Tecnologie del Linguaggio Naturale

Parte Prima

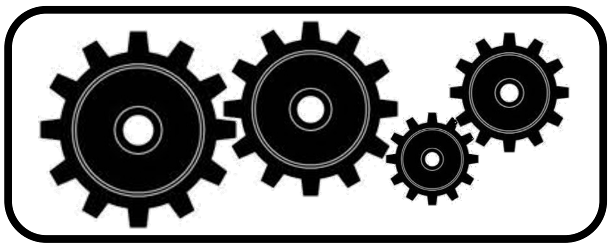
Lezione n. 04-3

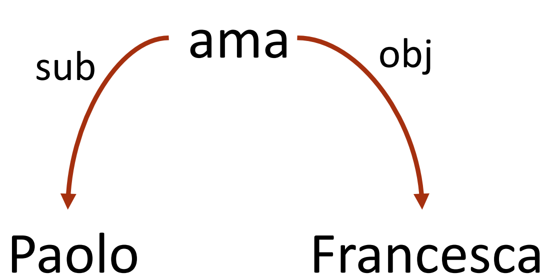
> Sintassi: grammatiche e parser a dipendenze

5 Marzo, 2024

Parser a dipendenze

Paolo ama Francesca





Outline

- Sintassi a dipendenze
- Parsing a dipendenze: approcci
- Parsing deterministico a “transizioni”
- Parsing a regole per dipendenze a vincoli

Outline

- **Sintassi a dipendenze**
- Parsing a dipendenze: approcci
- Parsing deterministico a “transizioni”
- Parsing a regole per dipendenze a vincoli

Lucien Tesnière (1959)

The sentence is an organized whole, the constituent elements of which are words. Every word that belongs to a sentence ceases by itself to be isolated as in the dictionary. Between the word and its neighbors, the mind perceives connections, the totality of which forms the structure of the sentence.

The structural connections establish dependency relations between the words. Each connection in principle unites a superior term and an inferior term. The superior term receives the name governor. The inferior term receives the name subordinate. Thus, in the sentence *Alfred parle*, *parle* is the governor and *Alfred* the subordinate.

Lucien Tesnière (1959)

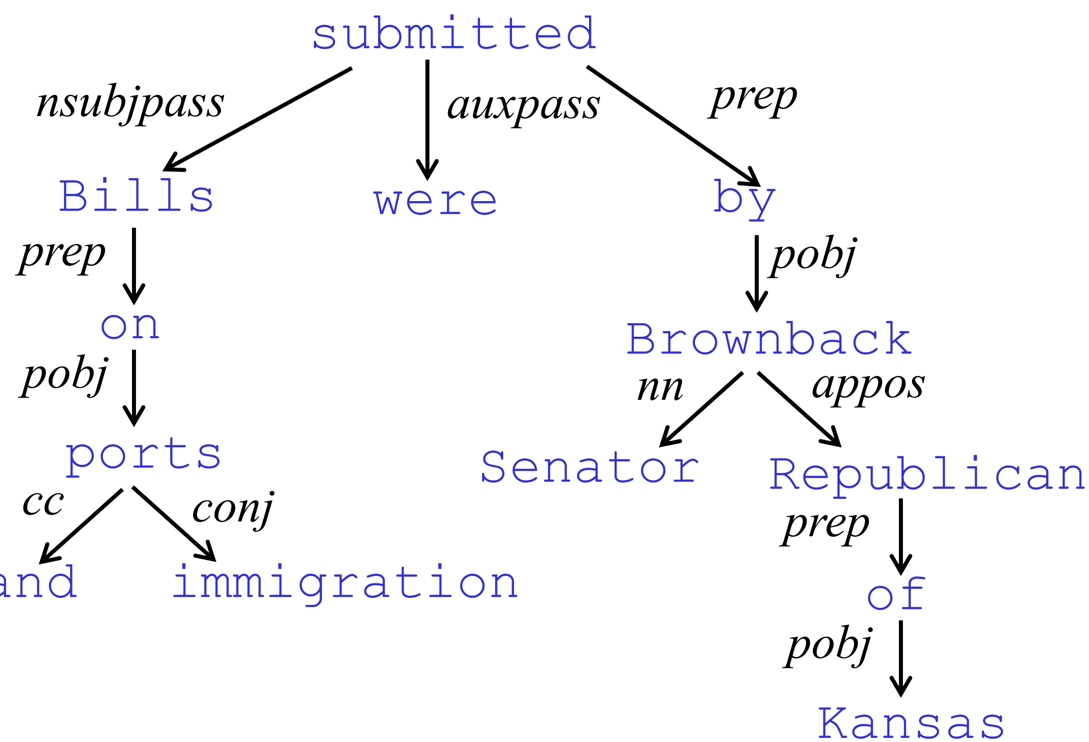
The sentence is an organized whole, the constituent elements of which are words. Every word that belongs to a sentence ceases by itself to be isolated as in the dictionary. **Between the word and its neighbors, the mind perceives connections, the totality of which forms the structure of the sentence.**

The structural connections establish dependency relations between the words. Each connection in principle unites a superior term and an inferior term. The superior term receives the name **governor**. The inferior term receives the name **subordinate**. Thus, in the sentence ***Alfred parle*, *parle* is the governor and *Alfred* the subordinate.**

Sintassi a dipendenze

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

Le dipendenze sono normalmente tipate con il nome si una relazione grammaticale (subject, prepositional object, apposition, etc.)

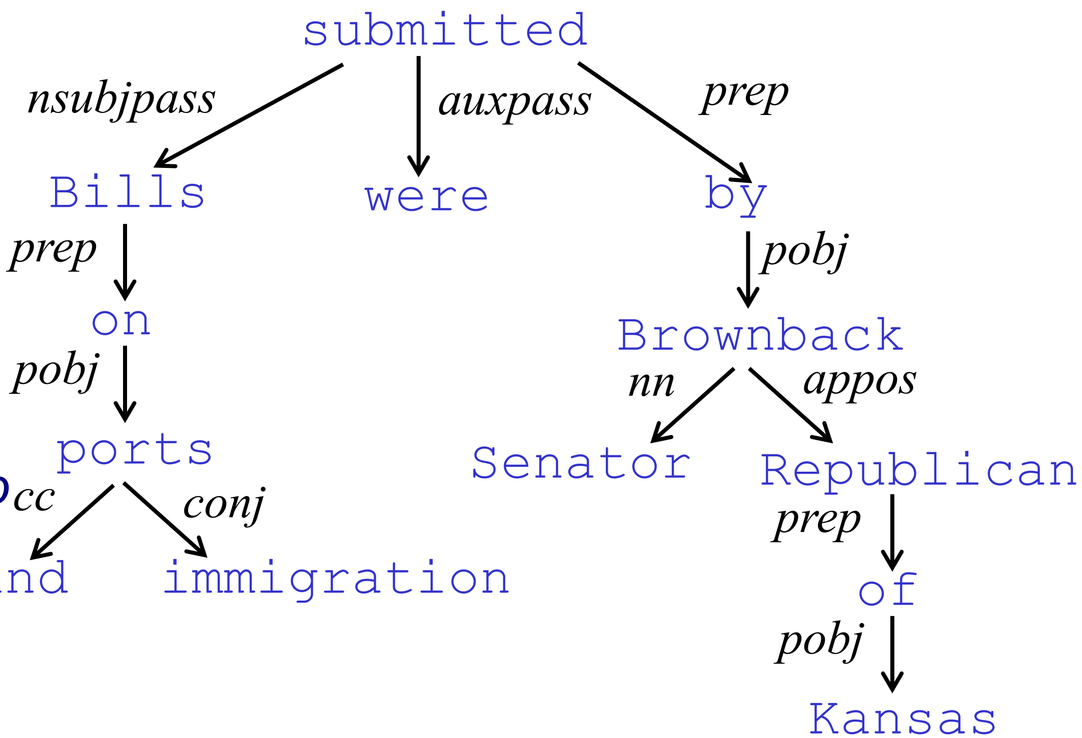


Sintassi a dipendenze

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

*Le dipendenze connettono una **head** (governor, superior, regent) con un **dipendente** (modifier, inferior, subordinate)*

Solitamente un albero (grafo connesso, aciclico, single-head)



Cosa è una testa?

