

---

ANNO ACCADEMICO 2024/2025

---

# Tecnologie del Linguaggio Naturale

---

## Teoria

Altair's Notes



**UNIVERSITÀ  
DI TORINO**



---

DIPARTIMENTO DI INFORMATICA

---



<b>CAPITOLO 1</b>	<b>INTRODUZIONE ALLE TECNOLOGIE DEL LINGUAGGIO NATURALE</b>	<b>PAGINA 5</b>
1.1	Prologo	5
	La Complessità del Linguaggio Naturale — 6 • I Livelli di Conoscenza del Linguaggio — 8 • Strutture Linguistiche e Ambiguità — 9 • Lo Stato dell’Arte — 10	
1.2	I Livelli Linguistici	13
	Da Frase a Significato — 13 • Il Livello Morfologico e l’Analisi Lessicale — 15 • Il Livello Sintattico — 18 • Il Livello Semantico — 20 • Il Livello Pragmatico e del Discorso — 23	
<b>CAPITOLO 2</b>	<b>SEQUENCE TAGGING</b>	<b>PAGINA 25</b>
2.1	Part of Speech (PoS) Tagging	25
	Perché studiare PoS? — 25 • Analisi Basata su Regole — 27 • ENGTWOL Lexicon — 27	
2.2	PoS Tagger Statistici	28
	HMM — 28 • MEMM e CRF — 32 • Tagging di Parole Sconosciute — 33	
2.3	NER Tagging	33
<b>CAPITOLO 3</b>	<b>SINTASSI</b>	<b>PAGINA 36</b>
3.1	Prologo	36
	Grammatiche Generative — 36 • Mildly-context-sensitive — 37	
3.2	La Performance e i costituenti	39
	Approcci al Parsing — 39 • Parsing Probabilistico — 41 • Valutazione — 42 • Parsing Parziale — 43	
3.3	Parsing a Dipendenze	43
	Sintassi a Dipendenze — 44 • Approcci — 45 • Parsing Deterministico a Transizioni — 46 • Parsing a Regole per Dipendenze a Vincoli — 49	
<b>CAPITOLO 4</b>	<b>SEMANTICA</b>	<b>PAGINA 51</b>
4.1	Fondamenti di Semantica Computazionale	51
	Algoritmo Fondamentale della Semantica Computazionale — 52	
4.2	Lambda Astrazione	52
	Beta riduzione — 53 • Semantica di Montague — 53 • Articoli e Nomi Propri — 54	
<b>CAPITOLO 5</b>	<b>NATURAL LANGUAGE GENERATION</b>	<b>PAGINA 56</b>
<b>CAPITOLO 6</b>	<b>DIALOG SYSTEMS E CHATBOTSS</b>	<b>PAGINA 58</b>

---

<b>CAPITOLO 7</b>	<b>SEMANTICA LESSICALE</b>	<b>PAGINA 60</b>
7.1	Introduzione Programma della Seconda Parte del Corso — 60	60

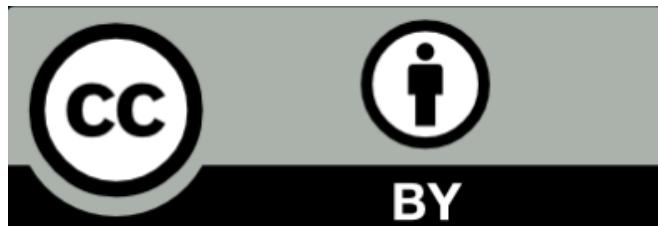




# Premessa

## Licenza

Questi appunti sono rilasciati sotto licenza Creative Commons Attribuzione 4.0 Internazionale (per maggiori informazioni consultare il link: <https://creativecommons.org/licenses/by/4.0/>).



## Formato utilizzato

Box di "Concetto sbagliato":

### Concetto sbagliato 0.1: Testo del concetto sbagliato

Testo contenente il concetto giusto.

Box di "Corollario":

### Corollario 0.0.1 Nome del corollario

Testo del corollario. Per corollario si intende una definizione minore, legata a un'altra definizione.

Box di "Definizione":

### Definizione 0.0.1: Nome delle definizioni

Testo della definizione.

Box di "Domanda":

### Domanda 0.1

Testo della domanda. Le domande sono spesso utilizzate per far riflettere sulle definizioni o sui concetti.

Box di "Esempio":

### Esempio 0.0.1 (Nome dell'esempio)

Testo dell'esempio. Gli esempi sono tratti dalle slides del corso.

**Box di "Note":**

**Note:-**

Testo della nota. Le note sono spesso utilizzate per chiarire concetti o per dare informazioni aggiuntive.

**Box di "Osservazioni":**

**Osservazioni 0.0.1**

Testo delle osservazioni. Le osservazioni sono spesso utilizzate per chiarire concetti o per dare informazioni aggiuntive. A differenza delle note le osservazioni sono più specifiche.



# 1

## Introduzione alle Tecnologie del Linguaggio Naturale

### 1.1 Prologo

La prima parte del corso sarà incentrata sulla linguistica computazionale generale, in cui ci si soffermerà sugli aspetti più tradizionali e linguistici<sup>1</sup>. In questa parte verrà anche trattato il parsing. Nella seconda parte si andranno a studiare la semantica lessicale e le ontologie. Infine, nella terza parte del corso si andrà a studiare NLP statistico e distribuzionale.

### Parte prima: keywords

- 
- NLP
  - CL
  - Lexicon
  - Morphology
  - Syntax
  - semantics
  - Conversational Interface
  - Conversational agent
  - Dialogue System
  - Parsing
  - NLG
  - MT
  - Grammar
  - Treebank
  - NL ambiguity
  - BOT
  - LLM

Figure 1.1: Il giorno prima dell'esame bisogna sapere cosa significano tutte queste parole :3

---

<sup>1</sup>Libro di riferimento: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. La prima e la seconda edizione, perché Jurafsky non riesce a finire il draft della terza :(

### Le 4 ere della linguistica computazionale:

1. 1940 - 1969: primi tentativi.
2. 1970 - 1992: formalizzazione.
3. 1993 - 2012: apprendimento automatico.
4. 2013 - 2018: deep learning.

**Note:-**

Tutto cambiò nel 2018, quando NLP fu il primo successo su larga scala di rete neurale autosupervisionata.

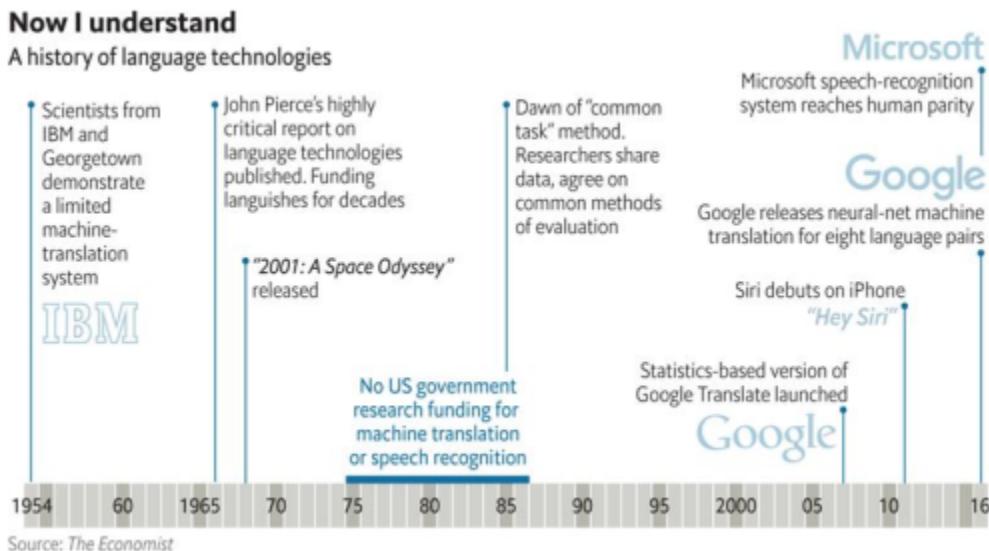


Figure 1.2: Il passato delle tecnologie del linguaggio naturale.

#### 1.1.1 La Complessità del Linguaggio Naturale

C'è un legame tra linguaggio umano e intelligenza. Già Turing sosteneva che se si potesse parlare in un certo modo si fosse intelligenti (test di Turing). La differenza tra il linguaggio umano e un linguaggio di programmazione è l'*ambiguità*: C o Java non sono ambigui.

##### Il linguaggio umano:

- *Discretezza* (esistenza di elementi):
  - Api: Ritmo, orientamento, durata.
  - Esseri umani: Fonemi, morfemi, parole.
- *Ricorsività*:
  - Scimpanze: Gesti atomici.
  - Uomo: Gianni vede Pietro, Maria vuole che Gianni veda Pietro, Paolo crede che Maria voglia che Gianni veda Pietro.
- *Dipendenza dalla struttura*:
  - Non "una parola dietro l'altra" ma c'è una struttura: La ragazza parte, I ragazzi di cui mi ha parlato la ragazza partono.
- *Località*:
  - Gianni lo ha guardato.
  - Gianni ha detto che Pietro lo ha guardato.

### Intelligenza e linguaggio nel il test di Turing:

- Possono le macchine pensare?
- Se riesco a parlare come un essere umano allora penso.
- Gioco dell'imitazione: un giudice deve capire se quello che ha davanti è un uomo oppure un computer.

**Note:-**

Ci sono una serie di obiezioni a questo test: teologia, matematica, coscienza, etc.

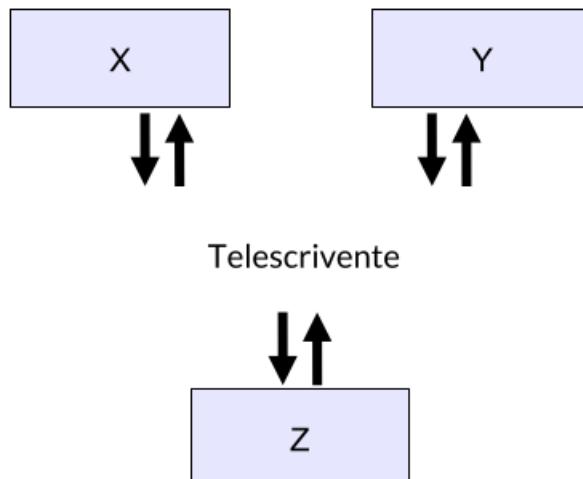


Figure 1.3: Il gioco dell'imitazione.

Nel 1966, Weizenbaum crea Eliza. Una macchina in grado di "comprendere" e ingannare gli esseri umani.

**Note:-**

Il punto debole del test di Turing e di Eliza è il giudice: se è coinvolto emotivamente potrebbe far passare un computer per un essere umano<sup>a</sup>.

<sup>a</sup>Blade runner moment

#### Definizione 1.1.1: Winograd Schema

Evoluzione del Turing test: un test a scelta multipla che utilizza domande con una specifica struttura. In questi test gli esseri umani sono molto bravi a rispondere, i computer no.

**Note:-**

Rimuove il giudizio, quindi tecnicamente più accurato.

#### Corollario 1.1.1 Captcha

Un test di Turing inverso per capire se l'interlocutore è umano. Non c'è linguaggio, ma riconoscimento cognitivo.

#### Corollario 1.1.2 Voight-Kampff Test

Test in Blade runner basato sulle emozioni, evoluzione del test di Turing.

### 1.1.2 I Livelli di Conoscenza del Linguaggio

HAL 9000, in "2001: Odissea nello spazio" mostra un esempio di comunicazione.

#### Domanda 1.1

Come fa HAL a rispondere?

- Riconoscimento vocale.
- Comprensione del linguaggio naturale.
- Generazione del linguaggio naturale.
- Sintesi vocale.
- Recupero ed estrazione di informazioni.
- Inferenza.

#### Livelli della conoscenza:

1. Il suono: HAL deve essere in grado di analizzare e produrre dei segnali audio che contengono le parole: foni e fonemi.
2. Le parole: HAL deve essere in grado di riconoscere le singole parole.
3. Raggruppare le parole: HAL deve essere in grado di distinguere la struttura della frase.
4. Significato: HAL deve conoscere il significato delle singole parole e deve essere in grado di comporre questi significati per trovare il significato complessivo della frase.
5. Contesto e scopi: HAL deve avere delle conoscenze del mondo che gli permettono usare il linguaggio in maniera contestuale: *I'm afraid, I can't* invece di *I won't*.
6. Conversazione: HAL deve avere deve essere in grado di conversare, dando delle risposte e facendo delle domande pertinenti al discorso.

#### A ogni livello corrisponde una parte del linguaggio:

1. Fonetica e Fonologia: lo studio del suono della lingua.
2. Morfologia: lo studio delle parti significative delle parole.
3. Sintassi: lo studio sulla struttura e sulle relazioni tra le parole.
4. Semantica: lo studio del significato.
5. Pragmatica: lo studio di come il linguaggio è usato per compiere goal. Il passivo serve per mettere in luce/enfatizzare alcune parti della frase.
6. Discorso: lo studio delle unità linguistiche rispetto alla singola dichiarazione.

