiche di WordNet, che è sempre stato il fulcro dell'intera esecuzione

dell'applicazione; infatti è stato sempre utilizzato sia nella fase di annotazione dello schema, che in quella per il calcolo della similitudine tra termini e per ultimo in quella relativa alla frequenza d'uso. Infatti la maggioranza degli algoritmi proposti sfruttano la sua struttura gerarchica, che fornisce informazioni aggiuntive rispetto ad un comune dizionario. Le caratteristiche del sistema non precludono, però, un'estensione ad altri tipi di dizionari, in quanto sono presenti algoritmi e tecniche alternative che lo svincolano dalla struttura gerarchica di WordNet. Per questo motivo, tra gli scopi principali della tesi, vi è quello di verificare l'applicabilità e testare il funzionamento di altri tipi di fonti di conoscenza esterne. È da notare che verranno introdotti nuovi generi sia da utilizzare nella fase di annotazione dello schema, che nella fase di calcolo della similarità tra termini.

È quindi possibile considerare STRIDER, come un sistema aperto al multilinguismo e a nuovi tipi di fonti di conoscenza esterne, che a loro volta possono avere caratteristiche simili a quelle di WordNet, ma anche strutture molto più semplici, che non possiedono altro che definizioni ed esempi. Gli stessi dizionari, utilizzati nella fase di annotazione e calcolo della frequenza d'uso, possono essere sia di tipo *knowledge-based*, i quali derivano da grandi quantità di dati elaborati e compilati da esperti linguisti, ma anche di tipo *corpus-based*, caratterizzati dall'estrazione automatica di conoscenza, da enormi corpora di documenti; di quest'ultimo genere ne è un esempio il dizionario machine-generated di Google™.

Anche il calcolo della similarità tra termini, come già accennato nel paragrafo 1.5.1, che ha lo scopo di determinarne la loro correlazione, non è così strettamente legato alla struttura ad ipernimi. Questo problema è stato infatti risolto utilizzando tecniche che sfruttano i motori di ricerca e le informazioni che forniscono; essi hanno la capacità di gestire la più grande quantità di documenti informatizzati, nel minor tempo possibile, rendendo così facile la loro interrogazione. Semplicemente servendosi della frequenza dei termini nel web, ed elaborando il risultato tramite funzioni appositamente studiate, è possibile determinarne la correlazione. Per questo motivo verrà testata la funzionalità già fornita da STRIDER, che prevede il

calcolo della similarità utilizzando la frequenza di un motore di ricerca, come Yahoo® o Google™.

Nel prossimo capitolo verranno descritti alcuni strumenti e tecnologie che sono state ricercate per adattare STRIDER alla lingua italiana. Le stesse ricerche possono essere eseguite in modo analogo per qualsiasi altra lingua, seguendo identici criteri di analisi e valutazione.

Capitolo 2

Strumenti linguistici

Per adattare STRIDER al multilinguismo, e più in particolare alla lingua italiana, è stato necessario ricercare alcuni strumenti linguistici che fungessero da fonte di conoscenza esterna per l'annotazione degli schemi e altri che fossero utili ad elaborare il testo contenuto nelle definzioni, in modo da estrarre solo le informazioni utili, nel nostro caso i sostantivi contenuti in frasi in lingua italiana.
Per questo motivo nelle prossime sezioni verranno descritti i dizionari elettronici che sono stati selezionati e scartati, gli strumenti linguistici elaborati quali *POS tagger* e *parser*, e i database di parole categorizzate che possono essere utilizzati nella manipolazione di testi.

2.1 Dizionari

Come è ben noto dall'esperienza comune, un dizionario è un'opera che raccoglie in lemmi, le parole di una o più lingue. Nel nostro caso è necessario la ricerca si

sofferma su vocabolari monolingue ed elettronici, che possono essere sia residenti sull'hard disk del computer che consultabili on-line.
Entrambi i generi hanno pregi e difetti: i primi a differenza dei secondi, consentono una consultazione immediata, non richiedendo una connessione ad internet; i secondi, generalmente, vengono aggiornati ed arricchiti più frequentemente. Nella maggioranza dei casi i dizionari on-line sono free, al contrario di quelli residenti, che necessitano di licenza spesso a pagamento e sono in quantità minore. Anche l'interrogazione differisce molto: nel primo caso, è necessario avere un'interfaccia, una libreria di funzioni, che si colleghi al programma esterno, nel secondo è sufficiente una richiesta web. Non è però detto che i dizionari on-line siano sempre più comodi da interrogare rispetto ai primi: alcuni dizionari residenti, come ad esempio WordNet, forniscono già un'interfaccia, mentre alcuni siti web, non permettono un'interrogazione diretta o limitano il numero di richieste al server o rendono difficile l'estrazione delle informazioni dai tag HTML.
Un'altra caratteristica generale che risulta determinante nella scelta del dizionario è la ricchezza dei vocaboli, caratterizzata sia dal numero di lemmi che dal numero di definizioni per ciascuno di essi.
Nei prossimi paragrafi verranno discusse la caratteristiche dei dizionari esaminati durante la ricerca, suddividendoli in due categorie: i dizionari che mantengono la struttura gerarchica di WordNet e i vocabolari non strutturati.

2.2 Interrogazione dei dizionari on-line

Per poter estrarre le definizioni fornite da un dizionario, è sempre necessario effettuare una richiesta al server web, in modo da ricevere la pagina contenente le informazioni. I metodi principali per effettuare questa operazione sono due:

- inviare una richiesta al server web con il metodo HTTP POST; questo permette di inviare informazioni all'URL specificato, sottoforma di coppie

attributo=valore che vengono processate dal server, sulla base di come è configurato e di come è fatto il file a cui vengono "postate"; lo scopo è quindi di inviare i dati al server, inserendoli nel corpo del messaggio di richiesta;

- inviare una richiesta, inserendo le informazioni che identificano la pagina direttamente nella URL, interrogando il server con il metodo HTTP GET; in questo caso la richiesta consiste in una singola riga di testo che inizia con la parola chiave GET seguita da un URL e dal numero di versione di HTTP.

Il linguaggio di programmazione Java, fornisce la libreria java.net, che permette di effettuare entrambi i tipi di richieste.
È da notare che il secondo metodo è quello già implementato e utilizzato in STRIDER; per questo motivo, nei prossimi paragrafi, quando verrà valutata la facilità di interrogazione del server, si considererà solo la facilità di inserire le informazioni utili nella URL, perché già presente nell'applicazione.

2.3 Dizionari con struttura ad ipernimi

Una delle valutazioni principali da fare nell'ambito della scelta del dizionario, è se sceglierne uno che mantenga la struttura gerarchica tipica di WordNet, oppure se preferirne uno non strutturato. I dizionari adatti che possiedono la prima caratteristica, sono in quantità minore rispetto agli altri, perché maggiormente strutturati e meno comuni. Gli esempi più significativi rintracciati nel web sono ItalWordNet e MultiWordNet.

2.3.1 ItalWordNet

WordNet [1], una risorsa per la lingua inglese, sviluppata più di dieci anni fa dal linguista George Miller presso l'Università di Princeton, è un dizionario interamente e gratuitamente scaricabile dal web; assieme ad esso è possibile effettuare il download di alcune librerie che ne consentono una libera ed agevole interrogazione. Dal 1996 un progetto finanziato dalla comunità europea ha stanziato i fondi per la creazione di database linguistici sullo stile di WordNet, per diverse lingue europee, creando il progetto EuroWordNet. Per l'italiano, assieme al progetto di ricerca Sistema Integrato per il Trattamento Automatico del Linguaggio (SI-TAL), è stato sviluppato dall'Istituto di Linguistica Computazionale del CNR di Pisa, il sistema ItalWordNet (IWN). Diversi approfondimenti relativi al progetto si trovano nel sito web [2].

Questa risorsa è ottenibile gratuitamente nel caso la si voglia utilizzare in campo di ricerca, sottoforma di export in formato XML, richiedendo una licenza allo staff del progetto; in più è anche disponibile una versione on-line [3], vedi Figura 9.

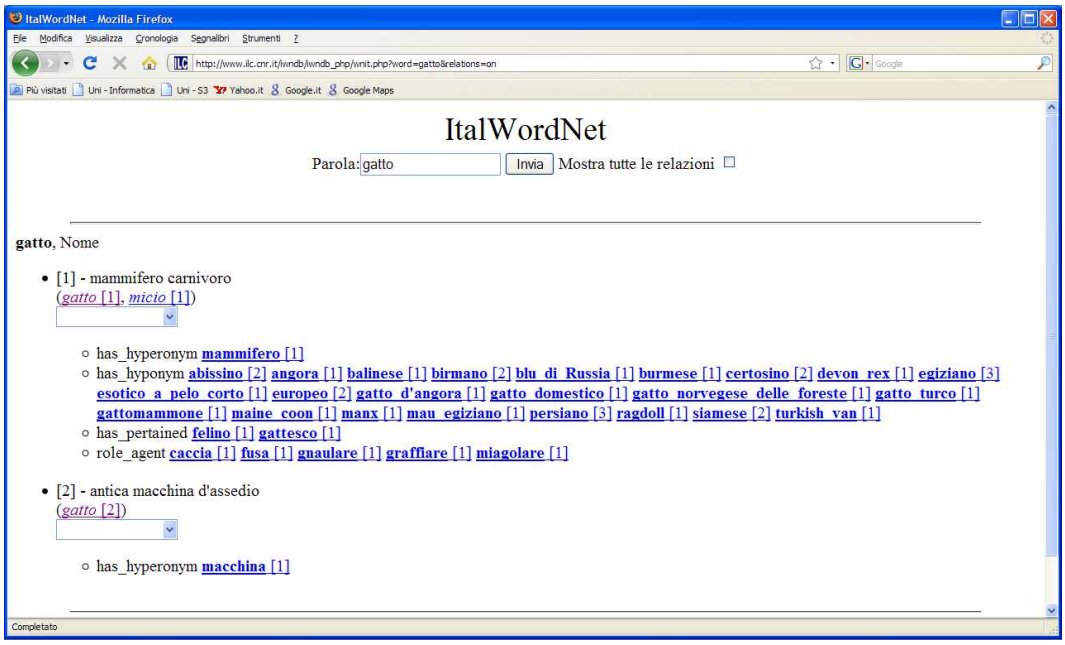


Figura 9. Screenshot di una ricerca effettuata con la versione on-line di IWN.

IWN possiede le stesse caratteristiche presentate in WordNet, seguendo la nozione di synset e mantenendo la struttura gerarchica ad ipernimi. Per questi aspetti risulterebbe decisamente consono all'arricchimento di STRIDER nell'ambito del multilinguismo, consentendo di mantenere attivi gli algoritmi principali. È da notare, però la scarsità di ricchezza linguistica rispetto ad altri dizionari, infatti possiede circa 47.000 lemmi, meno di un terzo di altri "concorrenti", anche se ciò può essere in parte compensato dalla struttura gerarchica. In più è possibile interrogarlo solamente tramite XML, in quanto, come già detto viene fornito sottoforma di export in questo formato.

2.3.2 MultiWordNet

Un altro dizionario che mantiene la struttura ad ipernimi di WordNet e che tenta di mantenere il più possibile la corrispondenza con i suoi synset è MultiWordNet (MWN) [4], un progetto creato da ITC-irst, un gruppo di ricerca di Povo-Trento.

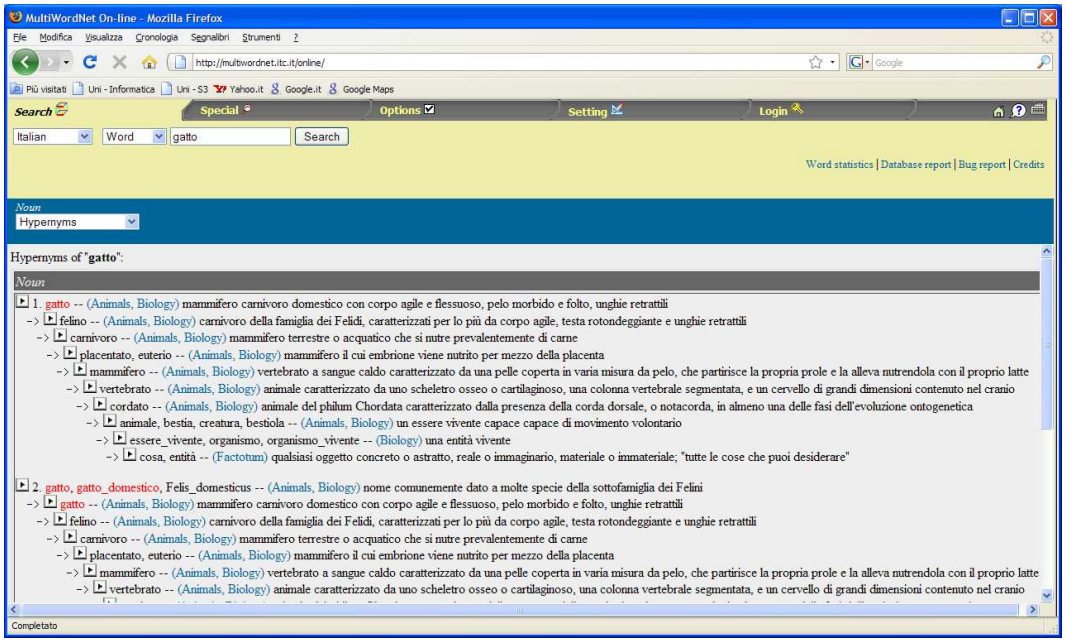


Figura 10. Screenshot di una ricerca effettuata con la versione on-line di MWN.

A differenza di IWN, questo sistema è scaricabile solamente acquistando una licenza, ma è presente anche on-line in una versione gratuita [5], vedi Figura 10.
Dal punto di vista linguistico possiede circa 42.000 lemmi, e per questo motivo sarebbe possibile riproporre lo stesso discorso dei vantaggi e degli svantaggi esposti nel paragrafo precedente, valutando l'importanza della struttura gerarchica nei confronti della scarsità linguistica. Tenendo presente che lo scopo della ricerca non prevedeva la scelta di un dizionario a pagamento, e notando d'altro canto che l'interrogazione della versione on-line non si presenta agevole, non è stato tenuto in considerazione nelle scelte finali.

2.4 Dizionari non strutturati

I dizionari che verranno proposti in questa sezione sono tutti disponibili e consultabili on-line. Il motivo di questa scelta, è stata l'impossibilità di trovare dizionari non strutturati e gratuiti da scaricare e salvare sull'hard disk. Gli esempi riportati sono suddivisi in due categorie: nella prima verranno descritti i vocabolari completi, derivati da grandi opere cartacee, mentre nella seconda quelli creati dagli utenti del web.

2.4.1 Dizionari classici

Nei prossimi paragrafi verranno elencati e descritti i dizionari on-line che sono versioni digitalizzate di grandi opere cartacee. Ne sono un esempio il "De Mauro Parvia", "il Sabatini Coletti" e "Garzanti".

Dizionario "De Mauro Paravia"

La versione on-line del dizionario della lingua italiana "De Mauro Paravia" è consultabile alla pagina web [6], e la Figura 11 ne mostra uno screenshot. Dal

punto di vista linguistico è molto ricco, infatti possiede circa 160.000 lemmi e le definizioni sono arricchite da esempi che possono essere estratti dalle definizioni. Anche l'interrogazione tramite richiesta web è facilitata, perché nonostante nella URL non venga visualizzato esplicitamente il termine richiesto, ma solo il numero del lemma, ad esempio per il termine "gatto" compare:

http://www.demauroparavia.it/47581

è possibile richiederlo ugualmente nel seguente modo:

http://www.demauroparavia.it/cerca?stringa=gatto

Anche l'estrazione delle definizioni dal formato HTML non è troppo laboriosa; per tutte queste considerazioni, il dizionario "De Mauro Paravia" verrà integrato nella versione italiana di STRIDER.

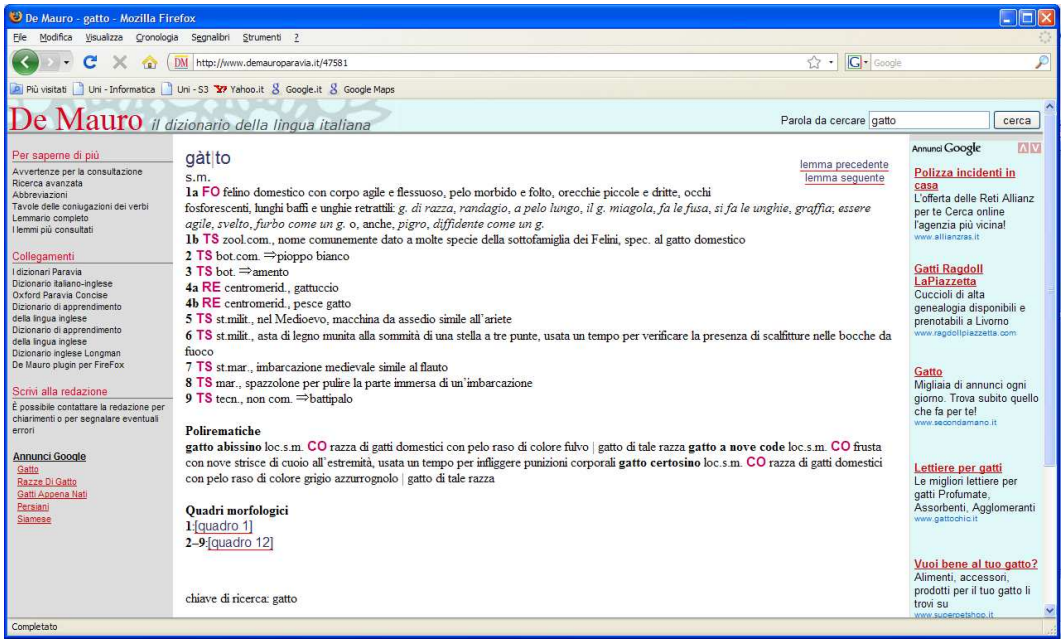


Figura 11. Screenshot di una ricerca effettuata con la versione on-line del dizionario "De Mauro Paravia".

Dizionario "il Sabatini Coletti"

Il dizionario "il Sabatini Coletti" è disponibile on-line nel sito web del quotidiano "Il corriere della sera", ed è consultabile alla pagina web [7]; la Figura 12 mostra lo screenshot di una ricerca. Rispetto al vocabolario precedente, questo è composto da una quantità leggermente minore di lemmi, circa 150.000, che sono ugualmente ricchi di definizioni e di esempi estraibili. L'interrogazione è maggiormente facilitata, in quanto direttamente nella URL, viene indicato il lemma richiesto:

http://dizionari.corriere.it/dizionario_italiano/G/gatto.shtml

Anche in questo caso l'estrazione delle sole definizioni dal formato HTML non risulta essere troppo faticosa. Per questi motivi, il dizionario "il Sabatini Coletti" diventa un altro ottimo candidato nell'integrazione in STRIDER.

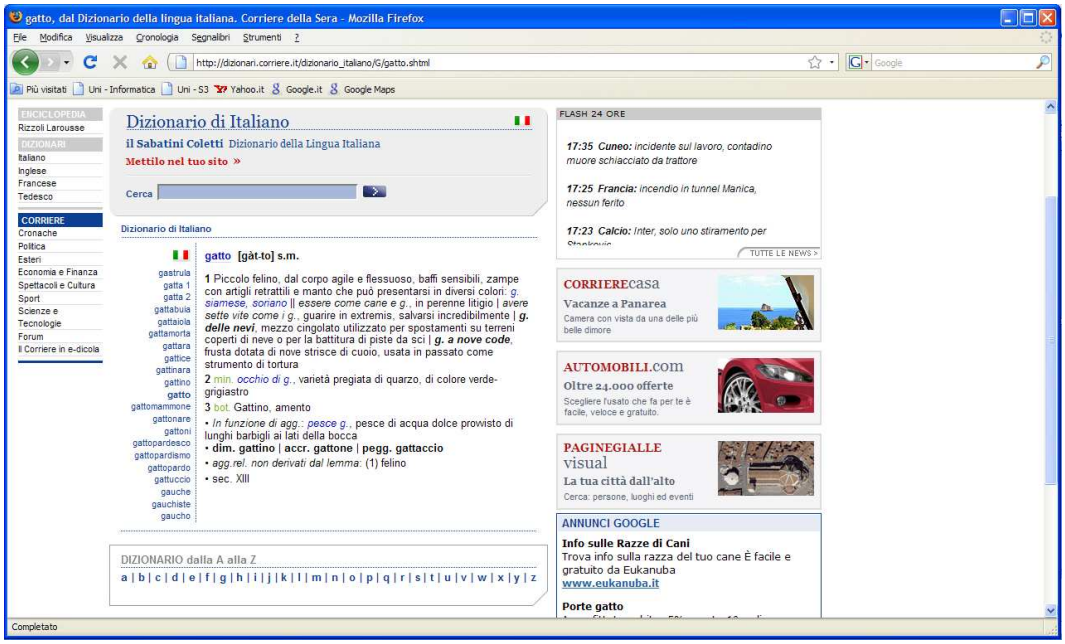


Figura 12. Screenshot di una ricerca effettuata con la versione on-line del dizionario "il Sabatini Coletti".

Dizionario "Garzanti"

Il dizionario "Garzanti" della lingua italiana è consultabile on-line al sito web [8]; la Figura 13 rappresenta lo screenshot della ricerca del termine gatto. Dal punto di vista linguistico, è un dizionario molto ricco, in quanto comprende 200.000 voci con esempi. A differenza dei vocabolari precedenti, però, quest'ultimo richiede l'iscrizione al sito e il login ogni volta che si accede. Questo fatto rende più complicata e lenta la richiesta della pagina e in più non è nemmeno possibile effettuare l'interrogazione modificando direttamente l'URL, nella quale non compare né il termine, né un codice identificativo. Per questi motivi, che non faciliterebbero l'integrazione in STRIDER, è stato scartato, anche se dal punto linguistico è uno dei dizionari più ricchi della lingua italiana, considerando che è on-line e gratuito.

