QuARS and the Natural Language Processing techniques: looking for a suitable syntax parser

Daniela Anastasio (anastas@cli.di.unipi.it**)**

Stefania Gnesi (stefania.gnesi@isti.cnr.it)

Giuseppe Lami (giuseppe.lami@isti.cnr.it)

Gianluca Trentanni (gianluca.trentanni@isti.cnr.it)

Consiglio Nazionale delle Ricerche - Istituto di Scienza e Tecnologie dell Informazione A. Faedo -System and Software Evaluation Center - Via Moruzzi 1, 56124 Pisa, Italy

Abstract. *This paper describes the QuARS (Quality Analyzer of Requirements Specifications) tool by means of its internal syntax parser (namely Minipar). Looking for a more expressive syntax parser, a little review will be presented and a proposal of integration will be suggested.*

Indice

Introduzione

QuARS (**Quality Analyzer of Requirements Specifications**) è un tool sviluppato presso l'ISTI - CNR di Pisa a partire dal 2000, per l'analisi e la valutazione della qualità di specifiche di requisiti software scritte in linguaggio naturale.
L'utilizzo del linguaggio naturale è in pratica il metodo più comune per la produzione dei documenti di Specifica dei Requisiti Software (SRS) perché è di facile comprensione e di immediata condivisione tra tutte le persone coinvolte nel processo di sviluppo software, dal progettista agli stakeholders. Per contro però l'intrinseca ambiguità del linguaggio naturale può condurre a definizioni incomplete o inconsistenti con l'introduzione di contraddizioni semantiche o mancanza di informazioni necessarie.
Lo scopo di QuARS è quello di ridurre le ambiguità che vengono introdotte nei documenti di requisiti scritti in Linguaggio Naturale. Esso individua i termini che possono essere causa di ambiguità e fornisce delle misure secondo degli indici di qualità per verificare l'accuratezza del documento.
Il fine del presente lavoro consiste nel miglioramento di alcune funzionalità di QuARS relative agli algoritmi di estrazione dell'informazione.
A tale proposito verrà effettuata un'analisi comparativa fra vari strumenti di analisi sintattica e lessicale al fine di determinare quello, o quelli, più adatti per una possibile integrazione all'interno di QuARS.

Capitolo 1

1. Processo di Sviluppo del Software

L' **Ingegneria del Software** è la disciplina che regola lo sviluppo di **prodotti** software di buona qualità, definendo le attività necessarie a gestire, organizzare e a portare a buon fine la **realizzazione**, **manutenzione** ed **evoluzione** di un sistema software di grandi dimensioni (**processo** di sviluppo software), è un approccio sistematico allo sviluppo, rilascio, manutenzione e ritiro di un prodotto software.
Il processo comprende tutte le attività che portano allo sviluppo di un certo prodotto, le cui fasi principali sono:

- **Specifica del software.**
- **Sviluppo del software.**
- **Validazione del software**
- **Evoluzione del software**

Prima di iniziare il processo si devono operare delle scelte (quali strumenti utilizzare, quale linguaggio di programmazione, etc…) e si devono individuare i requisiti necessari al prodotto. Per identificare i requisiti è importante riferirsi alle funzioni che il prodotto dovrà svolgere e alle caratteristiche che dovrà avere.
Il committente del prodotto quantifica e qualifica i suoi desideri circa il pacchetto software da realizzare. Da queste richieste nascono le caratteristiche (le funzionalità che deve sostenere, le possibilità da prevedere, …) e i requisiti (su quale hardware deve funzionare, in quali situazioni particolari deve essere usato, …) che il prodotto deve avere.

1.1 Ingegneria dei requisiti

L'I**ngegneria dei requisiti** è stata definita, come il lavoro necessario alla stesura di un documento di specifica dei requisiti il più possibile completo e preciso. I requisiti permettono di valutare il lavoro svolto: servono all'inizio per definire i termini di quello che deve essere costruito, durante lo sviluppo per le modifiche in corsa e alla fine per valutare la qualità di quello che è stato realizzato.
Nella fase dell'ingegnerizzazione, i requisiti devono essere definiti e individuati. Tali requisiti nascono dalle relazioni tra gli stakeholders, quest'ultimi sono tutti quegli attori che ruotano intorno al sistema software, dagli sviluppatori agli utenti finali, da coloro che commissionano il prodotto a coloro che lo progettano. L'interazione tra le parti avviene utilizzando il Linguaggio Naturale.
Prima di giungere alla stesura vera e propria del Documento dei Requisiti il processo attraversa altre frasi fondamentali:

- Studio di Fattibilità
- Analisi dei Requisiti
- Specifica dei Requisiti

1.1.1 Studio di Fattibilità

In questa fase si fa particolare uso del Linguaggio Naturale, poiché si cerca di definire il problema vero e proprio, attraverso l'interazione tra il committente ed il produttore.
E' una fase molto importante, poiché un mancato approfondimento potrebbe portare a dei problemi nella fase di sviluppo o di rilascio del prodotto software.

1.1.2 Analisi dei Requisiti

L'analisi dei requisiti è un'attività che ha come obiettivi la comprensione del dominio all'interno del quale deve essere sviluppato un sistema software e la definizione dei requisiti che tale sistema deve soddisfare. Senza un'adeguata comprensione del dominio è difficile identificare le interazioni che il sistema software avrà nello svolgimento delle sue funzionalità e senza una precisa definizione dei requisiti è impossibile effettuare correttamente la specifica e la successiva progettazione e realizzazione del sistema.
Ci sono numerosi metodi per analizzare il dominio e per definire i requisiti di un sistema software. Tutti hanno in comune la costruzione di un glossario (per caratterizzare il dominio) e la redazione di una lista di requisiti utente (funzionali, ovvero relativi alle funzionalità che il software deve garantire, strutturali o di qualità se relativi ad aspetti non funzionali del sistema).
Un metodo usato molto frequentemente è quello che si basa su un documento iniziale, chiamato "enunciato del problema", dal quale si estrae sistematicamente la conoscenza del dominio. Tale conoscenza è usata per costruire il "vocabolario di dominio", un documento che definisce univocamente tutti i termini (significativi) presenti nell'enunciato del problema, dove per significativi si intendono quei termini che hanno un particolare significato nello specifico contesto.
L'enunciato del problema è poi filtrato, eliminando le parti che sono andate a costituire il vocabolario di dominio. Il materiale residuo è ulteriormente analizzato, identificando tutti i termini relativi ai dati scambiati tra il dominio e il sistema software. Anche in questo caso si costruisce un vocabolario, detto "vocabolario dei dati", che comprende le definizioni dei termini identificati.
Il materiale residuo è ulteriormente filtrato, eliminando le parti relative al vocabolario dei dati. Ciò che rimane dopo questa operazione di filtraggio, opportunamente riadattato, costituisce la lista dei requisiti del sistema software, documento iniziale per la fase di specifica.

1.1.3 Specifica dei Requisiti

Definito il dominio di sviluppo del prodotto, si definiscono in modo preciso le caratteristiche del sistema e i suoi requisiti funzionali, cioè quello che il sistema deve offrire all'utente finale in termini di funzionalità.
Il documento prodotto in questa fase deve avere delle caratteristiche ben precise affinché la fase successiva possa essere affrontata correttamente. Il **Documento per la Specifica dei Requisiti (DSR)** deve essere:
- preciso
- completo
- consistente

Un requisito è una scrittura più o meno formale e rigorosa di una caratteristica del sistema, fatta dallo specialista. Nascono per questo motivo dei linguaggi per la specifica dei requisiti, più o meno formali.

La Specifica dei Requisiti in generale è un documento strutturato in cui è descritto il sistema, o meglio le caratteristiche funzionali.
Il suo scopo è quello di comunicare a tutti gli stakeholders i requisiti.
Dopo la stesura del DSR, prima che questo diventi operativo e venga utilizzato, deve essere validato. Occorre, cioè, verificare la sua idoneità come documento di base per lo sviluppo del sistema, ed eliminare gli eventuali difetti. Sono da risolvere:
- **inconsistenze**: Si hanno quando un requisito è in contraddizione con un altro.
- **ambiguità**: Si hanno quando un requisito si presta ad interpretazioni diverse; per cui a secondo di chi legge il DSR si potrebbero dare interpretazioni diverse e realizzazioni diverse del sistema.
- **incompletezze**: Si hanno quando nel DSR mancano le informazioni sufficienti a caratterizzare certe parti o vincoli del sistema.
- **imprecisioni**: E' un errore di terminologia, indica l'uso di termini in contraddizione con il glossario.

Oltre a quelli appena detti, si possono effettuare altri controlli sul DSR per definire altre caratteristiche, che non riguardano la correttezza o le funzionalità del documento ma ne migliorano la qualità e la usabilità:

- ***Verificabilità***: I requisiti così come sono espressi sono realmente verificabili?
- ***Comprensibilità*****:** I requisiti sono stati correttamente compresi da tutti gli stakeholders?
- ***Tracciabilità*****:** L'origine del requisito è definita in modo chiaro?
- ***Adattabilità***: Il requisito è facilmente modificabile?

Il DSR è un documento scritto in linguaggio naturale, quindi non sempre esaminabile in modo rigoroso ed esaustivo.
Questo è dovuto al fatto che non esiste una semantica associata al linguaggio naturale,

anzi questo si dice essere inerentemente ambiguo (semantica=dare un significato univoco al linguaggio).

L'utilizzo del linguaggio naturale è in pratica il metodo più comune per la produzione di documenti di Specifica dei requisiti Software(SRS) perché è di facile comprensione e di immediata condivisione tra tutte le persone coinvolte nel processo di sviluppo software, dal progettista agli stakeholders. Per contro però l'intrinseca ambiguità del linguaggio naturale può condurre a definizioni incomplete o inconsistenti con l'introduzione di contraddizioni semantiche o mancanza di informazioni necessarie.
Per risolvere questo problema si preferisce, prima di passare all'implementazione di un sistema, esprimere i requisiti funzionali di questo presentati nel DSR mediante linguaggi semiformali o formali.

1.2 NLP: Trattamento Automatico del Linguaggio Naturale

Il Trattamento Automatico del Linguaggio Naturale (noto anche come Natural Language Processing, NLP) è un ramo dell'informatica che studia i sistemi automatici per l'elaborazione del linguaggio naturale. Include lo sviluppo di algoritmi per il parsing, la generazione e l'acquisizione di conoscenza linguistica.
Storicamente la branca principale del NLP si è rivolta all'analisi, al riconoscimento e all'interpretazione del linguaggio naturale, svolto su tre diversi livelli:
• **lessicale**: analisi della struttura dei termini;
• **sintattico**: analisi della struttura delle frasi;
• **semantico**: analisi del significato a livello di frase o dell'intero documento.
Alcune applicazioni del NLP possono essere:
• traduzione automatica di testi;
• analisi grammaticale e ortografica;
• classificazione di documenti;
• riconoscimento automatico del parlato (speech recognition);
• sintesi vocale;
• question-answering;
• recupero delle informazioni (information extraction);
• riconoscimento di testi cartacei (OCR, Optical Character Recognition);
• sistemi di supporto alle decisioni e di data mining;

L'uso di tecniche di NLP è utile per cercare di ridurre l'ambiguità con cui vengono scritti i requisiti, rappresenta un importante tema di ricerca.
Esistono vari tipi di ambiguità che si possono trovare in una frase scritta in linguaggio naturale, per esempio:

- **Ambiguità Lessicale:** ovvero quando una parola ha più di un significato: *We found a bar along the way*"; in questo caso l'ambiguità è dovuta al doppio significato di *bar*: "bar"o "ostacolo."

- **Ambiguità Sintattica:** rappresenta un'ambiguità a livello di regole grammaticali: *"Tomorrow we will listen to the lesson of Shakespeare",* dove "of Shakespeare" è complemento di specificazione allora significa che un certo Shakespeare ci fa lezione, se "of Shakespeare" è complemento di argomento significa che ascolteremo la lezione sulla morale che Shakespeare ci ha impartito nell'Otello.

- **Ambiguità Semantica:** può essere generata da ambiguità sintattiche o lessicali.

L'eliminazione delle ambiguità è un problema di diagnosi: si ha un modello del proprio dominio di applicazione e si vuole interpretare un enunciato sulla base delle nostre conoscenze, per poter chiarire il significato.
Ci sono due approcci a questo problema:

1. **Tecniche di Natural Language Processing:** cercano di analizzare la struttura della frase, per ridurre le ambiguità usando delle basi di conoscenza e dei parser per il linguaggio naturale. Queste cercano di investigare profondamente nella struttura dell'enunciato e cercano di estrarre il significato.

2. **Tecniche di Analisi Lessicale:** Compiono questo tipo di analisi del documento, identificano i termini che sono portatori di ambiguità. Compiuta l'analisi si cerca di correggere la frase, da un punto di vista lessicale, sostituendo i termini ambigui. Chiaramente queste tecniche non operano a livello semantico, presumono che la riduzione dell'ambiguità semantica sia una proprietà "emergente" data dall'analisi lessicale.

La disponibilità oggi di software capace di analizzare documenti in linguaggio naturale offre un nuovo modo per ricercare, esplorare e classificare le informazioni. Tra questi, gli strumenti di *Linguistic Analysis (LA)* consentono l'individuazione degli elementi chiave di un documento combinando analisi morfologica, sintattica e semantica.
Quindi un sistema NLP deve essere capace di eseguire un processo di disambiguazione poiché le parole possono assumere significati differenti a seconda del contesto. Le parole dunque devono essere interpretate e diversificate nel significato.

1.3 QuARS

Il linguaggio naturale (NL) continua ad essere il metodo più comune di esprimere requisiti software, poiché nonostante la sua ambiguità ha il vantaggio di essere facilmente comprensibile da tutti.
L'ambiguità dei requisiti scritti in NL può essere ridotta con l'utilizzo di due tecniche:

1. La restrizione del grado di libertà nello scrivere i requisiti, utilizzando per esempio una lingua inglese limitata che evita condizioni ambigue.
2. L'analisi del testo in linguaggio naturale per identificare e correggere le ambiguità presenti.