#### Note:-

Jurafsky è un chad nerd.

### 1.1.3 Strutture Linguistiche e Ambiguità

Analizzando i vari livelli si trovano diverse *strutture linguistiche*.

#### Definizione 1.1.2: Struttura Linguistica

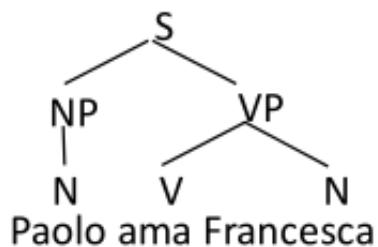
Una struttura è un insieme su cui è definita una relazione:

- Relazione fonetico-fonologica sull'insieme dei foni-fonemi.
- Relazione morfologica sull'insieme dei morfemi.
- Relazione sintattica sull'insieme delle parole.
- Relazione semantica sull'insieme dei significati delle parole.
- Relazione pragmatica sull'insieme dei significati delle parole e sul contesto.
- Relazione “discorsale” sull'insieme delle frasi.

#### Note:-

Ci sono relazioni tra i componenti della frase. Inoltre le relazioni cambiano a seconda della lingua.

#### Esempio 1.1.1 (Struttura Sintattica)



#### Definizione 1.1.3: Ambiguità

Il linguaggio naturale presenta frasi che possono essere interpretate in modi differenti.

#### Esempio 1.1.2 (Ambiguità)

"I made her duck"

- Ho cucinato una papera per lei.
- Ho cucinato una papera che apparteneva a lei.
- Ho creato una papera con la stampante 3D e gliel'ho data a lei.
- Ho fatto abbassare la sua testa.
- In Harry Potter<sup>a</sup>: Ho trasformato lei in una papera.

<sup>a</sup>Rowling merda.

### Osservazioni 1.1.1

- Le parole "duck" e "her" sono morfologicamente ambigue nella loro parte del discorso. "Duck" può essere un verbo o un nome, "her" può essere un pronome dativo o possessivo.
- Il verbo "make" è sinteticamente ambiguo: può essere transitivo o intransitivo.
- Inoltre "make" è anche semanticamente ambiguo: può significare creare o cucinare.
- In una frase parlata c'è un altro livello per cui "her" può essere udito come "eye" e "make" come "maid".

**Note:-**

Essere ambigui permette di essere brevi e concisi.

### Altre proprietà notevoli del linguaggio:

- Linguaggio non standard, evolve nel tempo.
  - Scialla bros → chill → è easy.



- Segmentazione.
  - Il treno Torino San Remo.
- Locuzioni, spesso l'interpretazione non è compositiva.
  - Pollica verde.
- Neologismi.
  - Twettare<sup>2</sup>
- Conoscenza del mondo.
  - Lucia e Carola erano sorelle.
  - Lucia e Carola erano madri.
- Meta-linguaggio.
  - La prima cosa bella ha avuto un grandissimo successo.

### 1.1.4 Lo Stato dell'Arte

- 1976: In Canada un sistema riesce a stampare due bollettini meteo in due lingue diverse.
- BabelFish, di Yahoo, era un sistema "a regole" di trascrizione automatica, basato su Systran.
- 2011: IBM costruisce un supercomputer per battere un essere umano a Jeopardy, Watson.
- Tecnologie vocali: Speech Recognition, TextToSpeech, HTML5 Speech API (pagine web vocali).

<sup>2</sup>Musk merda.

**Note:-**

Dopo sette milioni e mezzo di anni Pensiero Profondo fornisce la risposta: "42"<sup>a</sup>.

<sup>a</sup>Guida Galattica per gli Autostoppisti.

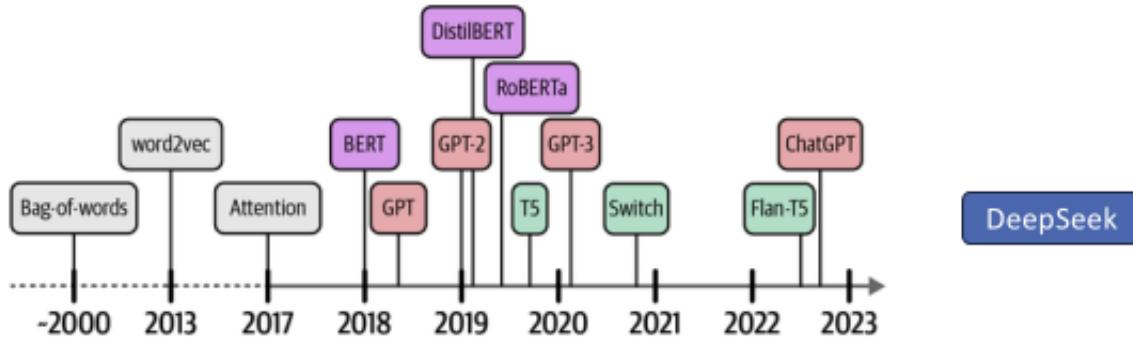


Figure 1.4: LLM. Tratto da "Hands-On Large Language Models", uscito nel Dicembre del 2024.

**Note:-**

Well, Deepseek è open source e funziona meglio di ChatGPT (a patto che non chiedi cosa sia successo a piazza Tienanmen nel 1989).

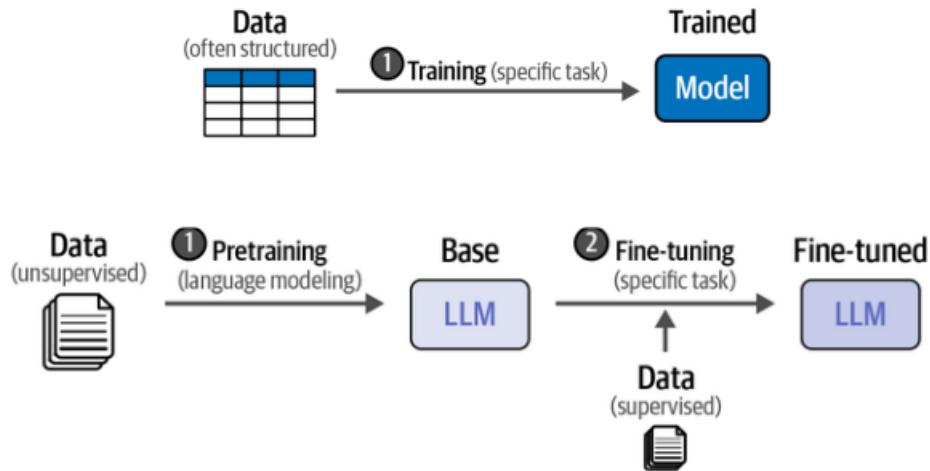


Figure 1.5: Shifting di paradigma dovuto al Machine Learning.

**Definizione 1.1.4: AI Generativa**

Modello di linguaggio di reti neurali multi-task basate sui transformer addestrati su una grande quantità di dati utilizzando self training e feedback umano.

- Modello di Linguaggio: Text prediction → T9.
- Multi-task: Google Translator, Siri.

### Domanda 1.2

Come fare un LLM (M. Lapata)?

1. Collezionare una grande quantità di dati.
2. Chiedere al LLM di predire la nuova parola in una frase.
3. Ripetere il tutto.

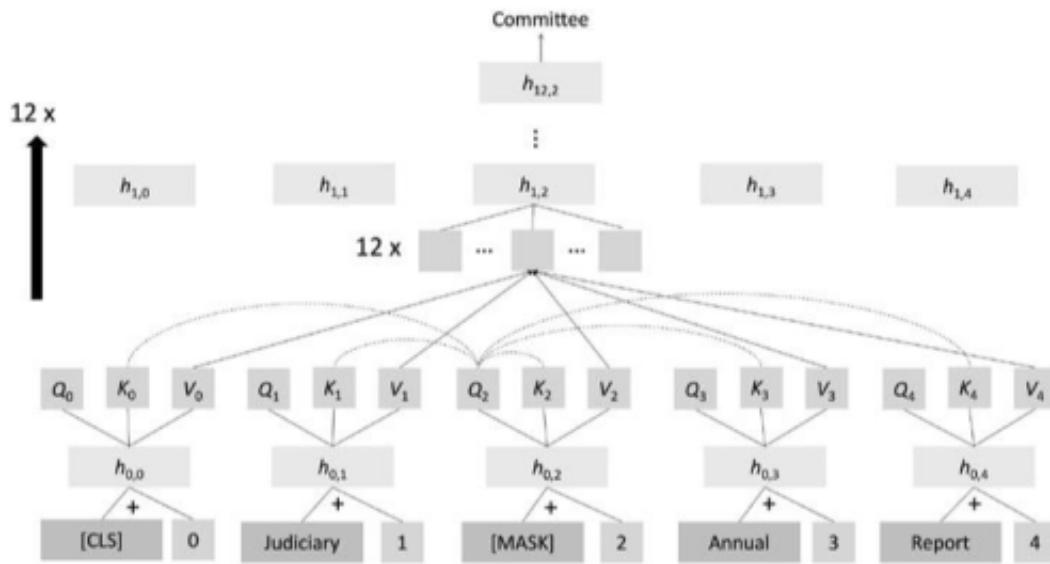


Figure 1.6: Auto addestramento di una rete neurale.

### Domanda 1.3

Come usare un LLM?

- Sintonizzazione a grana fine.
- Prompting.

Si può usare un LLM per:

- Search Engine.
- Writer/Code assistant.

**Note:-**

Noam Chomsky odia questi sistemi. Secondo lui servono per evitare l'apprendimento.

**DeepSeek:**

- Apprendimento rinforzato automatico (senza essere umani).
- Meno costoso → politicamente importante.

**Il problema fondamentale:** Convertire una frase o un testo in una forma che permetta l'applicazione di meccanismi di ragionamento automatico.

## 1.2 I Livelli Linguistici

### 1.2.1 Da Frase a Significato

**Problema:** Convertire una frase o un testo in una forma che permetta l'applicazione di meccanismi di ragionamento automatico.

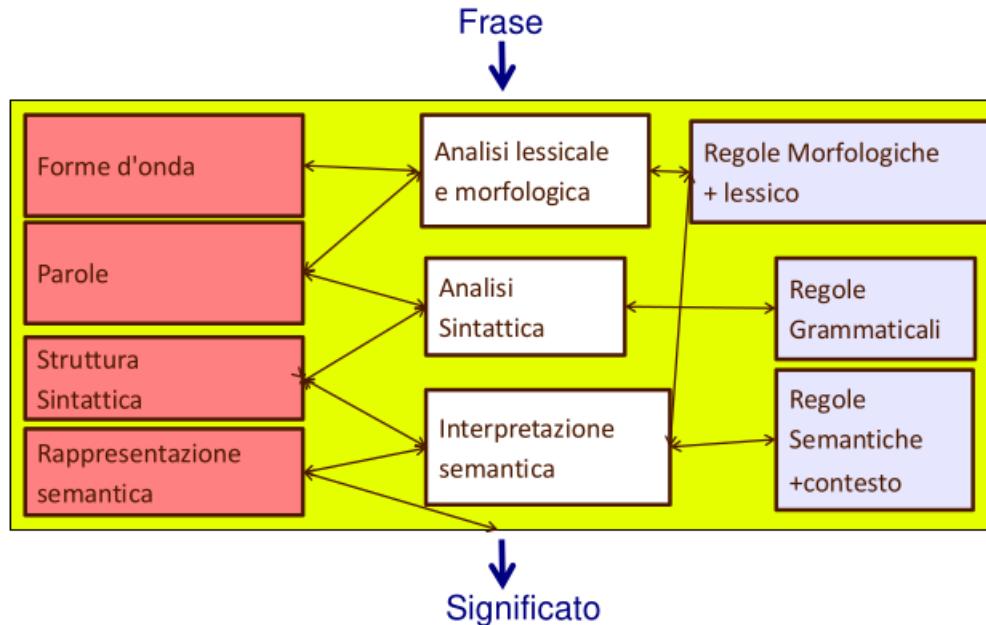


Figure 1.7: Passaggio da frase a significato.

**Note:-**

Però la situazione non è così semplice. Bisogna capire come funzionano i moduli e come comunicano

Nella linguistica computazionale c'è una divisione tra **regole** e **statistica**:

- Rules-driven.
- Data-driven.

**Note:-**

Steedman sostiene che i due aspetti dovrebbero convivere tra loro (2008). Evitare il regionamento tribale.

**Domanda 1.4**

Quando finisce una frase?

**Definizione 1.2.1: Sentence splitting**

Task in cui si deve capire quando una frase finisce.

- "!", "?" → Okay, pongono fine alla frase.
- ".":
  - Fine frase.
  - Abbreviazione (Doc., Mx.).
  - Numeri (0.2).

### Domanda 1.5

Quindi come si costruisce un classificatore binario che decida EoS (End of String) o not EoS?

