# Analisi di testi per il Semantic Web e il Question Answering

# Relazione Annuale

# Ottobre 2008

## 1. Introduzione

Il piano di lavoro prevedeva lo svolgimento delle seguenti attività nel primo anno:

**Fase 1: mesi 0-6**

1. Progetto dettagliato dell'architettura del sistema di QA: moduli software e relative interfacce
2. Analisi del corpus SI-TAL e definizione delle trasformazioni da attuare
3. Sviluppo di euristiche e di algoritmi per trasformare il corpus SI-TAL nel formato CoNLL
4. Creazione di uno strumento di annotazione visiva per l'inserimento e la correzione delle annotazioni
5. Progettazione architettura del sistema di First Topic Detection
6. Analisi delle tecniche di calcolo di similarità tra news

**Fase 2: mesi 7-12**

7. Progettazione di un parser induttivo a dipendenze
8. Studio del problema del trattamento di relazioni non projective presente nella lingua italiana
9. Sperimentazione di tecniche per risolvere le dipendenze non projective
10. Studio e scelta delle feature da usare nell'apprendimento del parser
11. Realizzazione del parser a dipendenze
12. Prove e verifiche di accuratezza del parser
13. Analisi degli errori nei risultati del parser e messa a punto
14. Messa a punto di sistema di estrazione di Named Entities
15. Sviluppo di algoritmi di calcolo di similarità tra news

Il piano di lavoro è stato seguito completamente, con una piccola variazione. Invece di utilizzare fonti di notizie (news) su cui applicare l'analisi linguistica, si è preferito orientarsi a utilizzare la Wikipedia italiana come fonte di materiale.

Questo è in linea con il crescente ruolo di Wikipedia come corpus di conoscenza enciclopedica, che viene continuamente aggiornato attraverso uno sviluppo cooperativo che è uno dei fondamenti del Web 2.0.

La fase di progetto dell'architettura del sistema di QA ci ha infatti portato a progettare un sistema di interrogazione avanzato in linguaggio naturale della Wikipedia italiana, in grado di utilizzare le relazioni semantiche espresse nei testi.

Nel progetto è stato quindi aggiunto un task relativo all'estrazione e all'analisi dei testi della Wikipedia italiana.

# 2. Attività Svolta

Il progetto è iniziato ufficialmente il 1 ottobre 2007. Le attvità svolte nel corso del primo anno si possono raggruppare nelle seguenti tematiche:

1. Specifiche Linguistica
2. Annotazione Linguistica
3. Specifiche Software
4. Visual Annotator
5. Dependency Parsing
6. Wikipedia Extraction
7. Wikipedia Analysis

Nel seguito forniremo il dettaglio dell'attività svolta in ciascuna di esse.

Il risultati prodotti nel progetto comprendono sia un insieme dettagliato di specifiche linguistiche, sia corpora annotati secondo tali specifiche, sia una serie di strumenti software per la creazione e l'uso di tali risorse.
Si è deciso di dare una denominazione collettiva a questo insieme di risultati, chiamandoli ***Tanl*** (Text Analysis and Natural Language) in quanto si tratta di strumenti idonei per compiti di *Text Analytics*, ossia che incorporano tecniche basate sull'analisi linguistica per l'interpretazione, l'estrazione di conoscenze, e la trasformazione di testi.

Pertanto descriveremo in seguito i seguenti prodotti:

1. Tanl POS tagset
2. Tanl ISST Corpus
3. Tanl Repubblica Corpus
4. Tanl POS Tagger
5. Tanl Parser
6. Tanl Annotated Wikipedia

## *2.1 Specifiche Linguistiche*

Il progetto ha stabilito la necessità di produrre una serie di specifiche linguistiche e di criteri di annotazione da usare nella costruzione delle risorse linguistiche da produrre nell'ambito del progetto, progettate a partire da revisioni e raffinamenti di specifiche per la lingua italiana sviluppate in precedenti progetti.

Nel corso del primo anno, l'attività si è incentrata su:

1. la definizione di un insieme di etichette (il cosiddetto "tagset") per l'annotazione del corpus di addestramento ai livelli morfo-sintattico e sintattico a dipendenze. Il tagset definito specifica l'ambito dei fenomeni linguistici coperti nell'annotazione del corpus e le relative modalità di annotazione;
2. la definizione di criteri per l'applicazione delle etichette definite per i livelli morfo-sintattico e sintattico a dipendenze ai contesti reali del corpus;
3. l'attività di annotazione vera e propria di due corpora ai livelli morfo-sintattico e sintattico a dipendenze. Questa fase di annotazione ha costituito un banco di prova importante della robustezza e adeguatezza descrittiva degli schemi e dei criteri di annotazione messi a punto nelle attività di cui a punti 1) e 2) che sono stati soggetti a periodiche revisioni.

Un punto di partenza di tali attività è stato identificato nella *Treebank Sintattico Semantica dell'Italiano* ("Italian Syntactic-Semantic Treebank", ISST) sviluppata tra

il 1999 e il 2001 nell'ambito del progetto nazionale SI-TAL ("Sistema Integrato per il Trattamento Automatico del Linguaggio"), finalizzato alla creazione di risorse per l'italiano e finanziato dal MURST nell'ambito della legge 46/82 art.10 (coordinamento prof. Antonio Zampolli).

In quanto segue, per ciascuna delle linee elencate sopra verrà fornito un rendiconto dettagliato del lavoro svolto.

### 2.1.1 Definizione del tagset Tanl

La definizione del tagset per l'annotazione del corpus è stata guidata dallo spettro di usi che si prospettano per un corpus annotato a vari livelli di analisi linguistica nell'ambito di applicazioni basate sull'elaborazione del linguaggio naturale. La tipologia degli usi di cui una risorsa di questo tipo si rende suscettibile è varia: si va dall'ambito più propriamente applicativo, per compiti quali l'addestramento automatico ("training/tuning") di sistemi per l'analisi sintattica automatica dell'italiano, alla valutazione di sistemi di elaborazione del linguaggio naturale. L'aspetto della valutazione dei risultati di diversi sistemi e tecniche è oggi cruciale, e anche per questo è essenziale la creazione di corpora annotati da usarsi come riferimento per la valutazione ("testbed"). Inoltre, corpora testuali annotati a diversi livelli di descrizione possono essere usati per l'induzione di modelli linguistici e per l'acquisizione di informazione linguistica.

Per entrambi i livelli morfo-sintattico e sintattico a dipendenze siamo partiti dalle specifiche di ISST che sono state riviste e integrate per soddisfare i desiderata del progetto.
I dettagli dei tagset definiti sono riportati nell'allegato A.

### 2.1.2 Definizione dei criteri per l'annotazione del corpus

Alla definizione degli schemi di annotazione per i livelli di descrizione linguistica morfo-sintattico e sintattico a dipendenze discussi nella precedente sezione ha fatto seguito una fase di specifiche dei criteri di applicazione dei tagset al testo, e in modo particolare a contesti e costrutti sintattici che possono porre particolari problemi di annotazione. Il risultato di questa attività è confluito in una sorta di guida all'applicazione degli schemi proposti, che fornisce criteri univoci di identificazione delle etichette morfo-sintattiche e delle relazioni di dipendenza definite e che illustra la rappresentazione di fenomeni sintattici lessicalmente governati (ad esempio schemi di sottocategorizzazione associati a classi di nomi, verbi e aggettivi) così come di costruzioni complesse che rispondono a principi generali della grammatica dell'italiano (ad esempio, costruzioni con frasi relative ed interrogative, fenomeni di ellissi, etc.). Nella fase di verifica e di addestramento all'annotazione del corpus (vedi sezione sotto) i criteri di annotazione sono stati rivisti ed estesi per fornire all'annotatore una casistica più dettagliata possibile che potesse servire da guida affidabile e che al contempo riducesse al minimo il margine di arbitrarietà dell'annotazione.

I criteri di annotazione sono accessibili nel wiki del progetto, nella sezione "Annotation Guidelines" all'interno delle sezioni "POS Tagset" (http://medialab.di.unipi.it/wiki/index.php/POS_Tagset) e "Dependency Tagset" (http://medialab.di.unipi.it/wiki/index.php/Dependency_Tagset) rispettivamente. In quanto segue, sono riepilogate le principali aree della grammatica per le quali sono stati forniti dettagliati criteri di annotazione (l'elenco non è da considerarsi esaustivo).