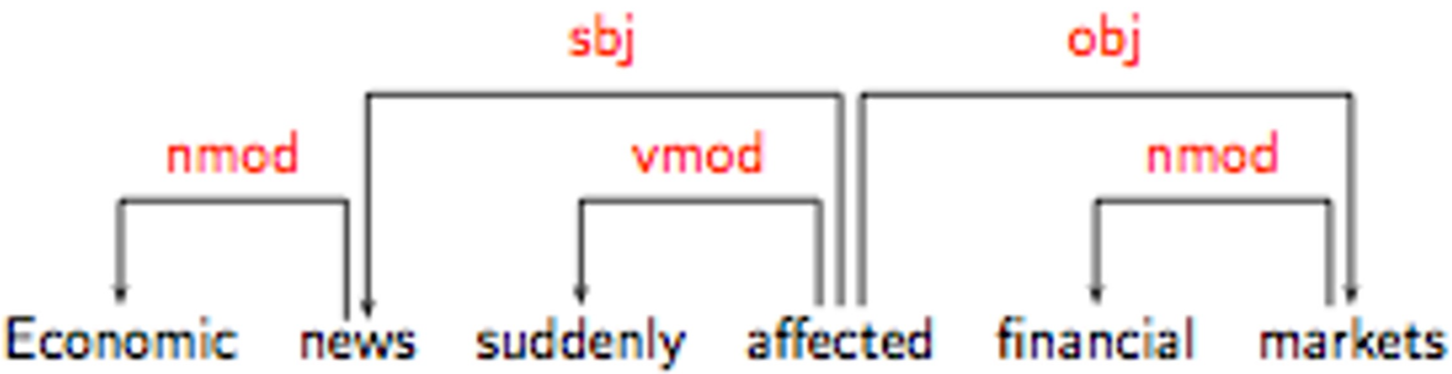
Criteri sintattici per distinguere una head **H** e un dependent **D** in una costruzione **C** [Zwicky 1985, Hudson 1990]:

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

Criteri: morfologici - sintattici - semantici

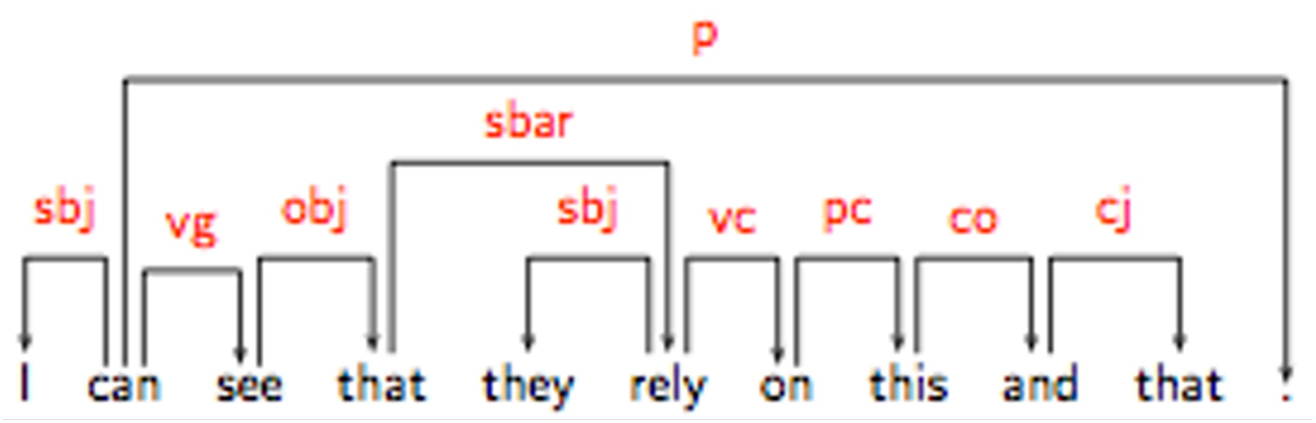
Teste evidenti

Head	**Dependent**
Verb	Subject (sbj)
Verb	Object (obj)
Verb	Adverbial (vmod)
Verb	Attribute (nmod)



Teste problematiche

Construction	Head?	Dependent?
Complex verb groups	auxiliary	main verb
Subordinate clauses	complementizer	verb
Coordination	coordinator	conjuncts
Prepositional phrases	preposition	nominals
Punctuation	verb	punct



2 *approssimazioni* sulla sintassi (Tesnière)

- *bottiglia* ***di*** *vino -> di* è una specie di funzione che

  trasforma il nome in un aggettivo

- *cani* ***e*** *gatti ->* la relazione è simmetrica

Dependency Grammar (2019)
de Marneffe and Joakim Nivre

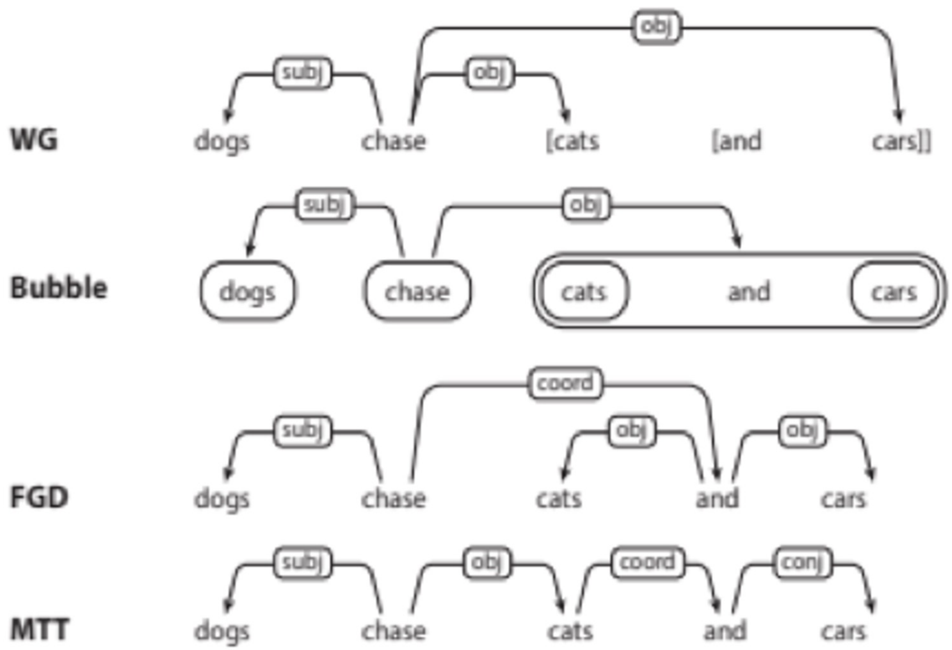


Figure 2

Treatment of coordination in four dependency frameworks: Word Grammar (WG), bubble tree (Bubble), Functional Generative Description (FGD), and Meaning–Text Theory (MTT).

Dependency Grammar (2019) de Marneffe and Joakim Nivre

In formal and generative linguistics, this term (*grammar*) is often understood in the sense of a formal declarative system capable of generating languages and making predictions about grammaticality. Most frameworks in the dependency grammar tradition do not make use of grammars in this sense but are better described as **syntactic analysis schemes that may be more or less formalized**. These characteristics carry over to most computational models for dependency parsing, which rely on machine learning techniques to learn statistical regularities from annotated corpora without inducinga formal grammar.

Dependency Grammar (2019) de Marneffe and Joakim Nivre

Vantaggi delle dipendenze:

- Generalization Across Languages
- Operationalization of Human Sentence Processing Facts
- Transparency and Simplicity of the Representation

Formalizzazione della sintassi a dipendenze

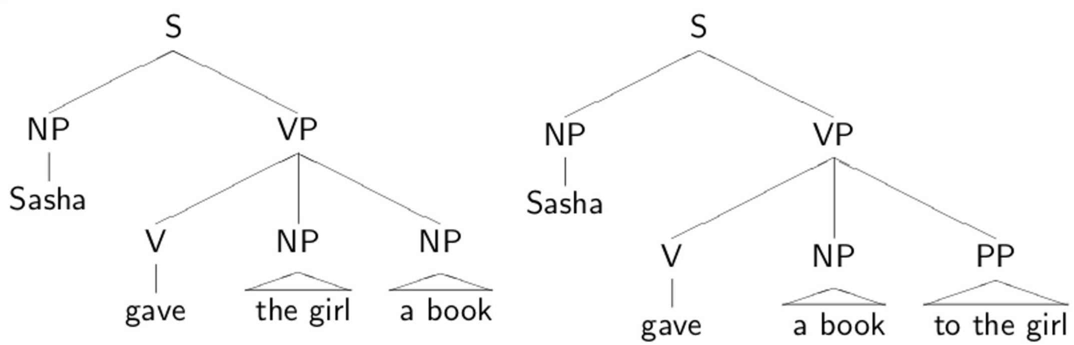
- Word Grammar (WG) [Hudson 1984, Hudson 1990]
- Functional Generative Description (FGD) [Sgall et al. 1986]
- Dependency Unification Grammar (DUG) [Hellwig 1986, Hellwig 2003]
- Meaning-Text Theory (MTT) [Melcuk 1988]
- (Weighted) Constraint Dependency Grammar ([W]CDG)
  - [Maruyama 1990, Harper and Helzerman 1995,
  - Menzel and Schro¨der 1998, Schroder 2002]
- Functional Dependency Grammar (FDG) [Tapanainen and Jarvinen 1997, Jarvinen and Tapanainen 1998]
- Topological/Extensible Dependency Grammar ([T/X]DG) [Duchier and Debusmann 2001, Debusmann et al. 2004]