- Si possono scrivere regole a mano:
  - Espressioni regolari.
  - Tokenizer (FA) e regole.
- Addestrare un sistema di machine learning.

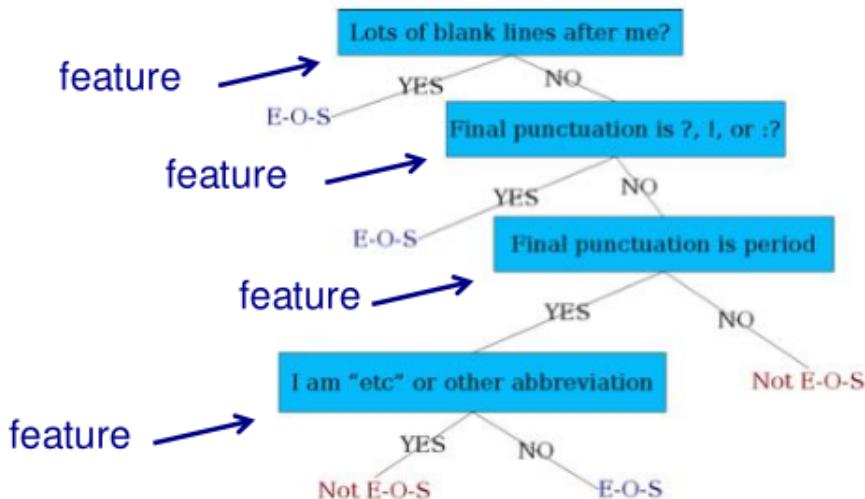


Figure 1.8: Albero di decisione.

### Features più complesse:

- Caso di parole con ".".
- Caso di parole dopo ".".
- Features numeriche:
  - Lunghezza di parole con ".".
  - Probabilità che una parola con "." avvenga alla fine della frase.
  - Probabilità che una parola dopo "." avvenga all'inizio di una frase.

### Domanda 1.6

Cos'è davvero un albero di decisione?

- Una serie di IF-THEN-ELSE encapsulati.
- Due possibilità per costruirlo:
  - *By-hand*: solo in contesti semplici.
  - *Machine learning*: su un training corpus.
- Il punto cruciale è la scelta delle features.

**Osservazioni 1.2.1**

- In questo corso ci concentreremo sullo studio delle feature linguistiche.
- In alcuni casi l'approccio by-hand verrà privilegiato poiché è didatticamente più chiaro/semplice e poiché è più semplice verificarne la fondatezza cognitiva mediante introspezione.

**Domanda 1.7**

Nei sistemi end-to-end cosa sono le feature linguistiche?

**Definizione 1.2.2: Features Linguistiche Neurali**

L'architettura neurale, ovvero il numero e il tipo di connessioni, codifica in maniera *implicita* le features linguistiche.

**Note:-**

La ricerca, in questo caso, si focalizza su quale scelta architettonica è più adatta alla modellazione implicita del fenomeno linguistico e alla creazione del corpus di training.

**1.2.2 Il Livello Morfologico e l'Analisi Lessicale**

Il lessico è fondato sul concetto di *parola*.

**Domanda 1.8**

Che cos'è una parola?

- Intuitivamente è una sequenza di caratteri delimitata da spazi o punteggiatura.
- Sequenze di più parole, Es. passammela = passa a me essa.
- Le parole hanno un significato unitario (semantica lessicale), ma volte sequenze di parole hanno un significato unitario. Es. di corsa, by the way.
- In altre lingue il problema è più grave.
  - In tedesco: Lebenversicherungsgesellschaftangestellter = impiegato di una società di assicurazione sulla vita.
  - In inglese: Wouldn't? = Would not.

**Presenza di suffissi:**

- CAPITANO (forma non declinabile).
- CAPITAN + O (nome o aggettivo o forma del verbo capitare).
- CAPIT + ANO (forma del verbo capitare).

**Note:-**

Non c'è una forma giusta a priori, ma c'è una forma giusta in base al contesto.

**Definizione 1.2.3: Forme Composte**

Generalmente una parola contenuto più una (o più) parole funziona.

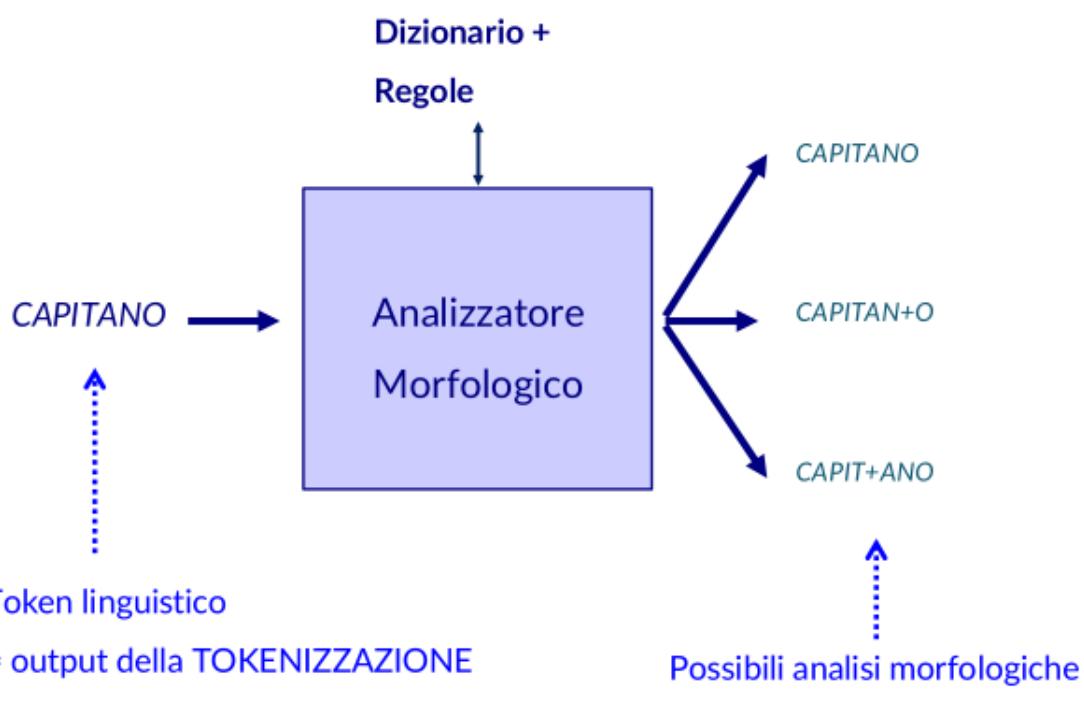


Figure 1.9: Analizzatore morfologico.

**Esempio 1.2.1** (Forme composte)

**STAMPAMELO:**

- STAMP è una radice verbale.
- A è un suffisso verbale.
- ME e LO sono forme pronominali<sup>a</sup>.

<sup>a</sup>Per triggerare gli Alt-Right.

**Definizione 1.2.4: Forme Multiple**

Le diversi componenti sono nel dizionario ma la semantica non è composituale.

**Esempio 1.2.2** (Forme multiple)

- Più o meno: puntatore tra le parole per recuperare la giusta semantica.
- Prendere un abbaglio: rimandare all'interprete semantico.

**Definizione 1.2.5: Lemmatizzazione**

Trasformare un lemma in forma normale.

**Note:-**

La forma normale non è stabile nel tempo.

**Definizione 1.2.6: Stemming**

Estrarre la forma radice (detta tema) da una parola.

**Definizione 1.2.7: Paradigmatico**

Si cambia una parte della parola con una equivalente si ha una frase morfologicamente corretta.

**Definizione 1.2.8: Sintagmatico**

I rapporti che intercorrono tra gli elementi che si succedono nella frase i rapporti che intercorrono tra gli elementi che si succedono nella frase.

**Nome:**

- Persone, oggetti, luoghi.
- Proprietà sintagmatiche:
  - Comparire dopo gli articoli.
  - Avere un possessivo.
  - Avere un singolare o un plurale.
- Comuni, propri, di massa, contabili.

**Verbo:**

- Eventi, azioni, processi.
- Molte forme morfologiche.
  - Tempo.
  - Modo.
  - Numero.
- Tante categorie (ausiliari, modali, copula, etc.).

**Aggettivi:**

- Proprietà.

**Avverbi:**

- Modificano qualcosa, spesso verbi, ma anche altri avverbi o intere frasi.

**Note:-**

Nomi, verbi, aggettivi e avverbi sono *di contenuto*, che puntano a oggetti reali.

**Definizione 1.2.9: Classi Aperte**

Classi che aumentano o scompaiono nel tempo costantemente (nomi, verbi, aggettivi, avverbi).

**Definizione 1.2.10: Classi Chiuse**

Classi che aumentano o scompaiono con tempi lunghissimi.

**Note:-**

Un esempio di classi chiuse sono i pronomi: una volta, in inglese, la seconda persona singolare era "thou", attualmente "you" ha assunto sia il ruolo di seconda persona singolare che plurale.

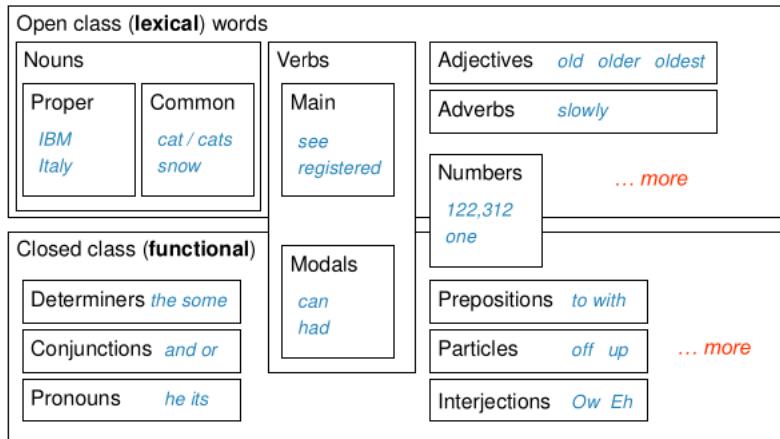


Figure 1.10: Parti aperte e parti chiuse.

**Google Unviversal PoS:** 12 PoS: NOUN (nouns), VERB (verbs), ADJ (adjectives), ADV (adverbs), PRON (pronouns), DET (determiners and particles), ADP (prepositions and postpositions), NUM (numerals), CONJ (conjunctions), PRT (particles), ‘.’ (punctuation marks) and X (a catch-all, e.g. abbreviations and foreign words).

### 1.2.3 Il Livello Sintattico

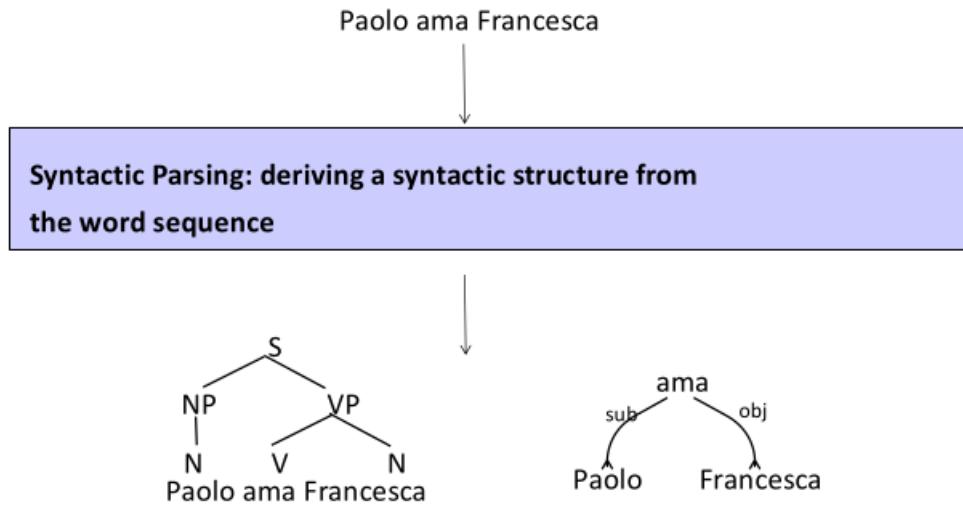


Figure 1.11: Parsing sintattico.

**Note:-**

Le due alternative derivano da prospettive diverse:

- Quella a sinistra è la struttura sintagmatica (o a costituenti).
- Quella a destra è la struttura a dipendenze (scuola di praga).

Entrambe le alternative sono equivalenti.

**Definizione 1.2.11: Costituenza**

La struttura della frase organizza le parole in costituenti annidati.

**Domanda 1.9**

Come si fa a sapere cos'è un costituente?

- Distribuzione: un costituente si comporta come un'unità che compare in differenti parti della frase.
- Sostituzione: test per verificare un constituent

**Note:-**

La cosa interessante è automatizzare il processo della costruzione di alberi.

**Osservazioni 1.2.2**

- NP: la parola più importante sintatticamente è un nome.
- VP: la parola più importante sintatticamente è un verbo.
- PP-LOC: la parola più importante sintatticamente è una preposizione.

