Laurea Specialistica in Informatica
a.a. 2007-2008

Interazione Uomo-Macchina II:

Interfacce Intelligenti

Fiorella de Rosis e Nicole Novielli

Laboratori di
Irene Mazzotta e Nicole Novielli

Programma del Corso

Introduzione
Unità 1: Ragionamento logico:
- Formalizzazione
- Risoluzione

Unità 2: Generazione di linguaggio naturale
- Teorie
- Metodi

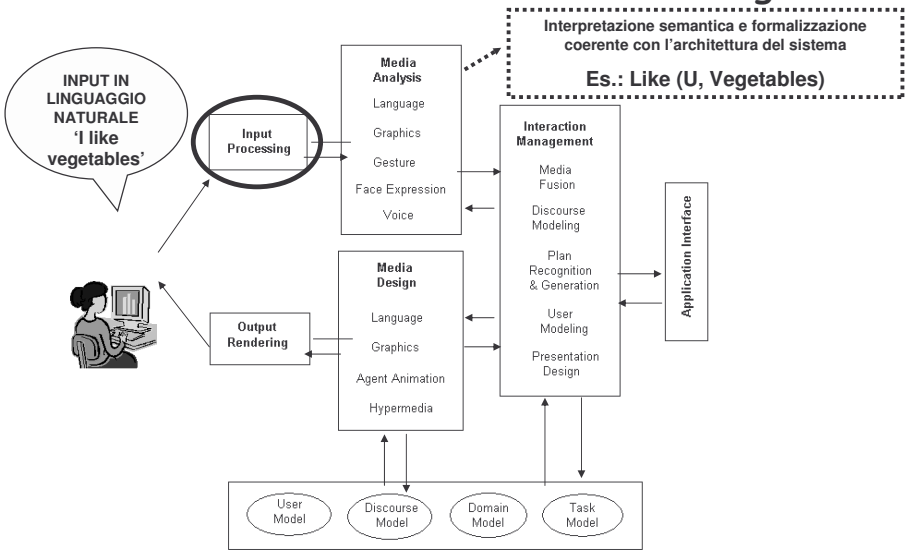
Unità 3: Ragionamento incerto
- Reti Causali Probabilistiche
- Reti dinamiche

Unità 4: Modelli di Utente
- Modelli logici
- Modelli con incertezza

Unità 5: **Comprensione del linguaggio naturale**
Unità 6: Simulazione di dialoghi
- Modelli basati su ATN
- Modelli basati su IS

Unità 7: Affective Computing

Architettura di una interfaccia intelligente



Natural Language Understanding

(Comprensione e interpretazione del linguaggio naturale)

Nell'analisi del linguaggio si distingue, classicamente, fra:

- *Morfologia:* studio della **struttura delle parole**
- *Sintassi:* studio della **struttura delle frasi**
- *Semantica:* studio del **significato delle frasi**
- *Pragmatica:* studio degli **obiettivi comunicativi** delle frasi

Morfologia: Le parole possono essere categorizzate in 'Parts of Speech':

Parts of speech	Esempio
Nouns (noun)	dog, equation, concert
Pronouns (pro)	I, you, it, they, them
Possessive (pos)	my, your
Verb (verb)	is, had, went, loves
Adjective (adj)	red, large, lovely
Determiner (det)	the, a, some
Proper noun (prop)	Alice, John
Conjunction (conj)	and, but, since
Preposition (prep)	In, to, into
Auxiliary verb (aux)	be, have
Modal verb (modal)	will, can, must, should
Adverb (adv)	closely, quickly,
Wh-word (wh)	who, what, where,
Final punctuation (fpunct)	. , ? !

Esempio per l'inglese

Le 'parts of speech' si chiamano anche 'simboli lessicali'

Sintassi:
La struttura delle frasi in un determinato linguaggio può essere definita con una Grammatica

Le Grammatiche 'context-free' (CFG) permettono di definire la struttura delle frasi 'corrette' in un linguaggio naturale semplice.