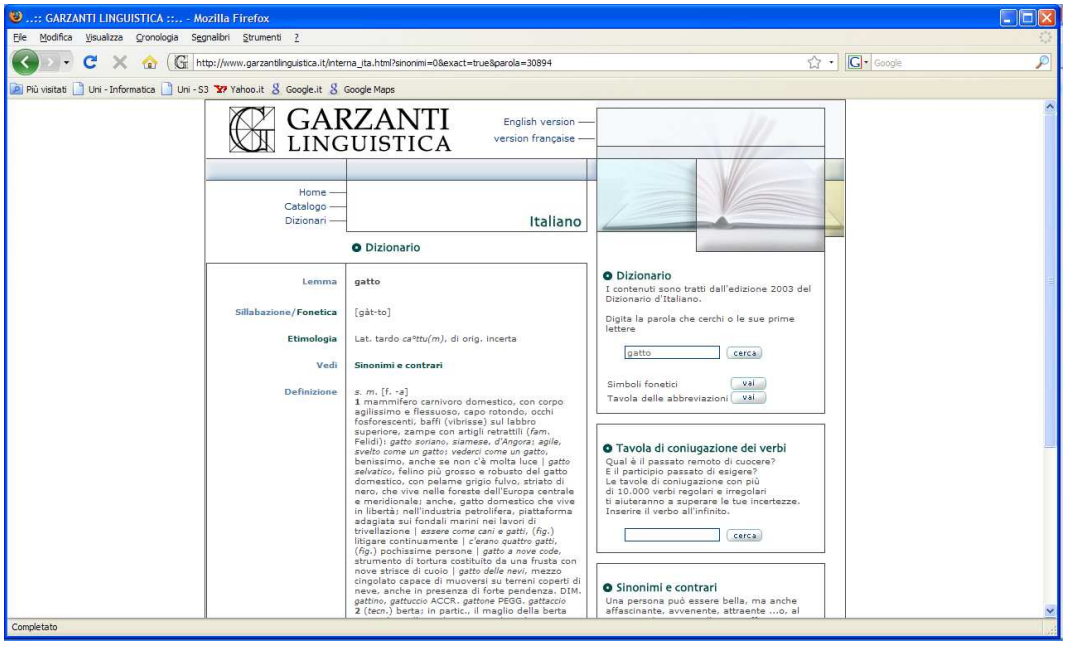


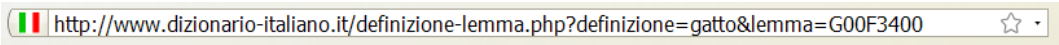
Figura 13. Screenshot di una ricerca effettuata con la versione on-line del dizionario "Garzanti".

2.4.2 Dizionari collaborativi

Con il termine dizionari collaborativi, si intendono tutti quei dizionari on-line che vengono aggiornati di giorno in giorno dagli utenti del web. Hanno la caratteristica di essere sempre in espansione, e quindi di essere aggiornati rispetto ai neologismi della lingua; anche se scritti da utenti, sono rigorosi e corretti, in quanto controllati da altre comunità di utenti o da staff di esperti. Questa tipologia di dizionario deriva probabilmente dal concetto di Wiki, che come definisce Wikipedia, il Wiki più famoso, "*è un sito web, (o comunque una collezione di documenti ipertestuali) che può essere modificato dai suoi utilizzatori e i cui contenuti sono sviluppati in collaborazione da tutti coloro che ne hanno accesso, come in un forum. La modifica dei contenuti è aperta e libera, ma viene registrata in una cronologia permettendo in caso di necessità di riportare la parte interessata alla versione precedente; lo scopo è quello di condividere, scambiare immagazzinare e ottimizzare la conoscenza in modo collaborativo.*".
Ne sono un esempio due dizionari: "Dizionario Italiano" e "Wikizionario".

Il "Dizionario Italiano"

Il "Dizionario Italiano" è consultabile on-line al sito web [9]; la Figura 14 rappresenta lo screenshot della ricerca del termine gatto. Come spiegato nella home page, questo vocabolario è un progetto ancora in corso di realizzazione, che ha l'obiettivo di realizzare il dizionario italiano più esteso, aggiungendo nuovi vocaboli e modi di dire ogni giorno. Già nell'anno 2006 erano presenti più di 100.000 lemmi. Le definizioni sono meno ampie rispetto a quelle dei tradizionali dizionari completi e non sempre contengono esempi. La richiesta della pagina non facilita l'interrogazione da STRIDER, perché anche se compare il nome del termine, è presente anche un codice del lemma non riproducibile a priori:



Per tutti questi motivi è stato scartato dall'integrazione nella versione per la lingua italiana di STRIDER.

Figura 14. Screenshot di una ricerca effettuata con la versione on-line del "Dizionario Italiano".

Il "Wikizionario"

Il dizionario di Wikipedia, chiamato in italiano "Wikizionario", o più internazionalmente "Wiktionary", è consultabile alla pagina web [10];]; la Figura 15 rappresenta lo screenshot della ricerca del termine gatto. Il progetto comprende dizionari per tutte le lingue. Attualmente la versione italiana è costituita da circa 95.000 voci, ma è importante tener conto che è in costante aggiornamento. Le definizioni sono abbastanza sintetiche e sono spesso arricchite da esempi; anche se l'aspetto non interessa direttamente STRIDER, questo dizionario fornisce informazioni aggiuntive, come sinonimi, parole derivate, traduzioni, proverbi e modi di dire. In più ha il vantaggio di poter essere interrogato direttamente modificando la URL:

Notando che l'importanza della ricchezza delle definizioni è fondamentale, è stato escluso dall'integrazione in STRIDER; però, in un futuro, è probabilmente il miglior candidato, in quanto è in continua espansione e offre informazioni aggiuntive che possono aiutare ad arricchire il contesto dei termini.

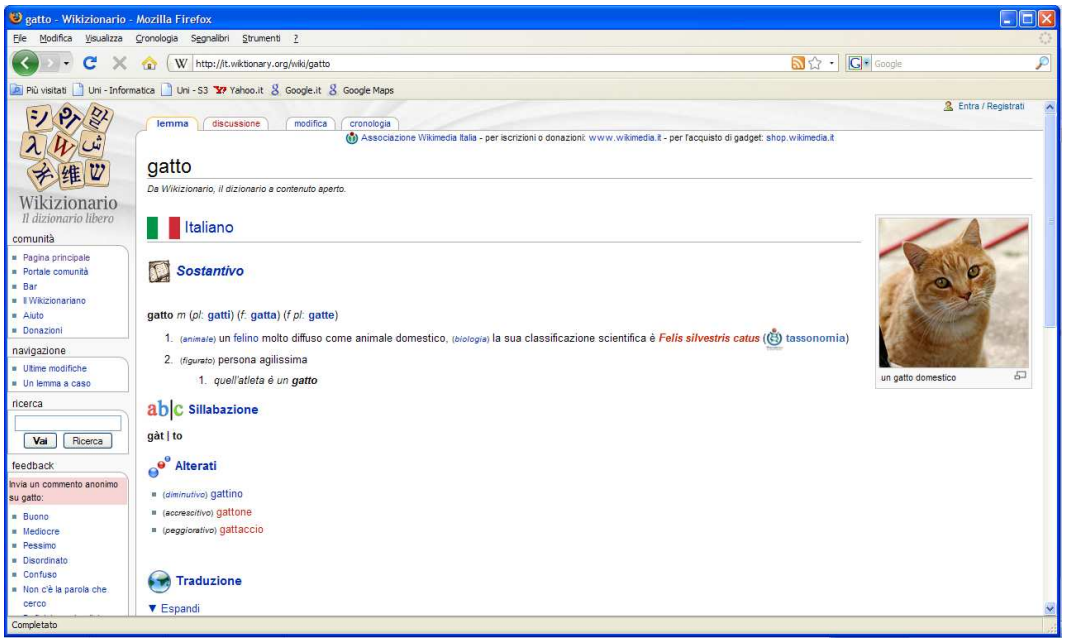


Figura 15. Screenshot di una ricerca effettuata con la versione on-line del "Wikitionary".

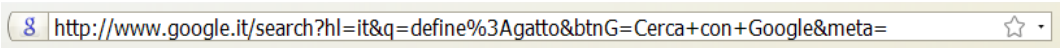
2.5 Dizionari machine-generated

Un ultimo tipo di dizionario valutato per l'integrazione in STRIDER, è il dizionario machine-generated di Google™. Esso, a differenza dei precedenti, non è elaborato e compilato da esperti linguisti, ma ricava la sua conoscenza dall'estrazione automatica di definizioni da enormi corpora di documenti, appunto dalle ricerche effettuare dal motore Google™. Per questo motivo non è possibile determinare il numero di lemmi presenti ed è da notare che le definizioni possono essere "imprevedibili". Questo significa che non sono rigorose e complete come

quelle dei dizionari precedenti, ma sono costituite da citazioni prelevate da siti web; non sempre possono essere complete e attendibili. Questo vocabolario è disponibile anche in tutte le altre lingue, essendo Google™ un motore di ricerca internazionale.

Per interrogare il dizionario è sufficiente inserire nella barra di ricerca del motore la seguente stringa:

oppure modificare la URL nel modo seguente:



Per le sue particolari peculiarità, verrà introdotto in STRIDER, in modo da poter testare un differente genere di dizionario.

2.6 Conclusioni

Concludendo, STRIDER verrà ampliato includendo le funzionalità per la lingua italiana, testando tre dizionari differenti:

- Dizionario “De Mauro Paravia”;
- Dizionario “il Sabatini Coletti”;
- Dizionario di Google™.

È da notare che i ragionamenti effettuati per la ricerca di una fonte di conoscenza esterna, sono applicabili a qualsiasi altra lingua, in quanto anche lo stesso STRIDER è un sistema adatto al multilinguismo.

2.7 Stemming

Lo *stemming*, o normalizzazione, è il processo che trasforma le parole declinate o derivate nel proprio *stem*, cioè nella loro forma base. Lo strumento linguistico che si occupa di questa elaborazione è lo *stemmer*; esso possiede inoltre una funzionalità aggiuntiva, che permette di eliminare le parole insignificanti al fine semantico, come articoli e preposizioni.
In STRIDER risulta molto importante l'uso di uno *stemmer*, prima della fase di categorizzazione delle parole, perché è comunemente noto che all'interno di una frase, le parole non compaiono nella loro forma base e, in caso contrario, la fase di estrazione dei sostantivi potrebbe non considerare tutti i termini coniugati.
Nei prossimi paragrafi verranno descritti alcuni strumenti linguistici, quali POS tagger e database di parole categorizzate; è da notare come entrambi non necessitino direttamente di questa pre-elaborazione, in quanto, i primi analizzati contengono già una funzionalità di *stemmer*, mentre i secondi includono anche le forme coniugate dei termini. Di conseguenza, non è stato necessario effettuare la ricerca di questo strumento per la lingua italiana, perché la sua funzione è già svolta dal POS tagger.

2.8 Part-of-speech tagger

Part-of-speech tagger (POS tagger o POST) è un software che esegue un processo di marcatura della parole contenute in un testo, identificando morfologicamente ciascun elemento della frase, utilizzando le relazioni tra le parole connesse o adiacenti all'interno di frasi o paragrafi. Gli elementi più comuni che vengono

identificati sono nomi, verbi, aggettivi, avverbi, articoli, congiunzioni, pronomi, punteggiatura e in alcuni casi anche simboli ed abbreviazioni. Proprio i sostantivi sarebbero gli elementi di nostro interesse.
Lo scopo di un POS tagger, è quindi quello di determinare il tag adatto ad ogni elemento della frase, tentando di eliminare le frequenti ambiguità presenti nelle parole. Quindi deve disambiguare restituendo un solo tag per elemento, utilizzando le caratteristiche del contesto e quelle probabilistiche fornite dai corpora già annotati. Come accennato nel paragrafo precedente, questo strumento linguistico pre-elabora il testo fornito, fungendo da *stemmer*, eliminando le parole insignificanti al fine semantico e riconducendo le altre alla forma base.
Il processo di *tagging* prende in input un Tagset, un dizionario con tag e una frase da annotare; il risultato fornito in output sarà quindi la frase annotata.
Esistono tre metodi differenti di POS tagger:

- Il metodo a regole, o *rule-based*, che utilizza un database di regole di disambiguazione;
- Il metodo probabilistico, o *stochastic*, che risolve le ambiguità con le evidenze probabilistiche estratte da un corpus già annotato ;
- Il metodo misto, che utilizza regole di disambiguazione ricavate con metodi stocastici.

Il metodo a regole, è basato su un funzionamento che prevede due fasi principali: nella prima vengono assegnati ad una parola tutti i possibili tag, utilizzando un dizionario; nella seconda si applicano delle regole di disambiguazione, create manualmente, per eliminare i tag ambigui.
Il metodo probabilistico, differisce dal precedente nella fase di disambiguazione: in questo caso il tag viene selezionato in base ad evidenze statistiche e alla teoria della probabilità. Esistono due tipi di approccio:

- Approccio semplice, di cui è un esempio il metodo *Most Frequent Tag*;

- Approccio complesso, come ad esempio *Hidden Macov Models* (HMM) e *Transformation-based tagging*.

L'approccio semplice si basa sull'idea che le parole ambigue utilizzano un tag più spesso di altri; per questo motivo il metodo segue tre fasi: nella prima è necessario creare un dizionario e annotare manualmente un corpus con tag non ambigui; nella seconda si calcola la probabilità di annotazione con un determinato tag, per ogni parola ambigua di un nuovo testo; nella terza si assegna il tag più probabile. Generalmente i risultati di quest'approccio sono buoni, cioè il POS tagger annota bene in molti casi; è però da notare che nei casi più rari spaglia sempre. Per migliorare le prestazioni si possono seguire alcuni accorgimenti, come ad esempio aiutarsi guardando i tag della parola precedente e successiva, oppure implementando automaticamente in versione probabilistica le regole dei sistemi *rule-based*. Per informazioni aggiuntive ed esempi, vedere [11].
Maggiori approfondimenti e nozioni riguardanti gli approcci probabilistici complessi, principalmente HMM, si trovano visitando [12].
La rete fornisce molti software gratuiti o con licenza per scopi di ricerca, purtroppo, però, la maggioranza è disponibile soltanto per la lingua inglese. Nei prossimi paragrafi verranno discusse la caratteristiche dei POS tagger disponibili per la lingua italiana e per ultimo verranno elencati anche altri per l'inglese.

2.8.1 TreeTagger

TreeTagger [13] è un tool opensource per l'annotazione testuale di tipo HMM, che contiene informazioni sulla struttura delle frasi e sui lemmi, ed è stato sviluppato assieme al progetto TC [14] all'Istituto di Linguistica Computazionale dell'Università di Stoccarda (*Institute for Computational Linguistico of the University of Stuttgart*). Questo tool è stato usato con successo per annotare testi in lingua tedesca, inglese, francese, italiana, olandese, spagnola, bulgara, russa, greca, portoghese, cinese e francese antico, ed è facilmente adattabile ad altre

lingue se sono disponibili un dizionario e un *training corpus* annotato manualmente.

TreeTagger è scaricabile direttamente dal sito [13], ed è disponibile per computer con sistema operativo Linux, Sparc-Solaris, Mac OS-X e Windows. I *parameter files* e i *tag set* per la lingua italiana sono stati forniti da Marco Baroni, ricercatore del gruppo CLIC del CIMeC (Center for Mind/Brain Sciences of the University of Trento), e membro del Dipartimento DISCoF dell'Università di Trento. Una versione on-line per la lingua italiana è presente sul sito [15], che può analizzare brani lunghi fino a 1000 parole; la Figura 16 ne mostra uno screenshot.

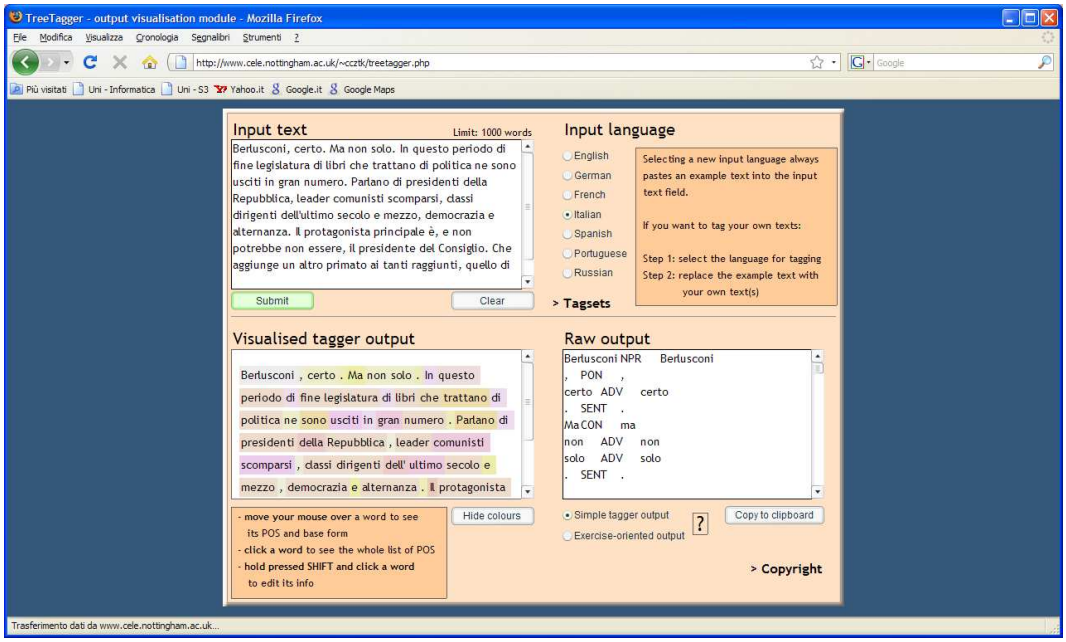


Figura 16. Screenshot di una delle versioni on-line del tool "TreeTagger".

Il software scaricabile, si presenta come un'applicazione scritta in linguaggio Perl, che può essere lanciata sia da linea di comando, sia utilizzando una GUI presente per alcuni sistemi operativi. In modo totalmente automatico, prende in input il *tag file* della lingua da utilizzare e il nome del file che contiene il testo da annotare; se si specifica anche il nome di un file di output, vi stampa il risultato, altrimenti lo mostra a video. Il programma contiene anche una funzione di *tokenization* per riportare le frasi del file di input, in un formato accettato da TreeTagger. Un

esempio di output per la lingua inglese della frase “The TreeTagger is easy to use.” è rappresentato nella Tabella 2, dove nella prima colonna è riportato il termine originale, nella seconda il tag corrispondente (le descrizioni delle sigle sono presenti nel *tag set*) e nella terza la forma base del lemma.

word	**pos**	**lemma**
The	DT	the
TreeTagger	NP	TreeTagger
is	VBZ	be
easy	JJ	easy
to	TO	to
use	VB	use
.	SENT	.

Tabella 2. Esempio di output dell’applicazione TreeTagger, per la frase inglese “The TreeTagger is easy to use.”.

È uno strumento molto valido, ma è da tener conto che essendo un analizzatore bastato su un metodo stocastico non può, però, essere infallibile. In più, per essere utilizzato in STRIDER, che è un’applicazione scritta in Java, dovrebbe essere richiamato come comando esterno, e questo rallenterebbe l’esecuzione. Per questi motivi non è stato utilizzato.

2.8.2 TextPro

Un altro POS tagger disponibile per la lingua italiana è TextPro, che è utilizzabile anche per l’inglese; è un progetto sviluppato da ITC-irst, un gruppo di ricerca di Povo-Trento, lo stesso gruppo del dizionario MultiWordNet. Tutta la documentazione e una versione demo on-line è presente nel sito [16]; la Figura 17 ne mostra uno screenshot. Il tool è gratuito, ma per essere scaricato, richiede una particolare licenza fornita per scopi di ricerca. È da sottolineare che TextPro non

ha sola la funzione di POS tagger, ma contiene anche altre funzionalità come quelle di tokenization, scissione di stringhe, analisi morfologica ed estrazione dei lemmi. L'output fornito dal programma è del tutto simile a quello del POS tagger proposto precedentemente; l'unica differenza resta nel *tag set*, che varia a seconda dei *training corpus* utilizzati per addestrare l'applicazione.

Il software si presenta in linguaggio Perl, e per lo stesso motivo per il quale il precedente TreeTagger è stato scartato, anche TextPro non è totalmente adatto alla versione italiana di STRIDER.