Universal Dependencies

Clausal Argument Relations	Description
NSUBJ	Nominal subject
OBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

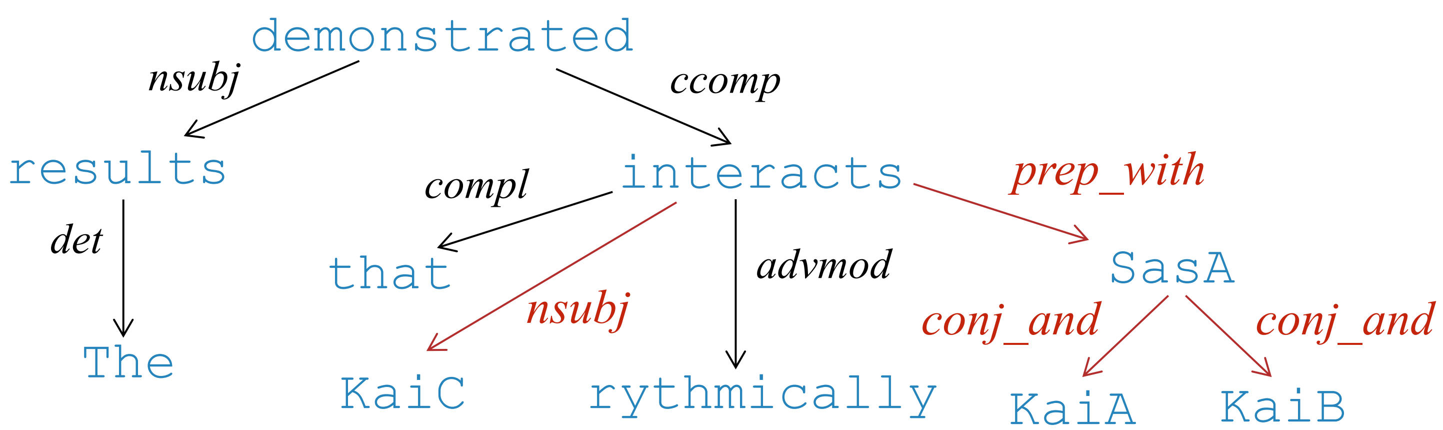
Figure 18.2 Some of the Universal Dependency relations (de Marneffe et al., 2021).

Why dependencies?



Information extraction e dipendenze: un esempio sulle proteine

The results demostrated that KaiC interacts rithmically with SasA, KaiA and KaiB.




```
KaiC <-nsubj interacts prep_with-> SasA
KaiC <-nsubj interacts prep_with-> SasA conj_and-> KaiA
KaiC <-nsubj interacts prep_with-> SasA conj_and-> KaiB
```

[Erkan et al. EMNLP 07, Fundel et al. 2007]

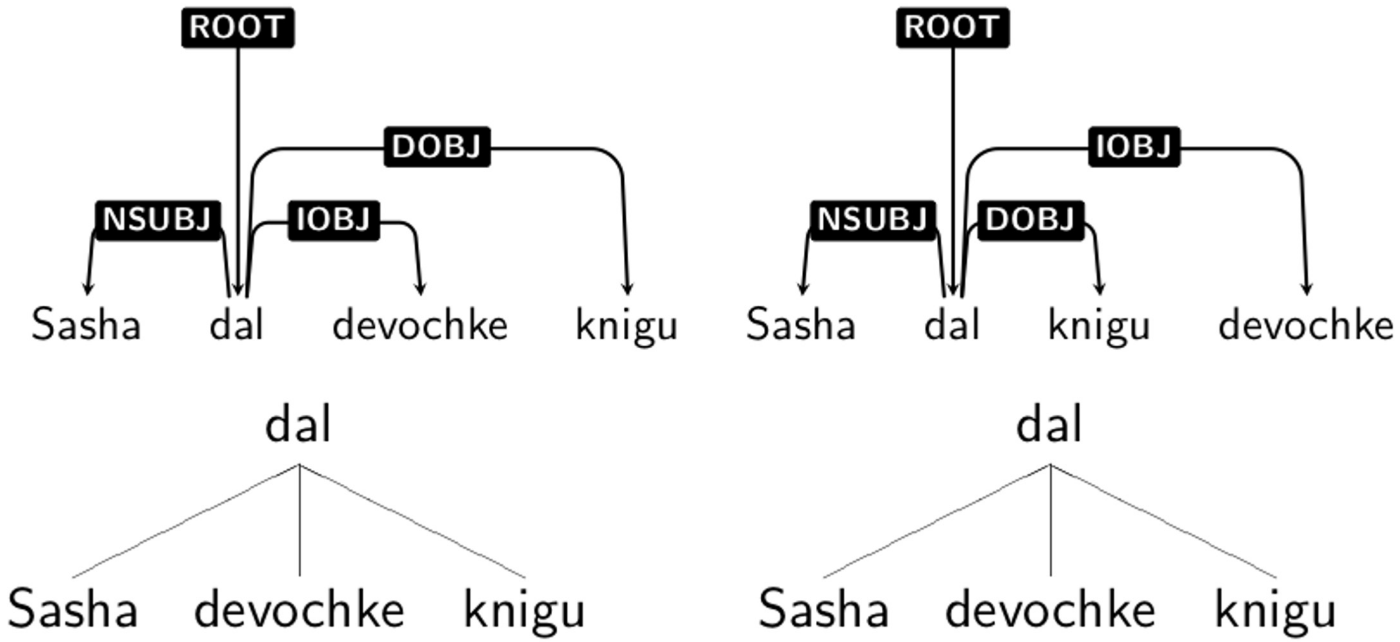
Vantaggio 1 delle dipendenze (Jurafsky)

This fact that the head-dependent relations are a good proxy for the semantic relationship between predicates and their arguments is an important reason why dependency grammars are currently more common than constituency grammars in natural language processing.

Free word-order (Vantaggio 2)

- Russian uses morphology to mark relations between words:
  - knigu means book (kniga) as a direct object.
  - devochke means girl (devochka) as indirect object (to the girl).
- So we can have the same word orders as English:
  - *Sasha dal devochke knigu*
  - *Sasha dal knigu devochke*
  - Sasha devochke dal knigu
  - Devochke dal Sasha knigu
  - Knigu dal Sasha devochke

Russian Trees



Pro vs. Cons

- Sensible framework for free word order languages.
- Identifies syntactic relations directly. (using CFG, how would you identify the subject of a sentence?)
- Dependency pairs/chains can make good features in classifiers, for information extraction, etc.
- Parsers can be very fast

- The assumption of asymmetric binary relations isn't always right -> **how to parse *dogs and cats*?**

Outline

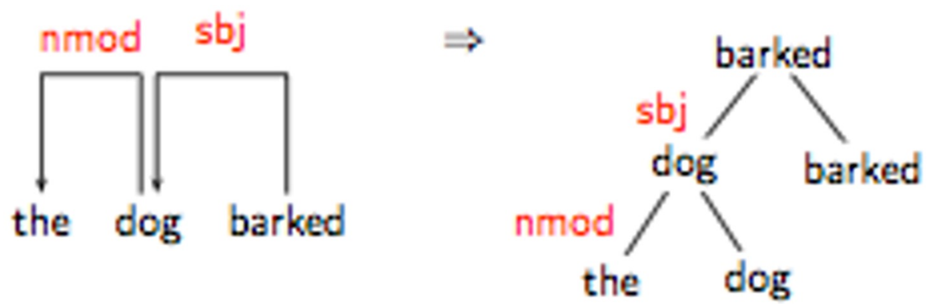
- Sintassi a dipendenze
- **Parsing a dipendenze: approcci**
- Parsing deterministico a “transizioni”
- Parsing a regole per dipendenze a vincoli

Tecniche per il dependency parsing

- Dynamic programming
- Graph algorithms
- Constituency parsing + conversion
- Transition-Based parsing
- Constraint Satisfaction

Tecniche per il dependency parsing

- Dynamic programming (simile a CKY)
- Algoritmo simile al parsing delle PCFG lessicalizzate: complessità $O(n^5)$
- Eisner (1996) ha "scoperto" un algoritmo migliore che riduce la complessità a $O(n^3)$



Tecniche per il dependency parsing

- Graph algorithms (ancora dynamic)
  - Si crea un Minimum-Spanning Tree per la frase
  - McDonald et al.'s (2005): MSTParser valuta le dipendenze indipendentemente unsando un ML classifier