Una CFG è composta da:

- un insieme di *simboli terminali*
- un insieme di *simboli non terminali*
- un non terminale designato come *'simbolo iniziale' (o 'seme')*
- un insieme di *regole di produzione* che hanno un unico non-terminale a sinistra ed una stringa di simboli terminali e/o non terminali a destra

Un esempio semplice di CFG

Simboli non terminali: le parts of speech che abbiamo visto prima, oltre a categorie più ampie: np, vp, s, s-maj e al *Seme:* s-maj

Regole:

1. **s-maj → s fpunc**
2. **s → np vp**
3. **np → noun**
4. **np → det noun**
5. **vp → verb**
6. **vp → verb np**
7. **vp → verb np np**
8. **det → the**
9. **noun → salespeople**
10. **noun → dog**
11. **noun → biscuits**
12. **verb → ate**
13. **verb → sold**
14. **fpunc → .**

Questa grammatica permette, ad Esempio, di generare la frase:

"Salespeople sold the dog biscuits"

Ma anche (estendendo il lessico):

"John sent Mary an email"

e simili

Usi delle Grammatiche

Le grammatiche possono essere utilizzate:

In modo generativo, per generare tutte e sole le stringhe che appartengono ad un linguaggio

In modo riconoscitivo, per riconoscere e classificare le frasi appartenenti a un linguaggio.

Uso generativo: Derivazione di una frase

a partire dal seme, si applicano le regole della grammatica una alla volta e in una sequenza opportuna, fino a ottenere la frase.

Un esempio di derivazione

"Salespeople sold the dog biscuits"

1. s-maj → s fpunc
2. s → np vp
3. np → noun
4. np → det noun
5. vp → verb
6. vp → verb np
7. vp → verb np np
8. det → the
9. noun → Salespeople
10. noun → dog
11. noun → biscuits
12. verb → ate
13. verb → sold
14. fpunc → .

Applico le regole una alla volta,
a partire dal seme,
fino ad ottenere la stringa:

s-maj (1)⇒ s fpunc (14)⇒ s . (2)⇒
np vp . (3)⇒
noun vp . (9)⇒ Salespeople vp . (7)⇒
Salespeople verb np np . (13)⇒
Salespeople sold np np . (4)⇒
Salespeople sold det noun np . (8) ⇒
Salespeople sold the noun np . (10) ⇒
Salespeople sold the dog np . (11) ⇒
Salespeople sold the dog biscuits.

Notiamo che la grammatica precedente genera anche frasi come:
'Biscuits ate the dog'
oppure
'Salespeople sold salespeople'
che hanno poco senso.

Ma (più che sulla semantica) una verifica importante da effettuare, nel definire una grammatica, è che non consenta di generare *anche* stringhe non corrette sintatticamente.

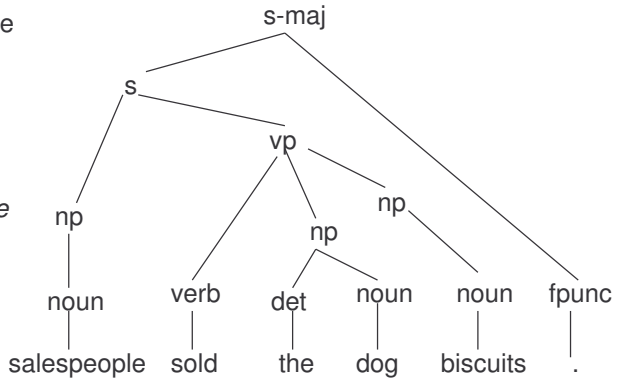
Vedremo degli esempi completi nel corso dell'esercitazione sul parsing probabilistico

La struttura di una frase può essere rappresentata in un albero sintattico

"Salespeople sold the dog biscuits"

np: predicato nominale
vp: predicato verbale
s: sentence

(il seme della grammatica è la radice dell'albero sintattico)



(s-maj s (np (noun *salespeople*)) vp ((verb *sold*) np ((det *the*) *(noun* dog))
(np (noun *biscuits*))) (fpunc .))

Esercizio

Aggiungi, alle regole della CFG del lucido n 9, la seguente:

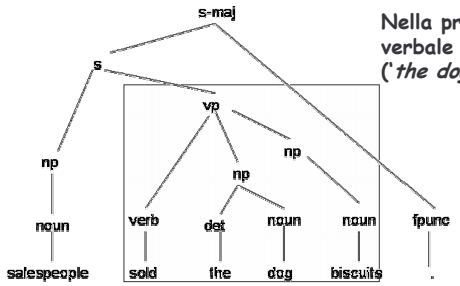
np → det noun noun

E prova a disegnare l'albero sintattico della frase:
"Salespeople sold the dog biscuits."