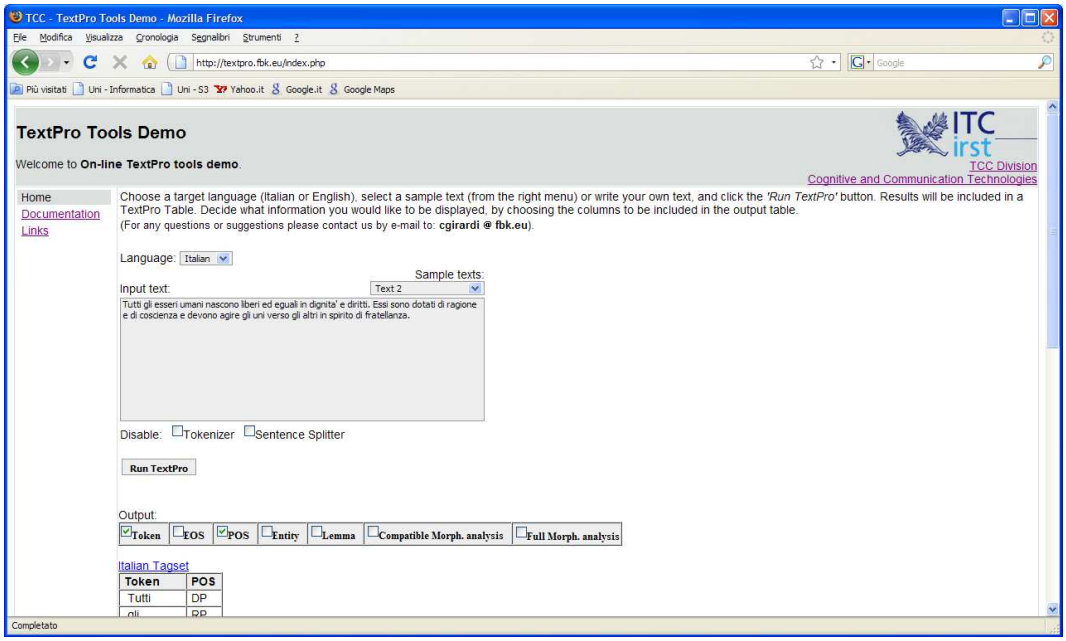


Figura 17. Screenshot della versione on-line del tool "TextPro".

2.8.3 Altri POS tagger

Come accennato precedentemente, molti POST non possiedono le funzionalità per la lingua italiana. Molti di questi, però, danno la possibilità all'utente di addestrare il programma, creando dei propri file sorgenti. Un esempio è QTag, un'applicazione di tipo stocastica sviluppata da Oliver Mason dell'Università di Birmingham, che possiede le caratteristiche di POS tagger per le lingua inglese e

tedesca. Nella home page del tool [17], sono fornite alcune informazioni che consentono di utilizzarlo, addestrandolo preventivamente con un file contenete del testo già annotato. Notando che Qtag è scritto in linguaggio Java, sarebbe stata un'ottima soluzione per STRIDER, ma è da tener presente che la creazione di un file di addestramento va al di fuori degli scopi della tesi; per questo motivo è stato scartato.
Solo per citarne alcuni, altri esempi di POS tagger presenti in rete per lingue diverse dall'italiano sono:

- Stanford Log-linear Part-Of-Speech Tagger, sviluppato dall'Università di Stanford [18];
- CLAWS part-of-speech tagger for English, sviluppato dal gruppo UCLER dell'Università di Lancaster [19];
- POS tagger, sviluppato dal gruppo Tsujii dell'università di Tokyo [20];
- MontyLingua e MontyTagger [21].

2.9 Parser

Una buona alternativa al Part-of-speech tagger è offerta dai parser, degli strumenti linguistici che oltre a fornire i risultati della tecnologia sopra descritta, offrono maggiori informazioni sulla struttura grammaticale delle frasi. Un parser, infatti, è un programma che data una grammatica formale riesce ad analizzare uno stream di dati in input con lo scopo di determinarne la sua struttura grammaticale.
Purtroppo, però, scarseggiano gli strumenti per la lingua italiana; l'unico degno di nota è CHAOS, "*a robust syntactic parser for Italian and for English*" [22], sviluppato dal gruppo AI-NLP (*Artificial Intelligence Research Group at Tor Vergata*) dell'Università di Roma Tor Vergata. Anche questo tool è scaricabile gratuitamente, ma richiede la registrazione nel sito web. Anche se scritto in Java, e applicabile sia per la lingua italiana che per l'inglese, risulta essere non adatto a STRIDER.

2.10 Morph-it!

Un'ultima soluzione al problema dell'estrazione dei sostantivi da frasi in lingua italiana è attuabile utilizzando dei database di parole categorizzate. Con questo termine si intende un grosso insieme di parole che caratterizzano una lingua tra cui nomi, verbi, aggettivi, avverbi, articoli, congiunzioni, pronomi, punteggiatura, e abbreviazioni sia nella forma base, che nelle possibili forme coniugate (maschile, femminile, plurale, singolare, coniugazione nei tempi verbali …).

Un buon database gratuito e scaricabile dalla rete è Morph-it!, sviluppato da Marco Baroni e Eros Zacchetta, presso SSLMIT (Scuola Superiore di Lingue Moderne per Interpreti e Traduttori) dell'Università di Bologna [23]. Questa base di dati che è costituita da 34.968 lemmi, coniugati in 504,906 parole, deriva in gran parte dal corpus "La Repubblica". Quest'ultimo è un grande corpus formato da articoli di giornale, prelevati appunto dal quotidiano "La Repubblica"; altre informazioni riguardanti il coupus e i campi nel quale è stato utilizzato, sono reperibili alla pagina web [24]. La Figura 18 mostra uno screenshot della versione demo on-line del database Morph-it!, raggiungibile dall'home page del progetto.

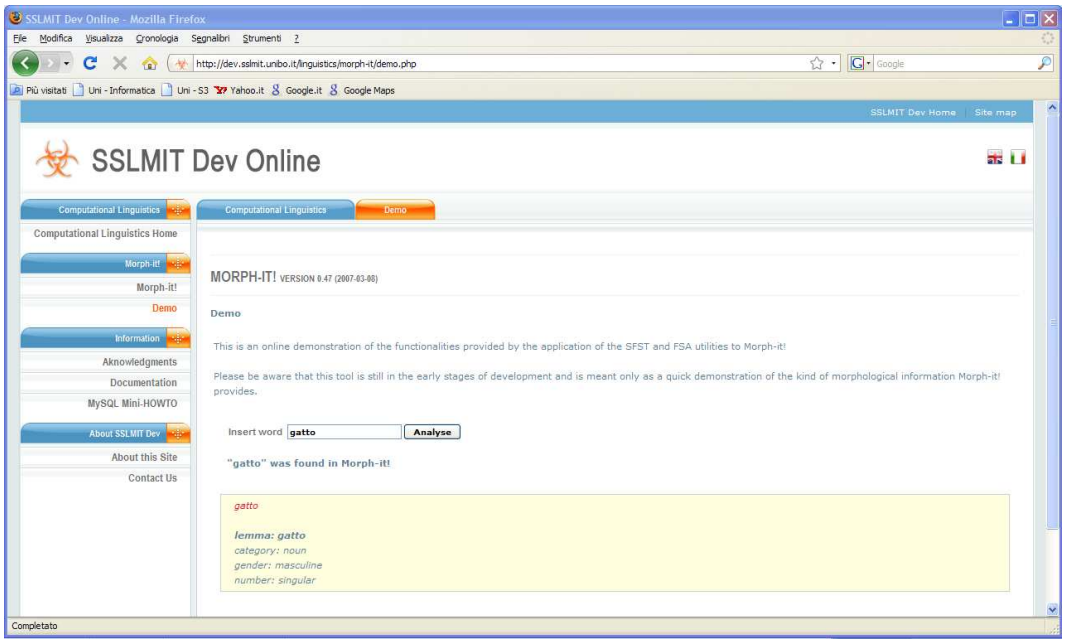


Figura 18. Screenshot della versione on-line del database "Morph-it!".

Le parole contenute all'interno del database sono categorizzate come viene rappresentato nella Tabella 3, dove "Form" identifica la forma della parola da cercare, "Lemma" il termine nella forma base e "Features" il tipo di vocabolo e la sua coniugazione.

Form	**Lemma**	**Features**
rimpinzeremmo	rimpinzare	VER:cond+pre+1+p
abominevole	abominevole	ADJ:pos+m+s
dabbenaggine	dabbenaggine	NOUN-F:s
ostensibilmente	ostensibilmente	ADV

Tabella 3. Esempio di categorizzazione delle parole presenti nel database Morph-it!.

L'elenco delle possibili "Features" sono descritte nel file readme [25] e nell'Appendice B.

La possibilità di ricercare parole in forma coniugata, è un grosso vantaggio, in quanto evita l'utilizzo di uno stemmer, come descritto nel paragrafo 2.7; in più, considerando che STRIDER impiega già il DBMS MySQL per memorizzare dei dati, diventa molto conveniente adottare la soluzione che coinvolge Morph-it!.

La scelta di utilizzare un database, elimina però, le funzionalità fornite dai veri POS tagger relative alle relazioni presenti in una frase; infatti essi determinano, a seconda del contesto, se un termine, che può essere ad esempio nome o verbo, è riconducibile alla prima o alla seconda categoria. Testando i precedenti POS tagger su alcune frasi di uso comune, come ad esempio "Oggi gioco a calcio" e "Il gioco del calcio è divertente", dove il termine "gioco" funge rispettivamente da verbo e da sostantivo, è stato notato che la scelta del tag non sempre avveniva in modo corretto. Questo è stato un altro punto a favore nella scelta di Morph-it!.

2.11 Conclusioni

Riassumendo, gli strumenti linguistici relativi alla lingua italiana che sono stati scelti per essere integrati nella versione multilingue di STRIDER sono i seguenti:

- Dizionario:
  - "De Mauro Paravia"
  - "il Sabatini Coletti"
  - Dizionario di Google™

- Database di parole categorizzate:
  - Morph-it!

Nella Parte II verrà prima presentata, tramite schemi UML, la struttura del progetto e delle nuove classi Java che sono state implementate, e successivamente verranno esposte le prove sperimentali effettuate per testare il funzionamento della nuova versione di STRIDER, su schemi in lingua italiana, utilizzando tre differenti dizionari on-line.

Parte II

Estensione di STRIDER

Capitolo 3

Progetto

La progettazione dell'estensione dell'applicazione è passata attraverso una serie di fasi. Per primo sono stati raccolti i requisiti funzionali da sviluppare e successivamente sono stati sviluppati alcuni casi d'uso, *activity diagram*, e *component diagram*, allo scopo di chiarificare la struttura delle nuove classi. Il tutto è stato creato seguendo le regole dello standard ANSI/IEEE 830 [26] relative all'SRS (*Software Requirements Specification*) e a quelle che definiscono l'UML (*Unifical Modelling Language*)

3.1 Requisiti funzionali

In questo paragrafo sono elencati e descritti i requisiti funzionali relativi all'ampliamento di STRIDER, che sono necessari per aggiungere le nuova funzionalità.

<u>F1: Preparazione di schemi in lingua italiana</u>
Questa funzionalità esamina gli schemi proposti in lingua italiana, e effettua tutte le operazioni intermedie, che portano alla produzione della struttura dati di input per gli algoritmi di disambiguazione.

RF01: Lingua, dizionario ed esempi

- ✓ *Descrizione:* Il sistema estrae i parametri inseriti dall'utente, riguardanti la lingua, il tipo di dizionario e gli esempi.
- ✓ *Input:* Tipo di lingua e dizionario scelti; valore booleano relativo agli esempi, che indica se utilizzare o no gli esempi all'interno delle definizioni nella creazione del contesto.
- ✓ *Processo:* A seguito della modifica dei dati nelle caselle predisposte, da parte dell'utente, il sistema provvede a controllare che la lingua indicata sia l'italiano; poi verifica che il numero del dizionario immesso sia tra quelli validi ed annota il valore booleano relativo agli esempi.
- ✓ *Output:* Solo in caso di incongruenza di almeno uno dei parametri, fornisce un messaggio di errore.

RF02: Connessione e richiesta pagina web

- ✓ *Descrizione:* Il sistema si connette al sito web contenente le informazioni da prelevare e richiede la pagina relativa al termine in questione.
- ✓ *Input:* Indirizzo del sito web, termine da ricercare.
- ✓ *Processo:* Una volta ricevuto il termine da ricercare, e identificato il sito web relativo al dizionario selezionato, viene creata la URL. Successivamente il sistema richiede la pagina web contenente le informazioni e salva tutto il codice HTML. Nel caso in cui il termine sia polisemico, è necessario determinare le caratteristiche delle nuove URL, riconnettersi al server per richiedere le informazioni e salvare tutto il codice HTML.
- ✓ *Output:* Se tutto va a buon fine, fornisce una variabile contenente tutto il codice HTML. In caso contrario segnala un errore, che può essere dovuto alla connessione, alla mancanza della pagina, etc.

RF03: Estrazione delle definizioni

- ✓ *Descrizione:* Dall'HTML, il sistema deve estrarre le definizioni dei termini, tralasciando tutte le altre informazioni contenute nella pagina salvata.
- ✓ *Input:* Indicazione del tipo di dizionario e variabile contenente il codice HTML.
- ✓ *Processo:* Una volta ricevuta la variabile contenente il codice HTML, vengono estratte le definizioni, applicando le regole create manualmente che differiscono per ogni tipo di dizionario.
- ✓ *Output:* Se tutto va a buon fine, viene creata una struttura dati contenente le definizioni estratte. In caso contrario, viene segnalato un errore.

RF04: Estrazione dei nomi

- ✓ *Descrizione:* Utilizzando uno strumento linguistico, il sistema estrae i sostantivi.

- ✓ *Input:* Struttura dati contenente le definizioni.
- ✓ *Processo:* Dopo aver ricevuto la struttura dati, il sistema preleva una definizione alla volta e utilizzando lo strumento linguistico predisposto, estrae e salva tutti i sostantivi che compongono le frasi. In questo modo termina la creazione della struttura dati di input per gli algoritmi di disambiguazione.
- ✓ *Output:* Se tutto va a buon fine, viene ampliata la struttura dati, inserendo tutti i sostantivi contenuti nelle frasi. In caso contrario, viene segnalato un errore.

3.2 Scenario: Estrazione dei sostantivi relativi al termine dato

In questo paragrafo verrà descritto a parole lo scenario che successivamente sarà esemplificato graficamente nello schema UCD (*Use Case Diagram*).

A seguito dell'immissione da parte dell'utente dei parametri relativi alla lingua, al tipo di dizionario e all'utilizzo di esempi, il sistema memorizza ed elabora i valori. Successivamente, preleva il termine da ricercare, che gli viene fornito da un'altra funzionalità dell'applicazione, e a seconda dei parametri memorizzati nella fase precedente, crea l'URL e richiede la pagina al server.

Se la connessione, la richiesta e la ricezione sono andate a buon fine, salva il contenuto della pagina HTML in una variabile, che verrà analizzata per estrarne le definizioni. Può capitare che il termine ricercato sia polisemico, e che quindi le sue diverse forme siano contenute in più pagine web, referenziate dalla principale, cioè da quella appena salvata. In questo caso vengono prelevati i codici relativi alle nuove ricerche da effettuare, e mediante il procedimento descritto poco fa, si estraggono i contenuti delle pagine HTML fornite.

Al termine di questa procedura, tramite regole create manualmente e caratteristiche di ogni dizionario, vengono estratte le definizioni dal codice HTML, che verranno poi salvate in una struttura dati ad oc.

Infine, utilizzando uno strumento linguistico (descritto nel paragrafo 3.6), vengono estratti tutti i sostantivi contenuti in ciascuna definizione, e poi memorizzati in una nuova struttura dati, che diventerà l'input degli algoritmi di disambiguazione.

3.3 Use Case Diagram

In Figura 19 è rappresentato il diagramma degli attori che partecipano allo scenario proposto nel paragrafo precedente, e che prenderanno parte agli UCD.

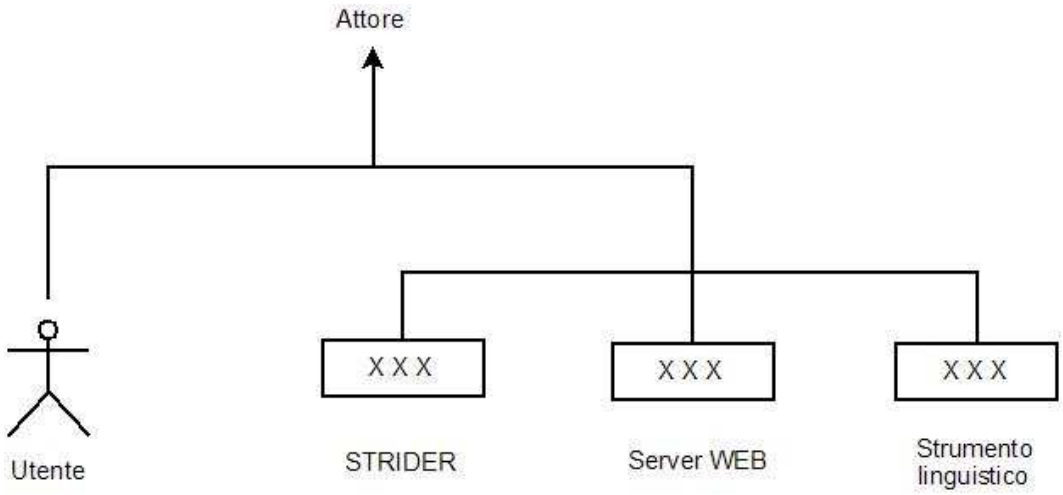


Figura 19. Diagramma degli attori.

Ora verranno mostrate due versioni dello stesso scenario; nella prima, Figura 20, è rappresento il "diagramma scheletro" mentre nella seconda, Figura 21, un diagramma maggiormente dettagliato. Per l'ampliamento è stato utilizzato il metodo top-down.

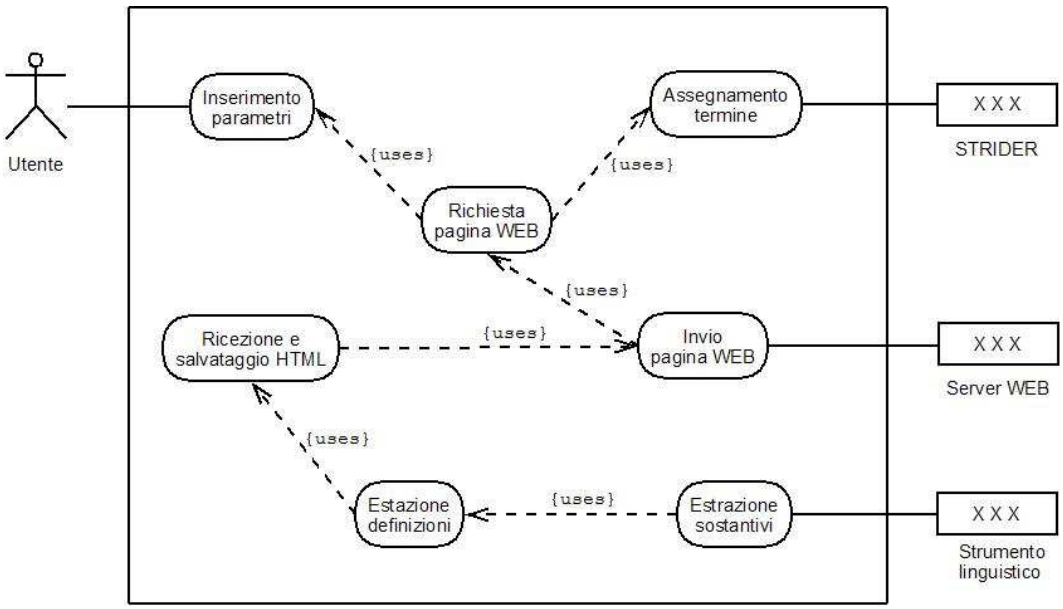


Figura 20. Prima versione di UCD relativo allo scenario "Creazione della struttura dati contenente i sostantivi relativi alle definizioni del termine dato".

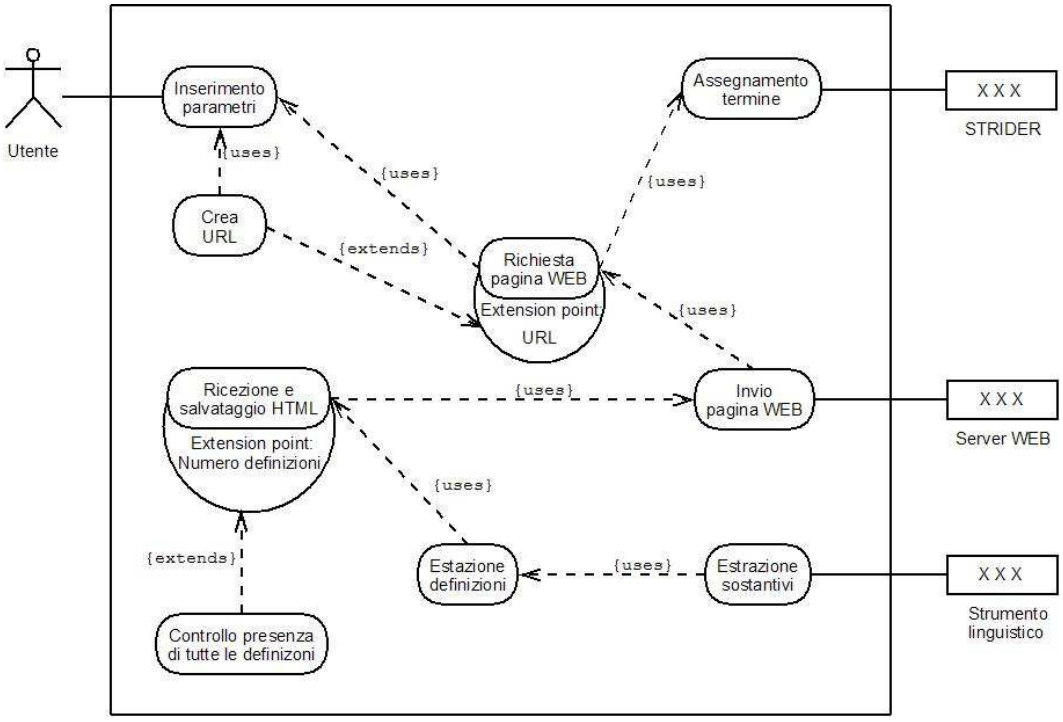


Figura 21. Seconda versione di UCD relativo allo scenario "Creazione della struttura dati contenente i sostantivi relativi alle definizioni del termine dato".