- Tagging morfo-sintattico
  i) Pronomi clitici;
  ii) Nomi propri. Si noti che la corretta etichettatura dei nomi propri ha un impatto sui livelli successivi di analisi, in particolare per quanto riguarda il riconoscimento e la categorizzazione di Named Entities;
  iii) Criteri per la gestione di ambiguità tra:
    a. l'accezione nominale e aggettivale (o participiale);
    b. l'accezione aggettivale e participiale;
    c. l'accezione come articolo o numero;
    d. l'accezione come congiunzione o avverbio;
  iv) Criteri per l'annotazione di casi di sottospecificazione di genere e numero;

- Tagging sintattico a dipendenze
  i) Annotazione delle dipendenze riguardanti:
    a. pronomi clitici;
    b. punteggiatura;
    c. nomi propri strutturalmente complessi;
    d. date.

## *2.2 Risorse Linguistiche*

Una volta definiti i tagset Tanl per l'annotazione e i criteri per la loro applicazione ad un corpus reale di testi, si è proceduto all'annotazione vera e propria dei testi ai livelli morfo-sintattico e sintattico a dipendenze.

L'annotazione è stata effettuata su due corpora distinti:

1. corpus della Treebank di SI-TAL (ISST) appartenente alla partizione "bilanciata" (ovvero rappresentativa di diversi generi testuali). In particolare, sono state selezionate le porzioni corrispondenti al "Corriere della Sera" e ai "Periodici", per un totale di 79.654 parole (tokens) distribuiti in 4.162 frasi.

2. corpus di La Repubblica, formato dal testo dalla pubblicazione *Gli anni di Repubblica*, e consistente di 108.874 parole, suddivise in 3.719 frasi.

Per trasformare il corpus ISST in modo conforme allo schema di annotazione definito per il presente progetto, sono stati sviluppate euristiche e algoritmi che a partire dal formato di rappresentazione ISST hanno generato il corpus annotato con i tag Tanl e nel formato CoNLL. Il risultato di questo processo di conversione è stato interamente rivisto manualmente con l'ausilio di uno strumento di annotazione visiva appositamente sviluppato finalizzato all'inserimento e alla correzione delle annotazioni.

In questa fase di controllo manuale sono state condotte due tipologie di revisioni:

I. sono state riviste tutte le dipendenze riguardanti parole funzionali, ovvero tutte quelle dipendenze che in ISST erano codificate mediante tratti associati ai partecipanti della relazione;
II. è stata rivista l'annotazione originaria, per renderla conforme allo schema di annotazione Tanl e ai criteri di annotazione definiti, e per la rimozione di inevitabili errori e incoerenze nell'annotazione originaria.

Questa fase di revisione manuale delle annotazioni del corpus ha svolto un ruolo centrale nella verifica sia dell'adeguatezza descrittiva del tagset messo a punto per i

due livelli di annotazione sia della copertura dei criteri di annotazione proposti; entrambi sono stati talora rivisti ed estesi sulla base dell'evidenza proveniente dai corpora.

Per quanto riguarda l'annotazione del corpus di Repubblica, esso era stato annotato unicamente con tag di Part-of-speech, dal gruppo di Marco Baroni all'Università di Bologna. Anche in questo caso sono stati sviluppati algoritmi di conversione in formato Tanl. Per quanto riguarda il tagging morfologico è stato dapprima applicato un tagger morfologico automatico (Morphit!) e quindi sono stati sviluppati ed applicati degli strumenti semiautomatici di revisione.

#### 2.2.1.1 Lessico Italiano

È stato costruito un lessico italiano, contenente oltre 1.250.000 forme, in formato Tanl.

Esso è stato generato mediante funzioni di espansione delle coniugazioni e declinazioni a partire dai lemmi di un dizionario contenente 65.000 voci, derivato dallo Zingarelli Minore, con il contributo di Achim Stein, dell'Institut für Linguistik/Romanistik, Stuttgart.

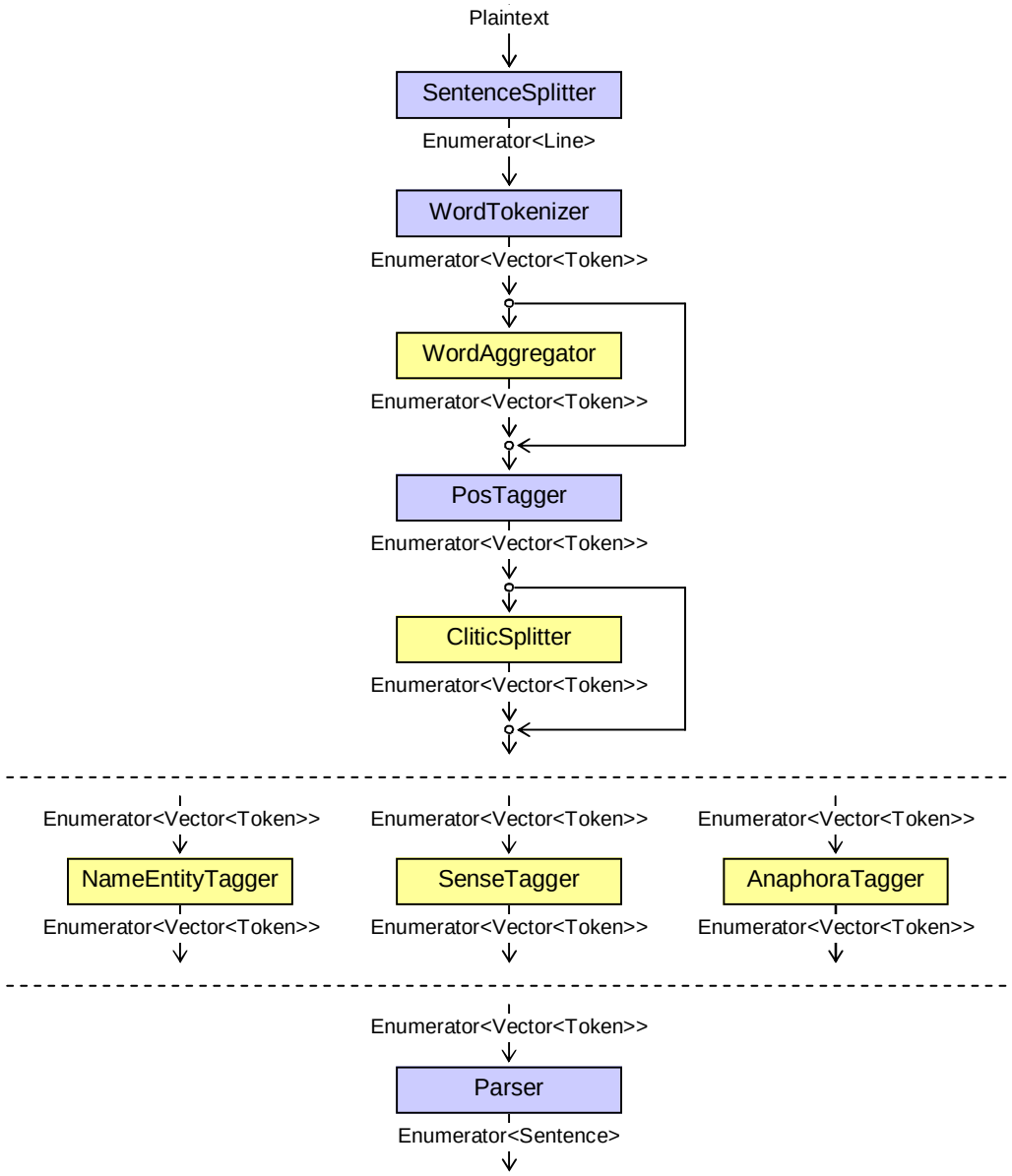
Le funzioni di espansione sono state riscritte tenendo conto delle specifiche Tanl e di ulteriori informazioni linguistiche, ad esempio verbi transitivi o intransitivi, per limitare la generazione di voci verbali clitiche: ad esempio è presente "diciamolo" ma non "corriamolo".