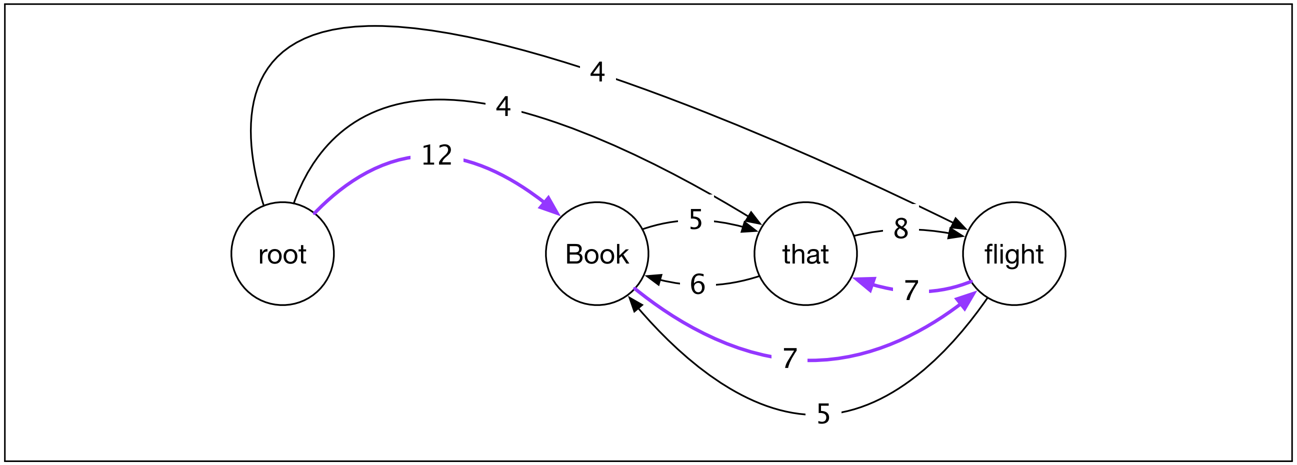


Figure 18.11 Initial rooted, directed graph for *Book that flight*.

Tecniche per il dependency parsing

- Constituency parsing + conversion
  - Parsing con una grammatica a costituenti e converto in un formato a dipendenze attraverso delle “tabelle di percolazioni” (teoria X-bar)
- Deterministic parsing
  - Scelte greedy per la creazione di dipendenze tra parole, guidate da machine learning classifiers
  - MaltParser (Nivre et al. 2008) -> Transition Based
- Constraint Satisfaction
  - Vengono eliminate tutte le possibili dipendenze che non soddifano a certi vincoli (hard). (Karlsson 1990), (Lesmo & Torasso 1985): FIDO -> TUP , etc.

Outline

- Sintassi a dipendenze
- Parsing a dipendenze: approcci
- **Parsing deterministico a “transizioni”**
- Parsing a regole per dipendenze a vincoli

Parser F - MALT

(1) Grammar

Dependency grammar (Automa from TB), ...

(2) Algorithm

I. Search strategy

top-down, ~bottom-up, left-to-right, ...

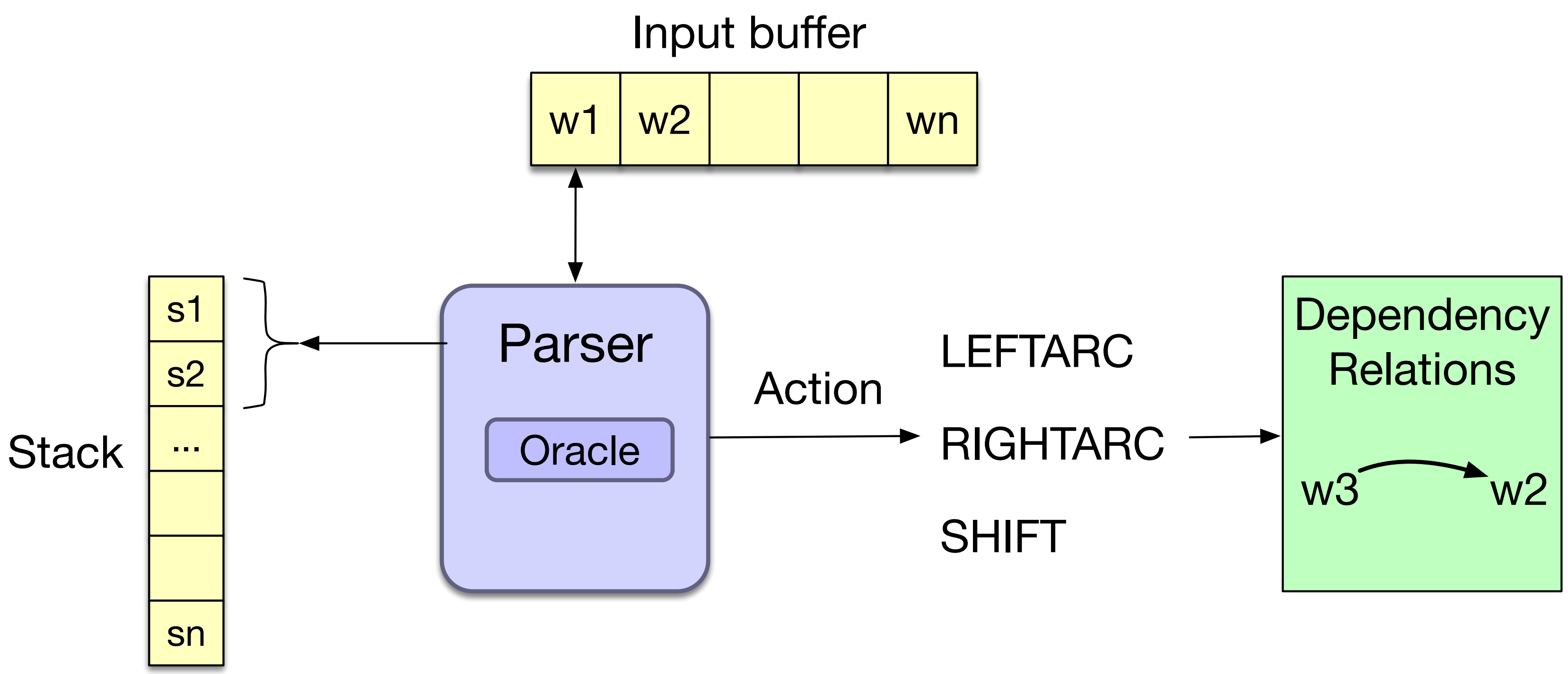
II. Memory organization

depth-first, back-tracking, dynamic programming, ...

(3) Oracle

Probabilistic, rule-based, ...

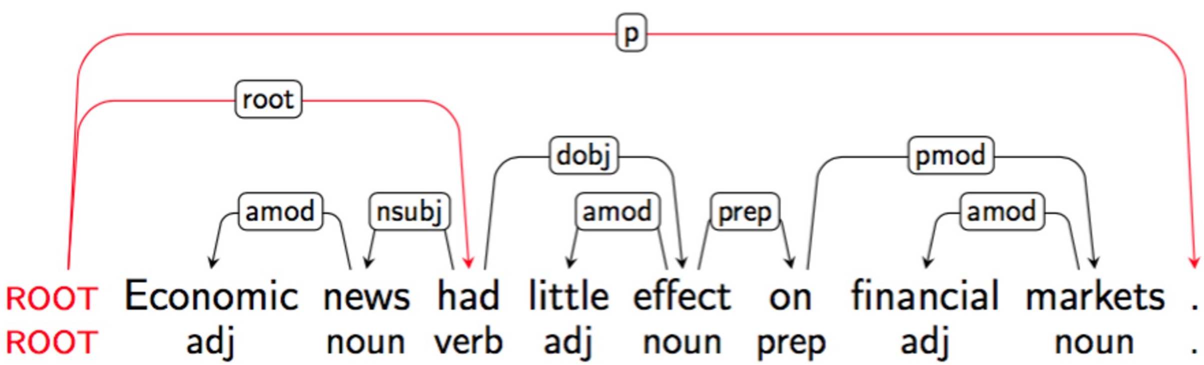
Transition-Based Parser (MALT)