3.3.1 Use case 01: Inserimento parametri

UC01: Inserimento parametri

- ✓ *Obiettivo:* La funzione ha lo scopo di valutare i parametri immessi dall'attore primario.
- ✓ *Attore primario:* Utente.
- ✓ *Ambito:* Lo UC01 è collocato nel sottosistema che crea i prerequisiti necessari per richiedere le informazioni sul termine.
- ✓ *Evento scatenante:* Inserimento e conferma, nella form, dei parametri immessi manualmente dall'utente.
- ✓ *Preposizione:* Sono già presenti, negli appositi campi, i valori di default, o i valori immessi al passo precedente dall'utente.
- ✓ *Sequenza:*
  1. L'utente inserisce il valore booleano relativo agli esempi e le stringhe che identificano la lingua e il tipo di dizionario.
  2. Il sistema verifica che i valori siano accettabili.
- ✓ *Post condizione di successo:* I valori vengono salvati e sono a disposizione del sistema per le fasi successive.
- ✓ *Post condizione di fallimento:* I nuovi valori non sovrascrivono i precedenti e viene mostrato un messaggio d'errore all'utente.

3.3.2 Use case 02: Richiesta pagina WEB

UC02: Richiesta pagina WEB

- ✓ *Obiettivo:* La funzione ha lo scopo di richiedere la pagina web contenente la definizione del termine fornito.
- ✓ *Attore primario:* Utente, STRIDER e server WEB.
- ✓ *Ambito:* Lo UC02 è collocato all'interno del sottosistema che raccoglie le informazioni relative ad un termine.
- ✓ *Evento scatenante:* Creazione della URL.

- ✓ *Preposizione:* Devono già essere stati forniti il nome del dizionario dall'utente e il termine da STRIDER.
- ✓ *Sequenza:*
  1. Il sistema preleva la URL creata dalla UC04 ("Crea URL") e si connette al server.
  2. Invia la richiesta HTML relativa al termine da cercare.
  3. Il sistema verifica che i valori siano accettabili.
- ✓ *Post condizione di successo:* Attende la risposta del server e l'invio del risultato.
- ✓ *Post condizione di fallimento:* Ritenta la richiesta con una URL modificata; in caso di nuovo fallimento invia un messaggio d'errore all'utente.

3.4 Activity Diagram

In questo paragrafo verrà descritta, utilizzando l'*activity diagram* proposto in Figura 22, la sequenza generica che coinvolge la ricerca e l'estrazione dei nomi. Successivamente, nella Figura 23, verrà proposta la successione di attività introdotte nel caso in cui il termine sia polisemico; infatti, come già accennato nel paragrafo 3.2, quando un termine è polisemico, le pagine web che contengono le sue definizioni sono più di una. Per questo motivo è stato creato un percorso differenziato per le parole che presentano queste caratteristiche. Il secondo diagramma è valido per tutti i dizionari utilizzati, anche se è da tener in considerazione che ciascuno possiede una struttura interna differente e al momento dell'implementazione, i dettagli varieranno, ma la struttura del codice sarà la stessa.

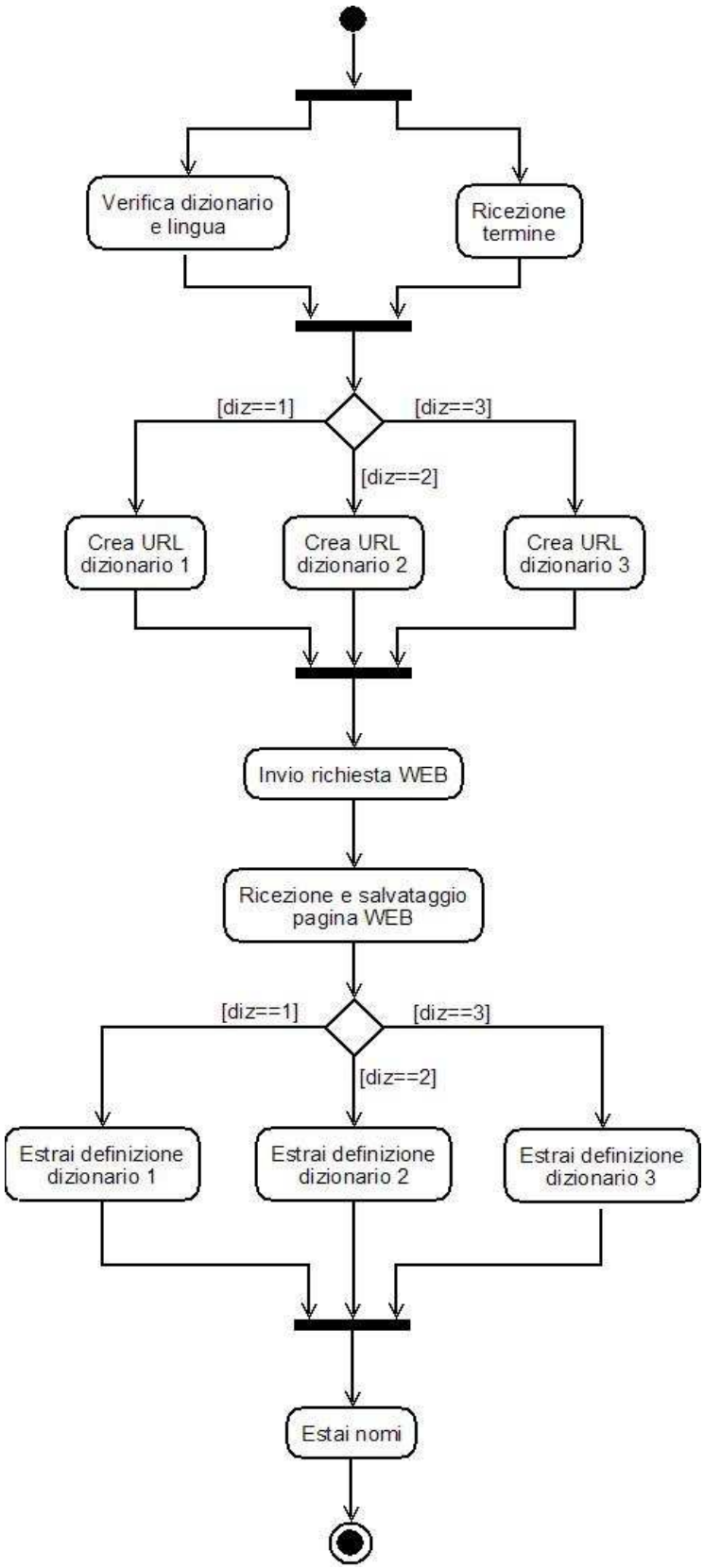


Figura 22. Activity diagram relativo alla sequenza di operazioni per estrarre i nomi.

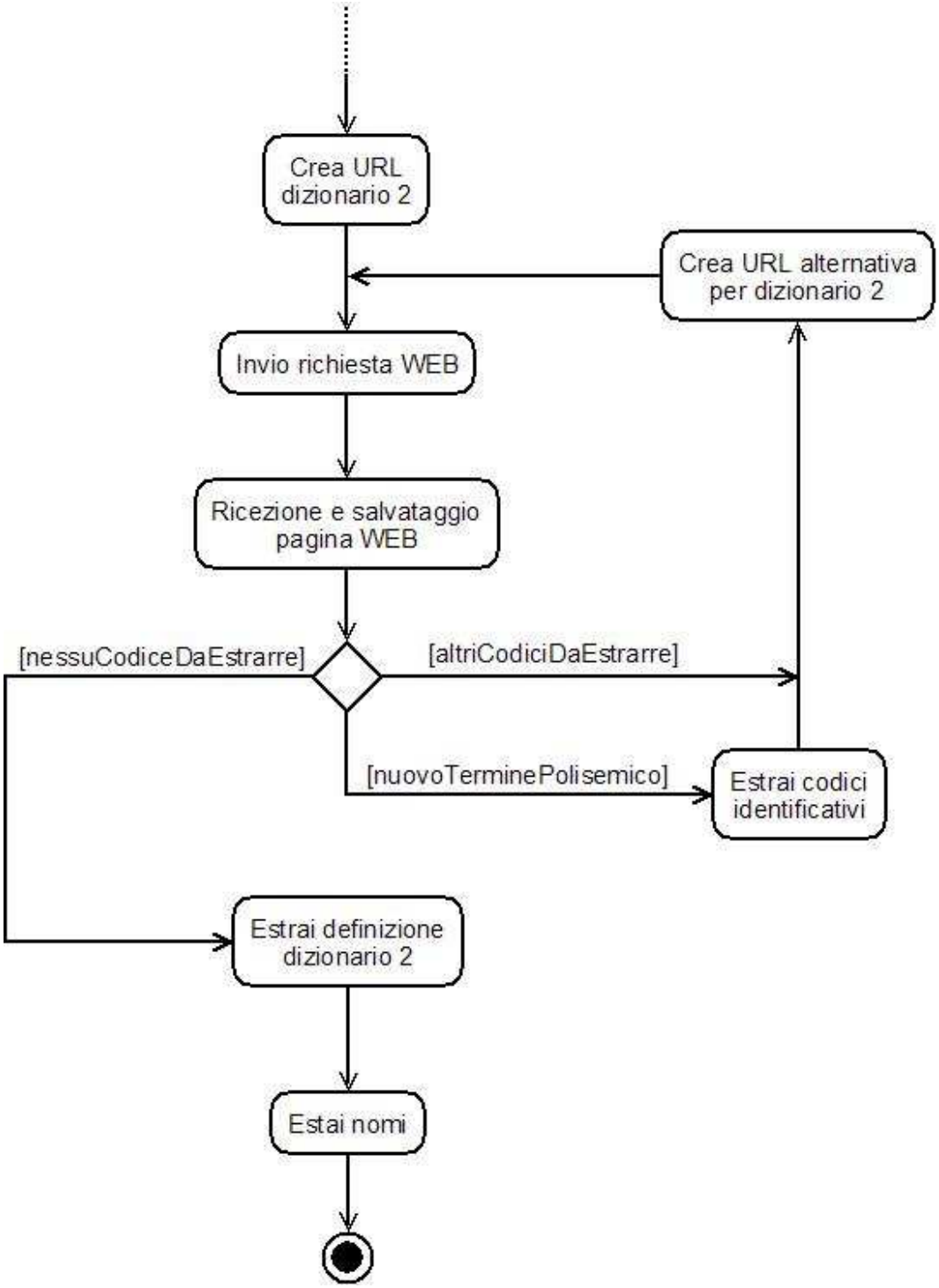


Figura 23. Activity diagram relativo all'estrazione dei nomi per i termini polisemici.

Al solo scopo esemplificativo, verrà utilizzato il dizionario 2, cioè "il Sabatini Coletti"; per non ripetere le prime attività, che sono identiche a quelle della Figura 12, l'*activity diagram* di Figura 23 inizierà da "Crea URL dizionario 2".

3.5 Component diagram e classi Java del progetto

In questo paragrafo, tramite il *component diagram* di Figura 24, verranno schematizzate le classi Java che sono state utilizzate nell'implementazione e le relazioni che le legano.
Le classi principali che sono state realizzate sono:

- **Dizionario:** l'interfaccia che verrà implementata per ogni dizionario che sarà testato; contiene i seguenti metodi:
  - lavora(), implementa il flusso centrale del programma, richiamando le altre classi che provvedono a creare la connessione, estrarre le definizioni e i nomi;
  - creaUrl(), crea l'URL ad oc per ogni termine;
  - statistiche(), calcola alcuni valori utili per le statistiche del programma;
  - ricerca(), nel caso in cui il POS tagger non riesca ad identificare nessun sostantivo all'interno della frase, questo metodo utilizza il dizionario che lo implementa per verificare se una parola passata come parametro è un nome; in questo caso la inserirà nella struttura dati contenente i sostantivi;
  - cercaGenere(), verifica che il termine sia un sostantivo;
  - getDefinizioni(), restituisce la struttura dati contenente le definizioni;

- **DizDeMauroParavia:** è l'implementazione dell'interfaccia Dizionario relativa al dizionario on-line "De Mauro Paravia". Oltre ad implementare i metodi elencati precedentemente, contiene anche:
  - creaUrlCodice(), crea la URL relativa al codice identificativo del termine;
  - ricercaTermineAccentato(), prepara i termini accentati da inserire nella URL, utilizzando i codici dei caratteri speciali HTML;

- **DizSabatiniColetti:** è l'implementazione dell'interfaccia Dizionario relativa al dizionario on-line "il Sabatini Coletti". L'unico metodo aggiuntivo che viene implementato, rispetto a quelli precedentemente elencati è:
    - eliminaSec(), elimina, a seconda dei casi, la ultima o le ultime due definizioni presenti nel dizionario, che riguardano il periodo in cui è avvenuto il primo utilizzo del termine;

- **DizGoogle:** questa classe implementa l'interfaccia Dizionario relativamente al dizionario di Google™. Anche in questo caso viene implementato un unico altro metodo aggiuntivo:
    - modificaParole(), sostituisce le lettere accentate presenti nel termine, con le rispettive lettere non accentate;

- **WebConnection:** questa classe ha lo scopo di creare la connessione web, utilizzando la URL che gli viene fornita dal dizionario che la istanzia. Il metodo centrale è:
    - connetti(), crea la connessione web e salva il contenuto della pagina HTML ricevuta in una stringa;

- **Definizioni:** questa classe ha lo scopo di estrarre le definizioni dalle variabili contenenti il codice HTML della pagina ricevuta, fornite dal dizionario che la invoca; ogni dizionario possiede le proprie regole per l'estrazione. I primi quattro metodi elencati sono i principali, mentre i successivi forniscono delle funzionalità necessarie per l'implementazione dei precedenti:
    - trovaDefPar(), implementa le regole per l'estrazione dal dizionario "De Mauro Paravia";
    - trovaDefSabCol(), implementa le regole per l'estrazione dal dizionario "il Sabatini Coletti";

- trovaDefGoogle(), implementa le regole per l'estrazione dal dizionario di Google™;
- estraiDef(), estrae le definizioni in un formato pulito, sostituendo i codici dei caratteri speciali HTML e altri tag, con caratteri in formato UNICODE;
- trovaSostPar(), estrae i codici dei termini polisemici per il dizionario "De Mauro Paravia";
- cercaSottoStringa(), ricerca le occorrenze di una sottostringa, all'interno della stringa contenente l'HTML dell'intera pagina;
- trovaFine(), ricerca le occorrenze un particolare tag di fine;
- trovaSep(), ricerca le occorrenze di un particolare tag separatore;
- trovaHref(), ricerca le occorrenze del tag HTML contenente "href";

- **Nomi:** questa classe ha lo scopo di interrogare il POS tagger che è stato creato, collegandosi al DBMS che lo contiene. I metodi principali sono:
  - dbConnection(), crea la connessione al database;
  - getLista(), chiama il metodo getNomi() e salva il risultato in una struttura dati contenente sostantivi; per ultimo chiude la connessione con il database;
  - getNomi(), estrae le singole parole dalla definizione, e per ciascuna di esse chiama il metodo isNome();
  - isNome(), verifica, interrogando il database, se una parola è un nome;
  - getAllName(), seleziona tutti i sostantivi contenuti nel database.

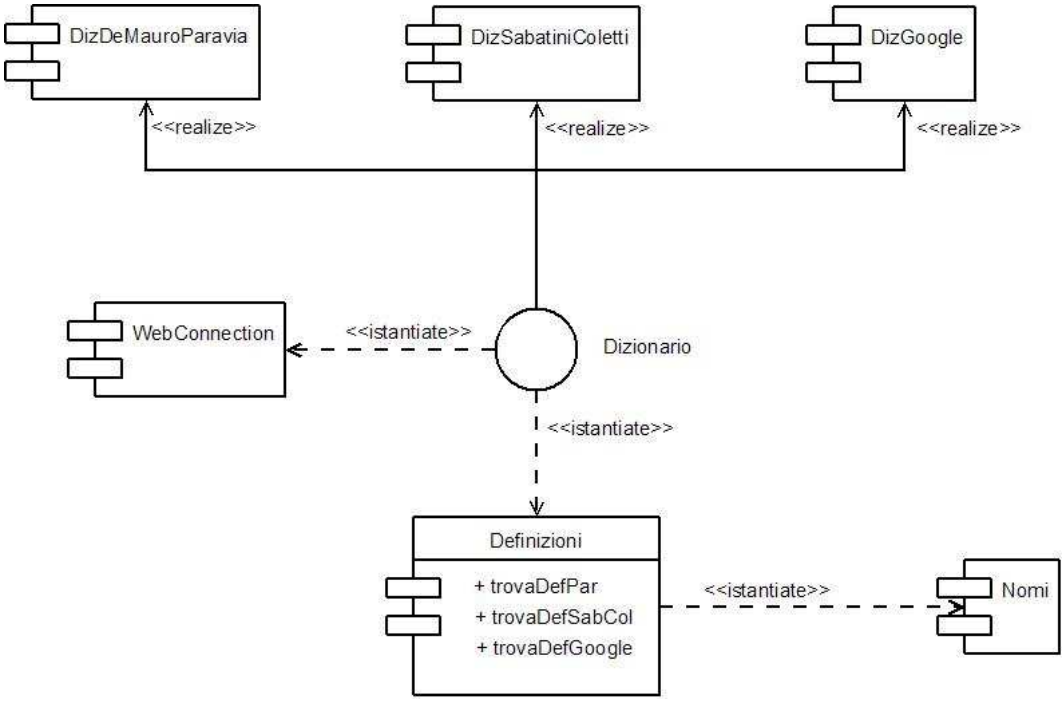


Figura 24. Component diagram relativo alle classi Java del progetto.

3.6 Morph-it! come POS tagger

Morph-it!, come è già stato accennato nel paragrafo 2.10, è un database di parole categorizzate, contenente nomi, verbi, aggettivi, avverbi, articoli, congiunzioni, pronomi, punteggiatura, e abbreviazioni sia nella forma base, che nelle possibili forme coniugate (maschile, femminile, plurale, singolare, coniugazione nei tempi verbali …). Per questo motivo è stato utilizzato nella creazione di uno strumento linguistico con funzionalità di POS tagger; in particolare, è stato utilizzato per identificare i sostantivi presenti all'interno delle definizioni.

A questo scopo, è stato inserito l'intero database all'interno del DBMS MySQL; la scelta di MySQL in alternativa ad altri DBMS, è dovuta al fatto che per il funzionamento di STRIDER, era già necessaria la sua installazione, in quanto

aveva lo scopo di memorizzare alcuni dati nel corso dell'esecuzione. È stata quindi mantenuta la versione 5.1.

L'inserimento di Morph-it! è possibile utilizzando il codice SQL presente nell'Appendice C; esso contiene le istruzioni che uniformano i tag relativi ai sostantivi, per agevolare la ricerca, e la creazione di un indice, per velocizzare l'accesso ai dati. Nell'Appendice C è anche presente il nuovo tag set.

A questo punto, le operazioni da effettuare per utilizzare lo strumento appena creato sono delle semplici query SQL, che controllano se il termine introdotto, lemma, è un nome:

```
SELECT DISTINCT lemma
FROM morphit
WHERE tipo = 'n';
```

Per fare queste interrogazioni dal programma scritto in Java, è stata utilizzata la libreria jdbc ufficiale, relativa a MySQL 5.1, "MySQL Connector/J".

Alcuni problemi possono sorgere quando una parola, che in realtà è un nome, non viene identificato come tale, perché non presente all'interno del database. Il problema si fa più serio, se per una definizione non viene estratto nessun nome. A questo scopo, è stata implementata una soluzione, che consiste nell'utilizzare il dizionario on-line come POS tagger. Infatti il programma creato applica il metodo *ricerca()*, già accennato nel paragrafo 3.5, ad ogni termine presente nella definizione e inserisce nella struttura dati finale, tutti quelli che risultano essere sostantivi nel dizionario on-line. Questa rimane una soluzione precaria, in quanto, per definizione, un vocabolario non contiene i termini nelle forme coniugate, e quindi non può distinguere un nome al plurale.

In ogni caso, è stato notato che l'unione dei due metodi fornisce una buona soluzione al problema, identificando la maggioranza dei sostantivi all'interno della frase, e arricchendo a sufficienza la struttura dati finale.

3.7 Integrazione, parametri e modifiche all'interfaccia grafica

A seguito della creazione dello strumento linguistico e delle classi Java, è stata necessaria l'integrazione del prodotto finito in STRIDER. Principalmente è stato necessario adattare parte del codice, a strutture dati e classi che non contenevano gli ipernimi di WordNet. Tutto questo è stato fatto inserendo dei controlli che verificano la lingua selezionata, creando sequenze di codice diverse per l'inglese e l'italiano, ed includendo nel secondo caso le nuove classi.