## *2.3 Specifiche Software*

Il software TANL consiste in una *pipeline* di moduli per:

1. Tagging
2. Parsing
3. Indexing

L'architettura dei moduli di *tagging* e di *parsing* è basata sulla metafora di streaming, come viene illustrato nel diagramma seguente:




Ciascun modulo consuma uno *stream* prodotto da un modulo precedente e produce a sua volta uno stream. Gli stream sono definiti da un'interfaccia generica definita come segue:

```
template <class T>
struct Enumerator {
  typedef T             ItemType;

  /** Advances to the next element of the collection.
      @return true if there is another item available. */
  virtual bool          MoveNext() = 0;

  /** @return the current element. */
  virtual ItemType      Current() = 0;

  /**   Optional reset to the beginning of the Enumerator. */
  virtual void          Reset() {}
};
```

Il primo modulo della pipeline, denominato *SentenceSplitter*, permette la suddivisione del testo dato in input in una sequenza di frasi. Il testo viene analizzato e all'occorrenza corretto dove l'eventuale già presente suddivisione risulti errata. Lo strumento fornisce in uscita un'enumerazione di stringhe, ognuna delle quali rappresenta una frase.

Il secondo modulo viene denominato *WordTokenizer* e si occupa della suddivisione delle parole che compongono una frase. Per ogni frase data in input, fornisce un vettore di token. La struttura *Token*, a questo stadio della pipeline, contiene le informazioni relative all'ID (un intero crescente che rappresenta la posizione del token all'interno della frase) e alla FORM (la parola stessa).
Il *WordAggregator*, il modulo successivo, permette di unire le espressioni di uso comune in un singolo sintagma (ad esempio "a meno che" diventa "a_meno_che"). Questo strumento è opzionale, a seconda che si desideri utilizzare o meno le multi-word. Allo stato attuale non abbiamo implementazioni di questo tool.
Il *PosTagger* è il modulo in grado di assegnare sia *part-of-speech* che lemma ai token di una frase. Lo strumento arricchisce la struttura *Token* con nuove informazioni, nella fattispecie POS e LEMMA.
Il modulo successivo, denominato *CliticSplitter*, permette il troncamento delle forme clitiche in due o più token (ad esempio il verbo "avercelo" diventa "aver- ce- lo"). Anche l'utilizzo di questo strumento nella pipeline è opzionale.
I moduli *NameEntityTagger*, *SenseTagger* e *AnaphoraTagger*, possono essere aggiunti alla pipeline, qualora si desideri, e senza un ordine particolare. Essi continuano ad usare sia in input che in output un'enumerazione di vettori di token, per cui non possono fare altro che arricchire gli attributi, senza alterare la struttura delle frasi.
Attualmente è in corso lo studio e l'implementazione di questi strumenti.

L'ultimo modulo della pipeline è rappresentato dal *Parser*, il quale permette la costruzione dell'albero delle dipendenze tra i token di una singola frase. Lo strumento prende in input un'enumerazione di vettori di token, e produce un'enumerazione di sentence. La struttura *Sentence* permette la rappresentazione dell'albero delle dipendenze mediante una lista di link.

I moduli sono stati implementati in linguaggi di programmazione differenti (C, C++, Python). Ciascun modulo implementa l'interfaccia `IPipe` che consente di attuare il meccanismo di composizione:

```
template <class Tin, Tout>
struct IPipe {
   Enumerator<Tout>* pipe(Enumerator<const Tin&);
};
```

Per esempio, il parser implementa l'interfaccia:

```
class Parser : IPipe<Enumerator<vector<Token*>*>,
Enumerator<Sentence*>
{ … };
```

Al fine di agevolare l'utilizzo dei componenti della pipeline anche da altri linguaggi, sono stati realizzati dei wrapper mediante SWIG, uno strumento di sviluppo software che permette di inter-connettere programmi scritti in C e C++ con una varietà di linguaggi ad alto livello.

In particolare abbiamo realizzato dei wrapper per Python, ini modo da poter creare applicazioni che usano i componenti della pipeline combinandoli mediante degli script Python.
Ecco ad esempio uno snippet di codice che utilizza i moduli principali della pipeline:

```
# crea un SentenceSplitter che legge da standard input
ss = SentenceSplitter(stdin)
# collega ad esso un tokenizer che opera sul suo output
wt = WordTokenizer().pipe(ss)
# collega ad esso un PosTagger per annotare le frasi
# tokenizzate
pt = PosTagger(italian.params).pipe(wt)
# collega ad esso un Parser configurato per l'italiano
pa = Parser(italian.conf).pipe(pt)

# attiva la pipeline
while pa.MoveNext():
   # estrai la prossima frase analizzata
   sentence = p.Current()
   print sentence
```

## *2.4 Visual Annotator*

L'attività del progetto richiede spesso di intervenire manualmente nell'annotazione di testi. Per facilitare questa attività, oltre che per consentire una agevole analisi visiva dei risultati del parsing, è stato sviluppato uno strumento grafico interattivo per annotare/visualizzare corpora di testi in formato di alberi a dipendenze.
Lo strumento sviluppato è denominato DGAnnotator ed è disponibile alla pagina:

http://medialab.di.unipi.it/Project/QA/Parser/DgAnnotator/

DGA fornisce una semplice interfaccia utente grafica che permette di creare e manipolare facilmente le strutture sintattiche delle frasi. DGA si basa sul formalismo delle Grammatiche a dipendenza, di conseguenza, la strutture sintattiche consistono in relazioni di dipendenza tra le parole di una frase. Relazioni di dipendenza sono rappresentati graficamente, come archi dalla parola di testa alla parola dipendente, etichettati con il nome della relazione grammaticale esistente tra le parole.

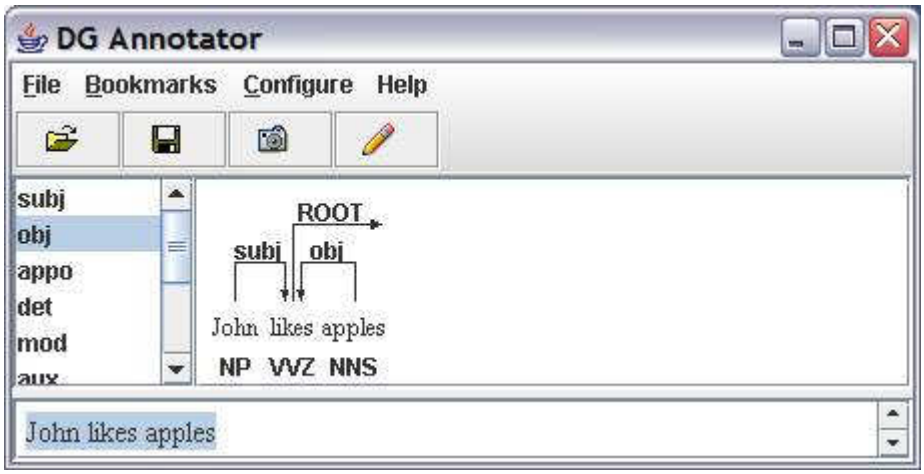



**Figure 1.** Finestra di annotazione.

L'utente può creare, modificare o eliminare archi di dipendenza tra i termini di un corpus di testi letto da file in tre possibili formati:

1. puro testo, una frase per riga
2. formato CoNLL X
3. formato XML

Tra le altre funzioni, menzioniamo la possibilità di confrontare diversi alberi sintattici diversi per le stesse frasi, utili per individuare ad esempio gli errori effettuati da un parser rispetto ai risutati corretti.