Non esistono molte tecniche per l'analisi dei requisiti in NL e solo alcune sono supportate dall'utilizzo di strumenti automatici, uno tra questi è QuARS.
QuARS è stato sviluppato per automatizzare e supportare l'analisi e la valutazione della qualità di un documento di requisiti in linguaggio naturale, che ha l'obiettivo di ridurre il problema delle ambiguità nella comunicazione dei requisiti. riducendo il problema dell'introduzione delle ambiguità nelle varie fasi della comunicazione dei requisiti.
La qualità di un documento di specifica dei requisiti software, si basa su alcune caratteristiche di qualità dei singoli requisiti e come abbiamo già detto precedentemente, un buon documento di specifica dei requisiti deve essere il più possibile:

- **Non Ambiguo**.
- **Completo.**
- **Comprensibile.**

L'obiettivo di QuARS è quello di analizzare i singoli requisiti che compongono un Documento dei Requisiti al fine di rilevare il più possibile difetti negli stessi dovuti all'interpretazione del Linguaggio Naturale. QuARS per valutare la qualità di un documento di specifica dei requisiti software, utilizza degli Indicatori, che sono presenti nel testo, che esprimono un difetto potenziale.

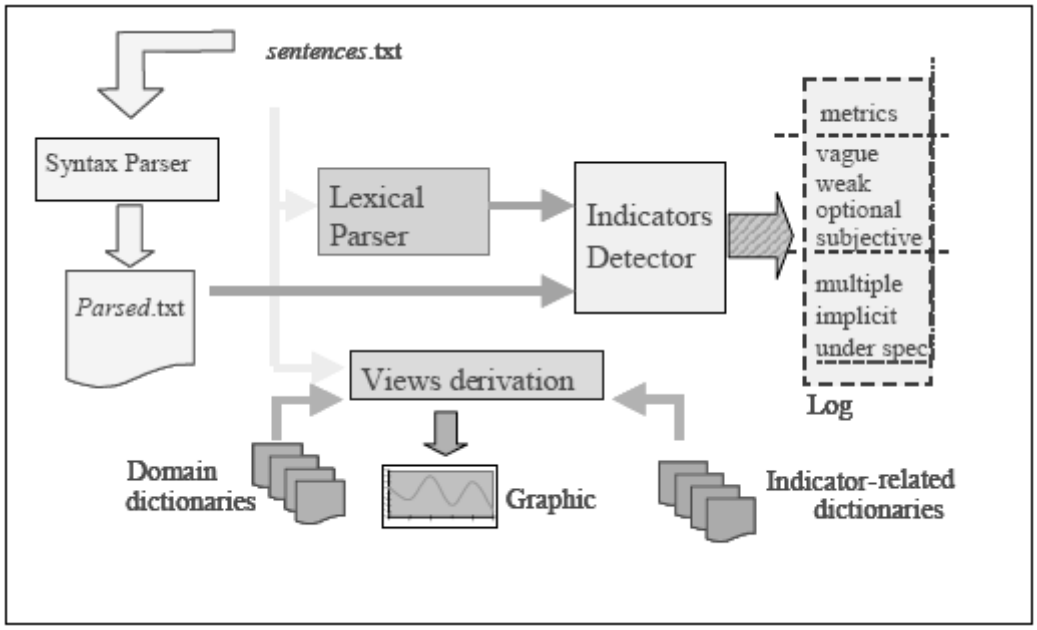


Figura 1.1 Schema dell'Architettura ad Alto Livello di QuARS.

Le principali funzioni di QuARS sono:

● Analisi lessicale delle frasi della specifica dei requisiti.
● Analisi sintattica delle frasi della specifica dei requisiti sulla base di una grammatica e costruzione degli alberi di derivazione.
● Analisi del documento sulla base del modello di qualità descritto prima.

I componenti principali di QuARS sono:

- **Analizzatore Lessicale:** compie un'analisi morfologica delle parole.
- **Analizzatore Sintattico:**
  - sulla base di una grammatica e dell'output dell'analizzatore lessicale compie un parsing della frase per controllarne la correttezza.
  - Costruttore dell'albero di derivazione: un albero di derivazione è un albero che rappresenta tutte le possibili derivazioni per una frase usando regole di un grammaticale.
- **Valutatore delle proprietà di qualità:**
  - **Valutatore degli indicatori di classe A**: gli indicatori di classe A sono quelli che hanno bisogno di avere informazioni sulla sintassi e sulla semantica del documento (tipo frasi non specificate).
  - **Valutatore degli indicatori di classe B**: gli indicatori di classe B sono quelli che non hanno bisogno di avere informazioni sulla sintassi e sulla semantica del documento (tipo le frasi vaghe).

Indicatori	Tecniche Lessicali	Tecniche Sintattiche
Vagueness	x	
Subjectivity	x	
Optionality	x	
Implicity		x
Weakness		x
Under-specification		x
Multiplicity		x
Readability	x	

Figura 1.2 Indicatori di Qualità vs. Tecniche Linguistiche

- **Vagueness (Frasi vaghe)**: una frase è vaga se contiene parole vaghe, tipo:
  - Aggettivi riferiti a caratteristiche intrinseche tipo: clear, well, strong, easy, good, efficient,...
  - Aggettivi riferiti a caratteristiche temporali tipo: old, new, recent, past ...
  - Aggettivi riferiti a caratteristiche ambientali tipo: useful, significant, adequate, fast, slow, ...
  - Aggettivi riferiti a disposizioni spaziali tipo: near, far, back, in front, ...

- **Subjectivity (Frasi soggettive)**: Se contengono opinioni personali o sensazioni: having in mind, take into consideration, similar, better, worse, as [...] as possible, ...

- **Optionality (Frasi opzionali)**:Una frase è opzionale se contiene una opzione, tipo: possibly, eventualy, if appropriate, if needed.

- **Implicity (Frasi con soggetto implicito)** sono le frasi dove il soggetto della frase:
  - Contiene un aggettivo dimostrativo : this, these, that, those.
  - E' espresso tramite un pronome: it, they.
  - E' espresso da una proposizione: above, below,...
  - E' espresso da un aggettivo tipo: previous, next, following, last, first,...

- **Weakness (Frasi deboli)**: contengono un verbo debole tipo: can, could, may, might.

- **Under-Specification (Frasi sotto specificate)**:
  - Se il suo soggetto contiene una parola che identifica una classe di oggetti senza che ci sia un modificatore che specifica un'istanza di quella classe. Per esempio: “The testers shall be indipendent from the software developers” è specificata.

- **Multiplicity (Frasi multiple)**:
  - Se una frase ha più di un soggetto o più di un verbo principale.
  - Se ha più di un complemento che specifica il soggetto in modo diretto o indiretto.

- **Readability (Leggibilità)**: é calcolata in base ad un Indice di leggibilità, che misura come una frase può essere facilmente letta.

- **I dizionari**: ogni indicatore ha un suo dizionario che contiene le parole tipiche per l'analisi. I dizionari sono modificabili.

L'Analisi Sintattica viene fatta con l'utilizzo del parser sintattico Minipar. Questi produce la struttura sintattica di ogni frase presente nel documento dei requisiti, associando ad ogni parola di una frase dei tag (etichette) che indicano il ruolo sintattico di ogni singola parola e altri che indicano le relazioni esistenti tra le parole stesse.
Minipar calcola e fornisce uno dei possibili alberi di derivazione per ogni frase in analisi usando le regole della lingua inglese, ma in realtà una singola frase potrebbe avere più alberi di derivazione, per cui potrebbe essere che il parser abbia fornito l'albero sbagliato, nel caso per esempio di frasi ambigue.
Minipar ha una percentuale di frasi correttamente dedotte molto alta (circa 85%), ma uno dei suoi limiti è proprio quello di non fornire più di un albero di derivazione per una frase ed è proprio per questo che abbiamo deciso, nei capitoli seguenti, di analizzare altri parser che fornissero questa funzione da poter integrare all'interno di Quars, eventualmente in sostituzione di Minipar.
Attualmente il tool QuARS è ancora ad uno stadio prototipale, in quanto non ancora industrializzato, anche se tutte le funzionalità sono pienamente sviluppate.
La piattaforma di sviluppo è, per quanto riguarda il “core”, in linguaggio C++ utilizzando in particolare il framework di programmazione Microsoft Visual Studio.
Le parti di interfaccia grafica, di interfaccia con il parser e di servizio sono state sviluppate nel linguaggio TCL/TK, utilizzando in particolare il framework di sviluppo TCL Development Kit della ActiveState.
Le persone attualmente coinvolte nel suo sviluppo e manutenzione afferiscono ai laboratori dell'ISTI-CNR FMT (Formal Methods and Tools) e SSEC (System and Software Evaluation Center) e sono: Fabrizio Fabbrini, Mario Fusani, Stefania Gnesi, Giuseppe Lami, Gianluca Trentanni.

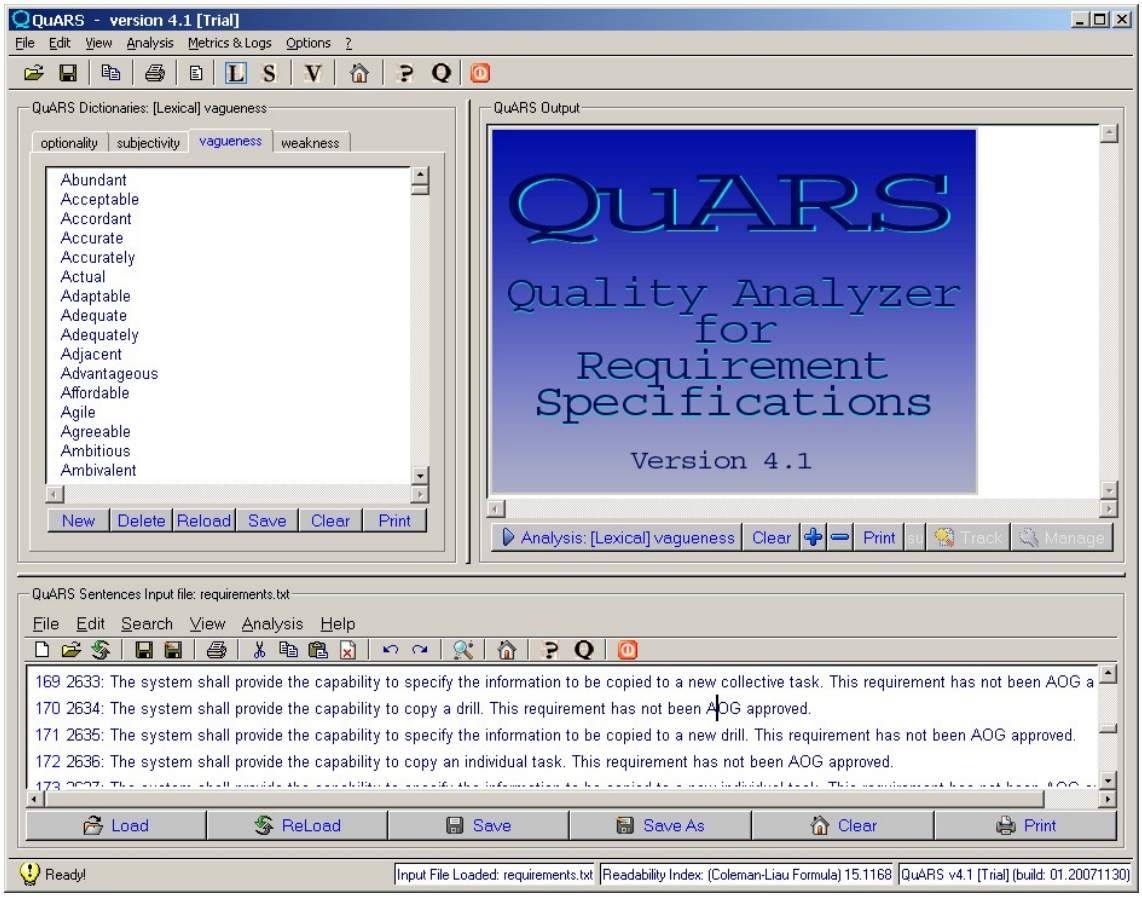


Figura 1.3 Interfaccia di QuARS.

Capitolo 2

Introduzione

In questo capitolo, vedremo nel dettaglio la struttura di alcuni parser e il loro funzionamento, poiché, come già detto nel capitolo precedente, il nostro fine è quello di riuscire a trovare un parser che riesca a superare i limiti di Minipar, quali per esempio, fornire per ogni frase in esame più alberi di derivazione.
Un parser è un programma che riceve una grammatica formale ed uno stream continuo in input (letto per esempio da un file o una tastiera) in modo da determinare la sua struttura grammaticale grazie alla grammatica data, per esempio stabilisce quali gruppi di parole sono unità (frasi) e quali parole sono il soggetto o l'oggetto di un verbo.
Il suo funzionamento è riassunto in Figura 2.1.

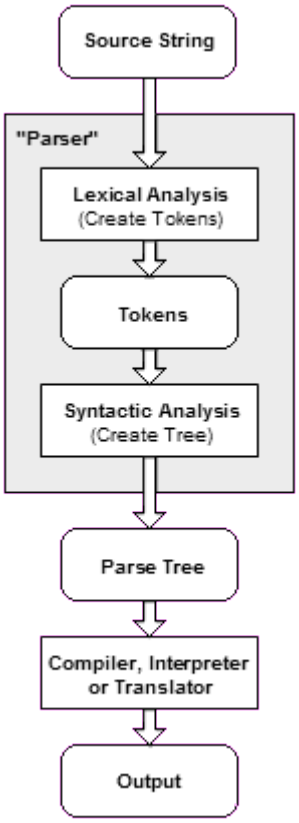


Figura 2.1 Funzionamento di un Parser.