Questo albero è unico?

No! Vediamo…

Due possibili interpretazioni per *"Salespeople sold the dog biscuits"*

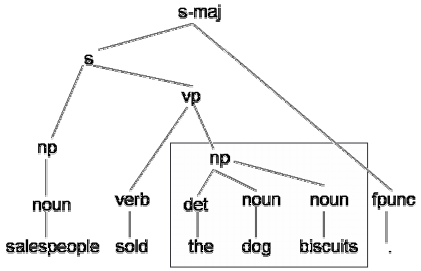


Nella prima trovo due complementi: la parte verbale vp è composta da un verb (*'sold'*) e due np (*'the dog'* e *'biscuits'*)

I biscotti sono stati venduti 'al cane' ☺

Nella seconda trovo un solo complemento np composto da un articolo e due nomi (noun), di cui il primo (*'dog'*) assume ruolo di aggettivo per il secondo (*'biscuits'*)

La semantica cambia: sono stati venduti 'biscotti per cani'



Il linguaggio naturale è ambiguo -> diverse possibili interpretazioni

Le Grammatiche possono essere 'ambigue'

Una *Grammatica* è *ambigua* quando ammette ***diverse interpretazioni della stessa frase*** (cioè diversi alberi sintattici che la rappresentano).

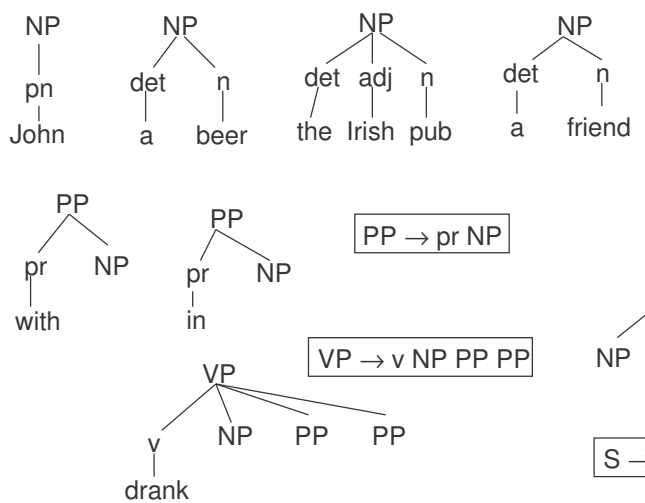
Le Grammatiche devono essere definite in modo da evitare le ambiguità di interpretazione!!! (vedi ad es le Grammatiche dei linguaggi di programmazione).

Tuttavia, alcuni *linguaggi* sono ***inerentemente ambigui***: Non è possibile, cioè, definire delle grammatiche non ambigue che li rappresentino.

Tutti i *linguaggi naturali* hanno questa caratteristica; vedi l'esempio della frase precedente (comprare 'biscotti per cani' vs 'i biscotti per il cane').

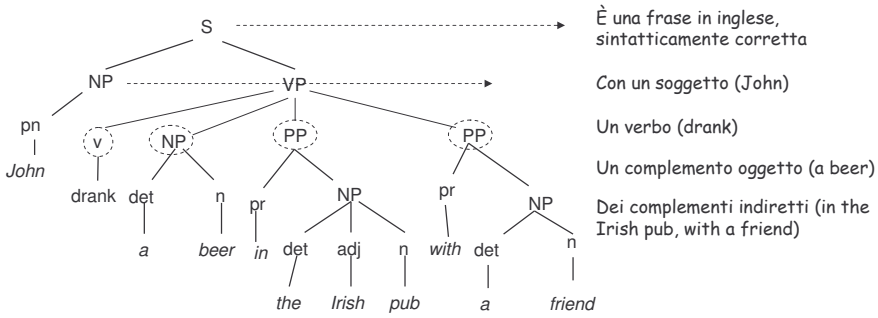
Impariamo a scrivere una Grammatica

Consideriamo la frase: *"John drank a bier in the Irish pub with a friend"* . Analizziamola a partire dalle 'parts of speech'.