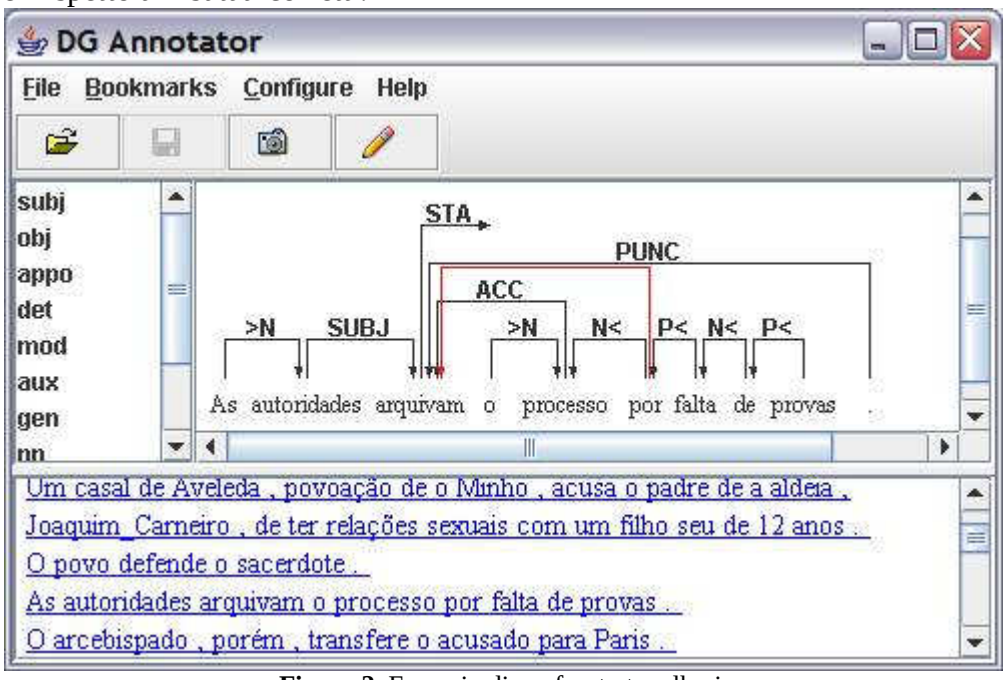



**Figure 2.** Esempio di confronto tra alberi.

Nella Figure 2 ad esempio, è evidenziato in colore rosso un arco errato.

Infine è possibile ottenere una immagine degli alberi visualizzati, in formato PNG per poter essere riutilizzata includendola in altri documenti o pagine Web.

DGAnnotator è configurabile per poter usare tagset diversi a seconda della lingua o del corpus prescelto: il tagset appare nel pannello a sinistra nella finestra dell'annotatore.

DGAnnotator è scritto in Java ed è disponibile su tutte le piattaforme che dispongono di Java 1.5.

## *2.5 Dependency Parsing*

La progettazione e lo sviluppo di un parser induttivo a dipendenze si è articolata nelle seguenti fasi:

1) sviluppo di uno Shift/Reduce dependency parser in grado di gestire relazioni non projective, molto comuni nella lingua italiana;

2) definizione di un insieme di feature da utilizzare nell'apprendimento del parser, per massimizzare l'accuratezza del sistema nell'analisi di testi scritti in lingua italiana;

3) accurata analisi degli errori tipici del parser di cui al punto 1) e di altri dependency parser che definiscono lo stato dell'arte. Sulla base di questa

analisi è stato progettata e sviluppata un tecnica di dependency parsing in due fasi, in grado di correggere questi errori e migliorare l'accuratezza del sistema;

4) sviluppo di un nuovo sistema di combinazione di parser in grado di combinare l'output di più sistemi e ottenere un ulteriore miglioramento rispetto ai sistemi di partenza.

I quattro punti elencati ci hanno permesso di sviluppare un parser a dipendenze statistico tra i più efficienti (fino a 200 frasi/sec) e tra i più accurati attualmente esistenti, secondo le risultanze delle valutazioni ufficiali effettuate in recenti competizioni internazionali.

### 2.5.1 Sviluppo del Dependency Parser

Il parser sviluppato in questo progetto è uno shift/reduce dependency parser denominato DeSR (Dependency Shift Reduce) [1], basato su una variante dell'approccio di Yamada e Matsumoto. L'algoritmo di DeSR costruisce le dipendenze tra le parole di una frase scorrendo il testo in una sola passata, decidendo a ciascun passo se compiere un'azione di *Shift* (avanzare nel testo) o un'azione di *Reduce*, creando una dipendenza tra due parole.

Nello sviluppo delle regole di *Reduce* sono stati risolti due problemi:

1. il trattamento di dipendenze *non projective*, ossia in cui gli archi di dipendenza si incrociano: situazione particolarmente frequente in lingue come l'italiano, che consentono un certo grado di libertà nell'ordine delle parole all'interno di una frase
2. efficienza, ottenendo un procedimento deterministico, che svolge l'intero procedimento con una scansione lineare della frase.

DeSR utilizza un classificatore per apprendere e predire quale azione compiere a ogni passo dell'analisi del testo, sulla base di un insieme di *feature* ricavate dalle parole sotto esame e dallo stato del parser.

Le *feature* da utilizzare possono essere specificate in un apposito file di configurazione, in modo tale da poter adattare il parser alle caratteristiche di lingue diverse.

Anche l'algoritmo di *machine learning* da utilizzare sia in fase di apprendimento che di analisi può essere scelto tra le diverse versioni implementate (in particolare: Averaged Perceptron, Maximum Entropy, Memory-Based Learning, Support Vector Machines).

### 2.5.2 Selezione delle feature

In questa fase è stato definito un insieme di *feature* da utilizzare nell'apprendimento del parser, per massimizzare l'accuratezza del sistema nell'analisi di testi scritti in lingua italiana.

La scelta delle *feature* è una delle fasi cruciali nello sviluppo di in un sistema basato su tecniche di *machine learning*.

Per ottenere l'insieme ottimale di *feature* per l'addestramento del parser sull'italiano abbiamo utilizzato sia un processo di selezione automatica, che una serie di esperimenti nei quali sono stati definiti insiemi di feature manualmente.

La selezione automatica è stata effettuata attraverso un processo di *backward selection*, che partendo da un insieme di *feature* il più ampio possibile valuta il sistema eliminando gradualmente le *feature* fino a quando non vengono ottenuti più ulteriori miglioramenti dall'eliminazione.

Mentre il processo di definizione manuale delle *feature* è stato effettuato per scoprire caratteristiche *nascoste* della lingua analizzata che utilizzate come feature nel processo di addestramento del sistema potessero apportare miglioramenti all'accuratezza del nostro parser.

Attraverso i due diversi processi di selezione è stato definito l'insieme ottimale di *feature* con le quali addestrare DeSR nelle varie lingue.

Il parser DeSR ha ottenuto risultati di accuretezza che sono tra i migliori nelle recenti competizioni della CoNLL (Conference on Natural Language Learning) dal 2006 al 2008, che costituiscono il punto di riferimento e lo stato dell'arte della ricerca sul parsing a dipendenze.

### 2.5.3 Reverse Revision Parser

Per ulteriormente migliorare l'accuratezza di DeSR, è stata effettuata un'accurata analisi degli errori, dei due tipi di parser migliori partecipanti allo shared task della CoNLL 2007 sul parsing multilingue a dipendenze.

La valutazione ha messo in evidenza nei parser shift/reduce un calo di accuratezza nell'analisi di dipendenze tra parole distanti all'interno della frase.

A seguito di questa analisi abbiamo introdotto un nuovo metodo di shift/reduce parsing, chiamato ***Reverse Revision Parsing***, in grado di migliorare nettamente le accuratezze del tradizionale algoritmo di shift/reduce parsing, preservandone comunque i costi computazionali.

L'approccio consiste nell'analizzare la frase con un primo parser shift/reduce deterministico, seguito da una seconda analisi della stessa frase scorrendola al contrario, ma sfruttando informazioni prodotte dalla prima analisi. L'intuizione alla base di questa tecnica è che il secondo parser, avendo a disposizione l'analisi del primo e scorrendo il testo in ordine inverso, sia in grado di riconoscere e correggere le predizioni sbagliate del primo parser. Questa intuizione si è rivelata corretta e come risultato abbiamo ottenuto un notevole miglioramento nelle accuratezze del parser, senza aumentare significativamente i costi computazionali dell'algoritmo di partenza.

Per sviluppare questo modello di parsing abbiamo modificato DeSR per renderlo in grado di analizzare una frase scorrendola in una qualunque direzione e di sfruttare informazioni estratte da una prima analisi del testo.

### 2.5.4 Combinazione di Parser

Un ulteriore miglioramento all'accuratezza del nostro parser è stato ottenuto introducendo una tecnica per combinare i risultati di più analisi e ottenerne una maggiormente accurata. La tecnica fa uso di un nuovo algoritmo di combinazione, chiamato ***Quasi-Linear Parser Combination***, che presenta un costo computazionale inferiore a quelli noti in letteratura e di mantenere efficiente l'intero sistema di parsing [6].