Come già accennato precedentemente, sono state apportate delle modifiche al pannello di configurazione, inserendo un campo per la lingua, "language", e uno per il tipo di dizionario, "dizionarioIta", come mostrato in Figura 25.

name	value	description
alfa	0.4	Peso del graph context. alfa + beta + gamma = 1; posso interagire solo su alfa e beta.
aTermsInContext	true	termini aggiuntivi del nodo stesso. es: desktopComponents -> term: desktop, aTerm: compon...
autoFeedDelta	0.01	soglia di stabilizzazione (auto feedback metodo 3)
autoFeedN	3	primi N sense da tenere attivi nell'autofeedback
autoFeedSel	3	1: 1 ciclo, tiene i primi autoFeedN sensi di ogni termine 2: a ogni ciclo toglie l'ultimo, si fe...
beta	0.4	Peso del expanded context (hidden context). alfa + beta + gamma = 1; posso interagire solo...
contextWindow	300	siglia di peso per un percorso all'interno della matrice. Il peso è sempre dell'ordine delle centinaia
directAllowed	true	direzione di navigazione permessa: direct
dizionarioIta	1	1: Dizionario De Mauro Paravia 2: Dizionario Sabatini Coletti 3: Dizionario automatico di Google
examples	true	espansione di contesto: uso gli esempi (se ci sono). es: examples=true
expandContext	false	mi dice se posso usare il contesto espanso
explanation	true	espansione di contesto: uso la spiegazione explanation=true
googleFunctSel	1	selezione funzione di correlazione 1: e^PMI 2: PMI
googleSimsInExp	false	true: usa frequenze google false: usa ipernimi wordnet
googleSimsInGraph	false	false: funzione di similarità originaria true: funzione di similarità google
kATerms	10	coefficiente termini aggiuntivi (aTerms)
language	ita	ita: lingua italiana eng: lingua inglese
lpSolve	true	abilita l'algoritmo del simplesso per ottenere i migliori alfa e beta per l'esecuzione
oppositeAllowed	true	mi dice se è permesso usare il verso di percorrenza opposto a quello specificato nel grafo
pathToBrowser	"C:/Pr...	path assoluto al browser da richiamare per la visualizzazione dei risultati

Figura 25. Screenshot della parte del pannello di configurazione con i nuovi parametri "dizionarioIta" e "language".

Sempre riguardo l'interfaccia grafica, quella originale di STRIDER conteneva, nella terzultima colonna sulla destra, le informazioni sulle gerarchie di ipernimi riguardanti il termine selezionato, come mostrato nello screenshot in Figura 26. Per la versione italiana, è stato scelto di inserire nella stessa colonna, gli esempi forniti dal dizionario, come mostrato in Figura 27.

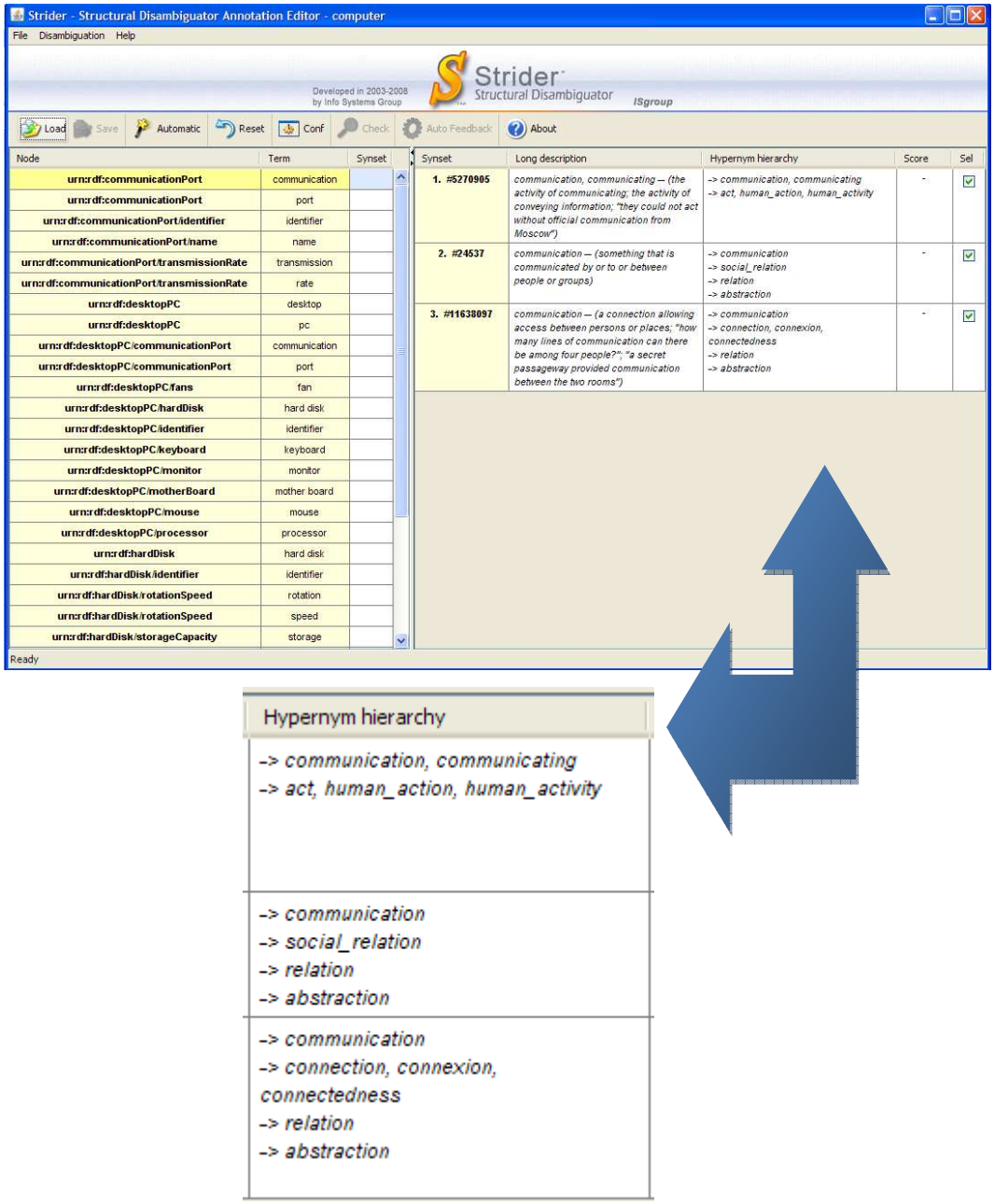


Figura 26. GUI originale di STRIDER. Particolare sulla terzultima colonna di destra, contenente le gerarchie di ipernimi.

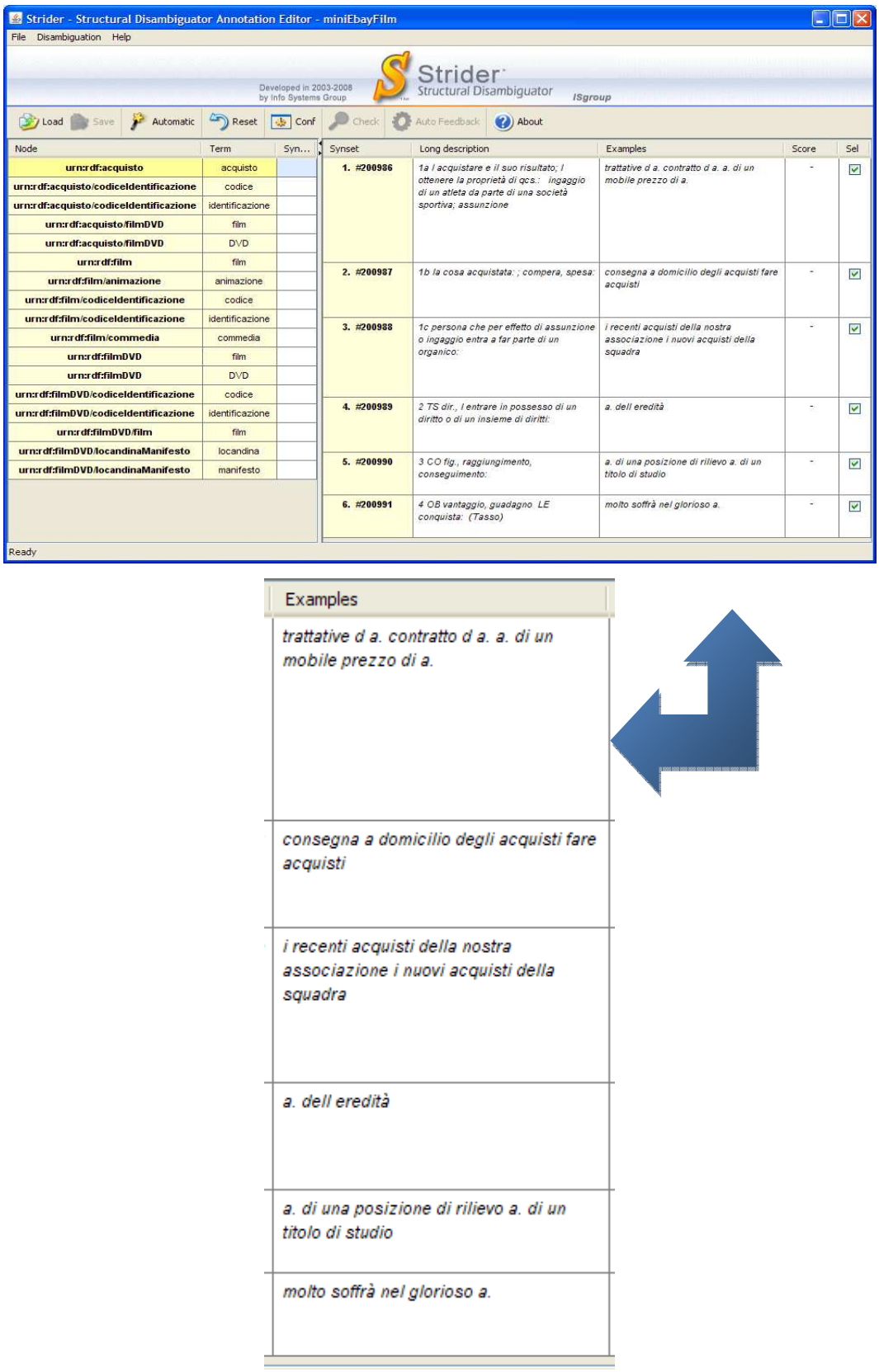


Figura 27. GUI per la lingua italiana, differisce dall'originale per la colonna contenente le gerarchie di ipernimi, sostituita con gli esempi delle definizioni.

Per poter far funzionare STRIDER con schemi in lingua italiana, è necessario modificare i parametri di configurazioni, in modo tale che vengano spenti tutti gli algoritmi e le funzioni che richiedono necessariamente un dizionario strutturato con ipernimi. Questo significa che diventa obbligatorio spegnere totalmente il contributo del contesto del grafo e utilizzare l'Algoritmo 2 per il contesto nascosto, descritto nel paragrafo 1.5.3, che si basa, per il calcolo della similitudine, su una versione esponenziale del PMI (paragrafo 1.5.1); relativamente al contributo della frequenza d'uso, è possibile utilizzare soltanto il primo metodo proposto nel paragrafo 1.5.4. Il nuovo schema relativo ai contributi è presente in Figura 28.

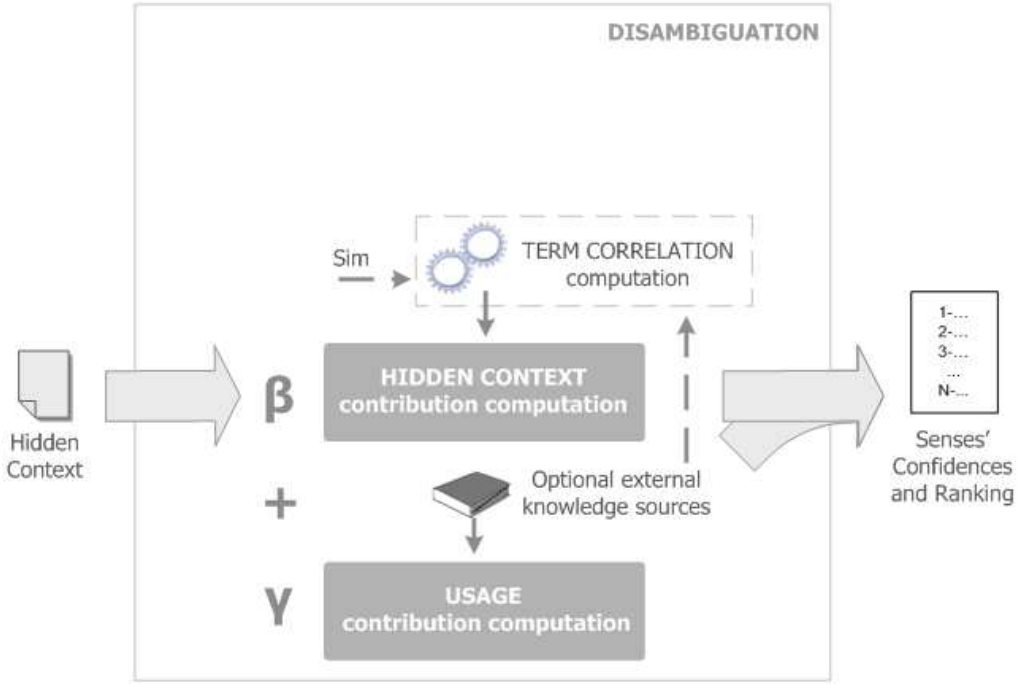


Figura 28. Contribuiti di disambiguazione per gli schemi in lingua italiana.

Dal punto di vista pratico sono da modificare i seguenti parametri nel pannello di configurazione, come in Figura 29:

- **alfa**, peso del graph context, uguale a **0.0**;
- **beta**, peso del hidden context, uguale a **0.7**;

- **dizionarioIta**, a scelta tra **1** ("De Mauro Paravia"), **2** ("il Sabatini Coletti") o **3** (di Google™);
- **examples**, **true** o **false**, a seconda che si voglia o no includere gli esempi nelle definizioni;
- **expandContext**, uguale a **true**, indicando di utilizzare il contesto nascosto;
- **googleFuncSel**, uguale a **1**, indica che si sceglie la versione esponenziale del PMI nel calcolo della similitudine tra termini;
- **googleSimsInExp**, uguale a **true**, indicando di utilizzare le frequenze di Google™;
- **googleSimsInGraph**, uguale a **true**, indica che si utilizza la funzione di similarità di Google™;
- **language**, uguale a **ita**;
- **IpSolve**, uguale a **false**, disattivando l'algoritmo che migliora i valori di alfa e beta;
- **resnikExp**, uguale a **false**, indica di utilizzare il secondo algoritmo nel calcolo del contesto nascosto;
- **wnFreqs**, uguale a **false**, indicando di utilizzare la decay lineare classica.

Gli altri parametri si possono variare a piacere, senza compromettere il funzionamento dell'applicazione nella nuova lingua.
Un'ultima modifica apportata, è stata la creazione e l'integrazione di una nuova classe che permettesse di salvare in cache le definizioni ricercate nella rete. Come già effettuato precedentemente per memorizzare i valori delle frequenze dei motori di ricerca, quest'aggiunta ha lo scopo di ottimizzare il tempo di esecuzione di uno stesso schema, dal suo secondo ciclo in poi. La classe creata è **MemorizeWord**, che prende spunto dalla già esistente "WNFreqCounts", e contiene i seguenti metodi:

- loadWord(), carica dal disco le parole salvate precedentemente, se presenti, altrimenti crea una nuova HashMap;
- saveWord(), salva su disco le parole correnti;
- getWord(), recupera la parola passata come parametro, nel caso in cui sia presente restituisce il suo oggetto dizionario, altrimenti null;
- putWord(), salva l'oggetto dizionario relativamente alla parola passata come parametro;
- encodeString(), codifica la stringa per essere inserita nella URL;
- encodeHtmlString(), codifica la stringa sostituendo i caratteri HTML.

name	value
alfa	0.0
aTermsInContext	true
autoFeedDelta	0.01
autoFeedN	3
autoFeedSel	3
beta	0.7
contextWindow	300
directAllowed	true
dizionarioIta	1
examples	true
expandContext	true
explanation	false
googleFunctSel	1
googleSimsInExp	true
googleSimsInGraph	true
kATerms	10
language	ita
lpSolve	false
oppositeAllowed	true
pathToBrowser	"C:/Programmi/Mozilla Firefox/firefox.exe"
resnikExp	false
rho	0.8
wnFreqs	false

Figura 29. Parametri del pannello di configurazione per l'esecuzione con schemi in italiano.

Capitolo 4

Prove sperimentali

In questo capitolo verranno descritte le prove sperimentali che sono state fatte per testare la nuova versione di STRIDER. Nei prossimi paragrafi si susseguiranno una descrizione dello *experimental setting*, relativamente ai dati e alle tecniche utilizzate, i risultati ottenuti, mostrati attraverso degli istogrammi e una loro interpretazione.

4.1 Experimental setting

Per testare la versione italiana di STRIDER, sono stati effettuati vari test, su schemi con diverse caratteristiche, che rappresentavano molteplici scenari; in questo *experimental setting*, verranno presentati gli schemi utilizzati e alcuni dei risultati più significativi.
La Tabella 2 mostra le caratteristiche degli schemi impiegati, che sono visibili interamente nell'Appendice D; come accennato precedentemente, sono tutti in formato SQL, e in più riguardano argomenti molto distanti l'uno dall'altro. La legenda relativa ai dizionari è la seguente:

- 1 : "De Mauro Paravia";

- 2 : “il Sabatini Coletti”;
- 3 : Google™.

Nome dello schema	Dizionario	Numero di		Numero di sensi	
		Termini	Nodi	Medio	Max
cinema	1	26	18	3,64	8
	2			4,00	20
ebay	1	50	32	4,80	12
	2			3,10	6
miniEbayAbbigliamento	1	20	13	4,65	12
	2			3,40	6
	3			3,95	15
miniEbayFilm	1	17	11	4,94	8
	2			2,88	6
	3			2,88	6
miniEbayMusica	1	19	12	5,11	11
	2			3,16	6
ordine	1	33	23	10,42	37
	2			4,79	12
università	1	31	23	7,19	23
	2			5,06	12
vacanza	1	41	33	5,49	15
	2			3,54	12

Tabella 4. Caratteristiche degli schemi utilizzati. Per la colonna dizionario, la legenda è la seguente: 1 “De Mauro Paravia”, 2 “il Sabatini Coletti”, 3 di Google™.

Gli schemi sono divisibili in due categorie principali, quelli a bassa specificità e di grandi dimensioni, come *ebay*, e quelli ad alta specificità e di medie dimensioni, come *cinema*, la serie *miniEbay*, *ordine*, *università* e *vacanza*.

Nel primo gruppo troviamo quel genere di schema che tratta di argomenti eterogenei, come appunto il catalogo delle categorie di ebay, che sono state riportate in piccola parte.

Il secondo gruppo contiene degli schemi di dimensione media, che trattano nello specifico di un argomento, quale il cinema, gli ordini commerciali, l'università e l'organizzazione di viaggi. La serie *miniEbay* è costituita da tre schemi che dividono quello del gruppo precedente; sono stati utili per fare test di confronto tra le due categorie e per utilizzare il terzo dizionario.

La Tabella 4 mostra, inoltre, delle informazioni relative al numero di termini e nodi presenti nello schema, e il numero di sensi medio e massimo, relativamente ad ogni dizionario.

Negli esperimenti effettuati, sono state valutate le performance del sistema e dei suoi algoritmi in base all'efficacia. L'efficienza, invece, non è stata considerata, in quanto non è mai stata negli obbiettivi di STRIDER.

Allo scopo di valutare la precisione dell'approccio utilizzato, è stato annotato manualmente ogni schema, salvando l'identificatore della definizione corrette per ogni termine, e successivamente sono stati confrontati questi risultati con quelli prodotti automaticamente dall'applicazione. In generale, la *precision* P, è il numero di termini correttamente disambiguati, diviso il numero totale dei termini presenti nello schema; nel nostro caso è stata valutata relativamente al numero di termini corretti entro la M-esima posizione, calcolando così P(M). Per esempio P(1) calcola la *precision* inserendo come corretti, solo i termini che possiedo il senso esatto nella prima posizione, P(2) nella prima o nella seconda, P(3), nella prima, nella seconda o nella terza, e così via.

È da notare, che in tutti gli schemi utilizzati, è possibile arrivare ad un valore di *precision* pari ad 1, disambiguando correttamente tutte le parole. Questo significa che per ogni termine, è stata trovata una definizione corretta. Il problema insorto nell'utilizzo del dizionario 3, cioè quello machine-generated di Google™, è relativo all'impossibilità di identificare sempre una definizione corretta per ogni termine nel suo contesto. In più non è possibile separare gli esempi dalla definizione, nelle rare volte in cui sono presenti. Per questi motivi è stato

utilizzato solo negli schemi in cui si poteva assegnare la definizione corretta ad ogni termine, perché altrimenti non sarebbe stato possibile valutare correttamente la sua efficacia.
Non sono stati riportati i singoli contributi degli algoritmi, in quanto non era rilevante valutarne le singole prestazioni, potendo utilizzare soltanto il contributo nascosto del grafo e quello della frequenza d'uso.

4.2 Prove sperimentali

Nei prossimi paragrafi, verranno descritte le prove sperimentali effettuate, riportando i risultati in istogrammi. Le tecniche seguite, comprendono la valutazione dell'efficacia variando il dizionario utilizzato, il tipo di schema, il contesto e valutando l'utilità di includere gli esempi. Tutti gli schemi, come detto precedentemente, sono presenti nell'Appendice D; in essa viene anche descritto il contesto standard al quale ci si riferirà nei test. In breve, $\alpha = 0$, $\beta = 0{,}7$ e $\gamma = 0{,}3$.