Nell'esempio precedente la grammatica è definita in accordo con la grammatica inglese e i suoi POS con il solo scopo di definire un metodo per il riconoscimento di frasi sintatticamente corrette

A partire dalla rappresentazione tramite albero sintattico deduco che



Riprendendo la convenzione già utilizzata nell'es. 4, posso effettuare il processo di mappatura inverso a quello di generazione superficiale:

Inform (Drank(John,a beer,in_Irish_pub,with_friend))

La grammatica precedente consente di riconoscere frasi in ***forma dichiarativa*** (es. Inform)

Quali modifiche suggerireste per permettere il riconoscimento di frasi in ***forma interrogativa*** (ad esempio un'AskIF)?

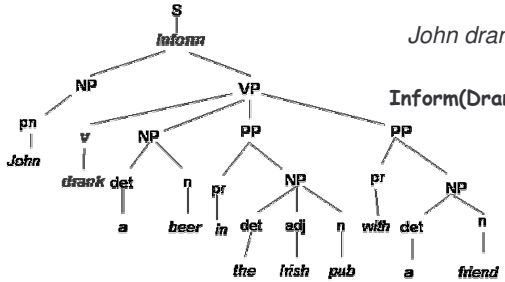
~~S → NP VP~~
NP → pn
NP → det n
NP → det adj n
VP → v NP PP PP
PP → pr NP
n → John / beer / …
adj → Irish

S → Inform / AskIF
Inform → NP VP
AskIF → AUX NP VP_pt
VP_pt → v_pt NP PP PP
AUX → did
v_pt → drink

Introduciamo la produzione che specifica le due possibilità di riscrittura di S

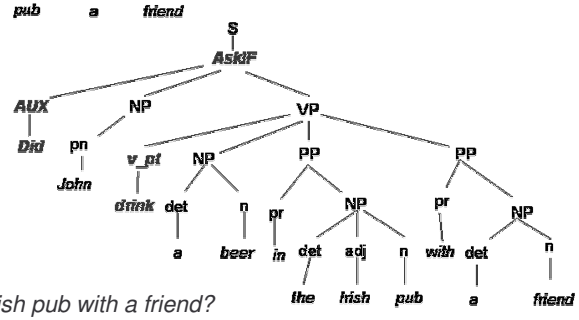
Quindi delle produzioni che definiscano le differenze tra la *struttura* di una *Inform* e la struttura di una *AskIF*

Inserisco nuove regole lessicali, assicurando la correttezza sintattica



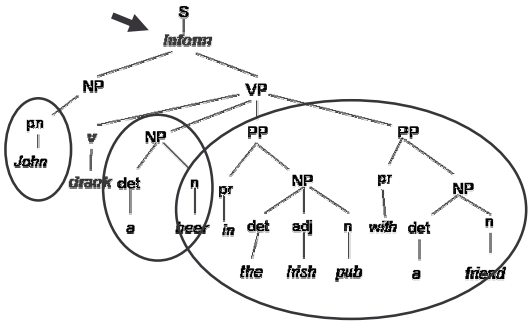
John drank a bier in the Irish pub with a friend

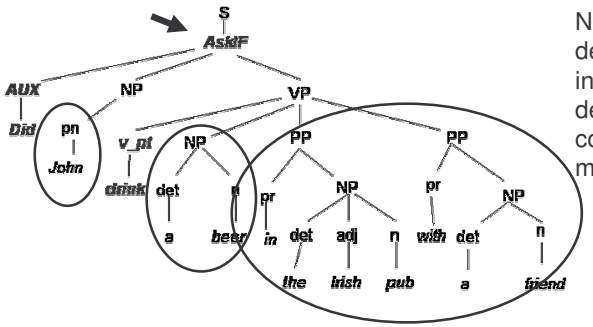
Inform(Drank(John,a beer,in_Irish_pub,with_friend))



Did John drank a bier in the Irish pub with a friend?

AskIF(Drank(John,a beer,Irish_pub,with_friend))





Il primo livello dell'albero sintattico dà informazioni sull'atto comunicativo

Esplorando l'albero in profondità posso ottenere informazioni sul soggetto

sul complemento diretto

e sui complementi indiretti

NOTA: il modo in cui è definita la grammatica influenza la struttura dell'albero sintattico e di conseguenza la procedura di mapping in forma logica!