Essenzialmente il lavoro di un parser si divide in 3 fasi:

1. **Analisi Lessicale (Lexical Analysis):** Prende in input una sequenza di caratteri e produce in uscita una sequenza di tokens, i tokens sono gli elementi minimi (non ulteriormente divisibili) di un linguaggio.
2. **Analisi Sintattica (Syntactic Analysis):** Prende in ingresso la sequenza dei tokens generata nella fase precedente e verifica che questi formino un'espressione ammissibile, ciò generalmente viene fatto con riferimento ad una Grammatica Libera dal Contesto, che definisce, in modo ricorsivo, le componenti che possono comporre un'espressione e l'ordine con cui devono comparire. Il risultato di questa fase è un **Albero di Sintassi**.
3. **Analisi Semantica**: Si occupa di assegnare un significato, un senso, alla struttura sintattica e di conseguenza all'espressione linguistica.

Il compito del parser è quello di determinare se e come l'input può essere derivato dal simbolo iniziale con le regole della grammatica formale.

I prodotti scelti per la nostra analisi sono i seguenti:

1. **Minipar**
2. **Link Grammar Parser**
3. **Stanford Parser**
4. **Tree Tagger**

Si dividono in due categorie:

1. **Syntactic parser (Minipar, Link Grammar, Stanford)**
   Con il termine Syntactic parser si intende la classificazione delle parole che compongono una frase in soggetti, verbi e complementi. Viene svolta quella che comunemente è chiamata **Analisi Logica**. Di solito questi strumenti effettuano anche l'analisi grammaticale.

2. **Part-Of-Speech tagger (Tree Tagger)**
   Con il termine Part-Of-Speech tagger si intende la classificazione delle parole che compongono una frase in: verbi, nomi, pronomi, aggettivi, etc.
   Quella che comunemente è chiamata **Analisi Grammaticale**.

2.1 MINIPAR

Minipar è un parser basato sui principi della lingua Inglese, sviluppato da Dekang Lin, professore presso il Dipartimento di Computer Science dell'università di Alberta, implementando alcune idee esposte nel “Minimalist Program”. Minipar rappresenta la grammatica come un network dove i nodi rappresentano le categorie grammaticali e i link rappresentano il tipo di relazione sintattica. Il diagramma della grammatica consiste di 35 nodi e 59 relazioni. In aggiunta, nodi e link, sono generati dinamicamente per rappresentare sotto categorie dei verbi.

Minipar impiega un algoritmo di parsing di grafi distribuiti e ogni nodo nel network della grammatica contiene un grafo contenente una struttura, parzialmente costruita, appartenente alla categoria grammaticale rappresentata dal nodo. Le regole della grammatica sono implementate come vincoli associati ai nodi e ai link. Il dizionario del Minipar è derivato da WordNet e contiene all'incirca 130.000 elementi nella loro forma base. Ogni elemento del dizionario elenca tutte le possibili varianti grammaticali di una parola. Le ambiguità lessicali sono gestite dal parser invece che da un tagger.

Minipar analizza le frasi e produce un albero delle dipendenze che rappresenta le relazioni di dipendenza tra le parole che compongono la frase. Una relazione di dipendenza è una relazione binaria asimmetrica fra una parola, che viene chiamata *head*, e un'altra parola, che viene chiamata *modifier*. Una parola in una frase può avere molti *modifier*, ma ogni parola può essere il *modifier* al massimo per una parola.

Minipar è sviluppato in C++ e viene fornito corredato del codice sorgente e delle API per poter essere integrato all'interno di altri software. Viene fornito anche un eseguibile binario per svariate piattaforme, utilizzato per effettuare le prove. Prende in input un file di testo contente la frase da analizzare, in realtà il file può contenere più frasi, basta che siano una per riga e l'output viene catturato in un secondo file di testo.

Per esempio prendiamo la frase: ***The system found a solution to the problem***.

il cui albero delle dipendenze è mostrato di seguito:

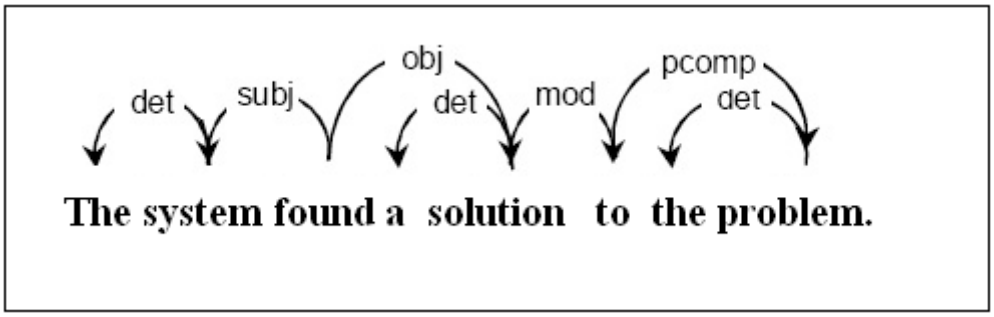


Figura 2.2 Albero delle dipendenze.

I collegamenti rappresentano le relazioni di dipendenza. La direzione dei collegamenti

è dalla parola head alla parola modifier. Le etichette associate ai collegamenti rappresentano il tipo di dipendenza.

Il programma viene eseguito nel seguente modo:

> pdemo -p ../data < file_input.txt >file_output.txt

La rappresentazione dell'albero delle dipendenze, che è il risultato dell'analisi del Minipar è il seguente:

```
> (
E0  (() fin C  *   )
1   (The       ~ Det 2   det (gov system))
2   (system    ~ N   3   s   (gov found))
3   (found     ~ V   E0  i   (gov fin))
E2  (()system N   3   subj    (gov found)   (antecedent 2))
4   (a         ~ Det 5   det (gov solution))
5   (solution  ~ N   3   obj (gov found))
6   (to        ~ Prep    5    mod   (gov solution))
7   (the       ~ Det 8   det (gov problem))
8   (problem   ~ N   6   pcomp-n    (gov to))
)
>
```

Figura 2.3 Output Minipar.

Per ogni riga possiamo individuare i seguenti elementi:

- identificatore della riga; (1)
- la parola analizzata; (The)
- la categoria grammaticale; (Det)
- il riferimento alla parola di cui è il *modifier;* (3)
- la categoria della relazione; (det)

Le parole che non hanno modificatore sono legate alle righe etichettate con Ex, che rappresentano dei nodi vuoti, che generalmente rappresentano la radice dell'albero ma che possono essere inseriti anche internamente all'albero per rappresentare la presenza di frasi subordinate.

Una breve descrizione dei significati delle etichette usate si trova nell'appendice A.

2.2 LINK GRAMMAR PARSER

Link Grammar Parser (LGP) è un parser sintattico della lingua inglese. E' stato sviluppato a partire dal 1991, presso la School of Computer Science della Carnegie Mellon University da Daniel D.K. Sleator e Davy Temperley.

La base del sistema formale grammaticale Link Grammar è la proprietà conosciuta con il termine Planarity, cioè tracciando degli archi che uniscono le parole in coppie tra loro correlate, questi non si incrociano.

Una link grammar consiste in un set di parole ognuna delle quali possiede un insieme di regole di collegamento. Una sequenza di parole è una frase (sentence) del linguaggio, definito dalla grammatica, se esiste il modo di collegare le parole tramite links (archi) così da soddisfare le seguenti regole:

- **Planarity**: i links non si devono incrociare mai;
- **Connectivity**: gli archi devono connettere tutte le parole della frase, o meglio tutte le parole e i relativi links formano un grafo completamente connesso;
- **Satisfaction**: i links di ogni coppia di parole sottostanno alle regole di collegamento di ciascuna parola, cioè gli archi soddisfano i requisiti definiti per ogni parola della frase.

Le regole di collegamento per ciascuna parola sono definite in un opportuno dizionario.

Il meccanismo di funzionamento del Link Grammar Parser è il seguente:

data una frase, il sistema assegna ad essa una struttura sintattica che consiste nell'insieme di links (archi), identificati da etichette, che collegano coppie di parole.

L'analizzatore sfrutta un dizionario di circa 60000 parole ed ha un'ampia varietà di costruzioni sintattiche, comprese molte forme rare ed idiomatiche.

Per ogni frase, l'analizzatore farà una ricerca esaustiva dei collegamenti (linkages), generando tutti i possibili linkages validi, ma il risultato dell'elaborazione sarà costituito dal linkage con minor costo trovato.

Il Link Grammar Parser è sviluppato in C++ e può essere impiegato su qualsiasi piattaforma; inoltre è correlato di API che rendono facile l'integrazione all'interno di applicazioni esterne. Viene fornito anche un eseguibile utile per effettuare le prove.

Il programma riceve in input un file di testo contente la frase da analizzare e genera un output che viene catturato e rediretto su di un file di testo:

>parse.exe <file_input.txt >file_output.txt

L'output standard del LGP è di tipo grafico:

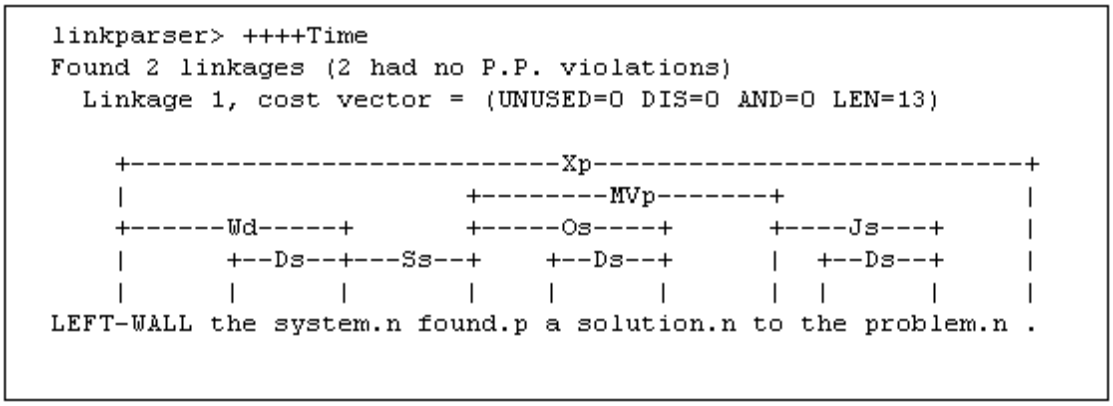


Figura 2.4 Output Grafico Link Grammar.

Il parser Link Grammar associa, nella fase di analisi, un'etichetta per ogni coppia di parole indicante la relazione esistente, cioè rappresenta il valore del connettore.

Utilizzando una delle funzionalità del Link Grammar , "**!links**", si può avere l'output in un formato testuale:

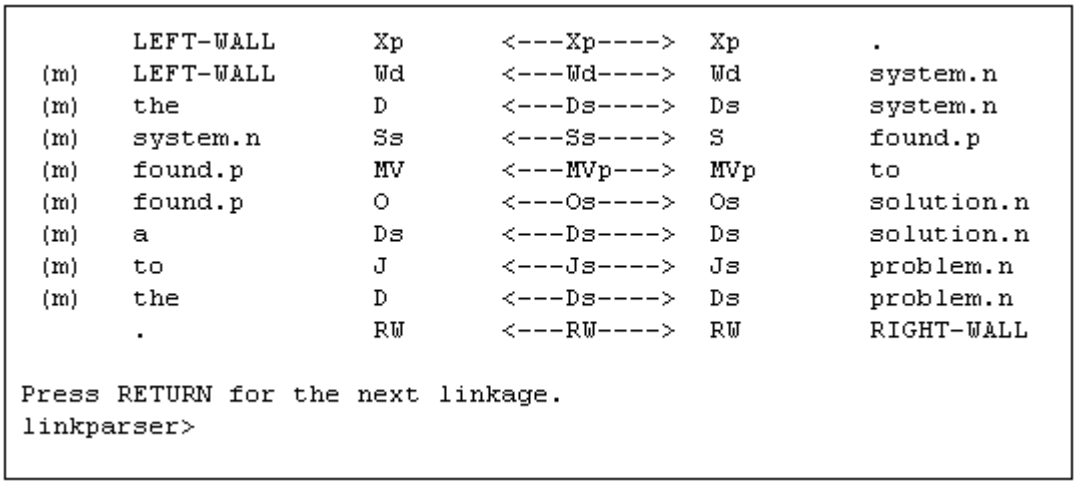


Figura 2.5 Output Testuale Link Grammar.

Il Link Grammar ha una documentazione abbondante e molto dettagliata, anche

relativamente alle regole che governano il formato dei dizionari, che ne facilita l'interpretazione dell'output.

Una breve descrizione dei significati delle etichette usate si trova nell'appendice B.

2.3 THE STANFORD PARSER

Stanford Parser è un package realizzato in Java di parser probabilistici del linguaggio naturale. Un parser probabilistico riceve in input una stringa in linguaggio naturale e fornisce in output la rappresentazione grafica di tutti i possibili alberi di parsing della stessa.
Stanford Parser è costituito essenzialmente da 3 parser :
optimized probabilistic parser, che fornisce un'analisi sintattica basata su una grammatica introdotta;
lexical dependency parser che fornisce una rappresentazione dell'input come dipendenze lessicali tra le parole;
lexicalized PCFG Parser (PCFG: **Probabilistic Context-Free Grammar)** che è utilizzato per estrarre le sotto strutture sintattiche.

E' stato sviluppato a partire dal 2003, presso l'università di Stanford, la prima versione è stata scritta da Dan Klein con il supporto di Christopher Manning, successivamente hanno partecipato al progetto Roger Levy, Cristopher Manning, Teg Grenager, Galen Andrew, Marie-Catherine de Marneffe, Bill MacCartney, Huishsin Tseng, Pi-Chuan Chang, Wolfgang Maier e Jenny Finkel.

Questo parser è stato realizzato sui principi del Factored Model per il parsing del linguaggio naturale, dove le dipendenze lessicali (semantica) e PCFG (sintassi, la struttura di frase) vengono ottenuti da modelli separati.

Dipendenze lessicali e Struttura di frase, sono due modi per rappresentare la struttura delle frasi:

- Struttura di frase, rappresenta l'insieme delle parole costituenti la frase
- Dipendenze Lessicali, rappresentano le relazioni tra le singole parole, che identifica con delle singole etichette.

La struttura sintattica (PCFG) è descritta in termini di categorie (definizioni di espressioni e classi di parole).