Projectivity

Definition: **G is projective**

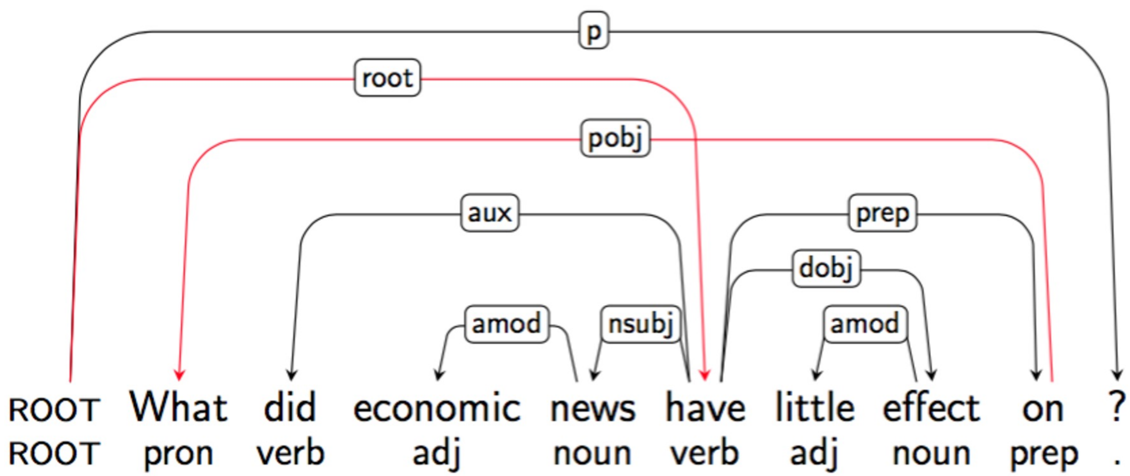
IF **i→j** THEN **i →* k** *for any* **k** *such that* **i<k<j** *or* **j<k<i**



NON Projectivity

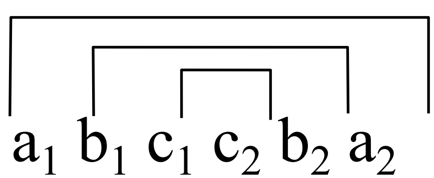
Definition: **G is projective**

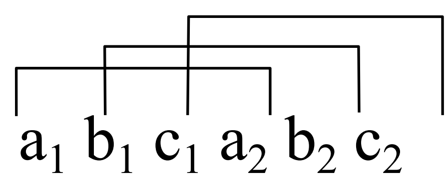
IF **i→j** THEN **i →* k** *for any* **k** *such that* **i<k<j** *or* **j<k<i**



NON Projectivity

- Most theoretical frameworks **do not** assume projectivity
- Non-projective structures are needed to account for long-distance dependencies (e.g. questions) and free word order
- MCSL -> cross-serial !





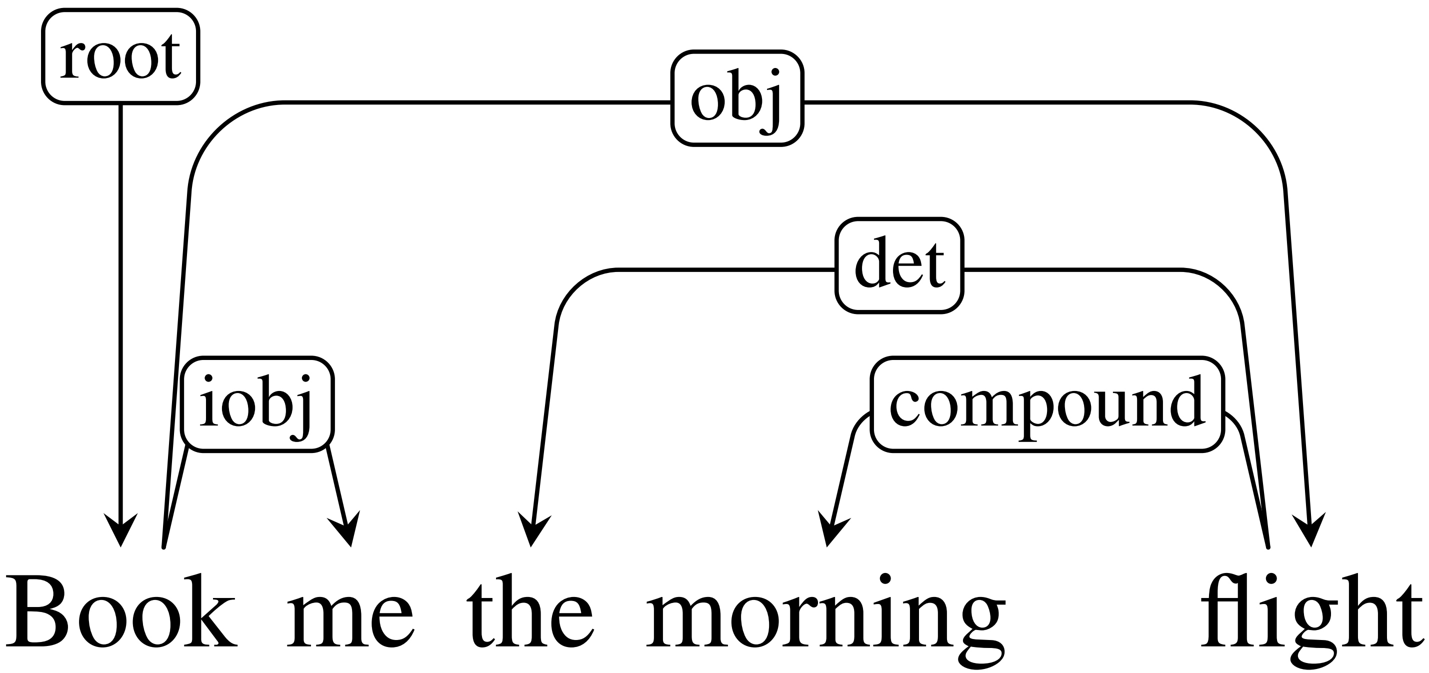
Transition-Based Parsing

- Ricerca in uno spazio di stati: similitudine con shift-reduce parsing
- Uno stato è composto da 3 cose:
  - Una stack che contiene le parole parzialmente analizzate
  - Una lista contentente le rimanti parole da analizzare
  - Un insieme contenente le dipendenze create fino a quel punto nell'analisi

Stati

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

Example



Esempio

- **Stato iniziale**

{[root], [Book, me, the, morning, flight], ()}

- **Stato finale**

{[root], [], ((book, me), (flight, morning), (flight, the), (book, flight), (root, book))}

Operatori di parsing

- Come arrivare dallo stato iniziale allo stato finale?
- Tre operatori di transizione tra stati:
  - **LeftArc:** Assert a head-dependent relation between the word at the top of the stack and the second word; remove the second word from the stack.
  - **RightArc:** Assert a head-dependent relation between the second word on the stack and the word at the top; remove the top word from the stack;
  - **Shift:** Remove the word from the front of the input buffer and push it onto the stack.

Parsing search algorithm

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

Greedy algorithm — the oracle provides a single choice at each step and the parser proceeds with that choice, no other options are explored, no backtracking is employed, and a single parse is returned in the end.

Example

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book $\rightarrow$ me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning $\leftarrow$ flight)
7	[root, book, the, flight]	[]	LEFTARC	(the $\leftarrow$ flight)
8	[root, book, flight]	[]	RIGHTARC	(book $\rightarrow$ flight)
9	[root, book]	[]	RIGHTARC	(root $\rightarrow$ book)
10	[root]	[]	Done	

Figure 18.6 Trace of a transition-based parse.

Another Example

Transition	State
	{[root] , [I, booked, a, morning, flight] , []}
Shift	{[root, I] , [booked, a, morning, flight] , []}
Shift	{[root, I, booked] , [a, morning, flight] , []}
LeftArc	{[root, booked]} , [a, morning, flight] , [(booked, I)]}
Shift	{[root, booked, a] , [morning, flight] , [(booked, I)]}
Shift	{[root, booked, a, morning] , [flight] , [(booked, I)]}
Shift	{[root, booked, a, morning, flight] , [] , [(booked, I)]}
LeftArc	{[root, booked, a, flight], [] , [(booked, I) , (flight, morning)]}
LeftArc	{[root, booked, flight], [] , [(booked, I), (flight, morning) , (flight, a)]}
RightArc	{[root, booked] , [] , [(booked, I), (flight, morning), (flight, a), (booked, flight)] }
RightArc	{[root] , [] , [(booked, I), (flight, morning), (flight, a), (booked, flight), (root, booked)] }

Bisogna provare provare provare

- *Paolo ama Francesca dolcemente*
- …

2 problemi

- Operatori e dipendenze -> relazioni “labelled”:

  subject, direct object, indirect object, etc.

- Quale operatore (L, R, S) usare ad ogni passo?