L'algoritmo sfrutta il fatto di combinare alberi piuttosto che grafi riuscendo a scendere a una complessità quasi-lineare rispetto al costo quadratico dei precedenti

algoritmi di combinazione che si basavano sul calcolo del maximum spanning tree di un grafo.

I risultati migliori sono stati ottenuti finora combinando tre alberi di parsing ottenuti utilizzando tre varianti di DeSR: Left-to-Right DeSR (che analizza il testo da sinistra verso destra) e Right-to-Left DeSR (che analizza il testo da destra verso sinistra) e Reverse Revision DeSR (che analizza il testo da destra verso sinistra utilizzando le informazioni prodotte dal Left-to-Right DeSR).

### 2.5.5 Valutazione del sistema

La metrica utilizzata nella CoNLL 2006 e 2007 per valutare l'accuratezza del parser è la *labeled attachment score* (LAS), ossia la percentuale di parole alle quali è stata assegnata sia la corretta testa che la corretta relazione di dipendenza. I risultati sono stati ottenuti utilizzando come algoritmo di machine learning le SVM e il sistema è stato testato sui test set ufficiali dello shared task della CoNLL-X.

Per l'italiano i diversi modelli di DeSR ottengono questi risultati:

| Modelli | LAS |
|---|---|
| **Left-to-Right** | 81.40 |
| **Right-to-Left** | 82.89 |
| **Reverse Revision** | 83.52 |
| **Parser Combination** | **84.16** |

Confrontando questi risultati con quelli ottenuti dai vari partecipanti della CoNLL-2007 si osserva che la combinazione di parser (Parser Combination, nella tabella) raggiunge il secondo risultato assoluto in termini di LAS, dopo l'84.40 % del miglior parser per l'italiano.

Da notare che il miglior parser per l'italiano della CoNLL-2007 utilizza una tecnica di combinazione di parser in cui vengono combinati gli output di sei shift/reduce parser (nel nostro lavoro solo tre), utilizzando per la combinazione l'algoritmo di *Chu-Liu-Edmonds* per il calolo di maximum spanning tree di un grafo, che ha un costo computazionale quadratico rispetto a quello quasi lineare del nostro sistema.
Per quanto riguarda le valutazioni dei costi computazionali, il sistema implementa un algoritmo lineare o quasi lineare nel caso della combinazione. Utilizzando un PC con una CPU Intel® Xeon™ da 2.80 GHz e 2 GB di memoria, i tempi per l'analisi delle 249 frasi (5096 parole) dei test precedenti sono stati:

| Modelli | Tempi |
|---|---|
| **Left-to-Right** | 4' 30" |
| **Right-to-Left** | 4' 25" |
| **Reverse Revision** | 4' 35" |
| **Parser Combination** | 0.4" |

La tabella seguente riporta i risultati del nostro sistema in termini di LAS per sedici lingue analizzate nello shared task della ConLL 2006 e 2007. La colonna **CoNLL**

riporta i risultati del miglior parser della competizione CoNLL-X per ogni lingua analizzata.

| Language | DeSR | CoNLL |
|---|---|---|
| Arabic | **68.36** | 66.91 |
| Bulgarian | **87.87** | 87.57 |
| Chinese | 87.77 | **89.96** |
| Czech-2006 | **81.76** | 80.18 |
| Czech-2007 | **80.57** | 80.19 |
| Danish | 84.71 | **84.79** |
| Dutch | **79.59** | 79.19 |
| English | 89.09 | **89.61** |
| German | 85.32 | **87.34** |
| Italian | 84.16 | **84.40** |
| Japanese | 91.53 | **91.65** |
| Portuguese | **88.14** | 87.60 |
| Slovene | **75.54** | 73.44 |
| Spanish | **82.33** | 82.25 |
| Swedish | 83.17 | **84.58** |
| Turkish | 65.25 | **65.68** |

## *2.6 Wikipedia Extraction*

La Wikipedia Italiana è stata scelta come fonte di materiale da cui estrarre i testi. I gestori dell'enciclopedia mettono a disposizione, con frequenza mensile, un *dump* dell'intero database degli articoli dell'enciclopedia: si tratta di un unico file in formato XML contenente l'intera enciclopedia, il quale viene normalmente usato allo scopo di effettuare elaborazioni in forma semiautomatica su Wikipedia quali analisi statistiche, estrazione di elenchi di servizio, ecc.

I dump della Wikipedia Italiana sono disponibili al seguente indirizzo:

http://download.wikimedia.org/itwiki/latest

Ai fini dell'analisi di testi è necessario estrarre il testo puro degli articoli, privo cioè di "decorazioni" sintattiche (grassetti, corsivi, sottolineature, ecc.).

Tale compito è affidato ad uno script di conversione, il quale "estrae" dal dump del database di Wikipedia soltanto i testi, scartando tutte le altre parti o annotazioni contenute in una pagina di Wikipedia, quali immagini, tabelle, liste, riferimenti ed elenchi.

Nel dump dell'enciclopedia, ogni articolo è rappresentato con un nodo XML, codificato come nel seguente esempio, relativo all'articolo intitolato "Armonium":

```
<page>
  <title>Armonium</title>
  <id>2</id>
  <timestamp>2008-06-22T21:48:55Z</timestamp>
  <username>Nemo bis</username>
  <comment>italiano</comment>
  <text
xml:space="preserve">[[Immagine:Harmonium2.jpg|thumb|right|300
px]]
```

```
L''''armonium'''' (in francese, ''harmonium'') è uno
[[strumenti musicali|strumento musicale]] azionato con una
[[tastiera (musica)|tastiera]], detta manuale. Sono stati
costruiti anche alcuni armonium con due manuali.

==Armonium occidentale==
Come l'[[organo (musica)|organo]], l'armonium è utilizzato
tipicamente in [[chiesa (architettura)|chiesa]], per
l'esecuzione di [[musica
sacra]], ed è fornito di pochi registri, quando addirittura in
certi casi non ne possiede nemmeno uno: il suo [[timbro
(musica)|timbro]] è molto meno ricco di quello organistico e
così pure la sua estensione.

...

==Armonium indiano==
{{S sezione}}

== Voci correlate ==
*[[Musica]]
*[[Generi musicali]]</text>
</page>
```

Per questo articolo il Wikipedia Extractor produce il seguente testo:

```
<doc id="2" url="http://it.wikipedia.org/wiki/Armonium">
Armonium.
L'armonium (in francese, “harmonium”) è uno strumento musicale
azionato con una tastiera, detta manuale. Sono stati costruiti
anche alcuni armonium con due manuali.
Armonium occidentale.
Come l'organo, l'armonium è utilizzato tipicamente in chiesa,
per l'esecuzione di musica sacra, ed è fornito di pochi
registri, quando addirittura in certi casi non ne possiede
nemmeno uno: il suo timbro è molto meno ricco di quello
organistico e così pure la sua estensione.
...
</doc>
```

Lo script di estrazione, denominato *Wikipedia Extractor*, è scritto in Python e si propone di raggiungere un’elevata accuratezza di estrazione.

Una delle maggiori difficoltà affrontate è stata quella di dover considerare la disomogeneità degli autori dei vari articoli dell’Enciclopedia.

La formattazione di una tipica pagina di Wikipedia prevede l’uso della *sintassi Wiki*, vale a dire un formalismo facile ed intuitivo con cui aggiungere meta-informazioni ai testi (grassetti, corsivi, sottolineature, link, immagini, tabelle, ecc.). Purtroppo non tutti gli autori sruttano questa notazione e inseriscono direttamente dei markup HTML all’interno degli articoli. Spesso tuttavia i markup Wiki e HTML non sono usati in modo corretto (tag che aperti e non chiusi, attributi errati, ecc.). Pertanto l’estrattore utilizza delle euristiche, cercando di massimizzare i casi di successo dell’estrattore. Resta quindi un margine di errore difficile da risolvere, il quale potrebbe essere oggetto di studio per la realizzazione di versioni successive dello strumento.