- VP -> ... VB\* ...
- NP -> ... NN\* ...
- ADJP -> ... JJ\* ...
- ADVP -> ... RB\* ...
- SBAR(Q) -> S|SINV|SQ -> ... NP VP ...
- Plus minor phrase types:
  - QP (quantifier phrase in NP), CONJP (multi word constructions: as well as), INTJ (interjections), etc.

Figure 1.12: Parti della sintassi.

**Note:-**

I costituenti si comportano come un'unità:

- Esperimento di Fodor-Bever.
- Esperimento di Bock-Loebell.

**Definizione 1.2.12: Context Free Grammar**

I CFG mettono in relazione i *simboli non terminali* e i *constituenti* (Chomsky).

### Definizione 1.2.13: X-barra

La teoria X-barra sostiene che se si costruisce un albero a costituenti con una determinata proprietà l'oggetto sarà presente internamente e il soggetto sarà presente esternamente.

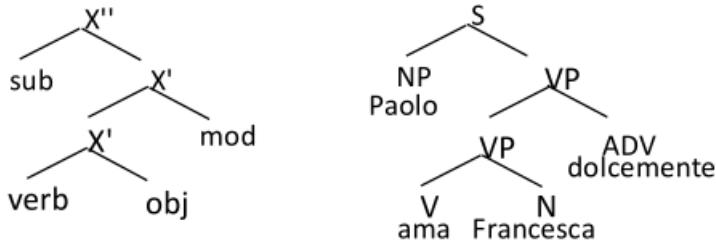


Figure 1.13: X-barra.

### Definizione 1.2.14: Dipendenza

Relazione tra due parole:

- *Head*: parola dominante.
- *Dipendenza*: parola dominata.

#### Note:-

La testa seleziona le sue dipendenze e determina le loro proprietà.

#### Corollario 1.2.1 Argomenti

Modificano in maniera sostanziale un evento (obbligatori).

#### Corollario 1.2.2 Modificatori

Modificano parzialmente un evento (facoltativi).

## 1.2.4 Il Livello Semantico

Esistono 2 approcci alla semantica lessicale:

- Classico.
- Distribuzionale (anni '60):
  - Statistico.
  - Neurale.

### Definizione 1.2.15: Semantica Lessicale Classica

Le connessioni sono legate al significato dei vari lessemi. La struttura interna dei lessemi è legata al significato.

#### Corollario 1.2.3 Lessema

Una coppia *forma-significato*, elemento del lessico.

**Note:-**

Il problema è che sono possibili definizioni ricorsive "infinite".

**Relazioni tra lessemi:**

- Omonimia: 2 lessemi con la stessa forma ortografica hanno due sensi diversi.
  - A bank can hold the investments.
  - We can go on the right bank of the river.
- Polisemia: lo stesso lessema ha due sensi diversi:
  - A bank can hold the investments.
  - He got the blood from the bank.
- Sinonimia: due lessemi con forma diversa hanno lo stesso senso (sostituibilità).
  - How big is that plane?
  - How large is that plane?
- Iponimia: due lessemi di cui uno denota una sottoclasse dell'altro:
  - Automobile è un iponimo di veicolo.
  - Veicolo è un iperonimo di automobile.

**Esempio 1.2.3 (Iponimia)**

- Quella è un automobile → quello è un veicolo.
- (?) Quello è un veicolo → quella è un automobile.

**Corollario 1.2.4 Syn-set**

Insieme di relazioni tra lessemi, usato per costruire le mappe in worldnet.

**Definizione 1.2.16: Semantica Distribuzionale (vettoriale)**

Il significato di una parola è collegato alla distribuzione delle parole attorno a sé.

**Esempio 1.2.4 (Semantica Distribuzionale)**

- A bottle of tesguino is on the table.
- Everybody likes tesguino.
- Tesguino makes you drunk.

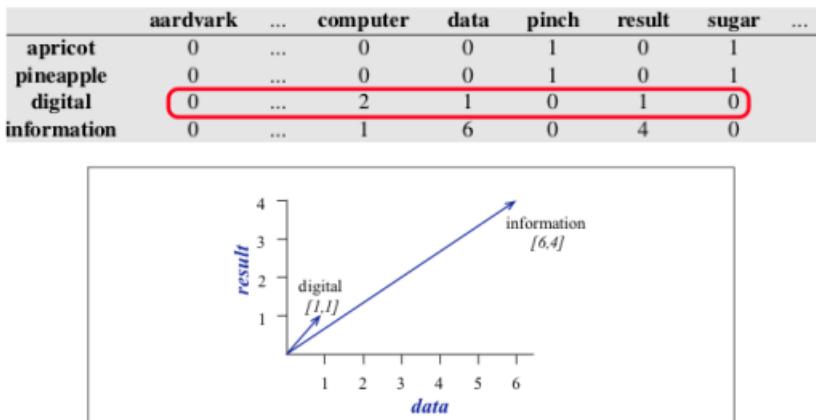
Si può inferire che *tesguino* sia un super alcolico.

**I vettori:**

- Lunghi (lunghezza 20.000-50.000).
- Sparsi (molti elementi sono zero).

### I lean vectors:

- Piccoli (lunghezza 200-1000).
- Densi (molti elementi sono non-zero).
- Vettori più corti sono più facili da usare come features nel ML.



**Figure 15.5** A spatial visualization of word vectors for *digital* and *information*, showing just two of the dimensions, corresponding to the words *data* and *result*.

Figure 1.14: Si può andare a determinare la "vicinanza" di parole con la semantica vettoriale.

### Osservazioni 1.2.3

- Con l'avvento delle reti neurali si ha un miglioramento.
- $\text{vector}(\text{'king'}) - \text{vector}(\text{'man'}) + \text{vector}(\text{'woman'}) = \text{vector}(\text{'queen'})$ .
- $\text{vector}(\text{'Paris'}) - \text{vector}(\text{'France'}) + \text{vector}(\text{'Italy'}) = \text{vector}(\text{'Rome'})$ .

### Definizione 1.2.17: Parole Contestualizzate

Costruire un vettore per ogni parola, condizionandolo al suo contesto. La rappresentazione per ogni token è una funzione dell'intera sequenza di input.

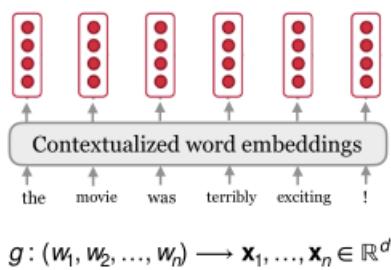


Figure 1.15: Contestualizzazione.

### Corollario 1.2.5 Semantica Composizionale

La semantica di un sintagma è funzione della semantica dei sintagmi componenti; non dipende da altri

sintagmi esterni al sintagma stesso.

**Note:-**

Conoscendo il significato di X, Y, e +, possiamo comporre il significato “X+Y”.

**Reasoning:**

- Deduzione: conseguenza logica.
- Induzione: basata su molti casi, si assume una regola generale.
- Abduzione: regionamento per indizi.

**Definizione 1.2.18: Metasemantica**

L'insieme di semantica compositiva e semantica lessicale distribuzionale. Serve per dare senso a parole sconosciute.

### 1.2.5 Il Livello Pragmatico e del Discorso

**Definizione 1.2.19: Pragmatica**

L'interpretazione di “io” (sottinteso) e “oggi” dipende da chi enuncia la frase e quando, rispettivamente.

**Note:-**

Il vero significato deve essere integrato da oggetti metalinguistici.

**Corollario 1.2.6 Anafora**

Sintagmi che si riferiscono a oggetti precedentemente menzionati.

**Esempio 1.2.5 (Anafora)**

- “La torta era sul tavolo. Giorgio la divorò”.
- “In giardino c'erano il cane e il gatto che giocavano con un pezzo di stoffa. Il felino lacero' la stoffa”.
- “Dopo essersi fidanzati, Giorgio e Maria trovarono un prete e si sposarono. Per la luna di miele, essi andarono ai Caraibi”.

Le **strutture dati** dei livelli:

- Livello morfologico e l'analisi lessicale: Lista.
- Livello sintattico: Alberi.
- Livello Semantico:
  - Semantica lessicale: Insiemi, Vettori.
  - Semantica formale: Logica, Alberi/Grafi.
- Livello paradigmatico e del discorso: Frame, Ontologie.



# 2

## Sequence Tagging

### 2.1 Part of Speech (PoS) Tagging

#### Definizione 2.1.1: PoS Tagging

Assegnare dei tag per distinguere le varie parti di una frase.

#### 2.1.1 Perché studiare PoS?

##### Domanda 2.1

Perché studiare PoS?

- Text-to-Speech: la pronuncia di alcune parole cambia in base alla loro parte nel discorso.
- Scrivere regexps: per cercare le frasi principali.
- Input per un parser completo.
- MT (Machine Translation): riordinare aggettivi e nomi nelle traduzioni.
- Si potrebbe volere distinguere tra aggettivi o altre parti del discorso.
- Si potrebbe voler studiare cambiamenti linguistici come la ceazione di nuove parole o shifting del significato.

##### Domanda 2.2

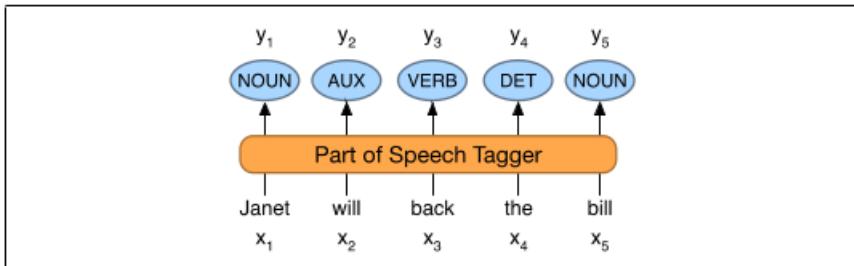
Quanto è difficile il PoS Tagging?

- 85% delle parole non sono ambigue.
- 15% delle parole sono ambigue e molto frequenti (il 60% delle parole che si ascoltano sono ambigue).

##### Domanda 2.3

Quanti tag sono corretti?

- Attualmente 97%.



**Figure 17.3** The task of part-of-speech tagging: mapping from input words  $x_1, x_2, \dots, x_n$  to output POS tags  $y_1, y_2, \dots, y_n$ .

Figure 2.1: Part of Speech Tagging.

- Una *baseline* del 92% è possibile con il metodo più banale:
  - Si dà un tag a ogni parola con il suo significato più frequente.
  - Si dà un tag nome alle parole sconosciute.

Tag	Description	Example
Open Class	<b>ADJ</b> Adjective: noun modifiers describing properties	<i>red, young, awesome</i>
	<b>ADV</b> Adverb: verb modifiers of time, place, manner	<i>very, slowly, home, yesterday</i>
	<b>NOUN</b> words for persons, places, things, etc.	<i>algorithm, cat, mango, beauty</i>
	<b>VERB</b> words for actions and processes	<i>draw, provide, go</i>
	<b>PROPN</b> Proper noun: name of a person, organization, place, etc..	<i>Regina, IBM, Colorado</i>
Closed Class Words	<b>INTJ</b> Interjection: exclamation, greeting, yes/no response, etc.	<i>oh, um, yes, hello</i>
	<b>ADP</b> Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	<i>in, on, by under</i>
	<b>AUX</b> Auxiliary: helping verb marking tense, aspect, mood, etc.,	<i>can, may, should, are</i>
	<b>CCONJ</b> Coordinating Conjunction: joins two phrases/clauses	<i>and, or, but</i>
	<b>DET</b> Determiner: marks noun phrase properties	<i>a, an, the, this</i>
	<b>NUM</b> Numeral	<i>one, two, first, second</i>
	<b>PART</b> Particle: a preposition-like form used together with a verb	<i>up, down, on, off, in, out, at, by</i>
	<b>PRON</b> Pronoun: a shorthand for referring to an entity or event	<i>she, who, I, others</i>
	<b>SCONJ</b> Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	<i>that, which</i>
Other	<b>PUNCT</b> Punctuation	<i>:, ;, ., !, ?</i>
	<b>SYM</b> Symbols like \$ or emoji	<i>\$, %</i>
	<b>X</b> Other	<i>asdf, qwfg</i>

Nivre et al. 2016

Figure 2.2: Tagset.

### Esempio 2.1.1 (*Janet will back the bill*)

- Probabilità a parole: "will" è di solito un ausiliario.
- Identità delle parole adiacenti: "the" implica che la prossima parola probabilmente non è un verbo.
- Morfologia e forma delle parole:
  - Prefissi.
  - Suffissi.
  - Capitalizzazione.

### Algoritmi di supervised learning:

- Hidden Markov Models (programmazione dinamica).
- Conditional Random Fields (CRF) / Maximum Entropy Markov Models (MEMM).
- Natural Sequence Models (RNNs o transformers).
- Large Language Models.