Il parser è stato implementato anche per altre lingue, oltre l'inglese, come cinese, italiano e bulgaro.

La versione corrente del parser richiede almeno Java 5 (JDK 1.5), richiede anche molta memoria, circa 100 Mb per essere eseguito come parser PCFG su frasi costituite da 40 parole e circa 500 Mb di memoria per essere eseguito come Factored Model.

Il pacchetto è munito di licenza GNU GPL, e il suo utilizzo non è consentito per scopi commerciali, solo per scopi di ricerca. Sono forniti i sorgenti, è correlato di API Java per l'integrazione all'interno di applicazioni esterne, include componenti per l'invocazione da riga di comando e un'interfaccia grafica.

Il programma può prendere in input vari formati di testo semplice contenti le frasi da analizzare e grazie alle varie opzioni fornite, è possibile avere un output sia in formato testo che grafico. Ogni riga del file può contenere più frasi, l'importante è che ogni frase venga delimitata da un punto.

Tra le varie opzioni abbiamo:

- Part-of-speech tagged text (utilizzando ***wordsAndTags***)
- Alberi a struttura di frase (utilizzando ***penn***)
- Relazioni Grammaticali (utilizzando ***typeDependenciesCollapsed***)
- ***-printPCFGkBest n***, il parser restituisce per ogni frase, gli *n* alberi con punteggio più alto.

Il risultato dell'elaborazione viene rediretto su un secondo file:

>java -server -mx150m -cp "stanford-parser.jar;" edu.stanford.nlp.parser.lexparser.LexicalizedParser -outputFormat "wordsAndTags,penn,typedDependenciesCollapsed" -printPCFGkBest n englishPCFG.ser.gz file_input.txt >file_output.txt

```
The/DT system/NN found/VBD a/DT solution/NN to/TO the/DT problem/NN ./.

(ROOT
  (S
    (NP (DT The) (NN system))
    (VP (VBD found)
      (NP (DT a) (NN solution))
      (PP (TO to)
        (NP (DT the) (NN problem))))
    (. .)))

det(system-2, The-1)
nsubj(found-3, system-2)
det(solution-5, a-4)
dobj(found-3, solution-5)
det(problem-8, the-7)
prep_to(found-3, problem-8)
```

Figura 2.6 Output Stanford Parser.

L'output essenzialmente è suddiviso in tre parti:

1) Analisi Grammaticale, cioè ad ogni parola è associata la categoria grammaticale a cui appartiene.

2) Albero di Struttura di Frase, cioè l'analisi logica della frase in formato albero.
3) Relazioni Grammaticali, viene specificata la relazione che intercorre tra coppie di parole, cioè la relazione che lega la seconda parola della coppia con la prima.

Inoltre utilizzando il comando:

java -mx800m -cp "stanford-parser.jar;" edu.stanford.nlp.parser.ui.Parser

viene caricata un'interfaccia grafica che può essere utilizzata per avere un output in formato "albero", come in Figura 2.7.

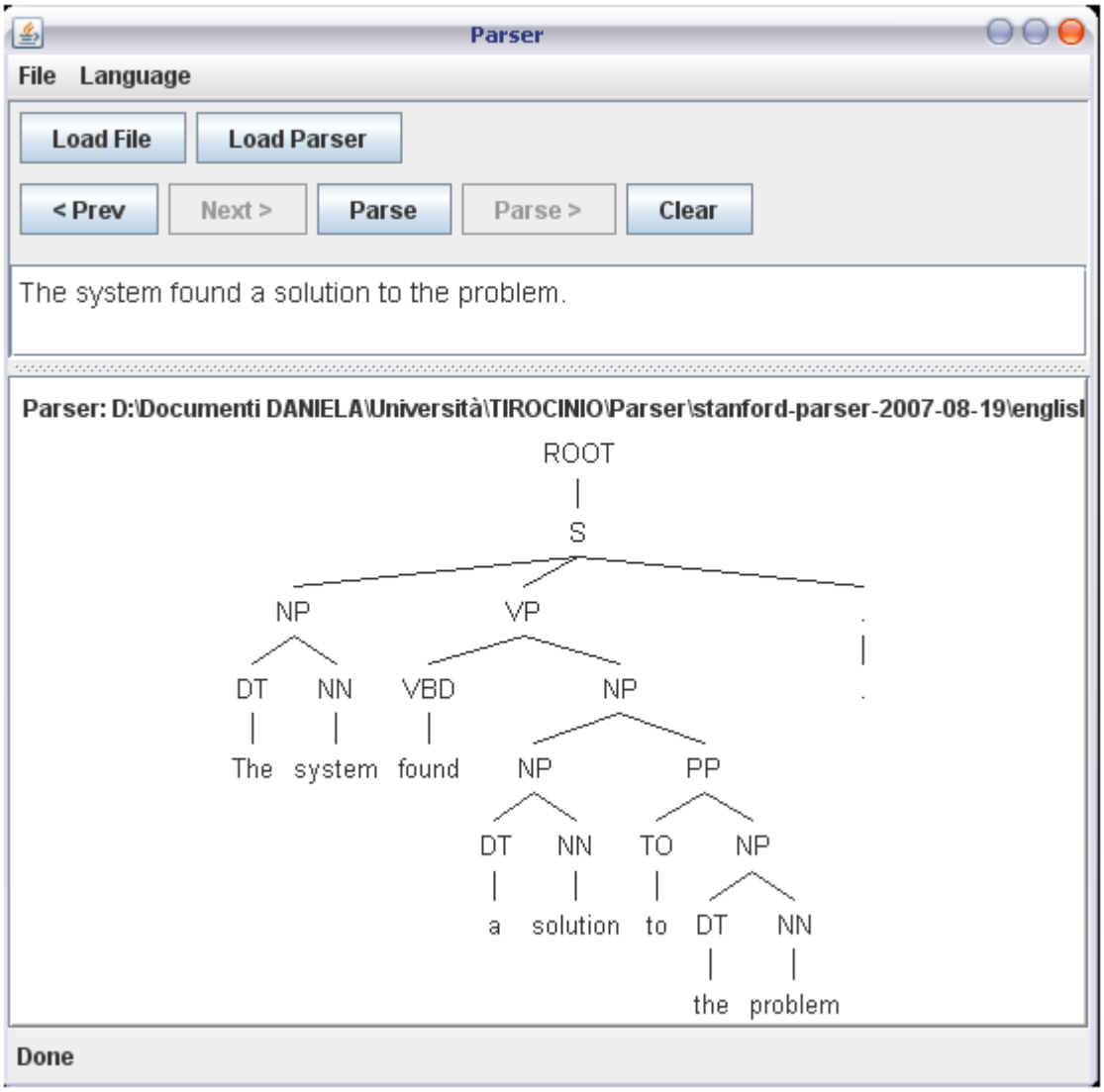


Figura 2.7 Rappresentazione Struttura ad Albero.

Una breve descrizione dei significati delle etichette usate si trova nell'appendice C.

2.4 TREE TAGGER

Tree Tagger è un tool per l'identificazione dei vocaboli basata sugli alberi decisionali, determina il lemma e le informazioni di Part-Of-Speech (POS-Tag, ovvero la categoria grammaticale del vocabolo analizzato). E' stato sviluppato all'interno del ***TC project*** presso l'istituto di linguistica computazionale dell'Università di Stoccarda. E' fornito con dizionari che gli permettono di essere utilizzato per etichettare testi in Inglese, Francese, Italiano, Tedesco, Greco, etc.

Tree Tagger è composto di due strumenti: uno è il tagger vero e proprio, che fornisce l'analisi lessicale e il part-of-speech tagging, il secondo è un programma che effettua il training per poter personalizzare il dizionario. Esso usa meccanismi probabilistici per determinare l'etichetta da assegnare ad una determinata parola. In particolare per stimare le probabilità delle transizioni sono usati degli alberi decisionali binari.

Il Tree Tagger è sviluppato in linguaggio C++, non viene fornito con i codici sorgenti ma esclusivamente con un file binario eseguibile per varie piattaforme, Sun-Solaris, PC-Linux, Mac-OSX, Windows. Per le prime tre non si ha alcuna limitazione d'uso mentre per la piattaforma Windows viene fornito un eseguibile con la limitazione di poter etichettare solo 200 parole alla volta.

In questo caso viene definito un file di parametri contenente le parole da lemmatizzare dette token. Ogni riga del file deve contenere un solo token.

Il risultato dell'elaborazione viene rediretto su un secondo file:

> tree-taggerr {-options} parameter_file {input_file.txt} {output_file.txt}

Un esempio di output del Tree Tagger, utilizzando come opzioni *-token* che stampa all'inizio della riga il token analizzato, *-lemma* che stampa il lemma della parola in esame e come dizionario *english.par,* per la frase *The largest system security system was produced from that oldest industry,* è il seguente:

```
The DT the
largest    JJS large
security   NN  security
systems    NNS system
was VBD be
produced   VVN produce
from   IN  from
that   DT  that
oldest JJS old
industry   NN  industry
.   SENT   .
```

Figura 2.8 Output Tree Tagger.

Per ogni riga viene specificata la parola analizzata, la categoria grammaticale a cui appartiene e il rispettivo lemma.

Una breve descrizione dei significati delle etichette (Categorie Grammaticali) usate si trova nella appendice D.

2.5 Confronto Parser

Le caratteristiche principali dei parser analizzati, sono stati riassunti nella Tabella di Figura 2.10, la quale mette in evidenza le maggiori differenze, che verranno viste nel dettaglio nel Capitolo 3:

PARSER	MINIPAR	LINK GRAMMAR PARSER	STANFORD PARSER	TREE TAGGER
Produttore	Department of Computer Science University of Alberta	School of Computer Science della Carnegie Mellon University	Stanford University Natural Language Processing Group	Istituto di linguistica computazionale dell'Università di Stoccarda
Link	http://www.cs.valberta.ca/~lindek/minipar.htm	http://www.link.cs.cmu.edu/link/	http://nlp.stanford.edu	http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger
Licenza	Free for use	Open Source	Gnu GPL, libera per scopi di ricerca o didattici	Libera per scopi di ricerca o didattici
Linguaggio	C++	C++	Java (version 5.0)	C ++
API	SI	SI	SI	NO
AnalisiLessicale	SI	SI	SI	SI
AnalisiSintattica	SI	SI	SI	NO
Lemmatizzazione	NO	NO	NO	SI
Parser Statistico	NO	SI	SI	SI
Tipo di Input	File	File	File	File
	Linea di comando	Linea di comando		
Limite input	Non c'è limite	Non c'è limite	Non c'è limite	Per Windows al max 200 righe
	Una frase per riga.	Una frase per riga.	Senza vincoli di riga.	Una parola per riga.
Delimitatore di frase	Ritorno Carrello	Ritorno Carrello	Punto	Punto
GUI	NO	NO	SI	NO
Interattivo	SI	SI	NO	NO
Output Grafico	NO	SI	SI	NO
Output Testuale	SI	SI	SI	SI
Alberidi Derivazione	NO	NO	SI	NO

Figura 2.10 Tabella Confronto tra i parser.

Capitolo 3

3.1 Criteri di confronto

Al fine di individuare la copertura di ogni singolo parser, cioè la percentuale delle frasi che vengono analizzate correttamente, sono stati definiti degli insiemi di frasi e per poter meglio valutare le prestazioni dei vari parser sono stati elaborati alcuni criteri di valutazione basati su approccio quantitativo. La valutazione è stata condotta comparando i risultati dell'analisi dei vari parser (le chiameremo *risposte*) con le soluzioni costruite a mano (che chiameremo *chiavi*).

Sono stati definiti due criteri diversi a seconda della tipologia di prodotto. Per i part-of-speech tagger o simili la *risposta* è costituita dal valore grammaticale associato alle singole parole, che è stato confrontato con le chiavi che sono costituite dal corretto valore grammaticale assunto dalle singole parole.

Per quanto riguarda i prodotti che effettuano una analisi sintattica completa la *risposta* è costituita, oltre che dal valore grammaticale per ogni parola, anche dalle etichette sintattiche (soggetti, verbi, complementi), espresse generalmente con relazioni di dipendenza tra coppie di parole. In questo caso si è andati a definire come *chiavi,* per ogni frase il soggetto, il verbo ed i complementi andando a valutare per ogni frase la corretta individuazione di ogni componente, sia grammaticale che sintattico.

Nel seguito si analizzeranno le risposte dei singoli prodotti a ciascun pattern predefinito, evidenziando le eventuali particolarità delle risposte. Le frasi che costituiscono ciascun pattern sono riportate nella appendice F.

3.2 Criteri di definizione dei pattern

Ciascun pattern è costituito da una frase principale, con predefinite caratteristiche sintattiche e da sue varianti. I pattern sono stati costruiti applicando variazioni alla frase principale nella maniera seguente:

1. sostituendo con nomi propri soggetto e/o complemento;
2. generando, quando possibile, la forma passiva;
3. spostando, quando possibile, la posizione di complementi.

Nella definizione del pattern si è partiti definendo semplici frasi costituite da un soggetto, un verbo e un complemento, fino ad arrivare, inserendo ad ogni passo qualche elemento di complicazione, alle situazioni più complesse, per esempio con frasi subordinate.

La frase più semplice di partenza è stata la seguente:

Soggetto	***Verbo***	***Complemento***
The system	supports	student testing

Da questa sono state ricavate le varie derivate:

- La forma passiva: *The student testing in supported by the system.*
- Sostituendo nomi propri: *The RASP system supports student testing*.

Sono stati poi inseriti altri complementi:

Soggetto	***Verbo***	***Complemento***	***Complemento***
The system	has brought	some important results	to me

Sono state analizzate frasi in cui il complemento di termine non viene introdotto da preposizioni:

Soggetto	***Verbo***	***Complemento***	***Complemento***

The system	has brought	me	some important results

Sono state inserite preposizioni che introducono particolari complementi:

Soggetto	***Verbo***	***Complemento***	***Complemento***
The group	could be heard	by the crowd	above the noise

Sono stati usati verbi al participio passato come aggettivi:

Soggetto	***Verbo***	***Complemento***
The principal required requirement	is	the simplicity

Sono stati introdotti pronomi relativi che congiungono due frasi:

The system shall provide an interface to the training developer that provides the capability to add extensive detail to the lesson outlines.