Informazioni tecniche e sull'utilizzo del Wikipedia Extractor sono disponibili al seguente indirizzo:

http://medialab.di.unipi.it/wiki/index.php/Wikipedia_Extractor

## *2.7 Text Analysis*

### 2.7.1 Sentence Splitter

Il *Sentence Splitter* si occupa della divisione del testo in frasi.

La nostra implementazione sfrutta la libreria open-source della NLTK, denominata *Punkt Sentence Tokenizer*. La tecnica usata da Punkt è basata su un algoritmo unsupervised (*Unsupervised Multilingual Sentence Boundary Detection*) il quale costruisce un modello di abbreviazioni, collocazioni ed inizi di frasi. Tuttavia una migliore accuratezza si ottiene calibrando i valori di soglia tramite l'addestramento su una collezione di testi nella lingua di interesse.

Maggiori informazioni sono disponibili ai seguenti indirizzi:

- NLTK (Natual Language Toolkit): http://nltk.org
- Unsupervised Multilingual Sentence Boundary Detection: http://www.linguistics.ruhr-uni-bochum.de/~strunk/ks2005FINAL.pdf

Il sentence splitter è stato dapprima addestrato su un'ampia collezione di testi in lingua italiana (articoli estratti dal giornale "La Repubblica" e "Il Corriere della Sera") e rivisti manualmente. Successivamente il corpus di addestramento è stato arricchito con diverse centinaia di frasi (estratte in modo semi-automatico da Wikipedia) contenenti abbreviazioni. Questa fase ha permesso di migliorare notevolmente la precisione del sentence splitter anche in presenza di abbreviazioni poco frequenti nell'uso comune della lingua italiana.

Anche il *SentenceSplitter* è realizzato in Python, e può essere utilizzato sia singolarmente (fornendogli un testo in input) o in cascata al Wikipedia Extractor (descritto in precedenza). Nel secondo caso lo script utilizza e mantiene lo stesso formalismo usato dall'estrattore per la rappresentazione dei documenti, ossia:

```
<doc id="..." url="...">
  ...
</doc>
```

Informazioni tecniche e sull'utilizzo del Sentence Splitter sono disponibili al seguente indirizzo:

http://medialab.di.unipi.it/wiki/index.php/Sentence_Splitter

### 2.7.2 Word Tokenizer

Il tokenizer è lo strumento che effettua la suddivisione delle frasi in unità di testo (appunto *token*) corrispondenti a forme linguistiche.

La versione per l'inglese è stata realizzato facendo uso del generatore di analizzatori lessicali *Flex* (http://flex.sourceforge.net/).

Tuttavia per l'italiano e altre lingue che usano caratteri in codifica UTF-8, Flex non era adatto e pertanto si è fatto uso di Quex (http://quex.sourceforge.net/). Lo strumento era tuttavia in una fase di sviluppo preliminare, pertanto si è dovuti

intervenire a correggere diversi errori nel trattamento di UTF-8. Tali correzioni sono state riportate nella più recente versione disponibile su Sourceforge.

Sia Flex che Quex sono generatori automatici di scanner, cioè sono programmi che generano codice C++ per effettuare l'analisi lessicale di testi, basandosi su delle regole di trasformazione descritte associando ad espressioni regolari degli snippet di codice che effettuano le relative trtasformazioni.

La seguente coppia, ad esempio, indica che quando si identifica un tag XML, sia in apertura che in chiusura, va restituito un token costituito dal tag stesso:

```
<\/?[A-Za-z!][^>]*> => TKN_XML(Lexeme);
```

Attualmente il tokenizer permette di riconoscere i seguenti lessemi (oltre naturalmente alle semplici parole):

1. Tag XML
2. Numeri (decimali e non, con . o , come separatore)
3. Parole con elisione (all')
4. Date
5. Valute (dollari, sterline, euro)
6. URL
7. Espressioni aritmetiche
8. Numeri di telefono
9. Puntini di sospensione
10. Parentesi aperte/chiuse
11. Sequenze contenenti i caratteri '*@-'

Le doppie single-quote (') vengono sostituite con " per dare uniformità al testo. I segni di punteggiatura vengono staccati dalla parola che li precede e lasciati su di una riga a sè stante.

Tra i numerosi problemi affrontati vale la pena di accennare alla gestione dei single-quote; questo simbolo viene infatti utilizzato con almeno tre significati:

1. semplice virgoletta
2. apostrofo
3. accento (nel caso sia stata utilizzata una tastiera con mappatura EN/US)

Non è possibile disambiguare tra questi casi facendo uso solamente di un sistema a regole, essendo la Wikipedia, per sua stessa natura, un testo non vincolato da regole di stile e formattazione

Segue un semplice esempio di tokenizzazione:

Input:

```
Mi chiamo Francesco. Sono nato a Pisa il 4/10, nell'anno 1980.
3a*2b+3c=56ab. <tag> TagTest </tag> ... ..
```

Output:

```
Mi
chiamo
Francesco
.
Sono
nato
```

```
a
Pisa
il
4
/
10
,
nell'
anno
1980
.
3a
*
2b
+
3c
=
56ab
.
<tag>
TagTest
</tag>
...
.
.
```

### 2.7.3 Part of Speech Tagging

I tag di Part-of-speech (POS) forniscono informazioni essenziali per i successivi livelli di analisi linguistica, quali il *chunking* (segmentazione in frasi nominali, preposizionali o verbali) e il parsing a dipendenze.

Per assegnare a testi italiani in modo automatico tag di POS secondo le specifiche Tanl, è stato necessario sviluppare un opportuno tagger.

Si è partiti da uno strumento (TreeTagger) messoci a disposizione da Helmut Schmid, University of Stuttgart. TreeTagger è un tagger ad apprendimento statistico, basato sulla tecnica dei *decision tree*, che quindi ha bisogno di un corpus di apprendimento e di un lessico della lingua. Per poterlo adattare ai nostri scopi, è stato necessario:

1. migliorarne le prestazioni, in particolare modificando le modalità di gestione della memoria, ottenendo una velocità di oltre 5.000 token/sec.
2. modificarlo per potere trattare testi in codifica UTF-8 (in particolare le lettere italiane accentate) e preservare la suddivisione in frasi separate da righe bianche.

Le modifiche al codice sorgente di TreeTagger sono state trasmesse a Schmid, che le ha incorporate nella più recente versione del tagger.

Il tagger così ottenuto è stato allenato utilizzando il lessico italiano e i due corpora descritti nella sezione 2.2. Si sono così prodotte due versioni del POS tagger per la lingua italiana. Quello allenato sul corpus Repubblica è in grado di assegnare tag di POS in formato Tanl con un’accuratezza di circa 95 %.

# 3. Infrastruttura di calcolo

L'esecuzione della pipeline completa sull'intera collezione di articoli della Wikipedia è un processo lungo e oneroso dal punto di vista computazionale.

Per ridurre i tempi di elaborazione è stata creata un'infrastruttura per il calcolo distribuito basata su *Hadoop* (http://hadoop.apache.org/), uno strumento di elaborazione parallela basato sulla metafora *Map*/*Reduce*.

Per poter sfruttare il Map/Reduce occorre un suddividere l'input in più parti che vengono inviati ai singoli nodi per l'elaborazione in parallelo. Nel caso dei testi di Wikipedia questo richiede di spezzare il testo complessivo in parti che contengono articoli interi della Wikipedia.

Allo stato attuale le fasi di PosTagging e Tokenizzazione possono essere eseguite su di un cluster costituito da alcune macchine presenti al Dipartimento di Informatica in maniera completamente trasparente all'utente.

# 4. Risultati

Il progetto prevedeva i seguenti obiettivi da raggiungere al termine del primo anno:

1. corpus di apprendimento per il parsing dell'italiano
2. prototipo di parser per l'italiano
3. partecipazione alla competizione CoNLL 2007

Tutti gli obiettivi sono stati raggiunti, in particolare:

1. è stata prodotta una versione annotata in formato CoNLL del corpus ISST
2. è stato realizzato il parser a dipendenze DeSR, che è stato allenato sul corpus ISST. Il parser è stato utilizzato nella nostra partecipazione alla competizione Evalita 2007, risultando il primo tra i parser statistici per l'italiano.
3. il nostro team ha partecipato alla competizione CoNLL 2008, in collaborazione con Yahoo! Research Barcelona. La competizione riguardava il *Joint Parsing of Syntactic and Semantic Dependencies*. Il nostro team ha avuto il ruolo principale di svolgere il parsing delle dipendenze sintattiche, utilizzando una versione di DeSR opportunamente allenata. Il risultato è stato un eccellente terzo posto, su 25 sottomissioni da tutto il mondo.