### 2.1.2 Analisi Basata su Regole

#### Idee di Base:

1. Assegnare tutti i possibili tags alle parole (analisi morfologica).
2. Rimuovere i tags in base a un *insieme di regole*.
3. Solitamente più di 1000 regole scritte a mano (ma possono essere apprese automaticamente).

### 2.1.3 ENGTWOL Lexicon

Word	POS	Additional POS features
smaller	ADJ	COMPARATIVE
entire	ADJ	ABSOLUTE ATTRIBUTIVE
fast	ADV	SUPERLATIVE
that	DET	CENTRAL DEMONSTRATIVE SG
all	DET	PREDETERMINER SG/PL QUANTIFIER
dog's	N	GENITIVE SG
furniture	N	NOMINATIVE SG NOINDEFDETERMINER
one-third	NUM	SG
she	PRON	PERSONAL FEMININE NOMINATIVE SG3
show	V	IMPERATIVE VFIN
show	V	PRESENT -SG3 VFIN
show	N	NOMINATIVE SG
shown	PCP2	SVOO SVO SV
occurred	PCP2	SV
occurred	V	PAST VFIN SV

Figure 2.3: Esempio di ENGTWOL.

- Utilizzare un analizzatore morfologico per ottenere tutte le parti del discorso.

Example: *Pavlov had shown that salivation ...*

Pavlov	PAVLOV N NOM SG PROPER
had	HAVE V PAST VFIN SVO
	HAVE PCP2 SVO
shown	SHOW PCP2 SVOO SVO SV
that	ADV
	PRON DEM SG
	DET CENTRAL DEM SG
	CS
salivation	N NOM SG

Figure 2.4: Primo passaggio.

- Si applicano i limiti.

Example: Adverbial "that" rule -> Given input: "that"

```

IF (and
    (+1 A/ADV/QUANT)          /* if next word is adj, adverb, or quantifier */
    (+2 SENT-LIM)             /* and following which is a sentence boundary, */
    (NOT -1 SVOC/A) )         /* and the previous word is not a verb like
                               'consider' which allows adjs as object
                               complements */
THEN
    eliminate non-ADV tags
ELSE
    eliminate ADV
  
```

Figure 2.5: Secondo passaggio.

## 2.2 PoS Tagger Statistici

### Domanda 2.4

Viene fornita una frase (un'osservazione o una sequenza di informazioni). Qual è la migliore sequenza di tags che corrisponde a questa sequenza di osservazioni?

**Vista probabilistica:**

- Considera tutte le possibili sequenze di tags.
- Di questo universo di sequenze viene scelta la sequenza più probabile.

**Terminologia:**

- *Modelling*: fornire un modello formale.
- *Learning*: un algoritmo per impostare i parametri del modello.
- *Decoding*: un algoritmo per applicare il modello per calcolare risultati.

### 2.2.1 HMM

Si vuole modellare, di tutte le sequenze di  $n$  tags  $t_1 \dots t_n$ , la sequenza di tags tale che  $P(t_1 \dots t_n | w_1 \dots w_n)$  è la maggiore.

$$\hat{t}_1^n = \operatorname{argmax} P(t_1^n | w_1^n)$$

**Note:-**

" $\hat{\cdot}$ " significa "la nostra stima del migliore".  
 $\operatorname{argmax}_x f(x)$  significa "la  $x$  che massimizza  $f(x)$ ".

### Domanda 2.5

Ma come si utilizza questa equazione?

**Inferenza Bayesiana:** si usa la regola di Bayes per trasformare quest'equazione in un insieme di probabilità che sono facilmente calcolabili.

**Note:-**

Nell'ipotesi di Markov sono presenti approssimazioni, per rendere il tutto più facile da calcolare.

$$\begin{aligned}
 \hat{t}_1^n &= \underset{t_1^n}{\operatorname{argmax}} \overbrace{P(w_1^n | t_1^n)}^{\text{likelihood}} \overbrace{P(t_1^n)}^{\text{prior}} \\
 P(w_1^n | t_1^n) &\approx \prod_{i=1}^n P(w_i | t_i) \\
 P(t_1^n) &\approx \prod_{i=1}^n P(t_i | t_{i-1}) \\
 \hat{t}_1^n &= \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \boxed{\underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-1})}
 \end{aligned}$$

Figure 2.6: Ipotesi di Markov, utilizzo dell'inferenza Bayesiana.

**Learning:**

## • PoS → PoS:

- Si calcola la probabilità che una determinata parte di una frase ne preceda un'altra.
- Esempio:  $P(NN|DT) = \frac{C(DT, NN)}{C(DT)}$  è la probabilità che un articolo (*DT*) preceda un nome (*NN*).
- In generale  $P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$ , dove  $C$  conta le occorrenze.

## • PoS → World:

- Calcola la probabilità che una certa parola assuma una certa valenza (likelihood, probabilità di verosimiglianza).
- Esempio:  $P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)}$ , la probabilità che un verbo alla terza persona presente (*VBZ*) sia *is*.
- $P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$ .

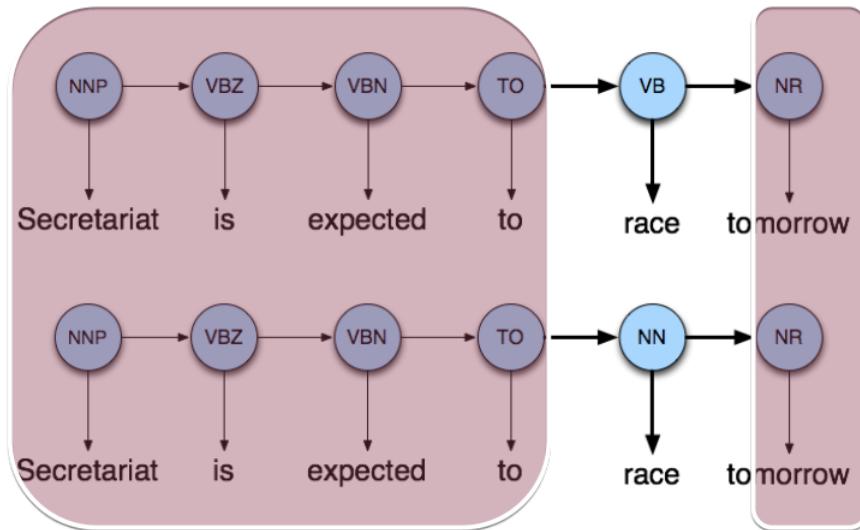


Figure 2.7: Disambiguare la parola "race".

**Note:-**

Si va a calcolare la probabilità della parola diversa.

**Domanda 2.6**

Come si fa a fare il decoding?

- L'algoritmo banale (che non funziona) porta a considerare tutte le possibili sequenze e andare a prendere la probabilità maggiore.
- Però con 30 tags e una frase di lunghezza media (20 parole) si hanno  $30^{20}$  casi.

Janet/NNP will/MD back/VB the/DT bill/NN

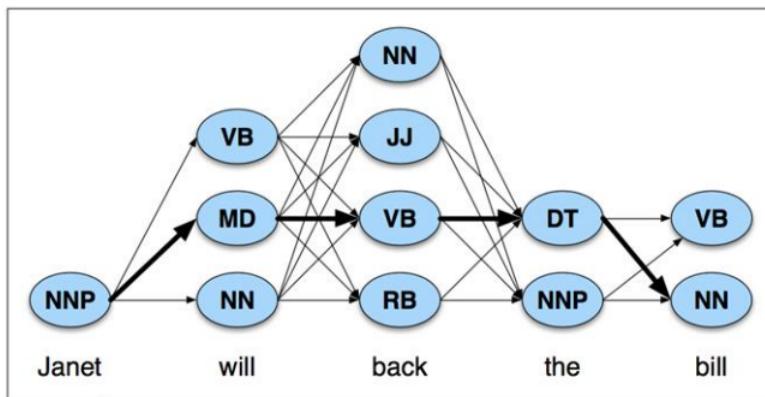


Figure 2.8: Tentativo di decoding.

**Idee della programmazione dinamica:**

- Si consideri una sequenza di stati (tag sequence) che termini con uno stato  $j$  con un particolare tag  $T$ .
- La probabilità che la tag sequence possa essere rotta in due parti:
  - La probabilità della migliore tag sequence attraverso  $j - 1$ .
  - Moltiplicata con la probabilità di transizione del tag alla fine della sequenza ( $j - 1$ ) rispetto a  $T$

**Sommario di Viterbi:**

- Crea una matrice:
  - Con colonne corrispondenti agli input.
  - Con righe corrispondenti ai possibili stati (PoS\_Tag +  $S_{ini} + S_{fin}$ ).
- Si attraversa la matrice riempendo le colonne a destra con i valori immediatamente a sinistra.
- Ogni cella della matrice,  $v_t(j)$ , rappresenta la probabilità che HMM sia nello stato  $j$  dopo aver visto le prime  $t$  parole e attraversato la sequenza di stati più probabile  $q_1, \dots, q_{t-1}$ .
- $v_t(j) = \max_{i=1}^N v_{t-1}(i)a_{i,j}b_j(o_t)$ .

Con i simboli:

- $v_{t-1}(i)$ : la probabilità della precedente iterazione dell'algoritmo di Viterbi.
- $a_{i,j}$ : la probabilità di transizione da un precedente stato  $q_i$  allo stato corrente  $q_j$ .



Figure 2.9: Transition probability or something, idk.

- $b_j(o_t)$ : la likelihood dello stato di osservazione del simbolo  $o_t$  dato lo stato corrente  $j$ .

```
function VITERBI(observations of len  $T$ ,state-graph of len  $N$ ) returns best-path
    create a path probability matrix viterbi[ $N+2,T$ ]
    for each state  $s$  from 1 to  $N$  do ; initialization step
        viterbi[ $s,1$ ]  $\leftarrow a_{0,s} * b_s(o_1)$ 
        backpointer[ $s,1$ ]  $\leftarrow 0$ 
    for each time step  $t$  from 2 to  $T$  do ; recursion step
        for each state  $s$  from 1 to  $N$  do
            
$$\begin{aligned} \textit{viterbi}[s,t] &\leftarrow \max_{s'=1}^N \textit{viterbi}[s',t-1] * a_{s',s} * b_s(o_t) \\ \textit{backpointer}[s,t] &\leftarrow \operatorname{argmax}_{s'=1}^N \textit{viterbi}[s',t-1] * a_{s',s} \end{aligned}$$

        viterbi[ $q_F, T$ ]  $\leftarrow \max_{s=1}^N \textit{viterbi}[s, T] * a_{s,q_F}$  ; termination step
        backpointer[ $q_F, T$ ]  $\leftarrow \operatorname{argmax}_{s=1}^N \textit{viterbi}[s, T] * a_{s,q_F}$  ; termination step
    return the backtrace path by following backpointers to states back in time from backpointer[ $q_F, T$ ]
```



Figure 2.10: Algoritmo di Viterbi.

**Note:-**

La chiave di programmazione dinamica è che abbiamo bisogno solo memorizzare il percorso prob MAX in ogni cella e non in tutti percorsi.

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \left[ \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-1}, t_{i-2}) \right] P(t_{n+1} | t_n)$$

$$P(t_i | t_{i-1}, t_{i-2}) = \frac{C(t_{i-2}, t_{i-1}, t_i)}{C(t_{i-2}, t_{i-1})}$$

Figure 2.11: Estensione a due tag.

Un tag può non dipendere solo dal precedente, ma a volte ne servono due:

**Note:-**

Tuttavia molte coppie sono linguisticamente inesistenti e per ciò avranno probabilità 0. Il problema è la presenza di "falsi zeri" dovuti al fatto che si lavora con un dataset finito.

#### Definizione 2.2.1: Sparsness

La sparsness di una matrice si riferisce alla proporzione di elementi nulli rispetto al totale degli elementi della matrice. Una matrice è considerata sparsa (sparse matrix) se la maggior parte dei suoi elementi è uguale a zero:

$$S = \frac{\text{numero di elementi nulli}}{\text{numero totale di elementi}} = \frac{\#\text{zeri}}{m \times n}$$

Problemi degli HMM:

- Quanto il corpus sia grande o sparso.
- Come si può valutare la probabilità per parole sconosciute?
  - Si possono usare alcune features: suffissi, capitalizzazione, etc.

#### 2.2.2 MEMM e CRF

Si può identificare un insieme di features su cui effettuare un modello probabilistico.

$$\begin{aligned} \hat{T} &= \underset{T}{\operatorname{argmax}} P(T | W) \\ &= \underset{T}{\operatorname{argmax}} \prod_i P(t_i | w_{i-l}^{i+l}, t_{i-k}^{i-1}) \\ &= \underset{T}{\operatorname{argmax}} \prod_i \frac{\exp \left( \sum_i w_i f_i(t_i, w_{i-l}^{i+l}, t_{i-k}^{i-1}) \right)}{\sum_{t' \in \text{tagset}} \exp \left( \sum_i w_i f_i(t', w_{i-l}^{i+l}, t_{i-k}^{i-1}) \right)} \end{aligned}$$

Figure 2.12: Il Maximum Entropy Markov Model (MEMM).