4.2.1 Confronto tra il dizionario 1 e il dizionario 2

La prima prova sperimentale è stata eseguita per valutare quale tra i due dizionari on-line è più adatto all'impiego in STRIDER, utilizzando il contesto standard. Il Grafico A, mostra i diversi valori di *precision*, valutati sullo schema *ordine*, utilizzando il dizionario 1, cioè il "De Mauro Paravia", e il dizionario 2, "il Sabatini Coletti", includendo gli esempi nelle definizioni.
Risalta facilmente che il dizionario 2 ha una netta supremazia nei confronti dell'1. Questo dato di fatto è comprensibile se si osserva la ricchezza delle definizioni; infatti, quello che risulta più adatto, possiede una completezza maggiore e un numero superiore di termini, che in riferimento al meccanismo degli algoritmi di disambiguazione risulta fondamentale.

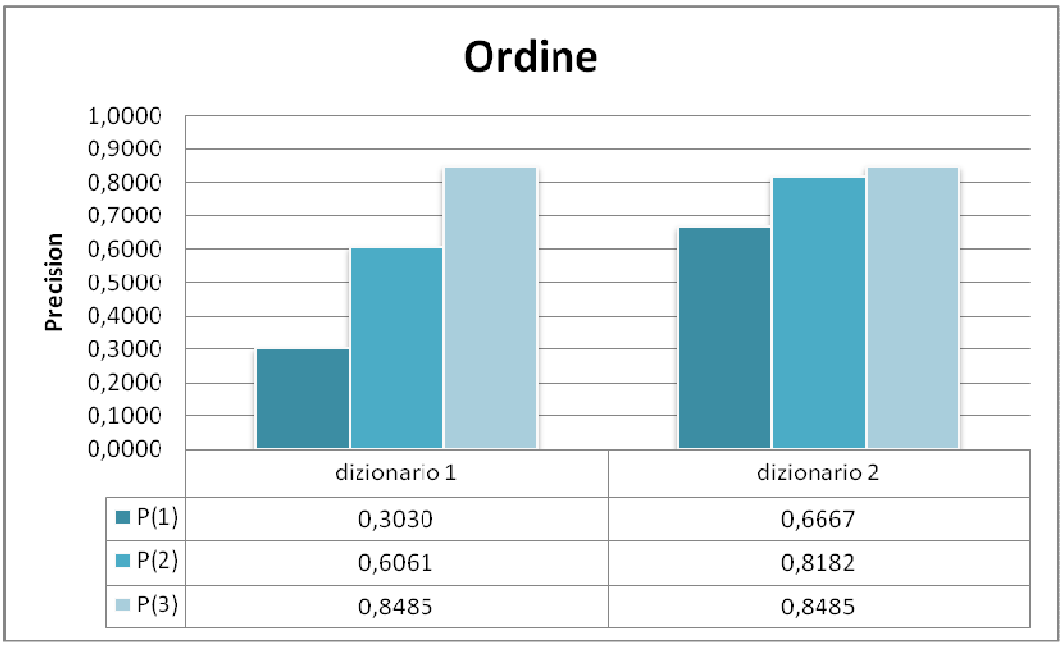
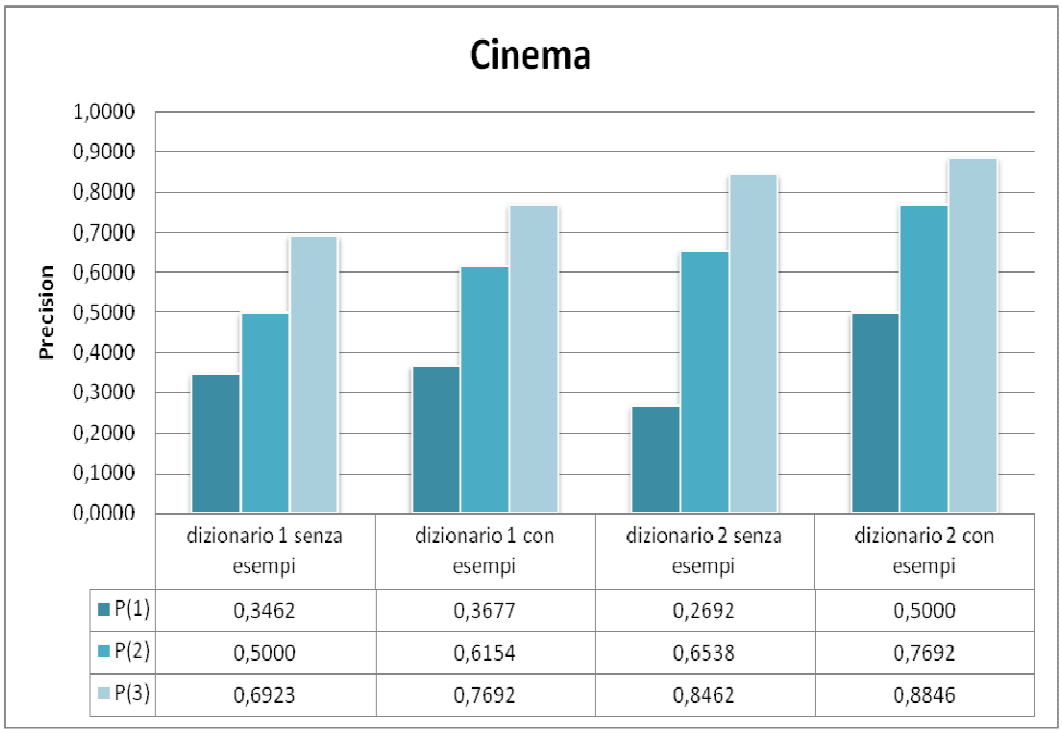


Grafico A. Confronto tra il dizionario 1 e 2 sullo schema *ordine*, utilizzando gli esempi nelle definizioni.

La netta supremazia si nota maggiormente nei valori bassi della *precision*, cioè per P(1) e P(2), che, come espresso nei grafico, valgono 0,3030 e 0,6061 per il dizionario "De Mauro Paravia" e, 0,6667 e 0,8182 per "il Sabatini Coletti". Il livello di P(3) che si raggiunge egualmente in entrambi i dizionari, cioè 0,8485, risulta essere molto soddisfacente.

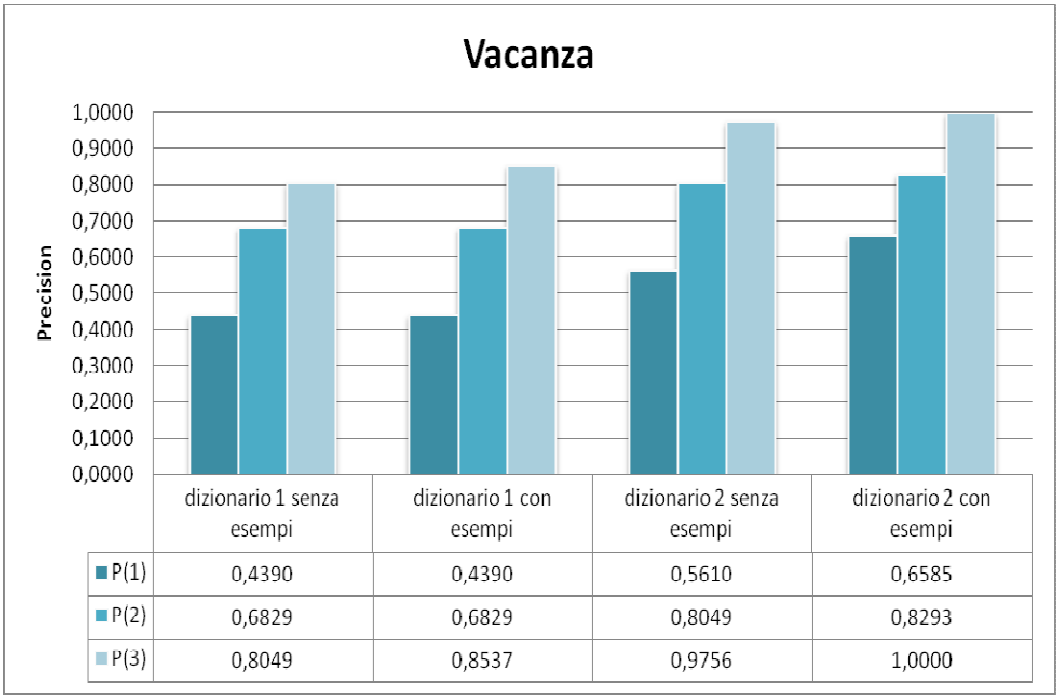
4.2.2 Valutazione del contributo degli esempi nei dizionari 1 e 2

Per valutare l'importanza degli esempi nelle definizioni, è stato analizzato il comportamento dell'applicazione facendo eseguire vari schemi, con dizionari diversi, con e senza esempi, nel contesto standard. I due casi proposti nel Grafico B e nel Grafico C, sono relativi agli schemi *cinema* e *vacanza*.



	dizionario 1 senza esempi	dizionario 1 con esempi	dizionario 2 senza esempi	dizionario 2 con esempi
P(1)	0,3462	0,3677	0,2692	0,5000
P(2)	0,5000	0,6154	0,6538	0,7692
P(3)	0,6923	0,7692	0,8462	0,8846

Grafico B. Confronto tra i valori di *precision* dei dizionari 1 e 2 valutati sullo schema *cinema*, con e senza esempi.



	dizionario 1 senza esempi	dizionario 1 con esempi	dizionario 2 senza esempi	dizionario 2 con esempi
P(1)	0,4390	0,4390	0,5610	0,6585
P(2)	0,6829	0,6829	0,8049	0,8293
P(3)	0,8049	0,8537	0,9756	1,0000

Grafico C. Confronto tra i valori di *precision* dei dizionari 1 e 2 valutati sullo schema *vacanza*, con e senza esempi.

La prima cosa da notare è come si comporta il singolo dizionario con o senza esempi. Considerando il numero 1, il “De Mauro Paravia”, si nota come la presenza degli esempi, nella maggioranza dei casi, migliori la *precision*: infatti nello schema *cinema*, la P(1) passa da 0,3462 a 0,3677, la P(2) da 0,5000 a 0,6154 e la P(3) da 0,6923 a 0,7692. Nello schema *vacanza*, invece, si nota come la presenza degli esempi possa non modificare sul risultato: infatti P(1) non varia da 0,4390 e P(2) da 0,6829. È da notare che la presenza degli esempi può migliorare o non modificare il valore di *precision*, ma generalmente non peggiora mai il suo risultato.

Diversamente dal primo, il dizionario 2, “il Sabatini Coletti”, dimostra di trarre molto vantaggio dall’uso degli esempi. Infatti nello schema *cinema*, tutti e tre i valori di *precision* subiscono un sostanziale aumento con l’inserimento degli esempi: P(1) passa da 0,2692 a 0,5000, P(3) da 0,6538 a 0,7692 e P(3) da 0,8462 a 0,8846. Lo stesso accade nello schema *vacanza*: P(1) passa da 0,5610 a 0,6585, P(2) da 0,8049 a 0,8293 e P(3) da 0,9756 a 1,000. In quest’ultimo si nota come l’introduzione degli esempi consenta, nel caso P(3), di disambiguare correttamente tutti i termini, ottenendo un risultato eccellente.

Si può quindi concludere che l’utilizzo degli esempi può determinare, nella maggioranza dei casi, un incremento di efficacia. In più è da notare che, anche in questo caso, il dizionario 2 ha un comportamento nettamente migliore del numero 1, e per queste ragioni verrà preso come punto di riferimento nell’esposizione dei prossimi test.

4.2.3 Confronto tra il contesto standard e il contesto completo

In questo paragrafo vengono confrontati e valutati i risultati ottenuti testando l’applicazione sullo schema ebay, utilizzando prima in contesto standard e poi quello completo; il dizionario usato è il “De Mauro Paravia” con esempi.

Il contesto generico, è caratterizzato da parametri che pesano l’attraversamento di un arco del grafo in modo differente a seconda che esso sia interno, che colleghi

due grafi (la foreign key degli schemi SQL) o dal verso in cui è percorso, diretto o opposto. Alcune prove sperimentali sono state incentrate sulla modifica di questi parametri, che hanno lo scopo di navigare il grafo in modo diverso, restringendo o allargando il contesto di ciascun termine, dando maggior peso ad un arco rispetto ad un altro.

Ad esempio, nel caso degli schemi del primo gruppo, quelli che trattano di argomenti eterogenei, si può facilmente immaginare che restringendo il contesto, vengano inclusi con maggiore probabilità solo i termini appartenenti ad uno stesso argomento, e quindi la disambiguazione riesca meglio. Al contrario, se si aumentano i parametri in modo tale che il contesto di un termine comprenda tutte le parole del grafo, si ipotizza che i risultati degli algoritmi di disambiguazioni siano peggiori. Questo ragionamento si può riscontrare nel Grafico D, relativo allo schema *ebay*.

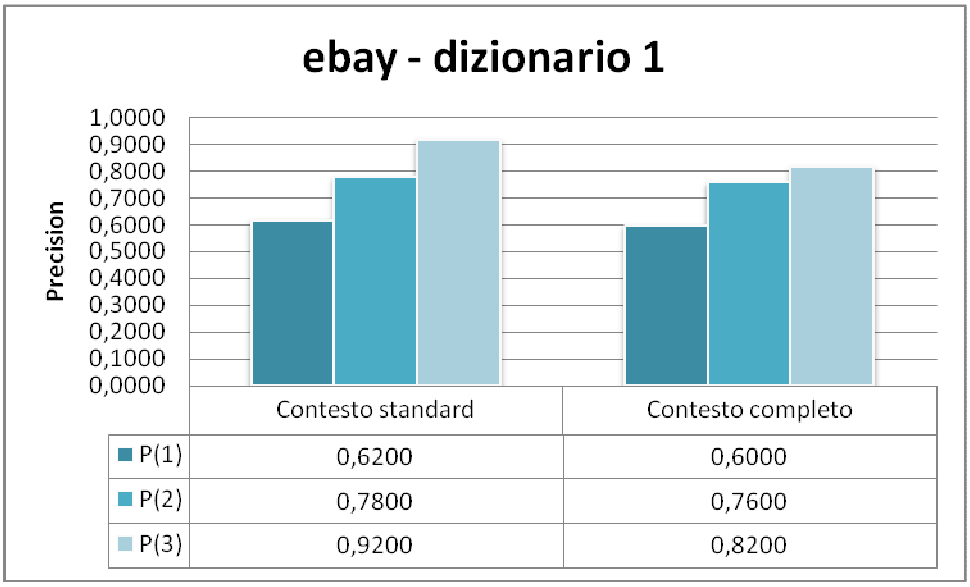


Grafico D. Confronto dei risultati per lo schema *ebay* valutato sul dizionario 1, variando il contesto.

Infatti si nota che allargando il contesto dei termini, cioè rendendolo completo, otteniamo dei risultati peggiori rispetto a quelli forniti dal contesto standard, che inserisce nella maggioranza dei casi, solo le parole relative ad uno stesso

argomento. La *precision* diminuisce nel seguente modo: P(1) cala de 0,6200 a 0,6000, P(2) da 0,7800 a 0,7600 e P(3) da 0,9200 a 0,8200.

Al contrario si comportano gli schemi del secondo gruppo, quelli che sono per definizione monotematici. Infatti, in questo caso, se si restringe il contesto di un termine che appartiene ad un grafo che tratta di un solo argomento, vengono eliminati alcuni termini che potrebbero essere fondamentali per la disambiguazione, ottenendo dei risultati peggiori. Al contrario, se si espande il contesto di un termine fino a comprendere tutte le parole del grafo monotematico, si contribuisce all'aumento dell'efficacia degli algoritmi. Questo ragionamento è ben visibile nel Grafico E, relativo allo schema *vacanza*, valutato utilizzando il dizionario 1 con gli esempi.

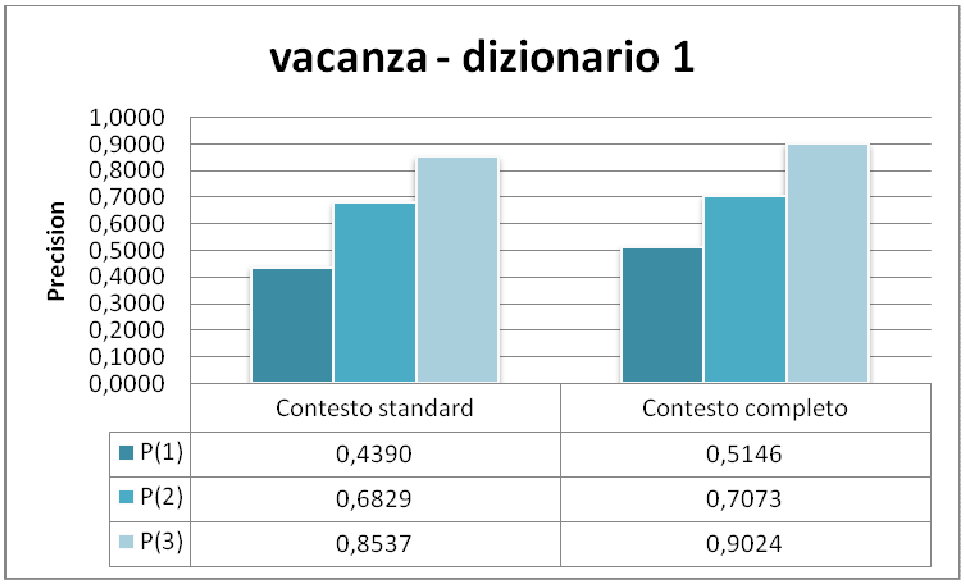


Grafico E. Confronto dei risultati per lo schema *vacanza* valutato sul dizionario 1, variando il contesto.

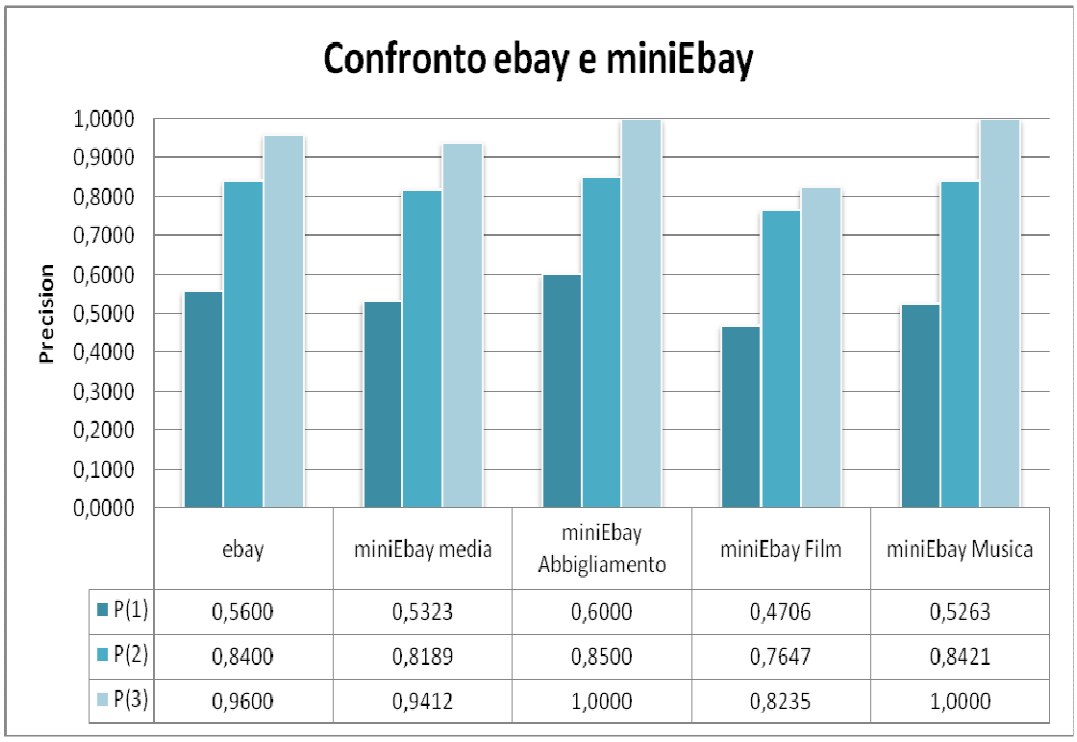
Infatti si nota come questo schema monotematico, ottenga dei risultati migliori se valutato con mediante il contesto completo. I valori di precision migliorano nel seguente modo: P(1) aumenta da 0,4390 a 0,5146, P(2) da 0,6829 a 0,7073 e P(3) da 0,8537 a 0,9024.

Gli stessi ragionamenti che portano a risultati migliori, si possono effettuare mantenendo costante il valore del contesto e variando il costo di attraversamento

degli archi, preferendo quelli intragrafo a quelli intergrafo (quelli delle foreign key SQL), per gli schemi del primo gruppo e l'opposto per quelli del secondo.