I prodotti del progetto sono di quattro tipi:

1. specifiche linguistiche
2. risorse linguistiche
3. software
4. pubblicazioni

## *4.1 Deliverables*

1. Specifiche annotazione Tanl
2. Corpus ISST in formato Tanl
3. parser DeSR: disponibile come Open Source Software su SourceForge, la più grande collezione di software OS: http://desr.sourceforge.net/
4. DGAnnotator, strumento visivo di annotazione: http://medialab.di.unipi.it/Project/QA/Parser/DgAnnotator/
5. Wikipedia extractor (http://medialab.di.unipi.it/wiki/index.php/Wikipedia_Extractor)
6. Sentence splitter (http://medialab.di.unipi.it/wiki/index.php/Sentence_Splitter)

7. POS tagger Tanl (http://medialab.di.unipi.it/wiki/index.php/POS_Tagger)

Tutti i deliverable sono disponibili sul sito del progetto:

http://medialab.di.unipi.it/wiki/index.php/Analisi_di_testi_per_il_Semantic_Web_e_il_Question_Answering

### *4.2 Pubblicazioni*


[1] G. Attardi, M. Simi. 2007. DeSR at the Evalita Dependency Parsing Task *Proc. of Workshop Evalita 2007. Intelligenza Aritificiale*, 4(2).

[2] M. Ciaramita, G. Attardi, F. Dell'Orletta and M. Surdeanu. 2008. DeSRL: A Linear-Time Semantic Role Labeling System. *Proceedings the Twelfth Conference on Natural Language Learning*, Manchester.

[3] H. Zaragoza, J. Atserias, M. Ciaramita, and G. Attardi. 2008. Semantically annotated snapshot of the English Wikipedia, *Proceedings of LREC 2008*, Marrakech.

[4] G. Attardi, F. Dell'Orletta. 2008. Chunking and Dependency Parsing. *Proceedings of LREC 2008 Workshop on Partial Parsing*, Marrakech.

[5] C. Bosco, A. Mazzei, V. Lombardo, G. Attardi, A. Corazza, A. Lavelli, L. Lesmo, G. Satta, M. Simi. 2008. Comparing Italian parsers on a common treebank: the Evalita experience *Proceedings of LREC 2008 Workshop on Partial Parsing*, Marrakech.

[6] G. Attardi, F. Dell'Orletta. 2008. Reverse Revision and Tree Combination for Dependency Parsing. *submitted*.


# 5. Spese

### *5.1 Borse di studio*

Sono state erogate borse di studio ai seguenti ricercatori:

1. Felice Dell'Orletta
2. Antonio Fuschetto
3. Francesco Tamberi
4. Eva Maria Vecchi

# Appendice A

# 1. Specifiche di annotazione lessicale e morfologica

## *1.1 Tagset morfo-sintattico*

Abbiamo assunto come punto di partenza il tagset morfo-sintattico di ISST, a sua volta basato sul tagset di ILC/PAROLE che soddisfa il requisito di conformità agli standard internazionali esistenti essendo conforme alle raccomandazioni di EAGLES. Tale tagset include 28 categorie, corrispondenti a 13 categorie morfo-sinttatiche di base (Nomi, Verbi, Aggettivi, Pronomi, Determinatori, Articoli, Avverbi, Preposizioni, Congiunzioni, Numerali, Interiezioni, Punteggiatura, Abbreviazioni alle quali si aggiunge una classe residua per i casi non riconducibili alle categorie previste).

Le revisioni hanno riguardato le seguenti categorie: verbi, punteggiatura, avverbi, congiunzioni, preposizioni e pronomi.

**Verbi**: si è passati da un'etichetta generica di verbo (V), a un'annotazione più dettagliata in cui si marca esplicitamente la funzione del verbo. La partizione adottata è la seguente:

- VA: tutte le occorrenze di ausiliari (*essere*, *avere* o *venire* nelle costruzioni passive);
- VM: tutte le occorrenze di verbi modali (*volere*, *potere*, *dovere*, *solere*);
- V: i verbi principali.

**Punteggiatura**: si è passati da un'etichetta generica di segno di interpunzione (marcato come PU, ora F) a un'annotazione più dettagliata in cui distinguono i seguenti casi:

- FS: punteggiatura "forte", ovvero di fine frase: . [ ! ? ];
- FC: punteggiatura "semi-forte", ovvero di fine clausola: [ ; : ];
- FE: punteggiatura "debole", ovvero le virgole: [ , ];
- FB: punteggiatura "bilanciata", ovvero: [ ( ) " " ' ' - - ].

**Avverbi**: all'interno della classe degli avverbi (marcati come B) viene esplicitamente distinta la classe degli avverbi di negazione etichettata come BN.

**Congiunzioni**: si è passati da un'etichetta generica di congiunzione (marcata come C) a un'annotazione più dettagliata in cui vengono distinte:

- le congiunzioni coordinanti, marcate come CC (es. *i libri **e** i quaderni, vengo **ma** non rimango*);
- le congiunzioni subordinanti, contrassegnate da CS (es. ***quando** ho finito vengo*, ***mentre** scrivevo ho finito l'inchiostro*).

**Preposizioni**: si è passati da un'etichetta generica di preposizione (marcata come E) a un'annotazione più dettagliata in cui vengono distinte le preposizioni semplici (marcate come E) da quelle articolate (contrassegnate da EA).

**Pronomi**: alle **sottoclassi** dei pronomi previste dal tagset ISST (ovvero pronome dimostrativo PD, pronome indefinito PI, pronome possessivo PP, pronome personale PE, pronome relativo PR e pronome interrogativo PQ) è stata aggiunta la sottoclasse dei pronomi clitici etichettati come "PC". Tale inserimento trova motivazione nel fatto che questa classe di pronomi è caratterizzata da un comportamento distribuzionale peculiare che la contraddistingue rispetto a tutte le altri classi di pronomi.

È stata inoltre aggiunta una nuova categoria per l'etichettatura dei Predeterminatori (T) in contesti del tipo *tutti gli studenti* dove *tutti* viene etichettato come T piuttosto che come DI, la cui combinazione con un articolo appare alquanto controversa.

Per tutti gli elementi del tagset è stata definita una versione "morfologizzata" in cui la categoria morfo-sintattica è combinata con tratti del tipo persona, numero, genere: es. Tfs (*tutta*, predeterminatore femminile singolare) vs Tms (*tutto*, predeterminatore maschile singolare) o VAip1s (*ho*, verbo ausiliare indicativo presente prima persona singolare).

### 1.1.1 Tagset delle relazioni di dipendenza

Anche nel caso del tagset sintattico a dipendenze è stato assunto come punto di partenza il tagset del livello di annotazione sintattico-funzionale di ISST, a sua volta basato su F.A.M.E (Functional Annotation Meta-scheme for Evaluation), lo schema elaborato nel progetto europeo ELSE (Lenci *et al.* 1999, 2000) ai fini della sua adozione nell'ambito di campagne di valutazione di analizzatori sintattici. Tale schema è stato rivisto e specializzato rispetto alle peculiarità della lingua italiana e alla sua adozione come schema di annotazione di una Treebank.