**MEMM:**

- Assegna le features alla parola stessa (non ha memoria).
- Costruisce un classificatore per predire il tag.

**Apprendimento per MEMM:**

- Regressione logistica multinomiale.
- Viene scelto il parametro  $w$  che massimizza la probabilità dell'etichetta  $y$  nei dati riguardanti un osservazione  $x$ .
- Per via dell'utilizzo di pesi è più lento dei modelli generativi e ha un comportamento random.
- Per il decoding si può usare il viterbi con alcune correzioni.

**Definizione 2.2.2: Conditional Random Fields (CRF)**

I CRF utilizzano features globali su un'intera sequenza. Sono modelli grafici molto complessi e potenti.

**Note:-**

In NLP sono catene lineari dove vengono accoppiate variabili ed etichette per tokens adiacenti.

**Osservazioni 2.2.1**

- In un CRF la funzione  $F$  mappa un'intera sequenza  $X$  e un'intera sequenza di output  $Y$  su un vettore di features.
- A differenza del modello MEMM si calcola il peso sull'intera sequenza.

**2.2.3 Tagging di Parole Sconosciute****Possibili approcci per gestire parole sconosciute in HMM:**

- Assumere che sono nomi.
- Assumere una distribuzione uniforme nel PoS.
- Dizionari esterni.
- Usare informazioni morfologiche (per esempio -are, -ere, -ire per i verbi in italiano).
- Assumere che le parole sconosciute abbiano una probabilità di distribuzione simile alle parole che sono comparse nel training set.

**2.3 NER Tagging****Definizione 2.3.1: Named Entity**

Una Named Entity è qualsiasi cosa che può essere indicata da un nome proprio. I 4 tags più comuni sono:

- PER (Person).
- LOC (Location).
- ORG (Organization).
- GPE (Geo-Political Entity).

**Note:-**

Spesso sono multi-word.

**Definizione 2.3.2: NER Tagging**

Il task di name entity recognition (NER) consiste in:

1. Trovare accorgimenti che costituiscono nomi propri.
2. Taggando il tipo di entità.

**Domanda 2.7**

Perché si usa NER?

- *Sentiment analysis*: i sentimenti del consumatore nei confronti di una particolare compagnia persona.
- *Rispondere alle domande*: su un'entità.
- *Estrazione di informazioni*: riguardo un'entità da un testo.

**Domanda 2.8**

Perché il NER è più difficile del PoS?

- Segmentation: bisogna dividere in segmenti che costituiscono le entità.
- Ambiguità dei tipi (la stessa named entity può assumere significati diversi).

**Definizione 2.3.3: BIO Tagging**

BIO (Begin Inside Out) è una tecnica per effettuare il NER Tagging. Si utilizza:

- B per indicare che sta iniziando una named entity.
- I per indicare che si è dentro una named entity.
- O per indicare che si è fuori da una named entity.

**Avendo  $n$  tipi di entità diversi si avranno:**

- 1 O tag.
- $n$  B tags.
- $n$  I tags.

**Note:-**

Quindi con  $n$  entità si utilizzeranno  $2n + 1$  tags diversi.



# 3

## Sintassi

### 3.1 Prologo

La sintassi è il fenomeno che permette di percepire come corrette certe frasi e come scorrette altre. I linguisti separano la sintassi in:

- Competence.
- Performance.

#### Definizione 3.1.1: Competence

La competence rappresenta la grammatica formale. Una conoscenza pura linguistica.

#### Note:-

La linguistica dovrebbe occuparsi della competence.

#### Definizione 3.1.2: Performance

La performance rappresenta un algoritmo di parsing. Come si utilizza la conoscenza pura.

#### Note:-

È importante conoscere la grammatica formale per poterla utilizzare con un algoritmo.

### 3.1.1 Grammatiche Generative

#### Definizione 3.1.3: Sistemi di Riscrittura

Sistema formale della tradizione matematica utilizzato come base da Chomsky.

- Le grammatiche generative modellano il linguaggio naturale come un linguaggio formale.
- L'albero di derivazione può modellare la struttura sintattica della frase.

Le grammatiche context free:

- Costituenza: i costituenti rappresentano i simboli non terminali V.
- Relazione grammaticale.
- Sottocategorizzazioni.


 $\Sigma = \text{alphabet}$ 
 $G = (\Sigma, V, S, P)$ 
 $V = \{A, B, \dots\}$ 
 $S \in V$ 
 $P = \{\Psi \rightarrow \theta, \dots\}$ 

Figure 3.1: Grammatiche Generative.

**Note:-**

Chomsky dimostrò che le lingue naturali sono *almeno* context free.

$$\begin{aligned} S \Rightarrow & NP \ VP \Rightarrow I \ VP \Rightarrow I \ V_1 S \Rightarrow \\ I \ saw \ S \Rightarrow & I \ saw \ NP \ VP \Rightarrow \\ I \ saw \ Harry \ VP \Rightarrow & \\ I \ saw \ Harry \ V_2 \Rightarrow & \\ I \ saw \ Harry \ swimming & \end{aligned}$$

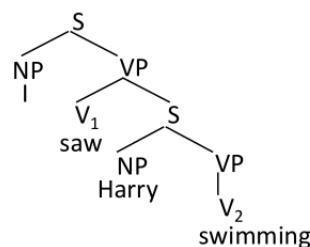
$$\begin{aligned} S \rightarrow & NP \ VP \\ VP \rightarrow & V_1 S \\ VP \rightarrow & V_2 \\ NP \rightarrow & I | John | Harry | Anna \\ V_1 \rightarrow & saw | see \\ V_2 \rightarrow & swimming \end{aligned}$$


Figure 3.2: Esempio con grammatica giocattolo.

**Note:-**

Questa gerarchia mette in "ordine" le complessità di vari tipi di linguaggi. Le lingue naturali si trovano nel *context-sensitive*, o meglio, nelle *mildly-context-sensitive* ( $a^n * b^n * c^n$ ).

### 3.1.2 Mildly-context-sensitive

Le grammatiche **mildly-context-sensitive** possiedono quattro proprietà:

- Includono le grammatiche Context-free.
- Hanno dipendenze annidate e incrociate.
- Sono parsificabili polinomialmente.
- Hanno la proprietà di crescita costante.

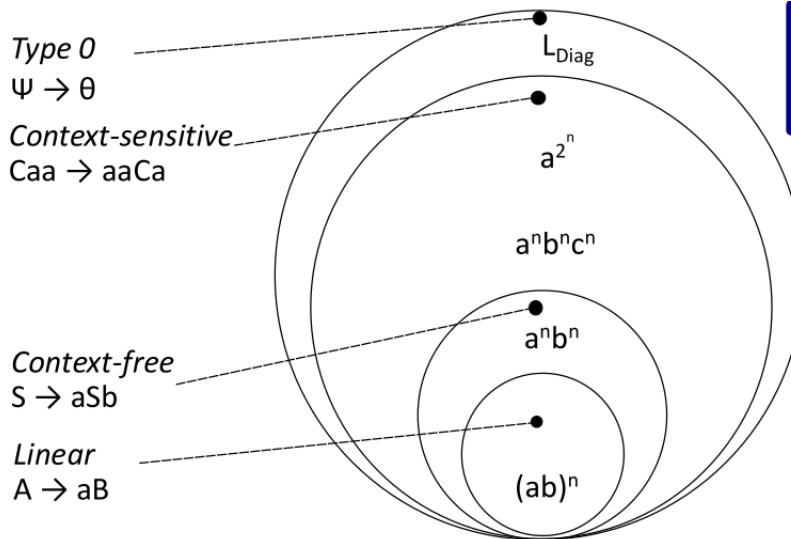


Figure 3.3: Gerarchia di Chomsky.

#### Definizione 3.1.4: Proprietà di Crescita Costante

Un linguaggio  $L$  cresce costantemente se c'è una costante  $c_0$  e un insieme finito di costanti  $C$  tale che per ogni  $w \in L$  dove  $|w| > c_0$  tale che  $|w| = |w'| + c$  per qualche  $c \in C$ .

#### Note:-

Questa proprietà è la versione formale dell'intuizione linguistica che una frase appartenente a un linguaggio naturale può essere costruita da un insieme finito di strutture usando la stessa operazione lineare.

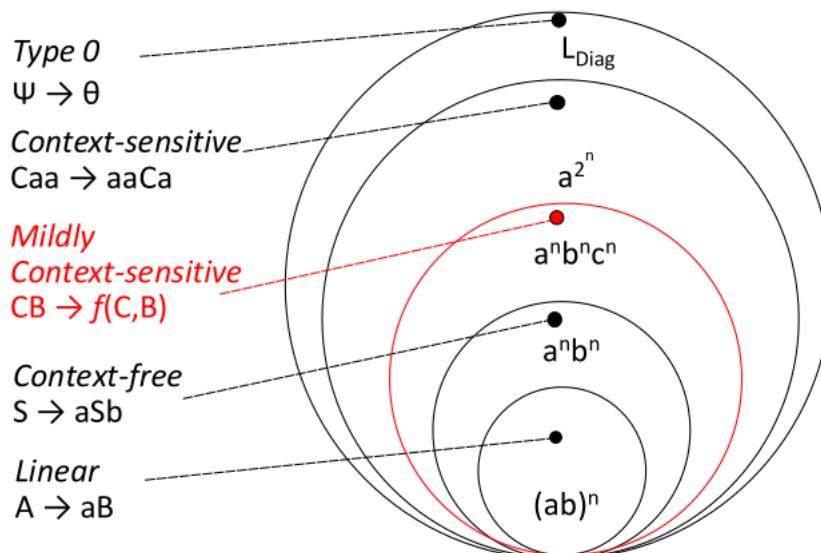


Figure 3.4: Gerarchia di Chomsky con mildly-context-sensitive.

## 3.2 La Performance e i costituenti

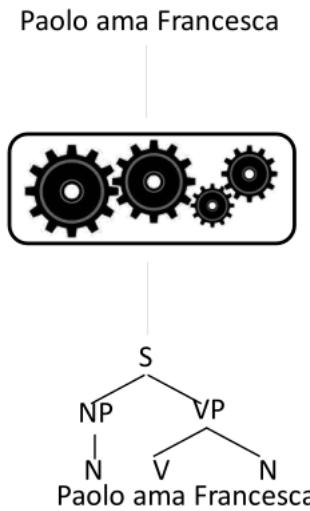


Figure 3.5: Un parser trasforma la frase in un albero.

### 3.2.1 Approcci al Parsing

**Parser anatomy (Steedman):**

- Competence: Context-free, TAG, CCG, Dipendenza, etc.
- Algoritmo:
  - Strategia di ricerca: top-down, bottom-up, left-to-right, etc.
  - Organizzazione della memoria: back-tracking (approccio in profondità, PROLOG), programmazione dinamica (approccio in ampiezza).
- Oracolo: probabilistico, a regole, etc.

**Note:-**

L'oracolo serve per via dell'ambiguità delle lingue naturali. Decide quale regola applicare prima secondo suoi criteri.

**Fonti di informazioni:**

- Grammatica → parsing diretto dai goal → top-down.
  - Solo ricerche che possono portare a risposte corrette, cioè con radice S.
  - Comporta la creazione di alberi non compatibili con le parole.
  - Razionalisti.
- Frase → parsing diretto dai dati → bottom-up.
  - Solo ricerche compatibili con le parole.
  - Comporta la creazione di alberi non corretti.
  - Empiristi.

**Parser A:**

- Context-free.
- Top-down, left-to-right, back-tracking.
- Rule-based.

**Problemi del parser A:**

- È lento: prova tutte le combinazioni e fa back-tracking.
- Va in loop: a causa della ricorsione se sono presenti regole che vanno in sé stesse.

**Domanda 3.1**

Come ri gestisce l'esplosione combinatoria dei sottoalberi?

**Con la programmazione dinamica (Richard Bellman):**

- Sottostrutture ottimali.
- Sottoproblemi sovrapponibili.
- Memoizzazione.

**Definizione 3.2.1: CKY**

Calcola tutti i possibili parse in tempo  $O(n^3)$ . Il difetto maggiore è che il caso peggiore e il caso medio coincidono (ma anche esige una forma normale per la grammatica).

**Note:-**

Questo algoritmo fu scoperto da tre persone nel giro di un anno e mezzo: Cocke, Kasami e Younger.

**Parser B (CKY):**

- Context-free.
- Bottom-up, left-to-right, programmazione dinamica.
- Rule-based.

**Idea del CKY:**

- Salva i sottoalberi perché possono essere riutilizzati.
- Si rende in forma normale di Chomsky (solo regole binarie):
  - Si copiano tutte le regole binarie nella nuova grammatica, senza cambiarle.
  - Si convertono i terminali in non terminali.
  - Si binarizzano tutte le regole e si aggiungono alla nuova grammatica.