Sono state definite frasi complesse, tipo subordinate congiunte :

Some grammars are better than others, as we have proved.

oppure coordinate congiunte:

The system shall capture the editor's user id and the system date the change history comment was generated.

He told me why he was here and what he was doing.

3.2.1 Analisi del Pattern 1

Il pattern 1 è composto da frasi estremamente semplici formate dalla sequenza **soggetto-verbo-complemento oggetto**. L'analisi di questo pattern da parte di tutti i prodotti analizzati è stata utile per prendere confidenza con la tipologia e la struttura delle risposte ottenute.

Tutti gli analizzatori grammaticali non falliscono questa prova.

Consideriamo una delle frasi del pattern1 ***The system supports student testing***, ed esaminiamo l'output dei nostri analizzatori.

Metteremo in evidenza le singole categorie grammaticali e sintattiche con l'utilizzo dei caratteri diversi:

Articolo: **Blu**

Sostantivo: **Verde**

Verbo: **Rosso**

Soggetto: **Azzurro**

Verbo: **Rosso**

Complemento Oggetto: **Fucsia**

Tree Tagger:

The	DT	the
system	NN	system
supports	VVZ	support
student	NN	student
testing	NN	testing
.	SENT	.

Figura 3.1 Output Tree Tagger

Minipar:

```
> (
E0  (()   fin C    *)
1    (The        ~ Det     2     det      (gov system))
2    (system     ~ N       3     s        (gov support))
3    (supports   support   V     E0 i     (gov fin))
E2  (()   system  N         3     subj     (gov support)
4    (student    ~ N       5     nn       (gov testing))
5    (testing    ~ N       3     obj      (gov support))
)
```

Figura 3.2 Output Minipar

Link Grammar:

```
    +------------------------Xp-----------------------+
    +------Wd-----+          +---------Os--------+     |
    |      +--Ds--+----Ss---+          +----AN---+     |
    |      |      |         |          |         |     |
LEFT-WALL the system.n supports.v student.n testing.n .

       LEFT-WALL       Xp      <---Xp---->  Xp        .
 (m)   LEFT-WALL       Wd      <---Wd---->  Wd        system.n
 (m)   the             D       <---Ds---->  Ds        system.n
 (m)   system.n        Ss      <---Ss---->  Ss        supports.v
 (m)   supports.v      O       <---Os---->  Os        testing.n
 (m)   student.n       AN      <---AN---->  AN        testing.n
       .               RW      <---RW---->  RW        RIGHT-WALL
```

Figura 3.3 Output Link Grammar

Stanford Parser:

```
The/DT system/NN supports/VBZ student/NN testing/NN ./.

(ROOT
  (S
    (NP (DT The) (NN system))
    (VP (VBZ supports)
      (NP (NN student) (NN testing)))
    (. .)))

det(system-2, The-1)
nsubj(supports-3, system-2)
nn(testing-5, student-4)
dobj(supports-3, testing-5)
```

Figura 3.4 Output Stanford Parser

3.2.2 Analisi del Pattern 2

Nel pattern 2 sono state analizzate frasi composte dalla sequenza ***soggetto-verbo-complemento_oggetto-complemento_indiretto*** o ***soggetto-verbo***

complemento_indiretto-complemento_indiretto, tipo ***The group could be heard by the crowd above the noise*** **e** ***The group could be heard above the noise by the crowd.***

L'analisi grammaticale viene svolta correttamente da tutti i parser.

Relativamente all'analisi sintattica, Link Grammar, Stanford e Minipar sono in grado di individuare la presenza di un complemento diretto (complemento oggetto), non introdotto da nessuna preposizione, e di complementi indiretti caratterizzati dalla presenza di preposizioni introduttive.

Quindi le risposte ottenute andrebbero considerate corrette per entrambi. In realtà Minipar, che vorrebbe offrire un'ulteriore informazione evidenziando nelle frasi passive l'elemento che compie l'azione (complemento d'agente), introdotto dalla preposizione by, indicandolo come soggetto, fallisce nel momento in cui questo complemento non è posizionato subito dopo il verbo. Ad esempio nella frase:

The group could be heard by the crowd above the noise.

"by the crowd" è il complemento d'agente e viene etichettato da Minipar con la stringa "by-subj". Mentre quando la posizione di tale complemento viene spostata, come nella frase seguente:

The group could be heard above the noise by the crowd.

Minipar individua i due complementi come indiretti, senza evidenziare l'elemento che compie l'azione della frase passiva.

In virtù di queste considerazioni, si è ritenuto opportuno non considerare completamente corretta la risposta di Minipar.

Per fare un esempio si riporta l'output generato per una frase di questo pattern dai parser sintattici in esame, cercando di spiegare a grandi linee il meccanismo che sta alla base del funzionamento.

La frase scelta per questo esempio è la seguente:

The group could be heard by the crowd above the noise.

Nel caso del Link Grammar l'output è riportato in Figura 3.5

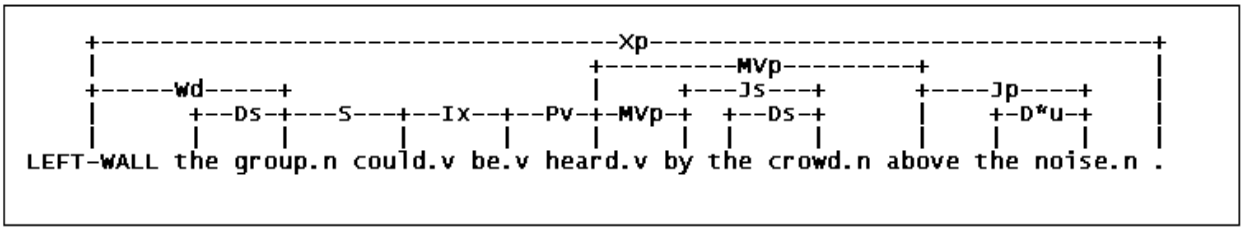


Figura 3.5 Spiegazione dell'output del Link Grammar Parser

Nella figura sono evidenziate e numerate le zone riconosciute dal parser:

1. Il soggetto viene individuato dall'etichetta *S* che collega la parola che ha la funzione di soggetto con il suo verbo.
2. Il verbo viene individuato sempre attraverso l'etichetta *S* e nel caso in cui la forma verbale sia complessa essa viene riconosciuta in base ad altre etichette. Nel caso in figura è individuata dalle etichette *I* (che collega le forme modali con l'infinito) e *Pv* (che definisce la forma passiva del verbo).
3. I complementi indiretti si individuano seguendo le etichette *Mv** e *J*. L'etichetta *Mv** lega il verbo al suo modificatore (preposizioni, avverbi etc…) e l'etichetta *J* lega le preposizioni al complemento.

L'output del Minipar è riportato in Figura 3.6.

```
> (
E0    (()      fin C   *       )
1     (The     ~ Det   2       det      (gov group))
2     (group   ~ N     5       s        (gov hear))
3     (could   ~ Aux   5       aux      (gov hear))
4     (be      ~ be    5       be       (gov hear))
5     (heard   hear V  E0      i        (gov fin))
E2    (()      group N 5       obj      (gov hear)      (antecedent 2))
6     (by      ~ Prep  5       by-subj  (gov hear))
7     (the     ~ Det   8       det      (gov crowd))
8     (crowd   ~ N     6       pcomp-n  (gov by))
9     (above   ~ Prep  8       mod      (gov crowd))
10    (the     ~ Det   11      det      (gov noise))
11    (noise   ~ N     9       pcomp-n  (gov above))
)
```

Figura 3.6 Spiegazione dell'output del Minipar

Nel caso del Minipar l'individuazione dei componenti avviene nel modo seguente:

1. Nel caso di una forma verbale passiva il soggetto viene etichettato con obj.
2. Il gruppo di parole che individuano la forma verbale è riconosciuto dal fatto che il verbo principale è legato alle ad una nodo dell'albero etichettato con Ex.
3. In presenza di forme verbali passive, il complemento d'agente viene introdotto dalla preposizione by etichettata con by-subj.
4. Il complemento è, in questo caso, riconosciuto dall'etichetta pcomp-n che lo lega alla preposizione che lo introduce.

L'output dello Stanford è riportato in Figura 3.7

```
(ROOT
  (S
    (NP (DT The) (NN group))
    (VP (MD could)
      (VP (VB be)
        (VP (VBN heard)
          (PP (IN by)
            (NP
              (NP (DT the) (NN crowd))
              (PP (IN above)
                (NP (DT the) (NN noise)))))))))
    (. .)))

det(group-2, The-1)
nsubjpass(heard-5, group-2)
aux(heard-5, could-3)
auxpass(heard-5, be-4)
det(crowd-8, the-7)
agent(heard-5, crowd-8)
det(noise-11, the-10)
prep_above(crowd-8, noise-11)
```

Figura 3.7 Spiegazione dell'Output dello Stanford.

Nello Stanford i componenti si distinguono nel seguente modo:

1. Nel caso di una forma verbale passiva il soggetto viene etichettato con nsubjpass.
2. Il gruppo di parole che individuano la forma verbale è riconosciuto dal fatto che il verbo principale è legato al soggetto e nel caso di verbi complessi questi sono

legati con opportune etichette al verbo principale.
3. Il complemento d'agente viene individuato dall'etichetta agent, che lo lega al verbo principale.
4. Il complemento indiretto viene introdotto dall'etichetta prep_above che indica la preposizione (above) che lo introduce.

3.2.3 Analisi del Pattern 3

Nel pattern 3 sono state analizzate frasi composte dalla sequenza **soggetto-verbo-complemento_oggetto.**
L'analisi grammaticale delle frasi che costituiscono il pattern 3, viene svolta correttamente, mentre nell'analisi di tipo sintattico, con questo pattern, riusciamo a mettere in evidenza la carenza da parte di tutti i prodotti analizzati per quanto riguarda l'interpretazione di particolari verbi con complemento oggetto introdotto da preposizione (es: display of, looking for, waiting for, etc…), in cui la preposizione va considerata parte integrante del verbo e non preposizione che introduce un complemento indiretto.

Quindi per la frase analizzata:

The system shall provide for display of a brief textual help message when the pointer is over an object.

sia Link Grammar che Minipar che Stanford etichettano erroneamente insieme *of message* come complemento indiretto.

L'altra frase tipica, analizzata in questo pattern, prevede la presenza di verbi che possono essere seguiti da complementi indiretti non introdotti da preposizione, con posposizione dell'oggetto. Ad esempio, una frase con il verbo *to bring* può essere costruita nei due modi seguenti:

The system has brought some important results to me.

oppure:

The system has brought me some important results.

La seconda costruzione è tipica della lingua inglese che permette, per alcuni verbi (to bring, to give, to make, to tell, etc.), di posizionare il complemento indiretto (in questo caso il complemento di termine) subito dopo il verbo stesso eliminando la preposizione e spostando in coda il complemento oggetto.

Sia Link Grammar che Minipar sono in grado di etichettare correttamente la prima formulazione della frase, ma non sono in grado di distinguere la diversa funzione dei complementi nella seconda formulazione, *The system has brought me some important results,* etichettando entrambi i complementi come diretti e quindi fallendo rispetto

all'individuazione del complemento indiretto (di termine). Questo risultato è dovuto al fatto che probabilmente entrambi gli analizzatori si basano semplicemente sulla presenza o meno di una preposizione introduttiva di complementi per distinguerli in diretti ed indiretti.

Stanford riesce ad etichettare in modo corretto i due complementi, nella frase *The system has brought me some important results,* individuando in *me* il complemento indiretto (**iobj**) e in *results* il complemento diretto (**dobj**), come si può vedere nella Figura 3.8.


```
(ROOT
  (S
    (NP (DT The) (NN system))
    (VP (VBZ has)
      (VP (VBN brought)
        (NP (PRP me))
        (NP (DT some) (JJ important) (NNS results))))
    (. .)))

det(system-2, The-1)
nsubj(brought-4, system-2)
aux(brought-4, has-3)
iobj(brought-4, me-5)
det(results-8, some-6)
amod(results-8, important-7)
dobj(brought-4, results-8)
```

Figura 3.8Output Stanford Parser.

3.2.4 Analisi del Pattern 4

Nel pattern 4 sono state analizzate frasi composte dalla sequenza **soggetto-verbo-complemento_oggetto.**

L'analisi grammaticale delle frasi del pattern4 non viene svolta correttamente da Tree Tagger, relativamente all'incapacità di riconoscere come forme aggettivali i participi passati dei verbi (*required requirement, factored product*).

L'analisi sintattica per le frasi presenti in questo pattern, richiede sostanzialmente la capacità di riconoscere come parte del soggetto o del complemento oggetto alcune

aggettivazioni espresse al participio passato di verbi e quindi, in sostanza, di distinguere l'uso in forma di aggettivo dall'uso in forma di verbo. Tale riconoscimento viene svolto in maniera corretta da Link Grammar, Minipar e Stanford.

3.2.5 Analisi del Pattern 5

Nel pattern 5 sono state analizzate frasi composte dalla sequenza **soggetto-verbo-complemento**.

Nel pattern 5 si richiede la capacità di riconoscere aggettivi composti da due parole separate dal trattino (*object-oriented, home-developed)*

L'analisi grammaticale del pattern 5, viene svolta regolarmente da tutti i prodotti che individuano correttamente le parole separate da trattino (in inglese *hyphenated adjective*), anche se TreeTagger in realtà poi non riesce ad associare a queste il lemma relativo, ma identifica il lemma con l'etichetta <unknown>.

A livello di analisi sintattica, Link Grammar, Minipar e Stanford sono in grado di assegnare le aggettivazioni composte all'elemento di competenza (soggetto o complemento).

3.2.6 Analisi del Pattern 6

Nel pattern 6 sono state analizzate frasi composte dalla sequenza **soggetto-verbo-complemento-complemento**.