  -> Oracolo e memory

Problema 1: Operatori e dipendenze

- *n* dipendenze -> *2n+1* operatori
  - {LeftArc, RightArc} X {tutte le relazioni} + Shift

- Es: LeftArcSubj, RightArcSubj, LeftArcObj,

  RightArcObj, Shift

I booked a morninhg flight

Transition	**State**
	{[root] , [I, booked, a, morning, flight] , []}
Shift	{[root, I] , [booked, a, morning, flight] , []}
Shift	{[root, I, booked] , [a, morning flight] , []}
LeftArcSubj	{[root, booked]} , [a, morning, flight] , [(Subj,booked, I)]}
…	…

Problema 2: Oracolo a regole

- Scelgo l'operatore da applicare usando un insieme di regole sul valore delle features dello stato corrente
- Ad esempio: **IF** *la parola al top dello stack c'è un pronome e la parte rimanente dello stack è solo "root" e la prima parola della lista è un verbo* **THEN** OP=LeftArcSubj

Transition	**State**
	{[root] , [I, booked, a, morning, flight] , []}
Shift	{[root, I] , [booked, a, morning, flight] , []}
Shift	{[root, I, booked] , [a, morning flight] , []}
LeftArcSubj	{[root, booked]} , [a, morning, flight] , [(Subj,booked, I)]}
…	…

Problema 2: Oracolo probabilistico

- Uso un sistema di ML per addestrare un classifier per scegliere l'operatore -> **oracolo=classifier**
- Il classifier sarà basato sulle feature dello stato: CL({[root, I], [booked, a, morning, flight], []}) -> OP=LeftArcSubj

Transition	State
	{[root] , [I, booked, a, morning, flight] , []}
Shift	{[root, I] , [booked, a, morning, flight] , []}
Shift	{[root, I, booked] , [a, morning flight] , []}
LeftArcSubj	{[root, booked]} , [a, morning, flight] , [(Subj,booked, I)]}
…	…

3 punti importanti

L'uso del ML comporta:

1. Trovare le features linguisticamente significative

2. Costruire il training data

3. Training algorithm

Punto 1.A: features template *by hand*

Tipiche features sono/riguardano:

- *posizioni* nello stato
  - stack
  - list (buffer)
  - albero parziale
- attributi di alcune parole
  - PoS del top dello stack
  - PoS della terza parola nella lista
  - Lemma del top della lista
  - Head del top della lista
  - dipendenze del top della lista
  - etc ...

Punto 1.A: features template *by hand*

Tipiche features sono/riguardano:

- *posizioni* nello stato
  - stack
  - list (buffer)
  - albero parziale
- attributi di alcune parole
  - PoS del top dello stack
  - PoS della terza parola nella lista
  - Lemma del top della lista
  - Head del top della lista
  - dipendenze del top della lista
  - etc …

stack, *b*uffer, *w*ord, *t*ag

- $s_1 .w$
- $s_2 .w$
- $s_1 .t$
- $s_2 .t$
- $b_1 .w$
- $b_1 .t$
- $s_1 .wt$

Punto 1.A: features template *by hand*

Stack	Word buffer	Relations
[root, canceled, flights]	[to Houston]	(canceled $\rightarrow$ United) (flights $\rightarrow$ morning) (flights $\rightarrow$ the)

$\langle s_1.w = \text{flights}, op = \text{shift}\rangle$

$\langle s_2.w = \text{canceled}, op = \text{shift}\rangle$

$\langle s_1.t = \text{NNS}, op = \text{shift}\rangle$

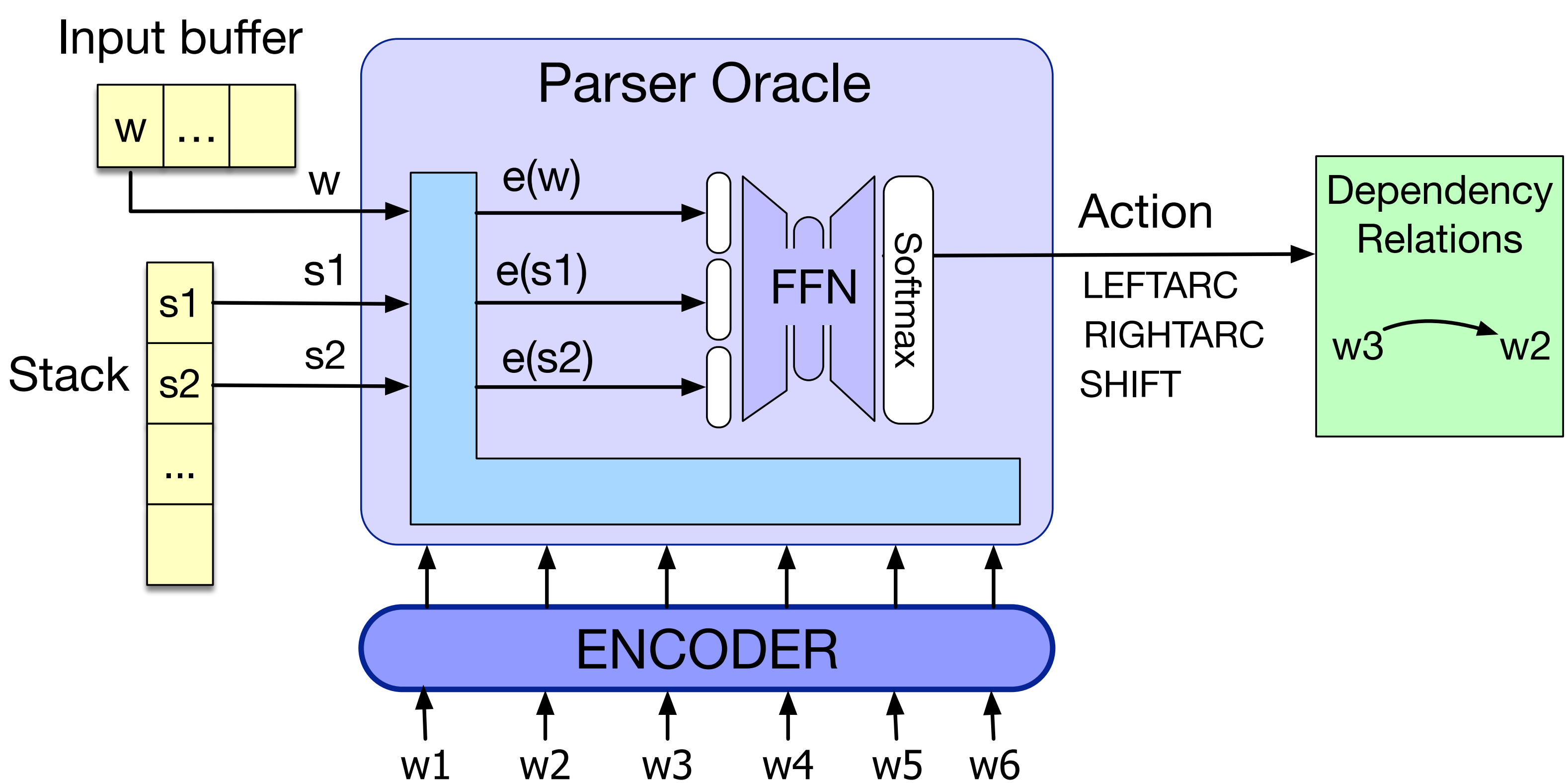
$\langle s_2.t = \text{VBD}, op = \text{shift}\rangle$

$\langle b_1.w = \text{to}, op = \text{shift}\rangle$

$\langle b_1.t = \text{TO}, op = \text{shift}\rangle$

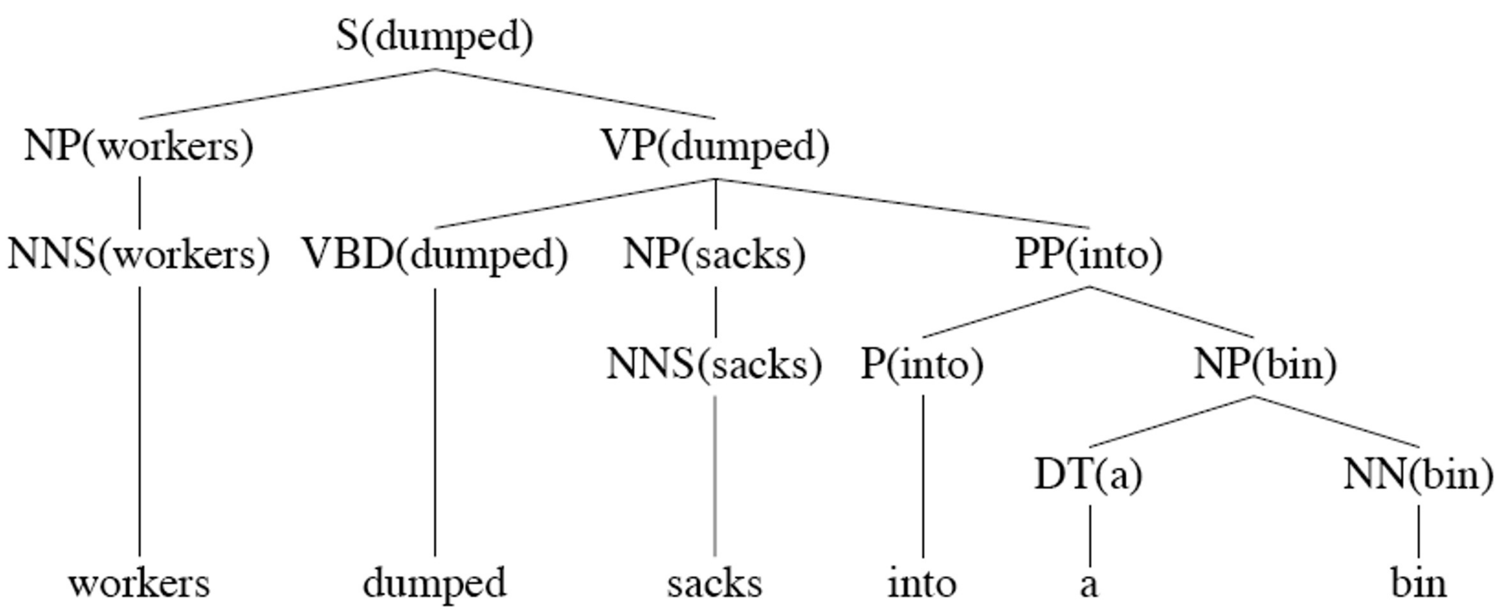
$\langle s_1.wt = \text{flightsNNS}, op = \text{shift}\rangle$