Vediamo degli esempi…

Un esempio semplice

Proviamo a definire una grammatica molto semplice per la generazione/riconoscimento delle seguenti frasi relative allo scenario di un dialogo tra un utente ed un sistema che svolge il ruolo di agenzia di viaggio:

I would like a hotel room in London

I need an accomodation in Bari

Importante: è necessario definire la grammatica tenendo conto del **dominio applicativo** oltre che dello scopo del riconoscimento (in questo caso distinguere tra due diversi tipi di richiesta dell'utente all'agenzia)

Si tratta di move in cui l'utente esprime le proprie preferenze riguardo alla sistemazione in hotel

È necessario che il sistema sia in grado di estrarre la semantica della mossa dell'utente in modo da rispondere adeguatamente

ES.:

U: I would like a hotel room in London -> **Need Hotel**
AskFor(U, HotelRoom(London))
S: Which hotel category would you prefer?

Una possibile grammatica per la Request

- -> definisco il non terminale NEED_HOTEL

  Il mio obiettivo è definire una grammatica che mi consenta di riconoscere le richieste di prenotazione per l'alloggio

Definisco la struttura della richiesta dell'utente che intendo accettare

- NEED_HOTEL:= WANT HOTEL WHERE

  a sua volta NEED_HOTEL si compone di una parte in cui viene espressa l'intenzione di prenotare un hotel (WANT) ed un'altra in cui si specificano i parametri (HOTEL)

In questo modo definisco il lessico della grammatica (simboli terminali, in minuscolo)

- WANT:= PRON VERB
- PRON:= i / we / …
- VERB:= want/ need / would like / …
- HOTEL:= hotel / motel / an accomodation / a place to stay / …

  i vari modi per formulare la mia richiesta

- WHERE:= PREP CITY
- PREP:= in
- CITY:= London / Bari / Milano …

Posso riconoscere frasi come:
I would like a hotel room in London
I want an accomodation in Bari
Etc…

Riconosco frasi corrette ed effettivamente appartenenti al nostro linguaggio (composto da tutte le possibili richieste formalmente corrette che possono pervenire all'agenzia)

I would like a hotel room in London
I need an accomodation in Bari
…

Nota: riconosco TUTTE E SOLE le stringhe del mio linguaggio!!!

Proviamo a complicare la situazione e immaginiamo che l'agenzia gestisca anche prenotazioni di voli. In questo caso voglio riconoscere anche move come

U: I would like a hotel room in London -> **Need Hotel**
AskFor(U, HotelRoom(London))
S: Which hotel category would you prefer?

U: I am leaving from Bari in the morning -> **Leaving From**
AskFor(U,Flight(from_Bari))
S: What city are you travelling to?

Distinguendo tra i due tipi di necessità dell'utente, in modo che il sistema possa rispondere adeguatamente ai due diversi tipi di inform

Modifico la grammatica

NEED_HOTEL:= WANT HOTEL WHERE
WANT:= PRON VERB
LEAVING_FROM := LEAVING WHERE WHEN
LEAVING:= PRON VERB

Introduco nuove produzioni per definire la struttura della nuova richiesta

WHEN:= TEMP_EXPR / PREP TEMP_EXPR
TEMP_EXPR:= TE / DET TE
DET:= the
TE:= morning / afternoon / today / first of May / Monday…
PRON:= i / we /…
VERB:= want/ need / would like / **am leaving /…**
HOTEL:= hotel / motel / an accomodation / a place to stay / …

WHERE:= PREP CITY
PREP:= in / **from / on /…**
CITY:= London / Bari / Milano …

Ed estendo il lessico esistente, in accordo con la definizione delle parti del discorso della lingua inglese, senza considerarne il ruolo all'interno della frase

Quali frasi permette di riconoscere questa grammatica?

Riconosco frasi corrette ed effettivamente appartenenti al nostro linguaggio (composto da tutte le possibili richieste formalmente corrette che possono pervenire all'agenzia)

I would like a hotel room in London
I am leaving from Bari in the afternoon
I am leaving from London the first of May
I need an accomodation in Bari
…

Ma non TUTTE E SOLE le stringhe del mio linguaggio!!! Riconosco anche

I would like a hotel room from London
I am leaving on Bari
I am leaving from Bari from today
…

Sintatticamente scorrette!!

Nel definire una grammatica è necessario assicurarsi che la grammatica consenta di generare TUTTE e SOLE le stringhe del mio linguaggio.