Il pattern 6 comprende alcuni periodi composti da due frasi di cui una introdotta dalla preposizione "by" seguita da verbo alla forma in **–ing**, che è quindi una subordinata.

In generale l'analisi grammaticale della forma in **–ing** viene svolta correttamente, a parte alcune eccezioni.

Per quanto riguarda l'analisi sintattica Link Grammar è piuttosto carente, in quanto non riesce ad individuare la frase subordinata, probabilmente a causa della particolarità di essere impersonale. Minipar, in alcuni casi, individua la frase subordinata come un'altra proposizione rispetto alla principale, introducendo anche un soggetto aggiuntivo per risolvere la frase impersonale. In altri casi, però, anche Minipar, come Link Grammar, non risolve l'analisi, sempre probabilmente a causa della difficoltà introdotta dalla

forma impersonale del verbo della subordinata.

A differenza dei due parser precedenti anche in questo caso lo Stanford risulta migliore, in quanto è in grado di individuare la frase subordinata.

3.2.7 Analisi del Pattern 7

Nel pattern 7 si sottopongono all'analisi frasi composte dalla sequenza **soggetto-verbo-complemento-complemento**, caratterizzate dalla presenza di più proposizioni, legate fra loro sia come principale e subordinata sia come coordinate congiunte.

Dal punto di vista grammaticale non si riscontrano particolari problemi.

Dal punto di vista sintattico, sia Link Grammar, che Minipar, che Stanford, individuano la presenza di più frasi legate dalla congiunzione *and*. In particolare Link Grammar e Stanford evidenziano, nell'output, in maniera molto intuitiva, la presenza di due frasi coordinate. Per esempio per la frase :

He told me why he was here and what he was doing.

Link Grammar fornisce l'output mostrato in Figura 3.9, dove è facile individuare le due frasi subordinate-coordinate *why he was here* e *what he was doing*, grazie alla restituzione caratteristica di due grafici distinti, che legano la principale prima con una subordinata-coordinata e poi con l'altra.

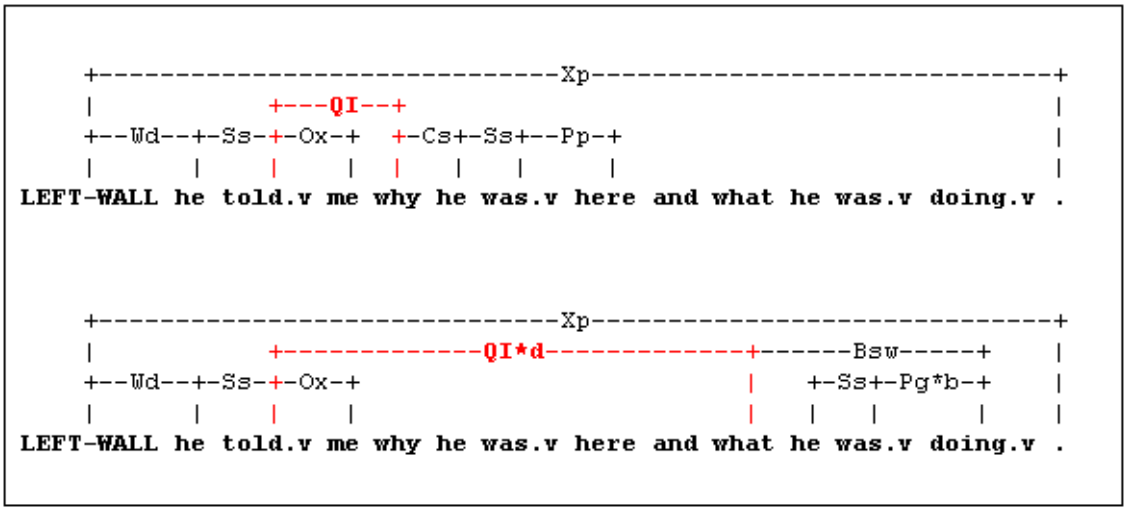
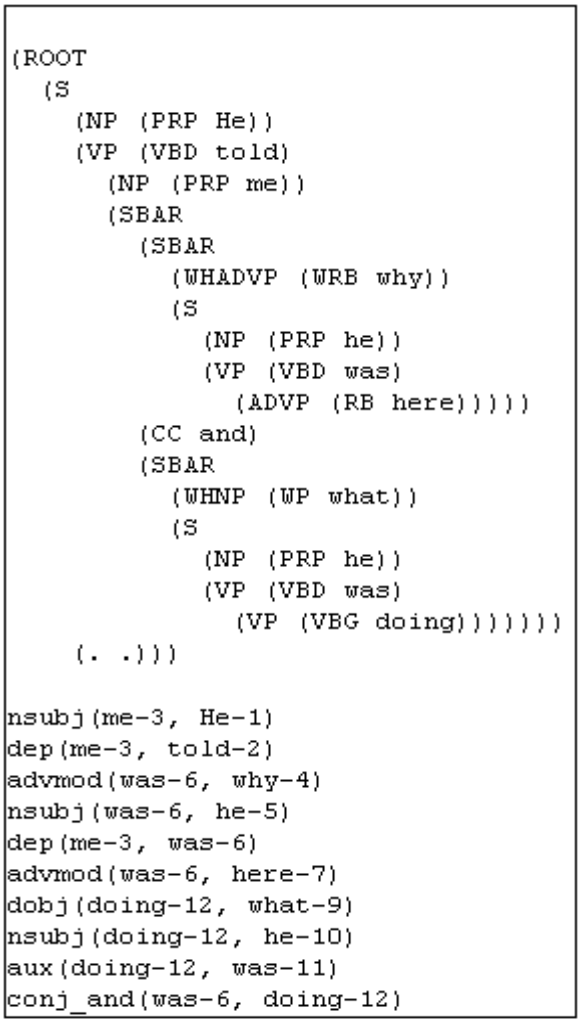


Figura 3.9 Output Link Grammar per frasi subordinate.

Stanford Parser fornisce l'output mostrato in Figura 3.10, dove si possono individuare le due frasi subordinate-coordinate *why he was here* e *what he was doing*, grazie alla struttura ad albero che lega il nodo principale (la frase principale) con due sottoalberi (subordinate-coordinate).

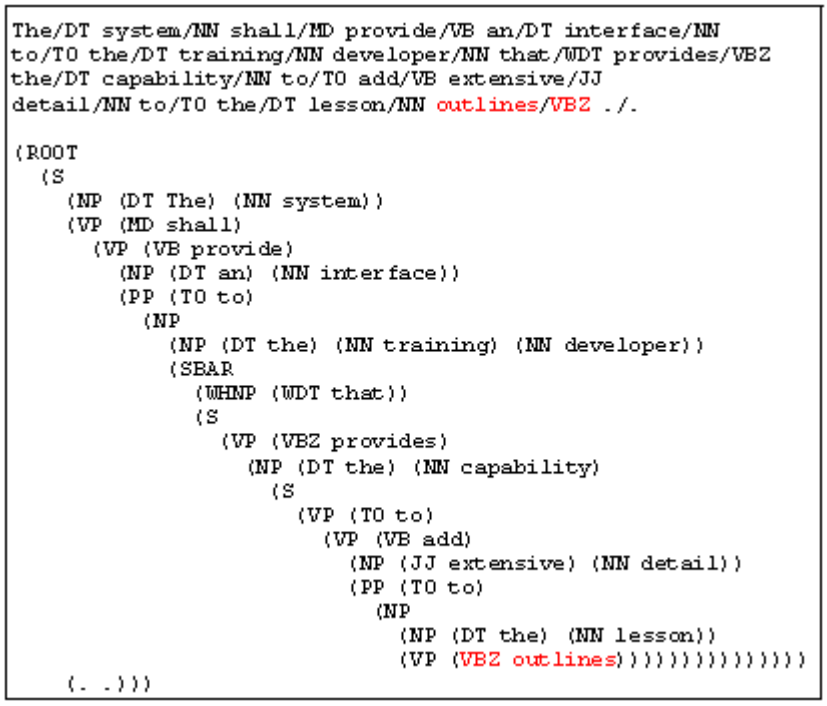


```
(ROOT
  (S
    (NP (PRP He))
    (VP (VBD told)
      (NP (PRP me))
      (SBAR
        (SBAR
          (WHADVP (WRB why))
          (S
            (NP (PRP he))
            (VP (VBD was)
              (ADVP (RB here)))))
        (CC and)
        (SBAR
          (WHNP (WP what))
          (S
            (NP (PRP he))
            (VP (VBD was)
              (VP (VBG doing)))))))
    (. .)))

nsubj(me-3, He-1)
dep(me-3, told-2)
advmod(was-6, why-4)
nsubj(was-6, he-5)
dep(me-3, was-6)
advmod(was-6, here-7)
dobj(doing-12, what-9)
nsubj(doing-12, he-10)
aux(doing-12, was-11)
conj_and(was-6, doing-12)
```

Figura 3.10 Output Stanford per frasi subordinate.

Esaminando la frase ***The system shall provide an interface to the training developer that provides the capability to add extensive detail to the lesson outlines***, possiamo notare che sia Stanford che Minipar, commettono lo stesso errore etichettando il termine *outlines* come verbo anziché come sostantivo come in Figura 3.11 e Figura 3.12, con la differenza che Stanford, grazie alla sua funzionalità di fornire più alberi di derivazione, riesce in un secondo albero a dare la corretta interpretazione, come è possibile vedere in Figura 3.13, mentre con Minipar non riusciamo ad accorgerci dell'errore visto che fornisce una sola analisi.



```
The/DT system/NN shall/MD provide/VB an/DT interface/NN
to/TO the/DT training/NN developer/NN that/WDT provides/VBZ
the/DT capability/NN to/TO add/VB extensive/JJ
detail/NN to/TO the/DT lesson/NN outlines/VBZ ./.

(ROOT
  (S
    (NP (DT The) (NN system))
    (VP (MD shall)
      (VP (VB provide)
        (NP (DT an) (NN interface))
        (PP (TO to)
          (NP
            (NP (DT the) (NN training) (NN developer))
            (SBAR
              (WHNP (WDT that))
              (S
                (VP (VBZ provides)
                  (NP (DT the) (NN capability)
                    (S
                      (VP (TO to)
                        (VP (VB add)
                          (NP (JJ extensive) (NN detail))
                          (PP (TO to)
                            (NP
                              (NP (DT the) (NN lesson))
                              (VP (VBZ outlines))))))))))))))
    (. .)))
```

Figura 3.11 Analisi Errata di Stanford.

```
> (
E3      (()     fin C   *       )
1       (The    ~ Det   2       det     (gov system))
2       (system         ~ N     4       s       (gov provide))
3       (shall ~ Aux    4       aux     (gov provide))
4       (provide        ~ V     E3      i       (gov fin))
E6      (()     system N        4       subj    (gov provide) (antecedent 2))
5       (an     ~ Det   6       det     (gov interface))
6       (interface      ~ N     4       obj     (gov provide))
7       (to     ~ Prep  6       mod     (gov interface))
8       (the    ~ Det   10      det     (gov developer))
9       (training       ~ N     10      nn      (gov developer))
10      (developer      ~ N     7       pcomp-n         (gov to))
E2      (()     fin C   10      rel     (gov developer))
11      (that   ~ N     12      s       (gov provide))
12      (provides       provide V       E2      i       (gov fin))
E7      (()     that N  12      subj    (gov provide) (antecedent 11))
E1      (()     fin C   12      fc      (gov provide))
13      (the    ~ Det   14      det     (gov capability))
14      (capability     ~ N     22      s       (gov outline))
E0      (()     inf C   14      rel     (gov capability))
E4      (()     ~ N     E0      s       (gov inf)       (antecedent 14))
15      (to     ~ Aux   16      aux     (gov add))
16      (add    ~ V     E0      i       (gov inf))
E8      (()     capability N    16      subj    (gov add)       (antecedent E4))
17      (extensive      ~ A     18      mod     (gov detail))
18      (detail         ~ N     16      obj     (gov add))
19      (to     ~ Prep  16      mod     (gov add))
20      ("the "         ~ N     21      nn      (gov lesson))
21      (lesson         ~ N     19      pcomp-n         (gov to))
22      (outlines       outline V       E1      i       (gov fin))
E9      (()     developer N     22      obj     (gov outline) (antecedent 10))
E10     (()     capability N    22      subj    (gov outline) (antecedent 14))
)
```

Figura 3.12 Analisi Errata di Minipar.

```
The/DT system/NN shall/MD provide/VB an/DT interface/NN
to/TO the/DT training/NN developer/NN that/WDT provides/VBZ
the/DT capability/NN to/TO add/VB extensive/JJ
detail/NN to/TO the/DT lesson/NN outlines/NN ./.

(ROOT
  (S
    (NP (DT The) (NN system))
    (VP (MD shall)
      (VP (VB provide)
        (NP (DT an) (NN interface))
        (PP (TO to)
          (NP
            (NP (DT the) (NN training) (NN developer))
            (SBAR
              (WHNP (WDT that))
              (S
                (VP (VBZ provides)
                  (NP (DT the) (NN capability)
                    (S
                      (VP (TO to)
                        (VP (VB add)
                          (NP (JJ extensive) (NN detail))
                          (PP (TO to)
                            (NP (DT the) (NN lesson) (NN outlines)))))))))))))
    (. .)))
```

Figura 3.13 Analisi Corretta di Stanford.

Link Grammar etichetta correttamente outlines, ma individua erroneamente *the* (the lesson outlines) come aggettivo di *outlines* anziché come Articolo, come possiamo vedere in Figura 3.14.