Punto 1.B: neural classifier *automatic*



Punto 2: training data

- TB a dipendenze
- TB a costituenti + conversione (percolazione)



Universal Dependency TB

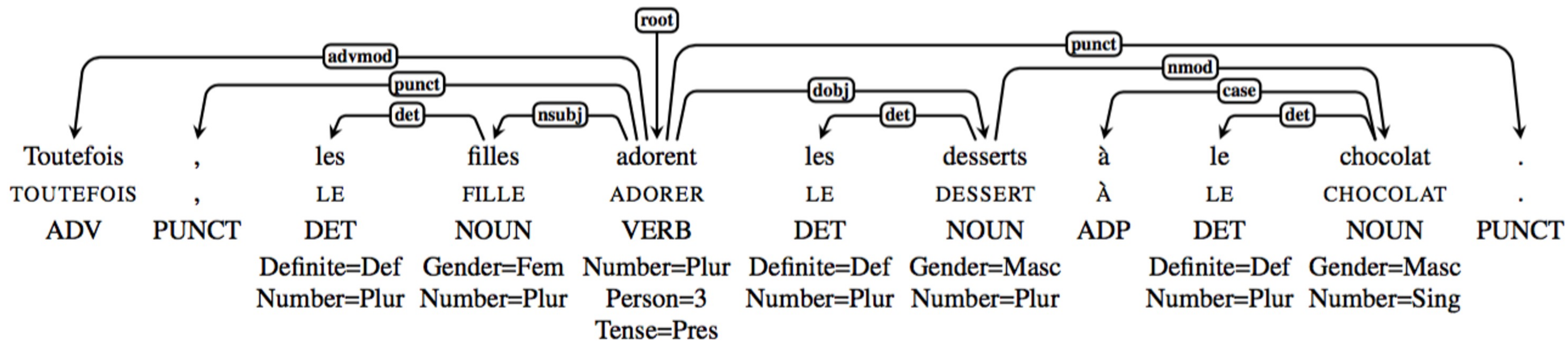
- 200 TBs , 100 languages
    - https://universaldependencies.github.io/docs/
- No-white space tokenization (cf. TUT later)
- Morphology: Lemma+PoS+FeatureSet
- 3 syntactic principles (40 **basic** relations)
    - content words are related by dependency relations
    - function words attach to the content word they further specify
    - punctuation attaches to the head of the phrase or clause in which it appear

UD relations

Relation	Examples with *head* and **dependent**
NSUBJ	**United** *canceled* the flight.
OBJ	United *diverted* the **flight** to Reno.
	We *booked* her the first **flight** to Miami.
IOBJ	We *booked* **her** the flight to Miami.
COMPOUND	We took the **morning** *flight*.
NMOD	*flight* to **Houston**.
AMOD	Book the **cheapest** *flight*.
APPOS	*United*, a **unit** of UAL, matched the fares.
DET	**The** *flight* was canceled.
	Which *flight* was delayed?
CONJ	We *flew* to Denver and **drove** to Steamboat.
CC	We flew to Denver **and** *drove* to Steamboat.
CASE	Book the flight **through** *Houston*.

Figure 18.3 Examples of some Universal Dependency relations.

Universal Dependency TB



Universal Dependency TB

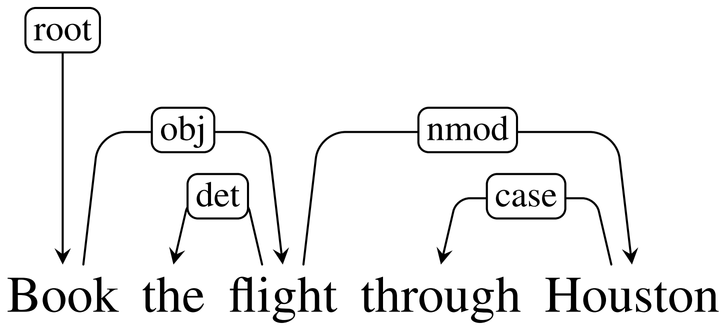
```
# isst_tanl sent_id  5

1   Inconsueto inconsueto   ADJ    A    Gender=Masc|Number=Sing   2   amod    _
_
2   allarme     allarme     NOUN   S    Gender=Masc|Number=Sing   0   root    _
_
3-4 alla        _           _      _    _          _              _   _       _
3   a           a           ADP    E    _                         5   case    _
_
4   la          il          DET    RD   Definite=Def|Gender=Fem|Number=Sing|PronType=Art
                                                                  5   det     _
_
5   Tate        Tate        PROPN  SP   _                         2   nmod    _
_
6   Gallery     Gallery     PROPN  SP   _                         5   name    _
_
7   :           :           PUNCT  FC   _                         2   punct   _
_
```

Punto 3: training algorithm

- Dato un albero del TB, devo ricostruire **tutte le scelte giuste** del mio parser per costruire quello specifico albero
- Ricostruisco ogni passo del parsing simulando il parsing e usando questo algoritmo:
  - **LeftArc** se ottengo la dipendenza che correttamente si trova nell'albero
  - **RightArc** se ottengo la dipendenza che correttamente si trova nell'albero AND tutte le altre dipendenze associate alla parola figlia si trovano già nell'albero
  - Altrimenti **Shift**

Reverse Engineering

- **LeftArc** se ottengo la dipendenza che correttamente si trova nell'albero
- **RightArc** se ottengo la dipendenza che correttamente si trova nell'albero AND se tutte le altre dipendenze associate alla parola figlia si trovano già nell'albero
- Altrimenti **Shift**



Step	Stack	Word List	Predicted Action
0	[root]	[book, the, flight, through, houston]	SHIFT
1	[root, book]	[the, flight, through, houston]	SHIFT
2	[root, book, the]	[flight, through, houston]	SHIFT
3	[root, book, the, flight]	[through, houston]	LEFTARC
4	[root, book, flight]	[through, houston]	SHIFT
5	[root, book, flight, through]	[houston]	SHIFT
6	[root, book, flight, through, houston]	[]	LEFTARC
7	[root, book, flight, houston]	[]	RIGHTARC
8	[root, book, flight]	[]	RIGHTARC
9	[root, book]	[]	RIGHTARC
10	[root]	[]	Done

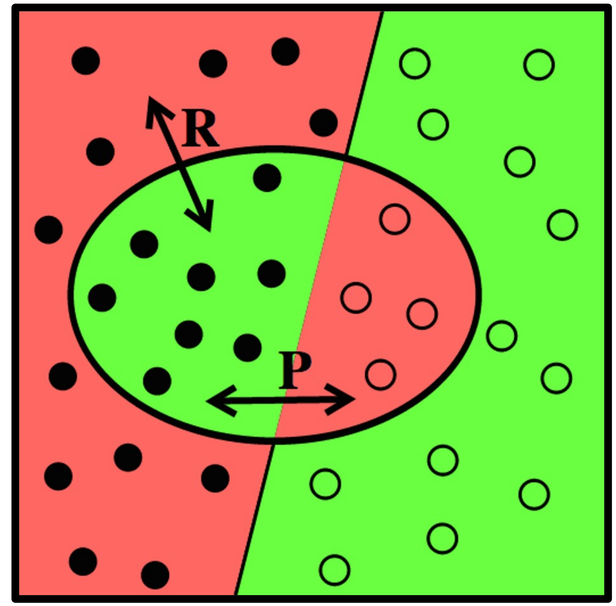
Figure 18.7 Generating training items consisting of configuration/predicted action pairs by simulating a parse with a given reference parse.