Nell'esempio dell'agenzia di viaggi, voglio riconoscere particolari move utente assicurandomi che siano sintatticamente corrette e semanticamente rilevanti per il mio dominio

Un esempio: per assicurare la CORRETTEZZA SINTATTICA uno degli aspetti dar gestire è l'accordo tra le varie parti del discorso all'interno della frase, rispettando le regole sintattiche, ortografiche, morfologice del linguaggio naturale di riferimento (nell'esempio, l'inglese)

Un esempio

NEED_HOTEL:= WANT HOTEL WHERE
LEAVING_FROM:= LEAVING WHERE WHEN
WHERE:= PREP CITY
PREP:= in /from /…

Come lo risolvereste?

Ad es. differenziando in modo opportuno in non terminali

NEED_HOTEL:= WANT HOTEL *IN_WHERE*
LEAVING_FROM:= LEAVING *FROM_WHERE* WHEN
IN_WHERE:= *in* CITY
FROM_WHERE:= *from* CITY

Esercizio

Provate a modificare la grammatica descritta nel lucido precedente per riconoscere anche frasi del tipo:

Politeness

Uso di aggettivi

I would like a nice, quiet hotel in the central part of the city.

I need a cheap flight in the early morning.

Please find me a room in a hotel close to Piazza di Spagna!

Hey baby, would you please find me a cheap flight?

Espressioni informali

Modo diverso di formulare la richiesta

e simili …

Noterete che ampliare una grammatica per il linguaggio naturale in modo da consentire questo livello di espressività è un lavoro molto oneroso!
Le grammatiche consentono un riconoscimento accurato ma pongono dei limiti alle capacità di espressione

Utilità delle Grammatiche

Le Grammatiche sono un modo tipicamente 'generativo' di definire i linguaggi. Possono essere utilizzate, quindi, per ***definire***, in un simulatore di dialogo, ***il linguaggio in cui si esprime il sistema*** (generazione 'superficiale' delle singole frasi).

Sono, però, anche alla base dei **sistemi di parsing**.

Possono essere utilizzate, quindi, anche per ***definire il linguaggio con cui l'utente può interagire con il sistema*** **(NB: ponendo delle limitazioni al suo modo di esprimersi!!!)**

Transition Network (TN)

Le Grammatiche sono un metodo *generativo* per la definizione dei linguaggi.
Permettono cioè di definire *come generare* **tutte e sole** le stringhe che appartengono ad un linguaggio.

Gli Automi sono, invece, un metodo *riconoscitivo* per la definizione dei linguaggi.
Permettono cioè di definire come *riconoscere* **tutte e sole** le stringhe che appartengono ad un linguaggio.

Per il Teorema di equivalenza fra Grammatiche e Automi (nei linguaggi di tipo 3), una Grammatica molto semplice può essere rappresentata, graficamente, con un *Diagramma di Transizione fra gli Stati (TN)*

Transition Network (TN): un esempio

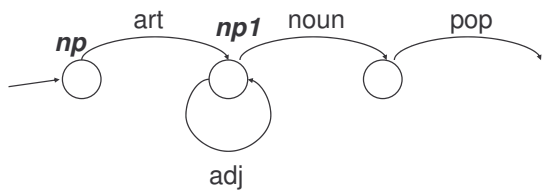
La grammatica G1:
np -> art np1
np1 -> adj np1
np1 ->noun

Ometto le regole lessicali di G1 per ragioni di spazio ma si assume che il lessico sia quello della lingua inglese

Es. noun -> city / cow / farmer / friend / …

Permette di generare **frammenti di frasi molto semplici** (predicati nominali) come:
"*a purple cow", "the large American city", "a good cheap hotel"…*

Il suo TN equivalente è il seguente:



Agli archi sono associati 'simboli lessicali'

Limiti dei Transition Network (TN)

I TN sono automi a stati finiti, equivalenti a **'grammatiche regolari'.**

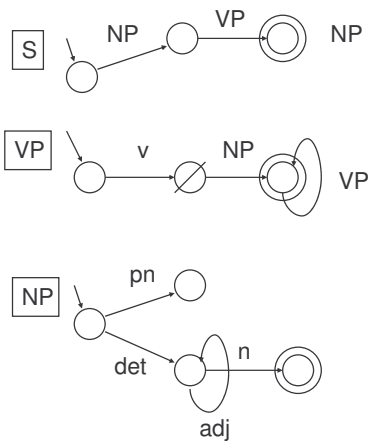
Permettono di fare analisi lessicali o di riconoscere frasi semplici, ma **non** di riconoscere **linguaggi più complessi** (come quelli definibili con CFG).