LEFT-WALL	Xp	<---Xp---->	Xp	.
LEFT-WALL	Wd	<---Wd---->	Wd	system.n
the	D	<---Ds---->	Ds	system.n
system.n	Ss	<---Ss---->	S	shall.v
shall.v	I	<---I----->	I	provide.v
provide.v	MV	<---MVp--->	MVp	to
provide.v	O	<---Os---->	Os	interface.n
an	Ds	<---Ds---->	Ds	interface.n
to	J	<---Js---->	Js	developer.n
the	D	<---Ds---->	Ds	developer.n
training.n	AN	<---AN---->	AN	developer.n
developer.n	Bs	<---Bs---->	Bs	provides.v
developer.n	R	<---R----->	R	that.r
that.r	RS	<---RS---->	RS	provides.v
provides.v	MV	<---MVi--->	MVi	to
provides.v	O	<---Os---->	Os	capability.n
the	D	<---D*u--->	D*u	capability.n
to	I	<---I----->	I	add.v
add.v	MV	<---MVp--->	MVp	to
add.v	O	<---Os---->	Os	detail.n
extensive.a	A	<---A----->	A	detail.n
to	J	<---Jp---->	Jp	outlines.n
the [?].a	A	<---A----->	A	outlines.n
lesson.n	AN	<---AN---->	AN	outlines.n
.	RW	<---RW---->	RW	RIGHT-WALL

Figura 3.14 Analisi Errata di Link Grammar.

Conclusioni

Il tool QuARS è stato progettato secondo principi di modularità, modificabilità e facilità d'uso:

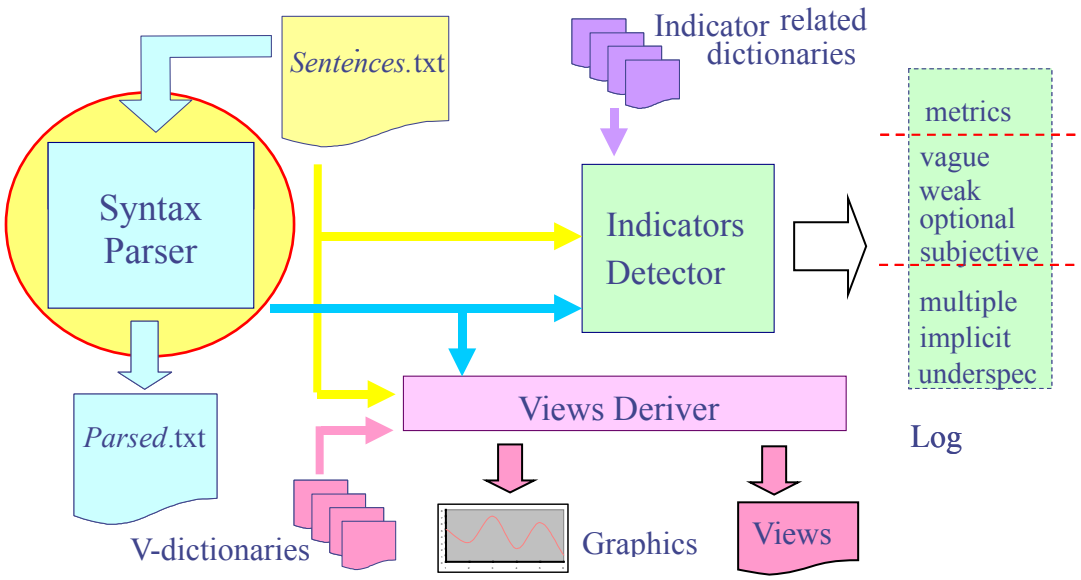


Figura 4.1 Architettura del tool QuARS.

La figura 4.1 mostra uno schema dell'architettura del tool. Un componente fondamentale, sia in termini di correttezza dell'analisi eseguita da QuARS sia in termini di tempi di esecuzione, è il *parser sintattico*.
La modularità dell'architettura adottata dal tool QuARS permette di poter integrare parser alternativi con minime modifiche al software. Questo è realizzato andando a sostituire nell'architettura del tool QuARS il componente relativo.

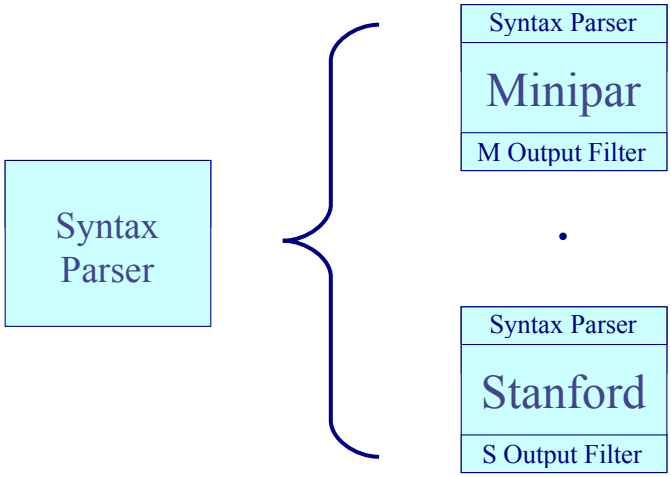


Figura 4.2 Parser Sintattico in QuARS.

L'intervento richiesto quando un diverso parser sintattico è integrato nel tool consiste nella modifica del componente "Parser Output Filter, come si può vedere in Figura 4.2. Si tratta di un modulo che modifica il formato dei dati in output del parser sintattico per renderli compatibili al formato dei dati in ingresso del componente "Indicators Detector" .
Lo scopo di questa tesi è stato di individuare un parser alternativo per l'analisi di documenti di requisiti scritti in linguaggio naturale, da integrare nel tool QuARS, che fosse in grado di fornire informazioni aggiuntive rispetto al parser Minipar, attualmente integrato in QuARS, nell'individuazione di frasi ambigue scritte in linguaggio naturale. A questo scopo è stato realizzato uno studio approfondito sulla struttura e sul funzionamento di alcuni parser attualmente presenti sul mercato e alla luce dell'analisi svolta nel Capitolo 3 abbiamo individuato lo Stanford parser, come il prodotto che potrebbe essere in grado di fornire l'informazione da noi cercata.

Appendice A – Tagset usato da Minipar

CATEGORIE GRAMMATICALI

Det	Determiners
PreDet	Pre-determiners (search for PreDet in data/wndict.lsp for instances)
PostDet	Post-determiners (search for PostDet in data/wndict.lsp for instances)
NUM	numbers
C	Clauses
I	Inflectional Phrases
V	Verb and Verb Phrases
N	Noun and Noun Phrases
NN	noun-noun modifiers
P	Preposition and Preposition Phrases
PpSpec	Specifiers of Preposition Phrases (search for PpSpec in data/wndict.lsp for instances)
A	Adjective/Adverbs
Have	have
Aux	Auxilary verbs, e.g. should, will, does, ...
Be	Different forms of be: is, am, were, be, ...
COMP	Complementizer
VBE	be used as a linking verb. E.g., I am hungry
V_N	verbs with one argument (the subject), i.e., intransitive verbs
V_N_N	verbs with two arguments, i.e., transitive verbs
V_N_I	verbs taking small clause as complement

RELAZIONI GRAMMATICALI

appo	"ACME president, --appo-> P.W. Buckman"
aux	"should <-aux-- resign"
Be	"is <-be-- sleeping"
c	"that <-c-- John loves Mary"
comp1	first complement
det	"the <-det `-- hat"
Gen	"Jane's <-gen-- uncle"
Have	"have <-have-- disappeared"
i	the relationship between a C clause and its I clause
inv-aux inverted auxiliary	"Will <-inv-aux-- you stop it?"
inv-be inverted be	"Is <-inv-be-- she sleeping"
inv-have inverted have	"Have <-inv-have-- you slept"
mod	the relationship between a word and its adjunct modifier
pnmod	post nominal modifier
p-spec	specifier of prepositional phrases
pcomp-c	clausal complement of prepositions
pcomp-n	nominal complement of prepositions
post	post determiner
pre	pre determiner
pred	predicate of a clause
rel	relative clause
vrel	passive verb modifier of nouns
wha, whn, whp	wh-elements at C-spec positions
obj	object of verbs
obj2	second object of ditransitive verbs
subj	subject of verbs
s	surface subject

Appendice B – Tagset usato da Link Grammar

Di seguito è riportato un elenco, estratto dalla documentazione, dei tag usati dal Link Grammar per etichettare i link utilizzati nella costruzione dei grafi associati alle frasi.

A connects pre-noun ("attributive") adjectives to following nouns: "The BIG DOG chased me", "The BIG BLACK UGLY DOG chased me".

AA is used in the construction "How [adj] a [noun] was it?". It connects the adjective to the following "a".

AF connects adjectives to verbs in cases where the adjective is fronted, such as questions and indirect questions: "How BIG IS it?"

AL connects a few determiners like "all" or "both" to following determiners: "ALL THE people are here".

AM connects "as" to "much" or "many": "I don't go out AS MUCH now".

AN connects noun-modifiers to following nouns: "The TAX PROPOSAL was rejected".

AZ connects the word "as" back to certain verbs that can take "[obj] as [adj]" as a complement: "He VIEWED him AS stupid".

B serves various functions involving relative clauses and questions. It connects transitive verbs back to their objects in relative clauses, questions, and indirect questions ("The DOG we CHASED", "WHO did you SEE?"); it also connects the main noun to the finite verb in subject-type relative clauses ("The DOG who CHASED me was black").

BI connects forms of the verb "be" to certain idiomatic expressions: for example, cases like "He IS PRESIDENT of the company".

BT is used with time expressions acting as fronted objects: "How many YEARS did it LAST?".

BW connects "what" to various verbs like "think", which are not really transitive but can connect back to "what" in questions: "WHAT do you THINK?"

C links conjunctions to subjects of subordinate clauses ("He left WHEN HE saw me"). it also links certain verbs to subjects of embedded clauses ("He SAID HE was sorry").

CC connects clauses to following coordinating conjunctions ("SHE left BUT we stayed").

CO connects "openers" to subjects of clauses: "APPARENTLY / ON Tuesday , THEY went to a movie".

CP connects paraphrasing or quoting verbs to the wall (and, indirectly, to the paraphrased expression): "///// That is untrue, the spokesman SAID."

CQ connects to auxiliaries in comparative constructions involving s-v inversion: "SHE has more money THAN DOES Joe".

CX is used in comparative constructions where the right half of the comparative contains only an auxiliary: "She has more money THAN he DOES".

D connects determiners to nouns: "THE DOG chased A CAT and SOME BIRDS".

DD connects definite determiners ("the", "his") to certain things like number expressions and adjectives acting as nouns: "THE POOR", "THE TWO he mentioned".

DG connects the word "The" with proper nouns: "the Riviera", "the Mississippi".

DP connects possessive determiners to gerunds: "YOUR TELLING John to leave was stupid".

DT connects determiners to nouns in idiomatic time expressions: "NEXT WEEK", "NEXT THURSDAY".

E is used for verb-modifying adverbs which precede the verb: "He is APPARENTLY LEAVING".

EA connects adverbs to adjectives: "She is a VERY GOOD player".
EB connects adverbs to forms of "be" before an object or prepositional phrase: "He IS APPARENTLY a good programmer".
EC connects adverbs to comparative adjectives: "It is MUCH BIGGER"
EE connects adverbs to other adverbs: "He ran VERY QUICKLY".
EF connects the word "enough" to preceding adjectives and adverbs: "He didn't run QUICKLY ENOUGH".
EI connects a few adverbs to "after" and "before": "I left SOON AFTER I saw you".
EL connects certain words to the word "else": something / everything / anything / nothing , somewhere (etc.), and someone (etc.).
EN connects certain adverbs to expressions of quantity: "The class has NEARLY FIFTY students".
ER is used the expression "The x-er..., the y-er...". it connects the two halfs of the expression together, via the comparative words (e.g. "The FASTER it is, the MORE they will like it").
EZ connects certain adverbs to the word "as", like "just" and "almost": "You're JUST AS good as he is."
FL connects "for" to "long": "I didn't wait FOR LONG".
FM connects the preposition "from" to various other prepositions: "We heard a scream FROM INSIDE the house".
G connects proper noun words together in series: "GEORGE HERBERT WALKER BUSH is here."
GN (stage 2 only) connects a proper noun to a preceding common noun which introduces it: "The ACTOR Eddie MURPHY attended the event".
H connects "how" to "much" or "many": "HOW MUCH money do you have".
I connects infinitive verb forms to certain words such as modal verbs and "to": "You MUST DO it", "I want TO DO it".
ID is a special class of link-types generated by the parser, with arbitrary four-letter names (such as "IDBT"), to connect together words of idiomatic expressions such as "at_hand" and "head_of_state".
IN connects the preposition "in" to certain time expressions: "We did it IN DECEMBER".
J connects prepositions to their objects: "The man WITH the HAT is here".
JG connects certain prepositions to proper-noun objects: "The Emir OF KUWAIT is here".
JQ connects prepositions to question-word determiners in "prepositional questions": "IN WHICH room were you sleeping?"
JT connects certain conjunctions to time-expressions like "last week": "UNTIL last WEEK, I thought she liked me".
K connects certain verbs with particles like "in", "out", "up" and the like: "He STOOD UP and WALKED OUT".
L connects certain determiners to superlative adjectives: "He has THE BIGGEST room".
LE is used in comparative constructions to connect an adjective to the second half of the comparative expression beyond a complement phrase: "It is more LIKELY that Joe will go THAN that Fred will go".
LI connects certain verbs to the preposition "like": "I FEEL LIKE a fool."
M connects nouns to various kinds of post-noun modifiers: prepositional phrases ("The MAN WITH the hat"), participle modifiers ("The WOMAN CARRYING the box"), prepositional relatives ("The MAN TO whom I was speaking"), and other kinds.
MF is used in the expression "Many people were injured, SOME OF THEM children".
MG allows certain prepositions to modify proper nouns: "The EMIR OF Kuwait is here".

MV connects verbs and adjectives to modifying phrases that follow, like adverbs ("The dog RAN

QUICKLY"), prepositional phrases ("The dog RAN IN the yard"), subordinating conjunctions ("He LEFT WHEN he saw me"), comparatives, participle phrases with commas, and other things.

MX connects modifying phrases with commas to preceding nouns: "The DOG, a POODLE, was black". "JOHN, IN a black suit, looked great".

N connects the word "not" to preceding auxiliaries: "He DID NOT go".

ND connects numbers with expressions that require numerical determiners: "I saw him THREE WEEKS ago".