4.2.4 Confronto tra *ebay* e *miniEbay*

Un altro esempio significativo in ambito di dimensione del contesto dei termini è quello esposto nel Grafico F. In questo caso vengono testati gli schemi *ebay*, *miniEbayAbbigliemento*, *miniEbayFilm* e *miniEbaMusica* utilizzando il contesto standard e il dizionario 2, "il Sabatini Coletti", con esempi. Siamo qui a dimostrate, come accennato nel paragrafo precedente, che lo schema *ebay*, abbinato al contesto standard, suddivide in modo equilibrato gli argomenti differenti che lo compongono, fornendo dei buoni risultati di disambiguazione.



	ebay	miniEbay media	miniEbay Abbigliamento	miniEbay Film	miniEbay Musica
P(1)	0,5600	0,5323	0,6000	0,4706	0,5263
P(2)	0,8400	0,8189	0,8500	0,7647	0,8421
P(3)	0,9600	0,9412	1,0000	0,8235	1,0000

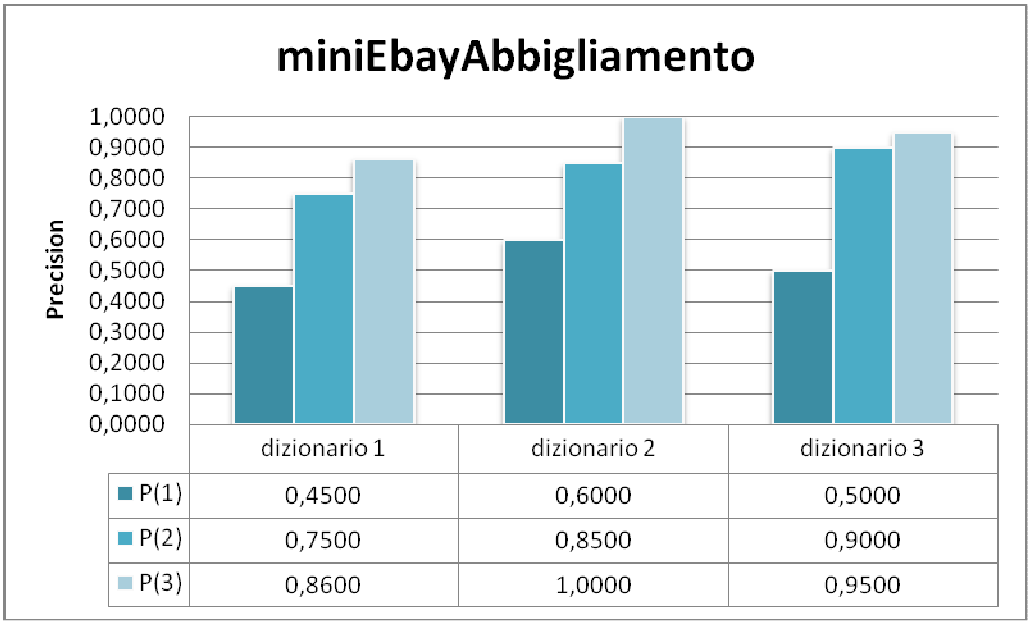
Grafico F. Confronto dei risultati ottenuti con il dizionario 2 per lo schema *ebay*, miniEbayAbbigliamento, miniEbayFilm e miniEbayMusica utilizzando il contesto standard. La seconda parte del grafico rappresenta una media di tutti i valori ottenuti con la serie miniEbay.

Applicando lo stesso contesto ai singoli *miniEbay*, che contengono gli stessi termini di *ebay*, ma già divisi per argomento, otteniamo ugualmente dei buoni risultati, paragonabili e molto vicini a quelli dello schema del primo gruppo.
È da notare come in *miniEbayAbbigliamento* e *miniEabyMusica* si arrivi con la *precision* P(3) alla totalità dei termini disambiguati correttamente.
Tutti questi dati confermano pienamente il fatto che il livello di contesto introdotto per lo schema *ebay*, suddivide equamente ed ottimamente gli ambiti che lo compongono.
Come ultima prova è stata calcolata la media dei risultati dei *miniEbay*, esposta seconda parte del Grafico F, e confrontata con il risultato di *ebay*. I valori sono molto simili, e quest'ultimo è di poco migliore del precedente: infatti P(1) cala da 0,5600 a 0,5323, P(2) da 0,8400 a 0,8189 e P(3) da 0,9600 a 0,9412.
Concludendo, si può affermare che mantenere un contesto basso negli schemi che trattano di argomenti eterogenei è equivalente a disambiguare i singoli schemi monotematici che lo compongono.

4.2.5 Confronto tra dizionario 1, dizionario 2 e dizionario 3

In questo paragrafo verranno confrontati i risultati ottenuti disambiguando gli schemi *miniEbayAbbigliamento* e *miniEbayFilm* con il dizionario 1, "De Mauro Paravia", il dizionario 2, "il Sabatini Coletti", e il dizionario 3, quello di Google™.
Come già accennato nei paragrafi e nei capitoli precedenti, il dizionario 3 ha una struttura e una conformazione molto particolare: infatti le definizioni relative al termine immesso, vengono ricercate e generate automaticamente dal motore di ricerca. Questo fatto comporta vari svantaggi, tra cui l'impossibilità di trovare sempre un senso adatto al termine nel particolare contesto. Per questo motivo è stato testato solo su questi due schemi, gli unici proposti nei quali è stato sempre possibile arrivare ad un livello di *precision* pari a 1.

Nel Grafico G vengono presentati i risultati ottenuti disambiguando lo schema *miniEbayAbbigliamento* utilizzando i dizionari 1, 2 e 3 con esempi e nel contesto standard. Come già concluso nel paragrafo 4.2.2, e come ben visibile anche in questo esempio, il dizionario 2 si comporta molto meglio del numero 1, raggiungendo in P(3) il valore 1,000, cioè la totalità di termini disambiguati correttamente. Relativamente al dizionario 3, si può affermare che fornisce degli ottimi risultati, nonostante le premesse poco invitanti, che lo definiscono come un dizionario non compilato: infatti, nella P(1) ottiene un valore intermedio tra quelli del dizionario 1, 0,4500 e quello del 2, 0,6000, cioè 0,5000; nella P(2) li supera entrambi, visto che il dizionario 1 ottiene 0,7500, il 2 0,8500 e il 3 0,9000; nella P(3) si ristabilisce ad un livello intermedio tra il numero 1, 0,8600 e il 2, 1,000, ottenendo 0,9500.



	dizionario 1	dizionario 2	dizionario 3
P(1)	0,4500	0,6000	0,5000
P(2)	0,7500	0,8500	0,9000
P(3)	0,8600	1,0000	0,9500

Grafico G. Confronto dei risultati ottenuti per lo schema *miniEbayAbbigliamento*, valutato sul dizionario 1, 2 e 3 con esempi, nel contesto standard.

Nel Grafico H, invece, vengono esposti i risultati ottenuti disambiguando lo schema *miniEbayFilm*. In questo caso il dizionario 1 raggiunge un valore di P(1) e P(3) superiore a quello del numero 1; a sostegno delle conclusioni tratte in

precedenza, è da notare che non è mai stata data una superiorità assoluta del 2 nei confronti del numero 1, e che un ribaltamento di risultati può sempre verificarsi.

Anche in questo schema è da valutare positivamente il risultato ottenuto dal dizionario 3: infatti, relativamente alla P(1) si posiziona ad un livello intermedio tra il dizionario 2, con 0,4706 e il numero 1, con 0,6471, ottenendo un valore di 0,5000; relativamente alla P(2) supera entrambi ottenendo un punteggio di 0,9000 rispetto allo 0,7059 del numero 1 e 0,7647 del 2; relativamente alla P(3) supera nuovamente ambedue i dizionari, ottenendo un valore di 0,9500, rispetto allo 0,9412 del dizionario 1 e lo 0,8235 del 2.

Concludendo, si può affermare nonostante non siano sempre disponibili le definizioni esatte per ogni termine nel suo contesto, e nonostante la diversa fonte di conoscenza, il dizionario machine-generated di Google™ fornisce degli ottimi risultati negli schemi in cui è applicabile.

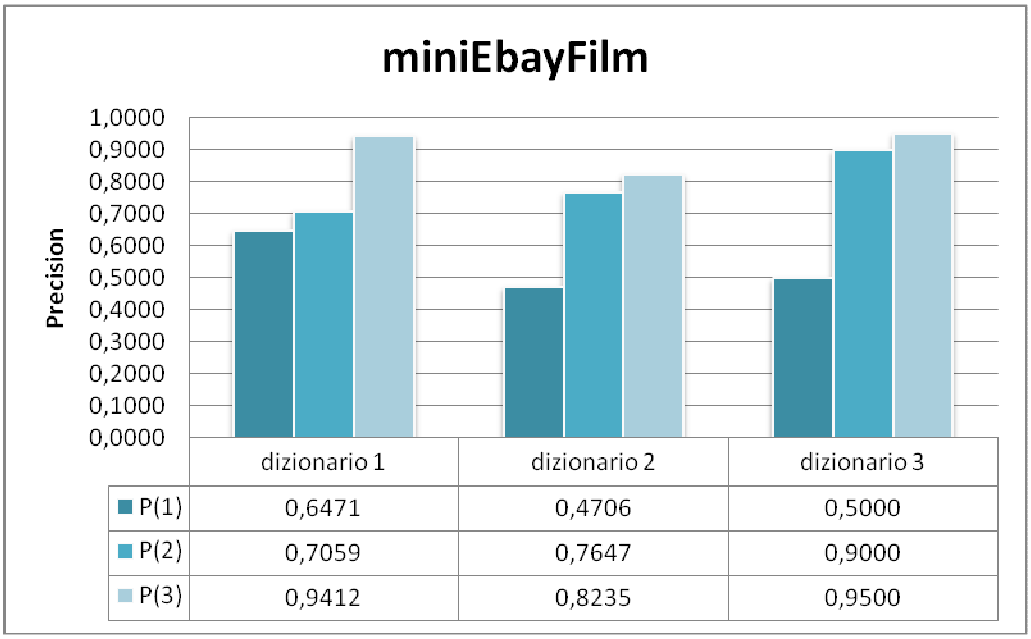


Grafico H. Confronto dei risultati ottenuti per lo schema *miniEbayFilm*, valutato sul dizionario 1, 2 e 3 con esempi, nel contesto standard.

Capitolo 5

Conclusioni e sviluppi futuri

Nel corso dell'analisi delle fonti di conoscenza esterne e degli strumenti linguistici, dell'implementazione e della stesura della tesi sono stati raggiunti i seguenti obiettivi:

- sono state effettuate diverse ricerche relative ai dizionari e ai POS tagger per la lingua italiana disponibili sulla rete; successivamente sono stati analizzati e valutati secondo le loro caratteristiche, scegliendo tre vocabolari di generi differenti e un database di parole categorizzate come POS tagger;
- è stato possibile creare una sorta di POS tagger partendo dal database Morph-it!, modificandone il *tag set* e inserendolo all'interno del DBMS MySQL;
- è stata implementata l'estensione per l'applicazione STRIDER per la lingua italiana, adattandola all'uso di tre differenti dizionari on-line; è da sottolineare che essa è stata effettuata seguendo i principi della modularità, in modo da permettere altre estensioni analoghe per lingue o dizionari differenti;
- sono state valutate le prestazioni della nuova versione, testando il

comportamento dei tre differenti dizionari on-line, su schemi SQL di molteplici argomenti e dimensioni, variando il contesto dei termini;

- infine è nata ed è stata sviluppata l'idea di adattare STRIDER al multilinguismo, in quanto è possibile mantenere nella stessa versione le implementazioni per lingue differenti; infatti, in questa, è possibile disambiguare schemi sia in lingua italiana che in inglese.

Relativamente agli sviluppi futuri dell'applicazione, sono nate le seguenti idee:

- per prima cosa potrebbe essere interessante selezionare altri dizionari per la lingua inglese che non possiedono la struttura gerarchica di WordNet, in modo da poter testare gli algoritmi utilizzati nella versione italiana, in modo equo con un dizionario per l'inglese;
- sarebbe anche interessante valutare il comportamento dell'applicazione utilizzando un dizionario italiano strutturato ad ipernimi, come ItalWordNet oppure MultiWordNet, potendo testare tutti gli algoritmi che sono implementati in STRIDER;
- sempre per la lingua italiana, come già accennato nel paragrafo 2.4.2, si potrebbero utilizzare i dizionari collaborativi, come il "Wikizionario" o il "Dizionario Italiano", una volta che essi avranno raggiunto un livello di completezza accettabile, in modo da poter paragonare le loro prestazioni con i generici vocabolari compilati o quelli generati automaticamente da un motore di ricerca;
- relativamente al concetto di feedback presentato nel paragrafo 1.6, diventerebbe molto utile sviluppare la stessa funzionalità anche per l'algoritmo che non utilizza la struttura ad ipernimi, cioè il secondo algoritmo per l'estrazione del contesto nascosto. Infatti questa parte non è stata implementata, in quanto richiedeva che il numero di interrogazioni al motore di ricerca, per determinare le frequenze dei termini, aumentasse in maniera esponenziale; per questo motivo sarebbe necessario valutare anche altre soluzioni alternative;

- per ultimo, sarebbe possibile rifare gli stessi ragionamenti effettuati per la lingua italiana, per qualsiasi altra lingua; infatti, il percorso seguito è totalmente universale e facilmente ripercorribile, e l'introduzione di una nuova aumenterebbe il multilinguismo che ci si proponeva di raggiungere.

Parte III

Appendici

Appendice A

Il *crossing set* di default utilizzato nella Parte I

La Tabella 5 mostra il *crossing set* di default che è stato utilizzato negli esempi proposti nella Parte I; esso viene proposto per ogni tipo di formato che STRIDER originariamente può prendere in input. La lista dei prefissi utilizzati è la seguente:

- PREFIX sql: <http://www.oim-converter.com/vocabulary/sqlddl/1.0>
- PREFIX xsd: <http://www.w3.org/2001/XMLSchema>
- PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns/>
- PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema>
- PREFIX owl: <http://www.w3.org/2002/07/owl>

Edge	Direct Direction	Opposite Direction	Applicability		
			relat.	tree	graph
SR	1	0.8	√	√	√
FK	0.1	0.1	√		
owl:differentFrom	1	0.8			√
owl:disjointWith	1	0.8			√
owl:complementOf	1	0.8			√
owl:inverseOf	1	0.8			√
owl:equivalentClass	0.1	0.1			√
owl:equivalentProperty	0.1	0.1			√
owl:someValuesFrom	0.1	0.1			√
rdfs:domain	0.1	0.1			√
rdfs:label	0.1	0.1			√
rdfs:range	0.1	0.1			√
rdfs:subPropertyOf	0.1	0.1			√

Tabella 5. *Crossing set* di default per ogni formato accettato da STRIDER.

Appendice B

Elenco delle possibili “features” di Morph-it!

In questa appendice vengono elencate le possibili “features” che categorizzano ogni termine contenuto all’interno del database Morph-it!. L’elenco è stato estratto dal file readme [25] presente nella documentazione della base di dati.

ABL
Abbreviated locutions, such as "a.C.", "ecc." and "i.e."

ADJ
Adjectives, with the following inflectional features:

pos/comp/sup
Thas is: positive, comparative, superlative. Although these are not true inflectional features, given their high productivity we decided to represent them as properties of inflected forms.

f/m
That is: feminine, masculine.

s/p
Thas is: singular, plural.

ADV
Adverbs.

ART
Articles, with gender as a derivational feature (F/M) and number as an inflectional feature (s/p).

ARTPRE
Preposition+article compounds ("col", "della", "nei"...), with gender as a derivational feature (F/M) and number as an inflectional feature (s/p).

ASP
Aspectuals ("stare" in "stare per"). Same inflectional features as VER (see below).

AUX
Auxiliaries ("essere", "avere", "venire"). Same inflectional features as VER (see below).

CAU
Causatives ("fare" in "far sapere"). Same inflectional features as VER (see below).

CE
Clitic "ce" as in "ce l'ho fatta".

CI
Clitic "ci" as in "ci prova".

CON
Conjunctions.

DET-DEMO
Demonstrative determiners (such as "questa" in "questa sera"), with inflectional gender (f/s) and number (s/p) features.

DET-INDEF
Indefinite determiners (such as "molti" in "molti amici") with inflectional gender (f/s) and number (s/p) features.

DET-NUM-CARD
Cardinal number determiners (e.g., "cinque" in "cinque amici"). Pure-digit numbers are not included (i.e., the list includes "100mila" but not "100000" nor "100,000", "100.000", etc.)

DET-POSS
Possessive determiners (e.g., "mio", "suo"), with inflectional gender (f/s) and number (s/p) features.

DET-WH
Wh determiners (e.g., quale in "quale amico"), with inflectional gender (f/s) and number (s/p) features.

INT
Interjections.

MOD
Modal verbs (e.g. "dover" in "dover ricostruire"). Same inflectional features as VER (see below).

NE
Clitic "ne" (as in: "ne hanno molte").

NOUN
Nouns, with gender as a derivational feature (F/M) and number as an inflectional feature (s/p).

PON
Non-sentential punctuation marks (e.g. , " $).

PRE
Prepositions.

PRO-DEMO
Demonstrative pronouns (e.g. "questa" in "voglio questa"), with both gender and number as derivational features (F/M, S/P).

PRO-INDEF
Indefinite pronouns (e.g., "molti" in "vengono molti"), with both gender and number as derivational features (F/M, S/P).

PRO-NUM
Numeral pronouns (e.g., "cinque" in "cinque sono sopravvissuti"). Pure-digit numbers are not included (e.g., the list includes "100mila" but not 100000 nor 100,000, 100.000, etc.)

PRO-PERS
Personal pronouns, such as "lui" and "loro". Clitic possessive pronouns (such as pronominal "lo" and "si") are marked by the derivational feature CLI. Person, gender and number are also encoded as derivational features (1/2/3, F/M, S/P).

PRO-POSS
Possessive pronouns, such as "loro" in "non era uno dei loro"), with gender and number encoded as derivational features (F/M, S/P).

PRO-WH
Wh-pronouns, such as "quale" in "quale e' venuto?"

SENT
End of sentence marker (! . ... : ?).

SI
Clitic "si" as in "di cui si discute".

TALE
"Tale" in constructions such as "una fortuna tale che...", "la tal cosa", "tali amici", ecc. Gender (f/m) and number (s/p) as inflectional features.

VER
Verbs, with the following inflectional features:

cond/ger/impr/ind/inf/part/sub
Conditional, gerundive, imperative, indicative, infinitive, participle, subjunctive.

pre/past/impf/fut
Present, past, imperfective, future.

1/2/3
Person.

s/p
Number.

f/m
Gender (only relevant for participles).

cela/cele/celi/celo/cene/ci/gli/gliela/gliele/glieli/glielo/gliene/la/le/li/lo/mela/mele/meli/melo/mene/mi/ne/sela/sele/seli/selo/sene/si/tela/tele/teli/telo/tene/ti/vela/vele/veli/velo/vene/vi
Clitics attached to the verb.

WH
Wh elements ("come", "qualora", "quando"...)

WH-CHE
"Che" as a wh element (e.g., "l'uomo che hai visto", "hai detto che").

Appendice C

Codice SQL per la creazione dello strumento linguistico che utilizza Morph-it! e il nuovo tag set

In questa appendice verrà presentato il codice SQL che permette di inserite Morph-it! all'interno di un DBMS e che ne modifica il tag set in modo da essere adatto alle ricerche effettuate da STRIDER.

```
CREATE DATABASE morphit;

USE morphit;

CREATE TABLE `morphit` (
  `parola` varchar(255) NOT NULL default '',
  `lemma` varchar(255) NOT NULL default '',
  `carat` varchar(255) NOT NULL default ''
) TYPE=MyISAM;

LOAD DATA LOCAL INFILE 'c:/morph-it_047/morph-it_047.txt'
INTO TABLE morphit
FIELDS TERMINATED BY '\t';

ALTER TABLE `morphit`.`morphit`
ADD COLUMN `id` INT(11) NOT NULL AUTO_INCREMENT FIRST,
ADD COLUMN `tipo` char(2) NOT NULL DEFAULT '' AFTER `lemma`,
ADD INDEX USING BTREE (`tipo`),
ADD PRIMARY KEY(`id`);

UPDATE morphit
SET tipo= 'n'
WHERE carat LIKE ('NOUN%');

UPDATE morphit
SET tipo= 'v'
```

```
WHERE carat LIKE ('VER%');

UPDATE morphit
SET tipo= 'v'
WHERE carat LIKE ('AUX%');

UPDATE morphit
SET tipo= 'v'
WHERE carat LIKE ('CAU%');

UPDATE morphit
SET tipo= 'v'
WHERE carat LIKE ('MOD%');

UPDATE morphit
SET tipo= 'v'
WHERE carat LIKE ('ASP%');

UPDATE morphit
SET tipo= 'p'
WHERE carat IN ('CI', 'CE', 'NE', 'SI', 'TALE', 'INT');

UPDATE morphit
SET tipo= 'pt'
WHERE carat IN ('PON', 'SENT', 'SMI', 'SYM');

UPDATE morphit
SET tipo= 'ab'
WHERE carat = 'ABL';

UPDATE morphit
SET tipo= 'ag'
WHERE carat LIKE ('ADJ%');

UPDATE morphit
SET tipo= 'ag'
WHERE carat LIKE ('DET%');

UPDATE morphit
SET tipo= 'ar'
WHERE carat = 'ADV';

UPDATE morphit
SET tipo= 'ar'
WHERE carat LIKE ('ART-%');

UPDATE morphit
SET tipo= 'pe'
WHERE carat = 'PRE';
```

```
UPDATE morphit
SET tipo= 'pe'
WHERE carat LIKE ('ARTPRE%');

UPDATE morphit
SET tipo= 'p'
WHERE carat IN ('CI', 'CE', 'NE', 'SI', 'INT');

UPDATE morphit
SET tipo= 'p'
WHERE carat LIKE ('TALE%');

UPDATE morphit
SET tipo= 'co'
WHERE carat = 'CON';

UPDATE morphit
SET tipo= 'po'
WHERE carat LIKE ('PRO%');

UPDATE morphit
SET tipo= 'wh'
WHERE carat LIKE ('WH%');

UPDATE morphit
SET tipo= 'np'
WHERE carat = 'NPR';
```

Ora verrà elencato il nuovo tag set a cui si riferisce lo strumento linguistico appena creato:

n → nome NOUN-M:s + NOUN-M:P + NOUN-F:s + NOUN-F:p
v → verbo VER + AUX + CAU + MOD + ASP
pt → punteggiatura e simboli PON + SENT + SMI + SYM
ab → abbreviazione ABL
ag → aggettivo ADJ + (DET-DEMO, DET-INDEF, DET-NUM-CARD, DET-POSS, DET-WH)
av →avverbio ADV
ar →articolo ART
pe →preposizione PRE + ARTPRE
p →particella CI + CE + NE + SI + TALE + INT
co → congiunzione CON
po → pronome PRO-DEMO + PRO-INDEF + PRO-NUM + PRO-PERS + PRO-POSS + PRO-WH
wh → wh element WH + WH-CHE
np → nome di persona NPR

Appendice D

Schemi SQL e contesto standard

In questa sezione verranno presentati gli schemi SQL che sono stati utilizzati nelle prove sperimentali; successivamente ci sarà una descrizione del contesto standard di riferimento su cui si basano molti test.