Transition-Based Parsing Variants

- Pseudo-projective parsing [Nivre and Nilsson 2005]
  - Preprocess training data, post-process parser output
  - Approximate encoding with incomplete coverage
  - Relatively high precision but low recall
- Beam search
- More Transition (e.g. *Arc-Eager*)

Valutazione

- Accuracy
- Precision vs. Recall
- F-score
- Unlabelled vs. Labelled



$$P = \frac{T^{\bullet}}{T^{\bullet} + F^{\bullet}} \qquad R = \frac{T^{\bullet}}{T^{\bullet} + F^{\circ}}$$

Outline

- Sintassi a dipendenze
- Parsing a dipendenze: approcci
- Parsing deterministico a “transizioni”
- **Parsing a regole per dipendenze a vincoli**

Parser G - TUP

(1) Grammar

Dependency grammar (constraints), ...

(2) Algorithm

I. Search strategy

top-down, ~bottom-up, left-to-right, ...

II. Memory organization

depth-first, back-tracking, dynamic programming, ...

(3) Oracle

Probabilistic, rule-based, ...

Turin University Parser

- Un parser a **dipendenze** bottom-up di ampia copertura basato su regole.
- **Regole** per: *Chunking*, *Coordination*, *Verb-SubCat*
- Dipendenze
  - Morpho-syntactic
  - Syntactic-functional
  - Semantic

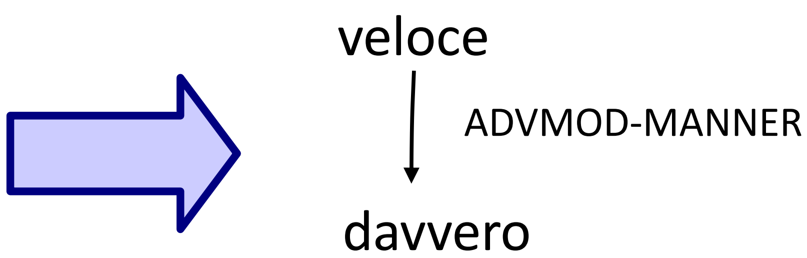
Turin University Parser

Chuncking nominale (non verbale)

```
(ADJ-QUALIF
    BEFORE (ADV (TYPE MANNER))
                              ADVMOD-MANNER )
```

IF un avverbio di tipo **`MANNER`** (modo) precedes immediatamente un aggettivo qualificativo **THEN** esso dipende da esso attraverso una dipendenza **`ADVMOD-MANNER`**

… davvero veloce …

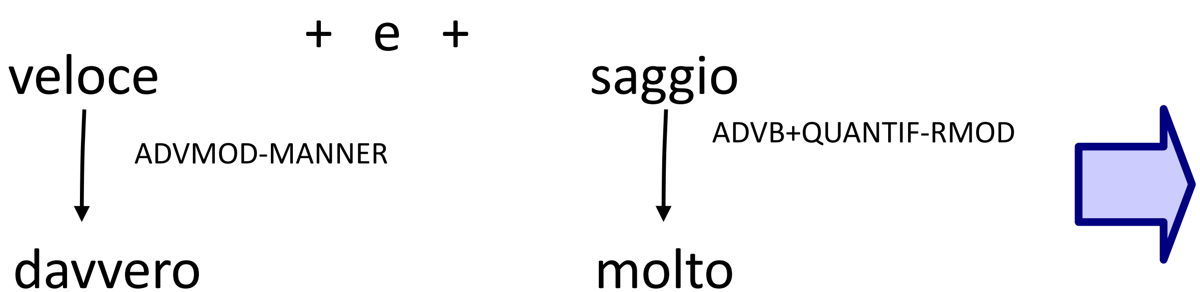


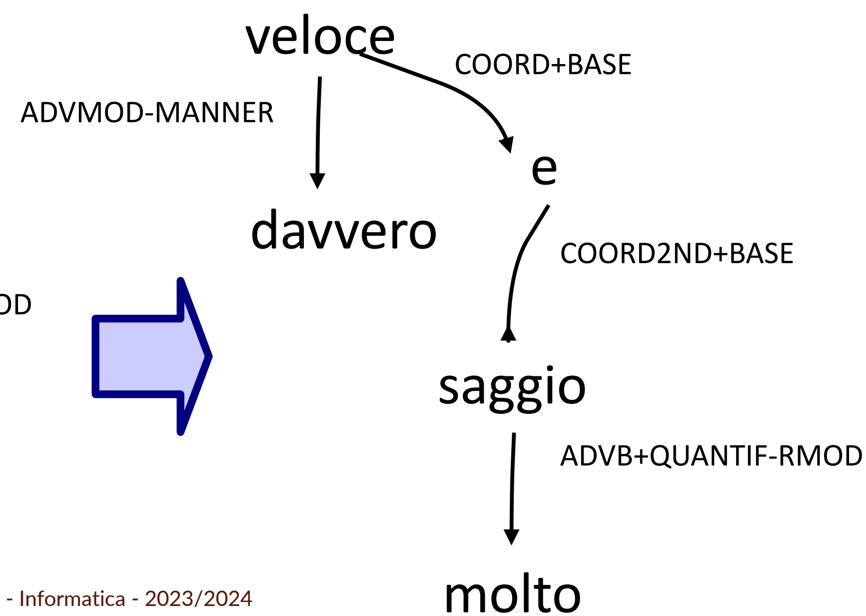
Turin University Parser

Coordination:

rimuovo l'ambiguità delle congiunzioni con delle regole

... davvero veloce e molto saggio ...





Turin University Parser

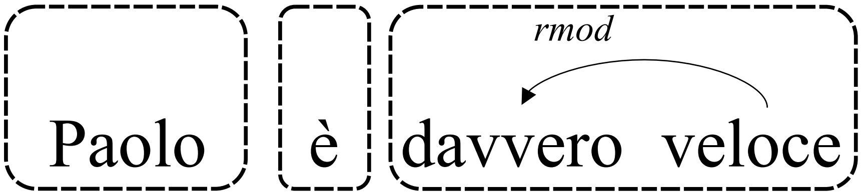
Verb-SubCat: regole verbali basate su una tassonomia di classi per la sottocategorizzazione

```
VERBS
   TRANS

      ...

   INTRANS

      ...

      INTRANS-INDOBJ-PRED  (Ex. "La casa gli sembra
        bella")

      ...
```

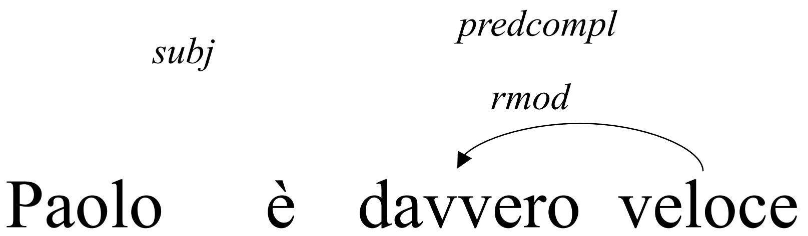
Turin University Parser

Paolo è davvero veloce

Chunking



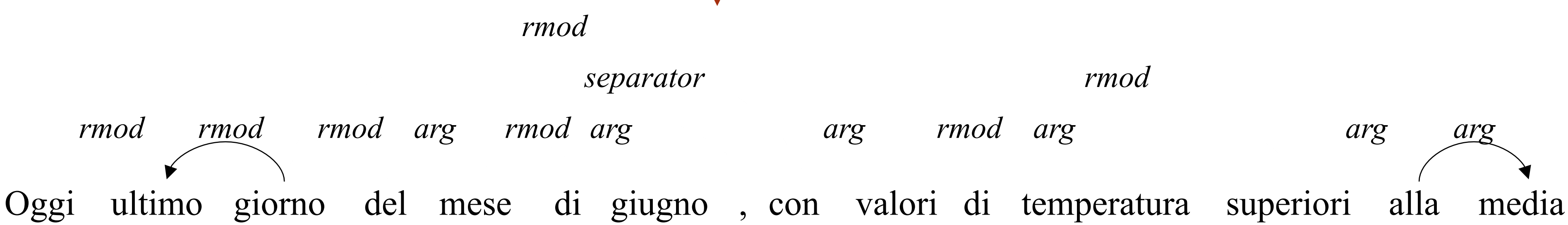
Verb-SubCat



Turin University Parser

Oggi ultimo giorno del mese di Giugno,

con valori di temperatura superiori alla media



Turin University Treebank

http://www.di.unito.it/~tutreeb/