NF is used with NJ in idiomatic number expressions involving "of": "He lives two THIRDS OF a mile from here".

NI is used in a few special idiomatic number phrases: "I have BETWEEN 5 AND 20 dogs".

NJ is used with NF in idiomatic number expressions involving "of": "He lives two thirds OF a MILE from here".

NN connects number words together in series: "FOUR HUNDRED THOUSAND people live here".

NO is used on words which have no normal linkage requirement, but need to be included in the dictionary, such as "um" and "ah".

NR connects fraction words with superlatives: "It is the THIRD BIGGEST city in China".

NS connects singular numbers (one, 1, a) to idiomatic expressions requiring number determiners: "I saw him ONE WEEK ago".

NT connects "not" to "to": "I told you NOT TO come".

NW is used in idiomatic fraction expressions: "TWO THIRDS of the students were women".

O connects transitive verbs to their objects, direct or indirect: "She SAW ME", "I GAVE HIM the BOOK".

OD is used for verbs like "rise" and "fall" which can take expressions of distance as complements: "It FELL five FEET".

OF connects certain verbs and adjectives to the word "of": "She ACCUSED him OF the crime", "I'm PROUD OF you".

ON connectors the word "on" to dates or days of the week in time expressions: "We saw her again ON TUESDAY".

OT is used for verbs like "last" which can take time expressions as objects: "It LASTED five HOURS".

OX is an object connector, analogous to SF, used for special "filler" words like "it" and "there" when used as objects: "That MAKES IT unlikely that she will come".

P connects forms of the verb "be" to various words that can be its complements: prepositions, adjectives, and passive and progressive participles: "He WAS [ANGRY / IN the yard / CHOSEN / RUNNING]".

PF is used in certain questions with "be", when the complement need of "be" is satisfied by a preceding question word: "WHERE are you?", "WHEN will it BE?"

PP connects forms of "have" with past participles: "He HAS GONE".

Q is used in questions. It connects the wall to the auxiliary in simple yes-no questions ("///// DID you go?"); it connects the question word to the auxiliary in where-when-how questions ("WHERE DID you go").

QI connects certain verbs and adjectives to question-words, forming indirect questions: "He WONDERED WHAT she would say".

R connects nouns to relative clauses. In subject-type relatives, it connects to the relative pronoun ("The DOG WHO chased me was black"); in object-type relatives, it connects either to the relative pronoun or to the subject of the relative clause ("The DOG THAT we chased was black", "The DOG WE chased was black").

RS is used in subject-type relative clauses to connect the relative pronoun to the verb: "The dog WHO CHASED me was black".

RW connects the right-wall to the left-wall in cases where the right-wall is not needed for punctuation purposes.

S connects subject nouns to finite verbs: "The DOG CHASED the cat": "The DOG [IS chasing / HAS chased / WILL chase] the cat".

SF is a special connector used to connect "filler" subjects like "it" and "there" to finite verbs: "THERE IS a problem", "IT IS likely that he will go".

SFI connects "filler" subjects like "it" and "there" to verbs in cases with subject-verb inversion: "IS THERE a problem?", "IS IT likely that he will go?"

SI connects subject nouns to finite verbs in cases of subject-verb inversion: "IS JOHN coming?", "Who DID HE see?"

SX connects "I" to special first-person verbs lke "was" and "am".

SXI connects "I" to first-person verbs in cases of s-v inversion.

TA is used to connect adjectives like "late" to month names: "We did it in LATE DECEMBER".

TD connects day-of-the-week words to time expressions like "morning": "We'll do it MONDAY MORNING".

TH connects words that take "that [clause]" complements with the word "that". These include verbs ("She TOLD him THAT..."), nouns ("The IDEA THAT..."), and adjectives ("We are CERTAIN THAT").

TI is used for titles like "president", which can be used in certain cirumstances without a determiner: "AS PRESIDENT of the company, it is my decision".

TM is used to connect month names to day numbers: "It happened on JANUARY 21".

TO connects verbs and adjectives which take infinitival complements to the word "to": "We TRIED TO start the car", "We are EAGER TO do it".

TQ is the determiner connector for time expressions acting as fronted objects: "How MANY YEARS did it last".

TS connects certain verbs that can take subjunctive clauses as complements - "suggest", "require" - to the word that: "We SUGGESTED THAT he go".

TW connects days of the week to dates in time expressions: "The meeting will be on MONDAY, JANUARY 21".

TY is used for certain idiomatic usages of year numbers: "I saw him on January 21 , 1990 ". (In this case it connects the day number to the year number.)

U is a special connector on nouns, which is disjoined with both the determiner and subject-object connectors. It is used in idiomatic expressions like "What KIND_OF DOG did you buy?"

UN connects the words "until" and "since" to certain time phrases like "after [clause]": "You should wait UNTIL AFTER you talk to me".

V connects various verbs to idiomatic expressions that may be non-adjacent: "We TOOK him FOR_GRANTED", "We HELD her RESPONSIBLE".

W connects the subjects of main clauses to the wall, in ordinary declaratives, imperatives, and most questions (except yes-no questions). It also connects coordinating conjunctions to following clauses: "We left BUT SHE stayed".

WN connects the word "when" to time nouns like "year": "The YEAR WHEN we lived in England was wonderful".

WR connects the word "where" to a few verbs like "put" in questions like "WHERE did you PUT it?".

X is used with punctuation, to connect punctuation symbols either to words or to each other. For example, in this case, POODLE connects to commas on either side: "The dog , a POODLE , was black."

Y is used in certain idiomatic time and place expressions, to connect quantity expressions to the head word of the expression: "He left three HOURS AGO", "She lives three MILES FROM the station".

YP connects plural noun forms ending in s to "'" in possessive constructions: "The STUDENTS ' rooms are large".

YS connects nouns to the possessive suffix "'s": "JOHN 'S dog is black".

Z coennects the preposition "as" to certain verbs: "AS we EXPECTED, he was late".

Appendice C– Tagset usato da Stanford Parser

dep	Dependent
aux	auxiliary
auxpass	passive auxiliary
cop	copula
conj	conjunct
cc	coordination
arg	argument
subj	subject
nsubj	nominal subject
nsubjpass	passive nominal subject
csubj	Clausal subject
comp	complement
obj	object
dobj	Direct object
iobj	Indirect object

pobj	Object of preposition
attr	attributive
ccomp	Clausal complement with internal subject
xcomp	Clausal complement with external subject
compl	complementizer
mark	Marker (word introducing an advel)
rel	Relative (word introducing a rcmod)
acomp	Adjectival complement
agent	agent
expl	Expletive (expletive *there*)
mod	modifier
advel	Adverbial clause modifier
purpel	Purpose clause modifier
tmod	Temporal modifier
rcmod	Relative clause modifier

amod	Adjectival modifier
infmod	Infinitival modifier
partmod	Participial modifier
num	Numeric modifier
number	Element of compound number
appos	Appositional modifier
nn	Noun compound modifier
abbrev	Abbreviation modifier
advmod	Adverbial modifier
neg	Negation modifier
poss	Possession modifier
possessive	Possessive modifier ('s)
prt	Phrasal verb particle
det	determiner
prep	Prepositional modifier
sdep	Semantic dependent
xsubj	Controlling subject

Appendice D – Tagset usato da Tree Tagger

CC	Coordinating conjunction
CD	Cardinal Number
DT	Determiner
EX	Existenzial *there*
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NP	Proper noun, singular
NPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PP	Personal pronoun
PP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	*to*

UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

Appendice E – Penn Treebank part-of-speech Tagset

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	and, but,or	**SYM**	symbol	+, %, &
CD	Cardinal number	one, two, three	**TO**	“to”	To
DT	Determiner	a, the	**UH**	interjction	ah, oops
EX	Existential ‘there’	there	**VB**	Verb, base form	Eat
FW	Foreign word	mea culpa	**VBD**	Verb, past tense	ate
IN	Preposition/sub-conj	of, in, by	**VBG**	Verb, gerund	eating
JJ	Adjective	yellow	**VBN**	Verb, past participle	eaten
JJR	Adj., comparative	bigger	**VBP**	Verb, non-3sg pres	eat
JJS	Adj., superlative	wildest	**VBZ**	Verb, 3sg pres	eats
LS	List item marker	1, 2, One	**WDT**	Wh-determiner	which, that
MD	Modal	can, should	**WP**	Wh-pronum	what, who
NN	Noun, sing. or mass	llama	**WP$**	Possessive wh-	whose
NNS	Noun, plural	llamas	**WRB**	Wh-adverb	how, where
NNP	Proper noun, singular	IBM	**$**	Dollar sign	$
NNPS	Proper noun, plural	Carolinas	**#**	Pound sign	#
PDT	Predeterminer	all, both	**‘**	Left quote	(‘ or “)
POS	Possessive ending	‘s	**’**	Right quote	(’ or ”)
PP	Personal pronum	I, you, he	**(**	Left parenthesis	([, (, {, <)
PP$	Possessive pronum	your, one’s	**)**	Right paranthesis	(],), }, >)
RB	Adverb	quickly, never	**,**	Comma	,
RBR	Adverb, comparative	faster	**.**	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	fastest	**:**	Mid-sentence punc	(: ; . -)
RP	Particle	up, off			

Appendice F – Sintagmi

NP	Noun phrase (SINTAGMA NOMINALE)
VP	Verb phrase (SINTAGMA VERBALE)
PP	Preposition phrase (SINTAGMA PREPOSIZIONALE)
AP	Adjective phrase (SINTAGMA AGGETTIVALE)
CP	Complementizer phrase (SINTAGMA DI COMPLEMENTAZIONE)
IP	Inflectional phrase (SINTAGMA DI FLESSIONE)
DT	Determiner phrase (SINTAGMA DETERMINANTE)
AdvP	Adverbial phrase (SINTAGMA AVVERBIALE)
TP	Tense phrase (SINTAGMA DEL TEMPO)
AspP	Aspect phrase (SINTAGMA DELL'ASPETTO)
FocP	Focus phrase (SINTAGMA DEL FOCUS)
TopP	Topic phrase (SINTAGMA DEL TOPIC)
QP	Quantifier phrase (SINTAGMA DEL QUALIFICATORE)
ForceP	Force phrase (SINTAGMA DELLA FORZA (FRASALE))
FinP	Finiteness phrase (SINTAGMA DELLA FINITEZZA)

Appendice G – Pattern Generici

Pattern 1

The system supports student testing
The student testing is supported by the system.
This shall provide capability to maintain grades and adjustments.
The system shall maintain the education/training product catalog.
The system shall support implementation of performance based testing.

Pattern 2

The above requirements shall be verified by test.
The witness said that the case was not brought before committee because of the incident the night before.
The system shall provide capability to remove student from training.
The group could be heard by the crowd above the noise.
The group could be heard above the noise by the crowd.

Pattern 3

The mean time needed to remove a faulty board and restore service shall be less than 30 minutes.
The system has brought me some important results.
The system has brought some important results to me.
The system shall not give out secrets and open files.
These parsers still make some mistakes.
The system shall provide for display of a brief textual help message when the pointer is over an object.

Pattern 4

The principal required requirement is the simplicity.
The simplicity is the principal required requirement.
The lexicalized probabilistic parser implements a factored product model.

Pattern 5

The software shall be designed according to the rules of object-oriented design.
The system shall provide for the re-use of content.
I read the well-known book about Natural Languages Processing.
The calculation was carried out by home-developed software.
The system shall maintain the education/training product catalog.

Pattern 6

The system shall associate a collective task with an echelon.
The system shall provide the capability to search for an individual task from a list by selectable criteria.
The overall cost was calculated by estimating the labour cost.
I prepared the solution by mixing different compounds.
We saw a movie about dancing bears.
I win by writing the best book

Pattern 7

The system shall provide an interface to the training developer that provides the capability to add extensive detail to the lesson outlines.
He told me why he was here and what he was doing.
Some grammars are better than others, as we have proved.
The system shall be able to run also in case of attack.
I saw Peter and Paul and Mary saw me.
The system shall provide an interface to the training developer that provides the capability to add extensive detail to the lesson outlines.

Bibliografia

[1] LinkGrammar On-Line Documentation
(http://www.link.cs.cmu.edu/link/)

[2] Minipar On-Line Documentation
(http://www.cs.valberta.ca/~lindek/minipar.htm)

[3] Tree Tagger On-Line Documentation
(http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger)

[4] Stanford On-Line Documentation
(http://nlp.stanford.edu)

[6] Roger S. Pressman, *Software engineering: a practitioner's approach —5th* ed., McGraw-Hill, 2001

[7] Leszek A. Maciaszek, *Requirements Analysis and System Design Developing Information Systems with UML*, Addison Wesley, 2001

[8] Lami G., *QuARS: A Tool for Analyzing Requirements,* CarnegieMellon Software Engineering Institute, 2005.

[9] Berry D.M., Bucchiarone A., Gnesi S., Lami G., Trentanni G., *A New Quality Model for Natural Language Requirements Specifications,* Cheriton School of Computer Science
University of Waterloo (Canada), Istituto di Scienze e Tecnologia dell'Informazione del C.N.R di Pisa (Italy).

[10] Santorini B., *Part-of-Speech Tagging Guidelines for the Penn Treebank Project*, 1990.

[11] Bies A., Ferguson M., Katz K., MacIntyre R., *Bracketing Guidelines for Treebank II Style Penn Treebank Project,*1995.

[12] Taylor A., *Bracketing Switchboard: An addendum to TREEBANK II Bracketing Guidelines*, 1996.

[12] Sleator D.D., Temperley D., *Parsing English with a Link Grammar*, School of Computer Science Carnegie Mellon University Pittsburgh e Music Department Columbia University New York.

[13] Dan Klein and Christopher D. Manning, *Fast Exact Inference with a Facrored Model for Natural Language Parsing*, Stanford University 2003.

[14] MacCartney B., Manning C.D., de Marneffe M.C., *Generating Typed Dependency Parses from Phrase Structure Parses,* Stanford University.

[15] Gnesi S., Dispense di Ingegneria del Software.

[16] Lami G., Ferguson R.W., *An empirical study on the impact of automation on the requirements analysis process.*