Gli schemi utilizzati sono i seguenti:

- cinema, Figura 30;
- ebay, Figura 31;
- miniEbayAbbigliamento, Figura 32;
- miniEbayFilm, Figura 33;
- miniEbayMusica, Figura 34;
- ordine, Figura 35;
- università, Figura 36;
- vacanza, Figura 37.

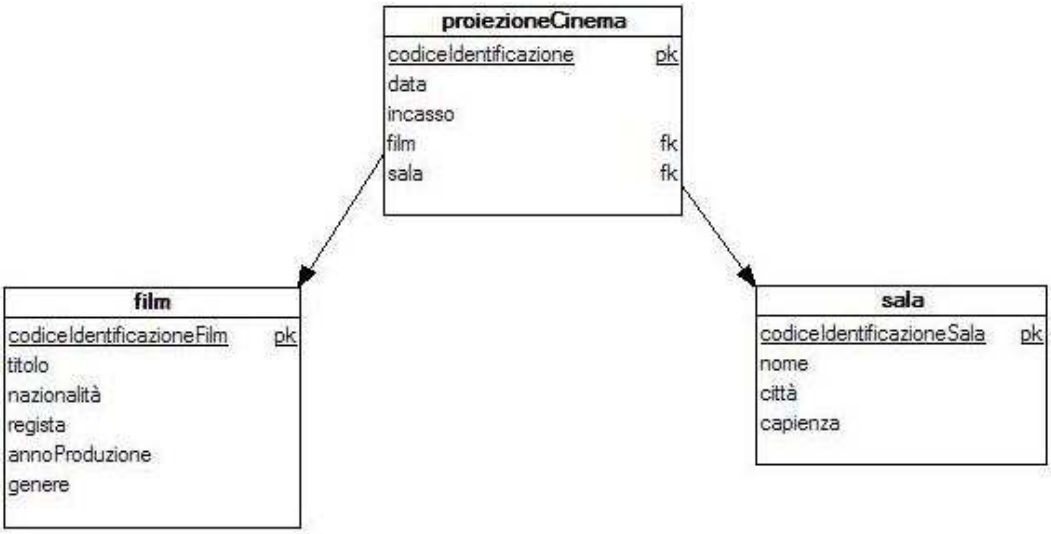


Figura 30. Schema SQL *cinema*.

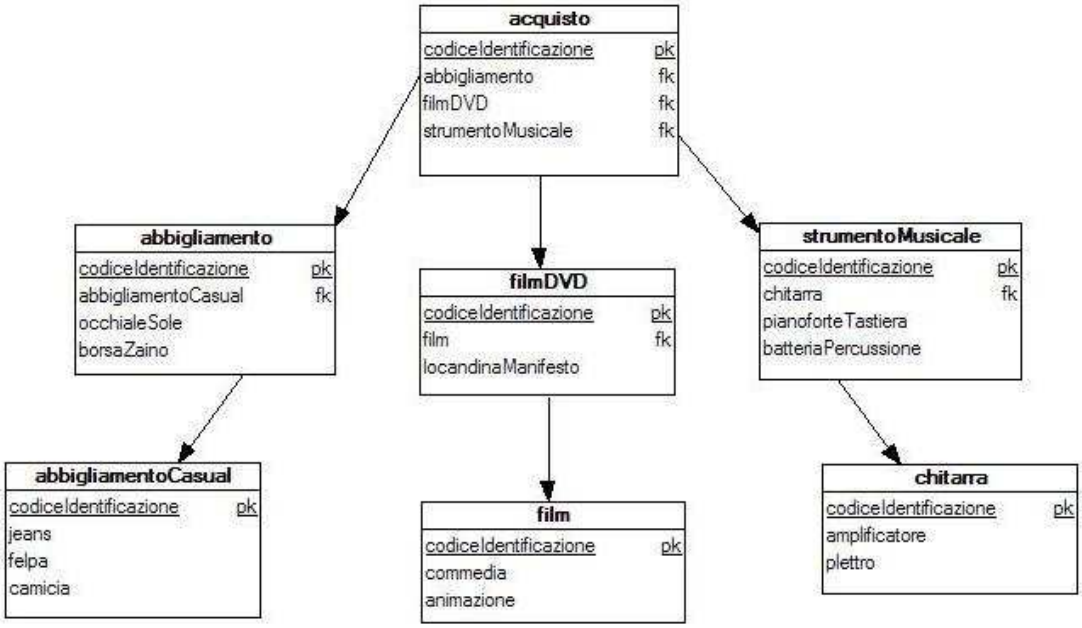


Figura 31. Schema SQL *ebay*.

Figura 32. Schema SQL *miniEbayAbbigliamento*.

Figura 33. Schema SQL *miniEbayFilm*.

Figura 34. Schema SQL *miniEbayMusica*.

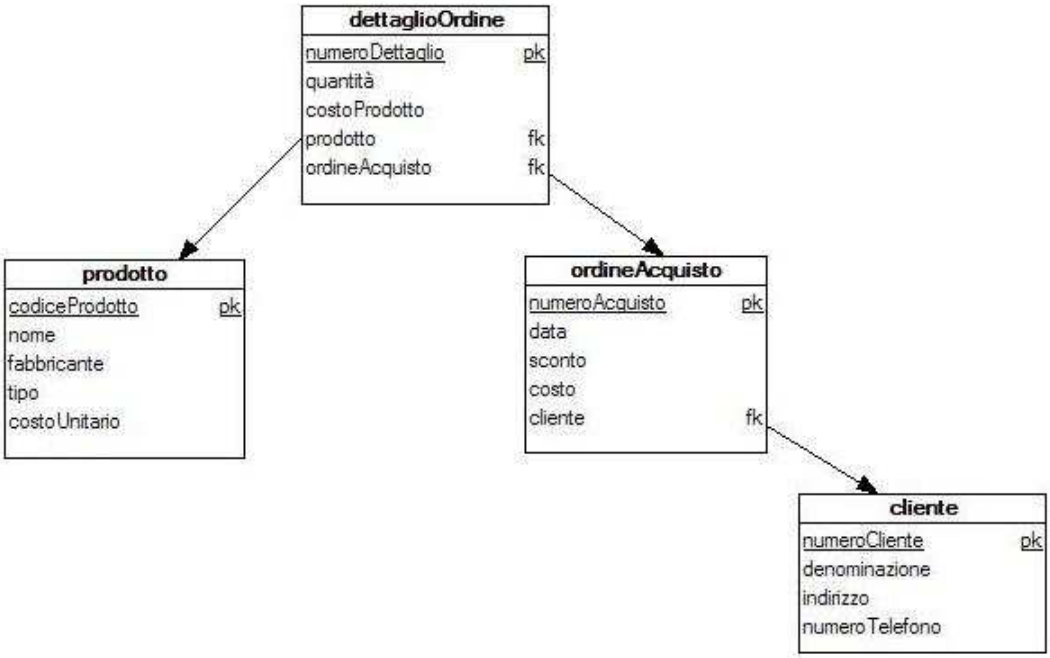


Figura 35. Schema SQL *ordine*.

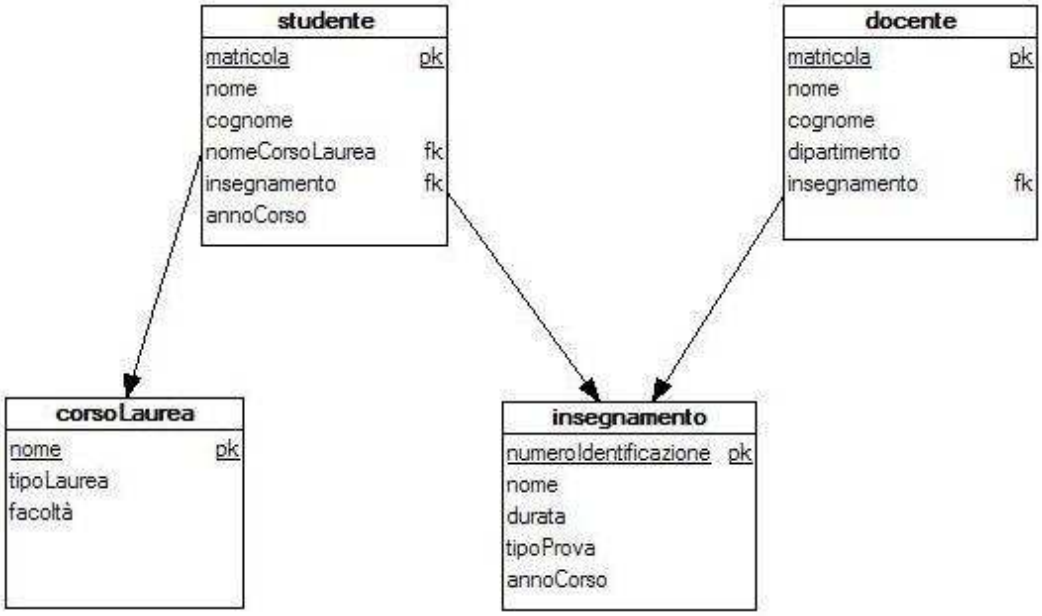


Figura 36. Schema SQL *università*.

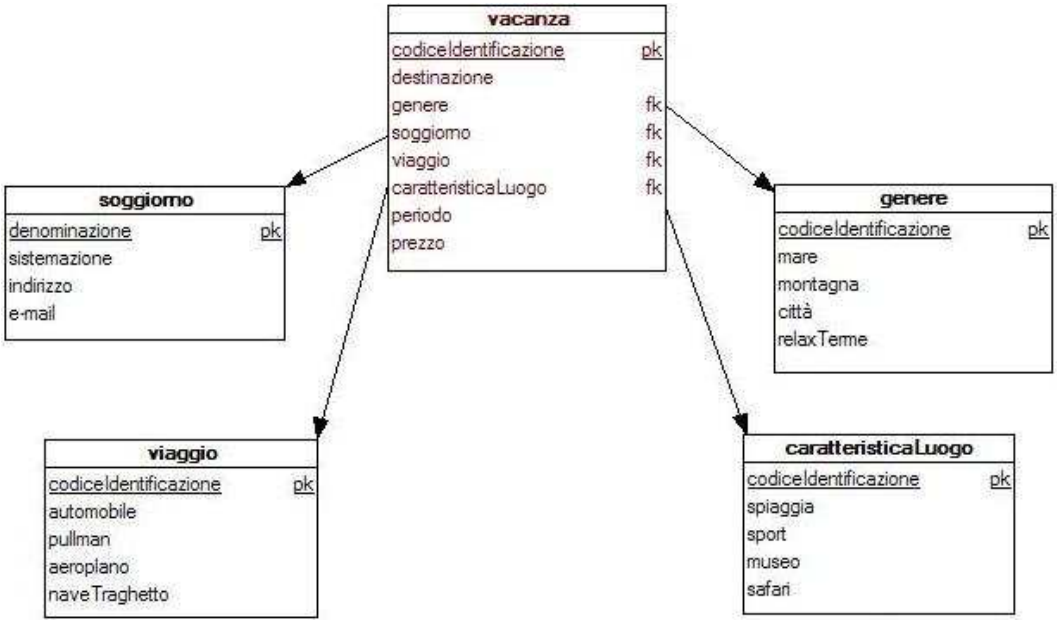


Figura 37. Schema SQL *vacanza*.

Con il termine contesto standard, si intende un insieme di valori assegnati a dei parametri, che vengono utilizzati come punto di riferimento per le prove sperimentali.
Nei test effettuati e spiegati nel capitolo 4, è stato preso di default il seguente contesto standard per gli schemi in formato SQL:

- ContextWindow = 300;
- FK: weight direct = 10;
- FK: weight opposite = 10;
- FK: enabled = true;
- STR: weight direct = 100;
- STR: weight opposite =100;
- STR: enabled = true;
- direct allowed = true;
- opposite allowed = true;
- googleFunctSel = 1.

indicando con FK le foreign keys e STR i collegamenti tra termini di una stessa tabella. Con i termini direct e opposite si intendono l'attraversamento rispettivamente in senso diretto e opposto degli archi del grafo.
Tutti questi parametri sono visibili e modificabile accedendo al pannello delle configurazioni dell'applicazione STRIDER.

Riferimenti e bibliografia

Ackoff, R. L. 1989. From Data to Wisdom. *Journal of Applied Systems Analysis 16*, 3-9.

Amitay, E., Nelken, R., Niblack, W., Sivan, R., and Soffer, A. 2003. Multi-resolution disambiguation of term occurrences. In *Proceedings of the 12th ACM International Conference on Information and Knowledge Management (CIKM'03)*. 255-262.

Artiles, J., Penas, A., and Verdejo, F. 2004. Word Sense Disambiguation based on term to term similarity in a context space. In *Proceedings of the 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (Senseval-3)*. 58-63.

Banerjee, S. and Pedersen, T. 2003. Extended Gloss Overlaps as a Measure of Semantic Relatedness. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*. 805-810.

Bouquet, P., Serafini, L., and Zanobini, S. 2006. Bootstrapping Semantics on the Web: Meaning Elicitation from Schemas. In *Proceedings of the 15th International Conference on World Wide Web (WWW'06)*. 505-512.

Budanitsky, E. and Hirst, G. 2001. Semantic distance in WordNet: An Experimental, Application-oriented Evaluation of Five Measures. In *Proceedings of the Workshop on WordNet and Other Lexical Resources, in the North American Chapter of the Association for Computational Linguistics (NAACL'01)*.

Cilibrasi, R. L. and Vitanyi, P. M. B. 2007. The Google Similarity Distance. *IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE) 19*, 3, 370-383.

Cimiano, P., Handschuh, S., and Staab, S. 2004. Towards the self-annotating web. In *Proceedings of the 13th WWW Conference (WWW'04)*. 462-471.

Ciravegna, F., Chapman, S., Dingli, A., and Wilks, Y. 2004. Learning to harvest information for the semantic web. In *Proceedings of 1st European Semantic Web Symposium (ESWS '04)*. 312-326.

Ciravegna, F., Dingli, A., Petrelli, D., and Wilks, Y. 2002. User-System Cooperation in Document Annotation Based on Information Extraction. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02)*. 122-137.

Do, H., Melnik, S., and Rahm, E. 2002. Comparison of schema matching evaluations. In *Proceedings of the 2nd International Workshop on Web Databases*. 221-237.

Ehrig, M. and Maedche, A. 2003. Ontology-Focused Crawling of Web Documents. In *Proceedings of the 2003 ACM Symposium on Applied Computing (ACM SAC'03)*. 1174-1178.

Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A., Shaked, T., Soderland, S., Weld, D. S., and Yates, A. 2004. Web-scale information extraction in KnowItAll: (preliminary results). In *Proceedings of the 13th International conference on World Wide Web (WWW'04)*. 100-110.

Ide, N. and Veronis, J. 1998. Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art. *Computational Linguistics 24*, 1, 2-40.

Leacock, C. and Chodorow, M. 1998. Combining local context and WordNet similarity for word sense identification. In *WordNet: An electronic lexical database*, C. Fellbaum, Ed. MIT Press, 256-283.

Liu, S., Yu, C., and Meng, W. 2005. Word Sense Disambiguation in Queries. In *Proceedings of 14th ACM International Conference on Information and Knowledge Management (CIKM'05)*. 525-532.

Madhavan, J., Bernstein, P. A., Doan, A., and Halevy, A. Y. 2005. Corpus-based Schema Matching. In *Proceedings of 21st International Conference on Data Engineering (ICDE'05)*. 57-68.

Madhavan, J., Bernstein, P. A., and Rahm, E. 2001. Generic Schema Matching with Cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*. 49-58.

Mandreoli, F., Martoglia, R., and Ronchetti, E. 2005. Versatile Structural Disambiguation for Semantic-aware Applications. In *Proceedings of 14th ACM International Conference on Information and Knowledge Management (CIKM'05)*. 209-216.

Mandreoli, F., Martoglia, R., and Ronchetti, E. 2006. STRIDER: A Versatile System for Structural Disambiguation. In *Proceedings of 10th International Conference on Extending Database Technology (EDBT'06)*. 1194-1197.

Mandreoli, F., Martoglia, R., and Ronchetti. Knowledge-Driven Sense Disambiguation (Almost.) For All Structures. Submitted for publication.

McDowell, L., Etzioni, O., Gribble, S., Halevy, A., Levy, H., Pentney, W., Verma, D., and Vlasseva, S. 2003. Mangrove: Enticing Ordinary People onto the Semantic Web via Instant Gratification. In *Proceedings of the 2nd International Semantic Web Conference (ISWC'03)*. 754-770.

Melnik, S., Garcia-Molina, H., and Rahm, E. 2002. Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*. 117-128.

Melnik, S., Rahm, E., and Bernstein, P. A. 2003. Rondo: A Programming Platform for Generic Model Management. In *Proceedings of the 22nd ACM International Conference on Management Of Data (SIGMOD'03)*. 193-204.

Mihalcea, R., Tarau, P., and Figa, E. 2004. PageRank on Semantic Networks, with Application to Word Sense Disambiguation. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*. 1126-1132.

Miller, G. A. 1995. WordNet: A Lexical Database for English. *Communication of the ACM 38*, 11, 39-41.

Navigli, R. and Velardi, P. 2005. Structural Semantic Interconnections: A Knowledge-Based Approach to Word Sense Disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 7, 1075-1086.

Resnik, P. 1995. Disambiguating Noun Groupings with Respect to WordNet Senses. In *Proceedings of the 3rd Workshop on Very Large Corpora*. 54-68.

Shadbolt, N., Berners-Lee, T., and Hall, W. 2006. The semantic web revisited. *IEEE Intelligent Systems 21*, 3, 96-101.

Shutt, D., Bernstein, P. A., Bergstraesser, T., Carlson, J., Pal, S., and Sanders, P. 1999. Microsoft Repository Version 2 and the Open Information Model. *Information Systems 24*, 2, 71-98.

Strube, M. and Ponzetto, S. P. 2006. WikiRelate! Computing Semantic Relatedness Using Wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06)*.

Svatek, V., Labsky, M., and Vacura, M. 2004. Knowledge Modelling for Deductive Web Mining. In *Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW'04)*. 337-353.

Tagarelli, A. and Greco, S. 2004. Clustering Transactional XML Data with Semantically-Enriched Content and Structural Features. In *Proceedings of the 5th Conference on Web Information Systems Engineering (WISE'04)*. 266-278.

Terra, E. and Clarke, C. L. A. 2003. Frequency Estimates for Statistical Word Similarity Measures. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL'03)*. 244-251.

Theobald, M., Schenkel, R., and Weikum, G. 2003. Exploiting Structure, Annotation, and Ontological Knowledge for Automatic Classification of XML Data. In *Proceedings of the International Workshop on Web and Databases (WebDB'03)*. 1-6.

Turney, P. D. 2004. Word Sense Disambiguation by Web Mining for Word Co-occurrence Probabilities. In *Proceedings of the 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (Senseval-3)*. 239-242.

Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., and Ciravegna, F. 2006. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Journal of Web Semantics 4*, 1, 14-28.

Wilkinson, K., Sayers, C., Kuno, H., and Reynolds, D. 2003. Efficient RDF storage and retrieval in Jena2. In *Proceedings of Very Large Data Bases Workshop on Semantic Web and Databases*. 131-150.

[1] http://wordnet.princeton.edu/

[2] http://www.ilc.cnr.it/viewpage.php/sez=ricerca/id=834/vers=ita

[3] http://www.ilc.cnr.it/iwndb/

[4] http://multiwordnet.itc.it/english/home.php/

[5] http://multiwordnet.itc.it/online/

[6] http://www.demauroparavia.it/

[7] http://dizionari.corriere.it/dizionario_italiano.shtml

[8] http://www.garzantilinguistica.it/

[9] http://www.dizionario-italiano.it/

[10] http://it.wiktionary.org/wiki/Pagina_principale

[11] http://www.uniroma2.it/didattica/IA0506/deposito/NLP_3.pdf

[12] http://www.cli.di.unipi.it/~carmigna/projects/POS%20Tagging%20con%20Metodi%20Statistici.pdf

[13] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

[14] http://www.ims.uni-stuttgart.de/projekte/tc/

[15] http://www.cele.nottingham.ac.uk/~ccztk/treetagger.php

[16] http://textpro.itc.it/ oppure http://textpro.fbk.eu/index.php

[17] http://www.english.bham.ac.uk/staff/omason/software/qtag.html

[18] http://nlp.stanford.edu/software/tagger.shtml

[19] http://ucrel.lancs.ac.uk/claws/

[20] http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/postagger/

[21] http://web.media.mit.edu/~hugo/montylingua/

[22] http://ai-nlp.info.uniroma2.it/external/chaosproject/

[23] http://dev.sslmit.unibo.it/linguistics/morph-it.php

[24] http://dev.sslmit.unibo.it/corpora/corpus.php?path=&name=Repubblica

[25] http://dev.sslmit.unibo.it/linguistics/downloads/readme-morph-it.txt

[26] http://standards.ieee.org/reading/ieee/std_public/description/se/830-1998_desc.html