**A → BC:**

1. Se c'è un A allora c'è un B seguito da un C.
2. Se A va da  $i$  a  $j$ , c'è un  $k$  tale che  $i < k < j$ .
3. Cioè B va da  $i$  a  $k$  e C va da  $k$  a  $j$ .
4. Usiamo una matrice per mettere B in  $\text{matrix}[i, k]$ , C in  $\text{matrix}[k, j]$ , A in  $\text{matrix}[i, j]$ .
5. Loop su  $k$ .
6. Se il parsing ha successo S è in  $[0, n]$ .

```

function CKY-PARSE(words, grammar) returns table
    for j from 1 to LENGTH(words) do
        for all {A / A ! words[j] 2 grammar}
            table[j - 1, j] = table[j - 1, j] [ A ]
    for i from j - 2 down to 0 do
        for k i + 1 to j - 1 do
            for all {A / A ! BC 2 grammar and B 2 table[i, k] and C 2 table[k, j]}
                table[i, j] = table[i, j] [ A ]

```

Looping over the columns  
Filling the bottom cell  
Filling row *i* in column *j*  
Looping over the possible split locations between *i* and *j*.  
Check the grammar for rules that link the constituents in [*i, k*] with those in [*k, j*]. For each rule found store the LHS of the rule in cell [*i, j*].

Figure 3.6: Algoritmo del CKY.

**Complessità:**

- $O(n^3)$ .
- $O(n^5)$  nella versione probabilistica.
- Troppo lento per il real-time (motori di ricerca).

### 3.2.2 Parsing Probabilistico

**Parser C (CKY probabilistico):**

- Probabilistic Context-free.
- Bottom-up, left-to-right, programmazione dinamica.
- Probabilistico.

$$\begin{array}{ll}
 A \rightarrow BC \quad [p_A] & P(1,4,A) = p_A * P(1,2,B) * P(3,4,C) \\
 D \rightarrow BC \quad [p_D] & P(1,4,D) = p_D * P(1,2,B) * P(3,4,C)
 \end{array}$$

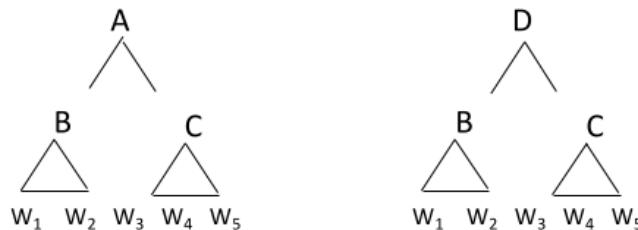


Figure 3.7: CKY probabilistico.

**Note:-**

Per fare ciò si realizza una distribuzione probabilistica sulle regole della grammatica.

**Domanda 3.2**

Da dove si prendono le probabilità?

- Treebanks → Banche di alberi:
  - Costituenti → Penn TB.
  - Dipendenze → TUT TB → Universal Dependency TB.
- Si creano con sistemi semiautomatici:
  - Prima con un parser.
  - Successivamente si corregge a mano.
- Guidelines di annotazione.
- Corpus Linguistics.

**Note:-**

I Treebanks sono più costosi del PoS.

**Training del modello probabilistico:**

- Serve  $P(\alpha \rightarrow \beta | \alpha)$ .
- Quindi per ogni albero nel TB si calcola:

$$P(\alpha \rightarrow \beta | \alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{Count}(\alpha \rightarrow \gamma)} = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

```
function PROBABILISTIC-CKY(words,grammar) returns most probable parse
   and its probability
   for j  $\leftarrow$  from 1 to LENGTH(words) do
      for all { A | A  $\rightarrow$  words[j] } in grammar
         table[j-1, j, A]  $\leftarrow P(A \rightarrow words[j])$ 
      for i  $\leftarrow$  from j-2 downto 0 do
         for k  $\leftarrow$  i+1 to j-1 do
            for all { A | A  $\rightarrow$  BC } in grammar,
               and table[i, k, B]  $> 0$  and table[k, j, C]  $> 0$ 
               if (table[i, j, A]  $< P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C]$ ) then
                  table[i, j, A]  $\leftarrow P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C]$ 
                  back[i, j, A]  $\leftarrow \{k, B, C\}$ 
   return BUILD_TREE(back[1, LENGTH(words), S]), table[1, LENGTH(words), S]
```

Figure 3.8: Algoritmo del CKY probabilistico.

**Note:-**

Attualmente si usano anche oracoli neurali dando vita a CKY neurali.

**3.2.3 Valutazione**

- Precision:
  - Quale percentuale di subtree del system tree sono anche nel gold tree?
  - Quanto di quello prodotto è giusto?
- Recall:
  - Quale percentuale di subtree del golden tree sono anche nel system tree?
  - Quanto di quello che avremmo dovuto produrre abbiamo realmente prodotto?

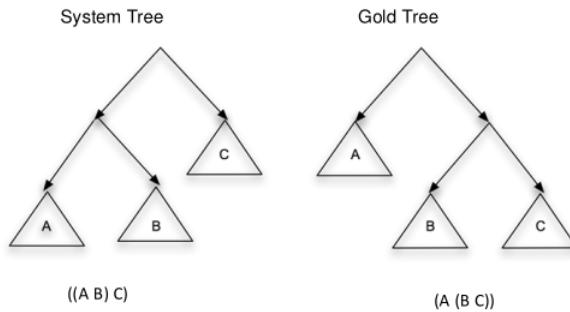


Figure 3.9: System tree e Gold tree.

**Problema con i parser probabilistici:**

- Non ci sono preferenze lessicali.

**Note:-**

Per questo motivo si utilizzano dei parser probabilistici *lexicalized* in cui si tiene anche conto dell'informazione lessicale. Purtroppo facendo così aumentano il numero di regole.

### 3.2.4 Parsing Parziale

**Parser E (Chunk parsing a regole):**

- Regular-Grammars (cascade).
- Bottom-up, programmazione dinamica.
- Rule-based.

**Note:-**

Si rimuove la ricorsione. I sintagmi possono essere ricorsivi, i chunk (unità sintattiche di base) no.

## 3.3 Parsing a Dipendenze

Paolo ama Francesca

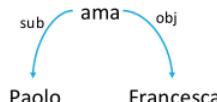
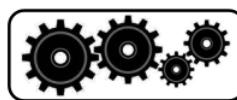


Figure 3.10: Un parser a dipendenze.

### 3.3.1 Sintassi a Dipendenze

#### Definizione 3.3.1: Sintassi a Dipendenze

La sintassi a dipendenze postula che la struttura sintattica consiste di elementi lessicali connessi da relazioni binarie asimmetriche (graficamente frecce) chiamate *dipendenze*.

#### Note:-

Non ci sono relazioni alla pari, c'è sempre un soggetto passivo e uno attivo.

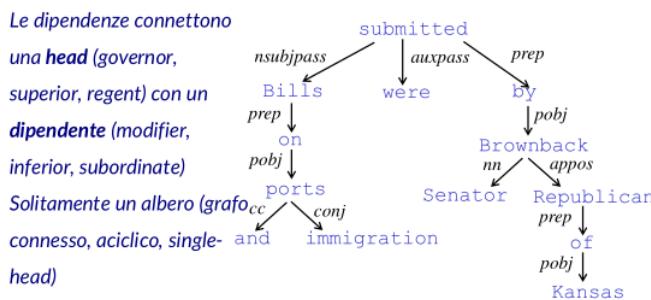


Figure 3.11: Esempio di sintassi a dipendenza.

#### Domanda 3.3

Cosa è una testa (head)?

- H determina la categoria sintattica di C; H può sostituire C.
- H è obbligatoria; D può essere opzionale.
- H seleziona D e determina quando D è obbligatoria.
- La forma di D dipende da H (agreement).
- La posizione nella frase di D è specificabile con riferimento a H.
- H determina la categoria semantica di C.

#### Osservazioni 3.3.1

- H: Head.
- D: Dependent.
- C: Costruzione.

#### Vantaggi delle dipendenze:

- Generalizzazioni tra lingue.
- Predizione psicolinguistica.
- Trasparenza e semplicità di rappresentazione.

#### Svantaggi delle dipendenze:

- L'assunzione di relazioni binarie asimmetriche non è sempre corretto (e. g. Cani e gatti).

**Note:-**

Le relazioni tra head e dipendenze sono una buona approssimazione della relazione tra predicati e i loro argomenti.

### 3.3.2 Approcci

Tecniche per il dependency parsing:

- Programmazione dinamica.
- Algoritmi per i grafi.
- Parsing a costituenza.
- Parsing transition-based.
- Constraint Satisfaction.

#### Definizione 3.3.2: Dynamic Parsing

Utilizza la programmazione dinamica (simile a CKY). L'algoritmo è simile al parsing delle PCFG lessicalizzate (con complessità  $O(n^5)$ ).

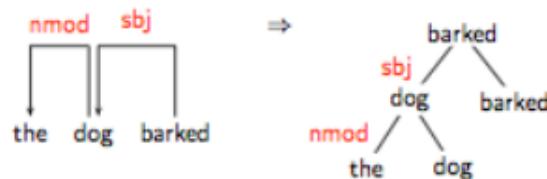


Figure 3.12: Dynamic Parsing.

#### Osservazioni 3.3.2

- Dynamic Programming.
- Eisner (1996) ha scoperto un algoritmo migliore che riduce la complessità a  $O(n^3)$ .

#### Definizione 3.3.3: Graph Algorithms

Si crea un minimum spanning tree per la frase. Le dipendenze vengono valutate usando un ML Classifier.

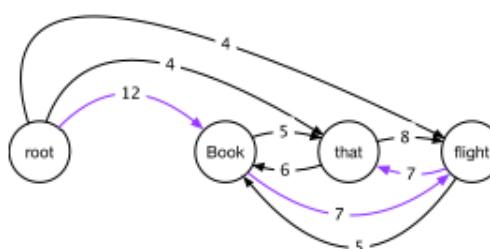


Figure 3.13: Graph Algorithm.

**Note:-**

Il primo algoritmo del motore di ricerca Google aveva un parsing di questo tipo.

**Definizione 3.3.4: Constituency Parsing**

Parsing con una grammatica a costituenti e converto in un formato a dipendenze attraverso delle “tabelle di percolazioni” (teoria X-bar). Scelte greedy per la creazione di dipendenze tra parole, guidate da ML classifiers. Vengono eliminate tutte le possibili dipendenze che non soddisfano a certi vincoli.

**Definizione 3.3.5: Deterministic Parsing**

Si fanno scelte greedy per la creazione di dipendenze tra parole, guidate da ML classifiers. Possono essere transition-based.

**Definizione 3.3.6: Constraint Satisfaction**

Vengono eliminate tutte le dipendenze che non soddisfano a certi vincoli.

### 3.3.3 Parsing Deterministico a Transizioni

#### Parser F (MALT):

- Dependency grammar.
- Bottom-up, depth-first.
- Probabilistico.

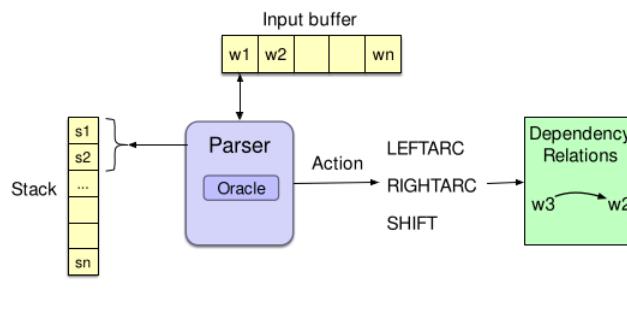


Figure 3.14: Transition-based parsing.

**Corollario 3.3.1 Proiettività**

IF  $i \rightarrow j$  THEN  $i \rightarrow *k$  for any  $k$  such that  $i < k < j$  or  $j < k < i$ .

**Note:-**

In parole povere: se non ci sono incroci.

**Osservazioni 3.3.3**

- La maggior parte dei framework non assume la proiettività.
- Le strutture non proiettive sono necessarie per tenere conto delle dipendenze a lunga distanza (e. g. le domande).

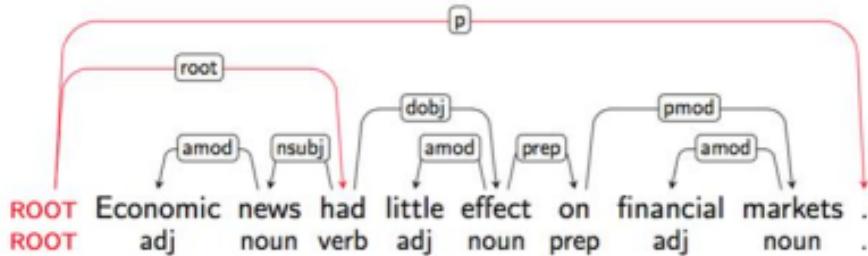


Figure 3.15: Struttura proiettiva.