Nell'ambito del presente progetto lo schema di annotazione sintattico-funzionale di ISST è stato rivisto alla luce dello stato dell'arte nella costruzione di corpora di addestramento e di valutazione. In particolare, si è deciso di aderire allo standard di rappresentazione a dipendenze CoNLL. Vale la pena evisenziare qui il fatto che non si è trattato di una semplice conversione di formato. La difficoltà principale era legata al fatto che mentre in ISST gli elementi di base dello schema di annotazione sono costituiti da relazioni funzionali espresse in termini di relazioni binarie sussistenti tra due partecipanti sempre corrispondenti a parole piene o lessicali (tipicamente, nomi, verbi, aggettivi e avverbi) e le relazioni funzionali riguardanti parole grammaticali (quali i determinativi, gli ausiliari, i complementatori, le preposizioni, etc.) sono codificate mediante tratti associati ai partecipanti della relazione, nello schema CoNLL l'annotazione delle relazioni di dipendenza non è circoscritta alle parole piene (sostantivi, aggettivi e verbi) ma riguarda tutti i token del testo, compresa la punteggiatura. Inoltre, mentre ISST prevedeva la presenza nell'annotazione di elementi "vuoti", che non trovavano una controparte al livello dei token del testo (tipicamente usati per la rappresentazione dei soggetti "ellittici"), nello schema CoNLL esiste il vincolo che le relazioni di dipendenza riguardano soltanto token espliciti del testo.

Per allinearsi a questo standard di annotazione, il tagset delle relazioni di dipendenza ha dovuto essere esteso per poter trattare relazioni riguardanti parole grammaticali. Segue la tipologia di relazioni che sono state aggiunte. Si noti che negli esempi che seguono il grassetto marca i partecipanti alla relazione esemplificata la cui testa è contrassegnata dal corsivo.

1. **aux** (ausiliare): la relazione intercorrente tra una testa verbale e il suo ausiliare. Ad esempio:
   Il corazziere è **stato *individuato***
   Il corazziere **è *stato*** individuato
2. **clit** (clitico): la relazione intercorrente tra un pronome clitico e una testa verbale usata nella forma pronominale. Ad esempio:
   La sedia **si** è ***rotta***
3. **cong_sub** (congiunzione subordinante): la relazione che lega la congiunzione subordinante alla testa verbale del complemento frasale. Ad esempio:
   Ha detto ***che*** non **intendeva** fare nulla
   Le autorità hanno annunciato ***che*** il blitz è **concluso**
   Venne ucciso ***mentre*** **cercava** di difendere la ragazza
4. **det** (determinante): la relazione intercorrente tra una testa nominale e il suo determinante. Ad esempio:
   **Una *sala*** ha dovuto essere sgomberata
   Rilevata **la *presenza*** di gas
5. **mod_rel** (modificatore relativo): la relazione intercorrente tra la testa verbale di una frase relativa e il suo antecedente. Lo stesso tipo di relazione è usato nel caso delle relative libere per collegare la testa verbale della relativa al pronome "chi". Ad esempio:
   ***Box*** che è stato **trovato** nel pomeriggio
   Gli utilizzatori del ***box*** dove sarebbe **avvenuta** la violenza
   Non è mai stato accertato ***chi*** **volle** la sua morte
6. **modal** (verbo modale): la relazione intercorrente tra una testa verbale e un verbo modale. Ad esempio:
   Una sala ha **dovuto** essere ***sgomberata***
7. **prep** (preposizione): la relazione che intercorre tra una testa preposizionale e il suo complemento, sia esso frasale o meno. Ad esempio:
   Un contributo ***alla*** **lotta** contro la criminalità
   Un contributo alla lotta ***contro*** la **criminalità**
   Hanno un modo ***di*** **ragionare** rozzo
   ***Prima_di*** **partire** ho telefonato
8. **con** (congiunzione coordinante copulativa): la relazione che intercorre tra una congiunzione coordinante copulativa (sia essa costituita da "e" oppure da una virgola) e il primo congiunto della struttura coordinata, che rappresenta la testa dell'intera struttura. Ad esempio:
   Una ragazza ***violentata*** **e** sequestrata da due slavi
   ***Gabriella*** **e** Paolo sono partiti
   ***Scontri*** **,** assalti e centinaia di feriti
   ***Scontri*** , assalti **e** centinaia di feriti
9. **dis** (congiunzione coordinante disgiuntiva): la relazione che intercorre tra una congiunzione coordinante disgiuntiva (tipicamente "o") e il primo congiunto della struttura coordinata, che rappresenta la testa dell'intera struttura. Ad esempio:
   Cassonetti dell'immondizia ***rovesciati*** **o** incendiati
   Partecipa ***a*** manifestazioni politiche **o** a dibattiti
10. **punc** (interpunzione): la relazione che intercorre tra un segno di interpunzione e un token del testo. Ad esempio:
    ***Teatro*** della tragedia **,** ...
    Una polemica è ***scoppiata*.**

Inoltre, sulla base di analisi accurate del corpus annotato di ISST nell'ambito di studi linguistico-computazionali, lo schema di annotazione originario è stato rivisto come segue:

1. è stata neutralizzata la distinzione tra modificatori e argomenti non sempre applicabile in modo univoco e non controverso, ricorrendo a una relazione sottospecificata che le copre entrambe, che può essere poi ulteriormente arricchita da specificazioni di tipo semantico; come conseguenza di questo punto è stata circoscritta la corpertura della relazione di tipo modificazione;
2. è stato esplicitamente marcato il soggetto in costruzioni passive, creando i presupposti per una rappresentazione semantica della frase.

Per quanto riguarda 1):

- **comp** (complemento) è andato a designare la relazione tra una testa e un complemento, sia esso modificatore o argomento. Secondo questa accezione più larga, la relazione **comp** va anche a coprire il caso di **obl** (complemento obliquo), ovvero la relazione tra un predicato e un complemento non frasale, non diretto e non indiretto, realizzato tipicamente in forma di sintagma preposizionale, il caso di **ogg_i** (oggetto indiretto), che indica la relazione tra un predicato e un oggetto indiretto, cioè il complemento che esprime l'entità che accoglie l'azione espressa dal verbo, e anche alcune relazioni di tipo **mod**. Seguono alcuni esempi:
  - Si è ***trasformato*** **in** gas
  - Era ***uscito*** **di** casa alle 10
  - Sono ***volati*** **nel** burrone
  - **Al** magistrato ***ripetevano***
  - **Mi** ha ***elargito*** uno sguardo
  - Un ***contributo*** **alla** lotta contro la criminalità
  - Fu ***assassinata*** **da** un pazzo
  - Il padre è stato ***ucciso*** **dai** banditi
  - E' più ***interessante*** **del** libro
  - Si ***valorizzano*** **nella** luce mediterranea
  - La ***produzione*** **di** Vietri
  - ***Trionfo*** di Didoni **nei** 20 km di marcia
  - Trionfo di Didoni nei 20 ***km*** **di** marcia
- **mod** (modificatore): in conseguenza del cambio di cui sopra, la copertura di questa relazione è stata circoscritta alla relazione sussistente tra una testa e modificatori di tipo aggettivale, avverbiale e frasale. Ad esempio:
  - I colori ***sono*** **sempre** gli stessi
  - ***Colori*** **intensi**
  - Trionfo di Didoni nei **20** ***km*** di marcia
  - **Quando** urla non mi ***piace***
  - ***Lavoravano*** come volontari **per** costruire una centrale

Riguardo alle revisioni di cui al punto 2), la relazione "sogg" viene assegnata al soggetto superficiale della testa verbale usata con diatesi attiva, mentre viene usato "sogg-pass" per marcare il soggetto superficiale in costruzioni passive. Ad esempio:

SOGG:

- il **testimone** ha ***parlato*** subito
- le **vittime** ***seguivano*** gli aiuti
- ***è*** opportuno **dire** due parole
- ***è*** difficile **che** la gente se ne vada presto

- ***sarà*** difficile **negare**

SOGG-pass

- i **missionari** erano stati ***rapiti*** la mattina presto
- circa 83.000 **franchi** furono ***spesi***

Inoltre, per le relazioni PRED(icativa), MOD(ificatore) e COMP(lemento) sono state introdotte distinzioni semantiche riguardanti il ruolo del complemento all'interno della frase, ovvero "loc(ativo)", "temp(orale)" o "ind(iretto)" (quest'ultima specificazione si combina solo con relazioni di tipo COMP). Seguono alcune esemplificazioni:

- Ho ***dato*** il libro **a** Luigi (comp-ind)
- Si ***trovava*** **in** un parco (comp-loc)
- **Nel** 1985 nella stessa zona è stata ***uccisa*** un'antropologa (comp-temp)
- L'allarme è ***scattato*** la scorsa **settimana** (mod-temp)
- ***Traffico*** intenso **ovunque** (mod-loc)