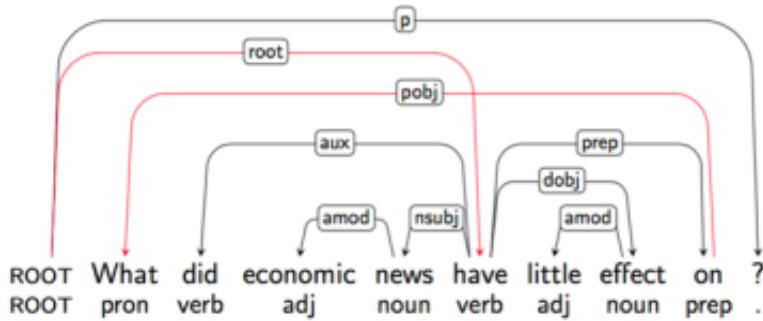


Figure 3.16: Struttura non proiettiva.

Uno **stato** è composto da tre cose:

- Uno stack che contiene le parole parzialmente analizzate.
- Una lista contenente le rimanenti parole da analizzare.
- Un insieme contenente le dipendenze create fino a quel punto nell'analisi.

**Stato iniziale:**

- {[root], [sentence], ()}.

**Stato finale:**

- {[root], [], (R)}.
- R è l'insieme delle dipendenze costruite.
- [] è la lista vuota poiché tutte le parole sono state analizzate.

#### Osservazioni 3.3.4 Operatori di parsing

- LeftArc: asserisce una relazione testa-dipendenza tra la parola sulla cima dello stack e la seconda parola nello stack (rimuove la seconda parola dallo stack).
- RightArc: asserisce una relazione testa-dipendenza tra la seconda parola nello stack e la parola sulla cima dello stack (rimuove la cima dallo stack).

- Shift: rimuove la parola dall'inizio dell'input e fa "push" sullo stack.

```

function DEPENDENCYPARSE(words) returns dependency tree
    state  $\leftarrow$  {[root], [words], []} ;initial configuration
    while state not final
        t  $\leftarrow$  ORACLE(state) ;choose a tr. operator to apply
        state  $\leftarrow$  APPLY(t,state) ;apply it, creating a new state
    return state

```

Figure 3.17: Algoritmo per fare il parsing delle dipendenze.

**Note:-**

Questo algoritmo è greedy: l'oracolo fornisce una singola scelta a ogni step e il parser procede con quella scelta (non c'è back-tracking).

## 2 Problemi:

- Operatori e dipendenze: come tipare?
- Quale operatore (L, R, S) si deve utilizzare a ogni passo? Bisogna avere un buon oracolo.

### Problema 1:

- $n$  dipendenze  $\rightarrow 2^n + 1$  operatori, ossia Left e Right in combinazione con tutte le relazioni + Shift.

### Problema 2:

- Oracolo a regole: si sceglie l'operatore da applicare usando un insieme di regole sul valore delle features dello stato corrente.
- Oracolo Probabilistico: si usa un sistema di ML per addestrare un classifier per scegliere l'operatore.

### Osservazioni 3.3.5

Per usare il ML bisogna:

- Trovare le features linguisticamente significative.
- Costruire il training data.
- Implementare un training algorithm.

### Trovare le features:

- A mano.
- In modo automatico utilizzando un classificatore neurale.

**Training Data:**

- TB (Treebank) a dipendenze.
- TB a costituenti e conversione (percolazione).

**Training Algorithm:**

- Dato un albero del TB si devono ricostruire tutte le scelte giuste del parser per costruire quello specifico albero.
- Si usa questo algoritmo:
  - LeftArc: se ottengo la dipendenza che correttamente si trova nell'albero.
  - RightArc: se ottengo la dipendenza che correttamente si trova nell'albero e tutte le altre dipendenze associate alla parola figlia si trovano già nell'albero.
  - Shift: in tutti gli altri casi.

**3.3.4 Parsing a Regole per Dipendenze a Vincoli****Parser G (TUP, Turin University Parser):**

- Dependency grammar (constrains).
- Bottom-up, depth-first.
- Rule-based.

**Questo parser:**

- Un parser a dipendenze bottom-up di ampia copertura basato su regole.
- Regole per:
  - Chunking.
  - Coordination: si rimuove l'ambiguità delle congiunzioni con delle regole.
  - Verb-SubCat: regole verbali basate su una tassonomia di classi per la sottocategorizzazione.
- Dipendenze:
  - Morfosintattiche.
  - Sintattico-funzionali.
  - Semantiche.



# 4

## Semantica

### Domanda 4.1

Si possono costruire le grammatiche a mano?

- Per l'italiano: L. Lesmo (1985 → 2014), Common Lisp.
- Per l'inglese:
  - SHRDLU (Winograd 1972).
  - CHAT-80 (1979 → 1982), sviluppato in Prolog. È un linguaggio naturale per un database sulla geografia.

### 4.1 Fondamenti di Semantica Computazionale

#### Rappresentare il significato:

- Quale forma per il significato?
  - Tavole di un database, logica descrittiva, logica modale, AMR.
- Blackburn e Bos → Logica del primordine.
- Il problema nasce per le frasi incomplete.
- Lambda e logica del primordine per parole e sintagmi.

#### Logica del primordine:

- Simboli:
  - Costanti.
  - Predicati e relazioni.
  - Variabili.
  - Connettivi logici.
  - Quantificatori.
  - Altri simboli.
- Formule:
  - Atomiche.
  - Composte.
  - Quantificate.

### 4.1.1 Algoritmo Fondamentale della Semantica Computazionale

1. Parsificare la frase per ottenere l'albero sintattico.
2. Cercare la semantica di ogni persona nel lessico.
3. Costruire la semantica per ogni sintagma:
  - Bottom-up.
  - Syntax-driven: traduzione rule-to-rule.

**Definizione 4.1.1: Principio di Composizionalità di Frege**

Il significato del tutto è determinato dal significato delle parti e dalla maniera in cui sono combinate.

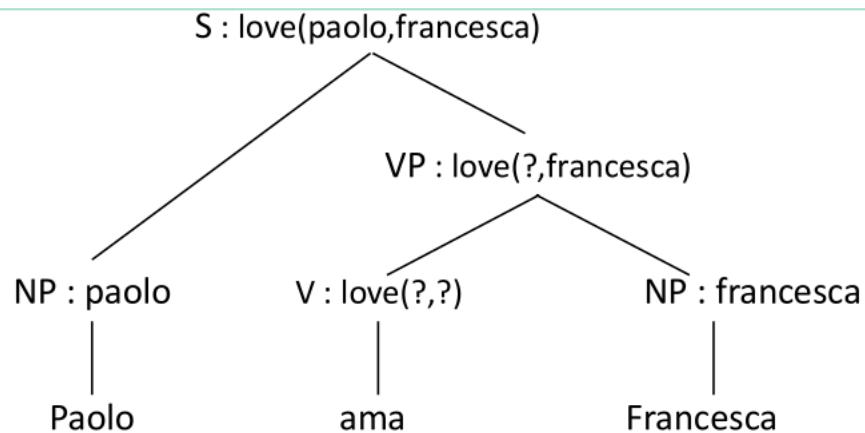


Figure 4.1: Applicazione dell'algoritmo.

**Il significato della frase è costruito:**

- Dal significato delle parole → Lessico.
- Risalendo le costruzioni semantiche → Regole semantiche.

**Sistematicità:**

- Come costruiamo il significato del VP?
- Come specificare in che maniera combinare i pezzi?
- Come rappresentare pezzi di formule?

## 4.2 Lambda Astrazione

**Si utilizza il lambda  $\lambda$ :**

- Il nuovo operatore  $\lambda$  si usa per legare (bind) le variabili libere.
- Questo simbolo meta-logico segna l'informazione mancante, ossia l'informazione generica.

### 4.2.1 Beta riduzione

$$\lambda(x.\text{love}(x, \text{mary}))(john)$$

- Si elimina il  $\lambda$ :
  - $(\text{love}(x, \text{mary}))(john)$
- Si rimuove l'argomento:
  - $\text{love}(x, \text{mary})$
- Si rimpiazzano le occorrenze della variabile legata dal  $\lambda$  con l'argomento in tutta la formula:
  - $\text{love}(john, \text{mary})$

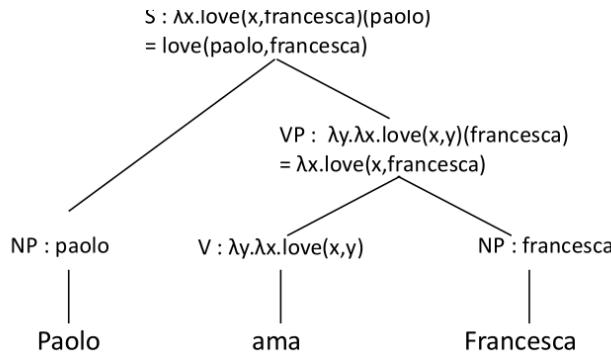


Figure 4.2: Beta riduzione.

### 4.2.2 Semantica di Montague

Montague propose un collegamento tra i linguaggi naturali e la logica: rifiutò dell'assunzione che ci sia una differenza importante tra linguaggi formali e linguaggi naturali. Inoltre viene mostrata l'influenza del tempo a livello del significato (logica temporale):

- U: utterance.
- R: reverence.
- E: event.

**Note:-**

Queste tre parti si combinano insieme per modificare il significato.

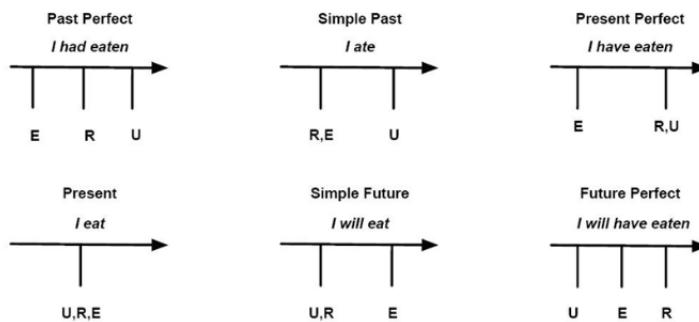


Figure 4.3: Rappresentazione del tempo.

### 4.2.3 Articoli e Nomi Propri

DT :  $\lambda P. \lambda Q. \exists z (P(z) \wedge Q(z)) \rightarrow \text{un}$

DT :  $\lambda P. \lambda Q. \forall z (P(z) \rightarrow Q(z)) \rightarrow \text{tutti}$

DT :  $\lambda P. \lambda Q. \forall z (P(z) \rightarrow \neg Q(z)) \rightarrow \text{nessun}$

Figure 4.4: Rappresentazione degli articoli nel lambda calcolo.

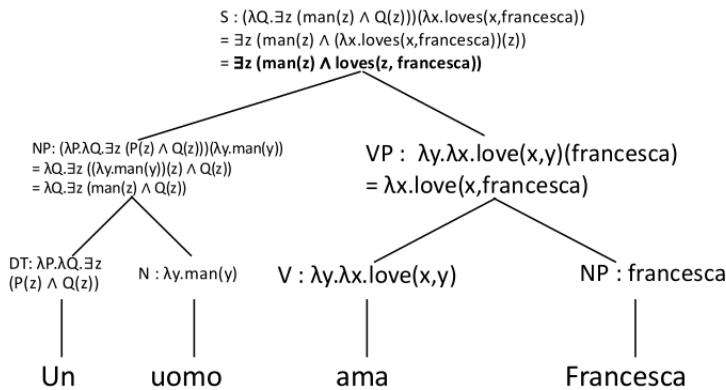


Figure 4.5: Esempio di frase con l'articolo "un".

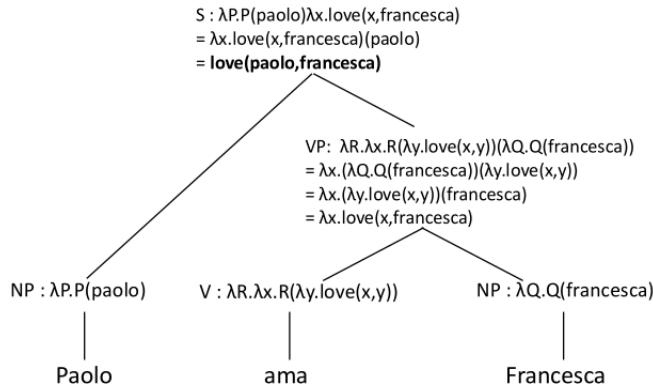


Figure 4.6: Il nuovo modo di trattare i nomi propri.

**Note:-**

Attenzione alle ambiguità: *Every man loves a woman.*

**Lettura 1:**

$$\forall x (\text{man}(x) \rightarrow \exists y (\text{woman}(y) \wedge \text{love}(x, y)))$$

**Lettura 2:**

$$\exists y (\text{woman}(y) \forall x (\text{man}(x) \rightarrow \text{love}(x, y)))$$

$\lambda Q \lambda P (\forall x (Q(x) \rightarrow P(x))) \rightarrow \text{Every}$



# 5

Natural Language Generation



# 6

Dialog Systems e ChatBOTSs



# 7

## Semantica Lessicale

### 7.1 Introduzione

#### 7.1.1 Programma della Seconda Parte del Corso