Per questi linguaggi, occorre introdurre una *memoria a stack* e associare ad ogni arco, oltre ad un carattere letto, operazioni di lettura/scrittura in questa memoria.

Occorre cioè passare da 'Transition Network' a '**Recursive** Transition Network'

Recursive Transition Networks (RTN)

S'introduce uno stack che permette di saltare dal TN principale ad uno secondario (innestamento).



John drank
John went to the pub
John went to the Irish pub
John drank a bier in the new Irish pub with a friend
…

Agli archi possono essere associati simboli lessicali oppure non terminali. Quando si attraversa un arco a cui è associato un simbolo non terminale, si memorizza il nome del simbolo in un registro e si salta ad un sub-RTN che ha questo nome. Alla fine dell'esecuzione del sub-RTN, si torna all'RTN principale.

Augmented Transition Networks (ATN)

Una ATN è un diagramma di transizione fra gli stati, ai cui archi sono associati una label e *una condizione.*

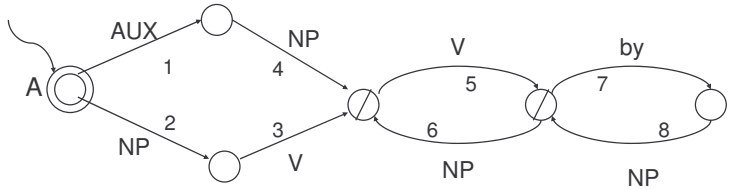
Le *label* possono denotare **categorie di parole** oppure **altre ATN**: si tratta quindi di diagrammi ricorsivi.

Le ***condizioni*** devono essere soddisfatte perché l'arco possa essere attraversato.

Un insieme di ***registri*** permette di memorizzare risultati intermedi oppure lo stato dell'esplorazione dell'ATN.

Un esempio di ATN

"The rice was eaten by the cat"

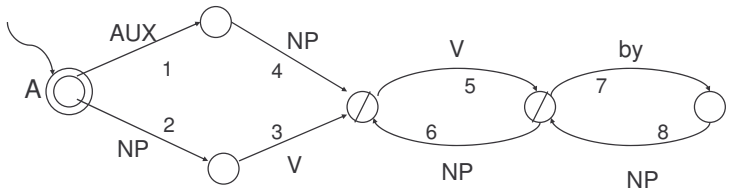


	test	*action*
1	T	SETR V* SETR TYPE'QUESTION
2	T	SETR SUBJ* SETR TYPE'DECLARATIVE
3	agrees SUBJ*	SETR V*
4	agrees* V	SETR SUBJ*
5	AND(GETF PPRT)(=V'BE)	SETR OBJ SUBJ SETR V* SETR AGFLAG TRUE SETR SUBJ'SOMEONE
6	TRANS V	SETR OBJ*
7	AGFLAG	SETR AGFLAG FALSE
8	T	SETR SUBJ*

Tabella Condizioni-azioni

Riconosciamo la frase

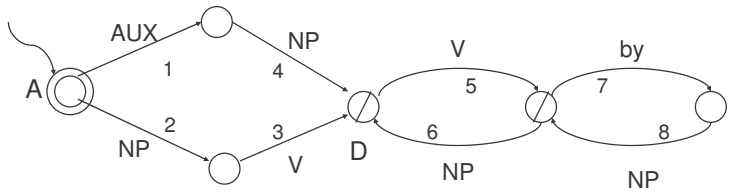
"The rice was eaten by the cat"



Iniziamo dallo start node A.
Ci sono due cammini possibili; ma poiché la prima parte della frase ('the rice') è un NP, si sceglie l'arco 2. Prima di attraversarlo, però, si controlla se ci sono altre condizioni o azioni da soddisfare (vedi tabella).
'T' significa che non ci sono altri test.
Si esegue l'azione che consiste nel memorizzare 'rice' nel registro SUBJECT e TYPE come 'declarative'.
Si passa all'esame di 'was'.
E' un verbo: quindi si traversa l'arco 3. Il test indica di controllare che questo verbo si accordi con il soggetto. Se questa condizione è verificata, si memorizza 'was' nel registro VERB.

Riconosciamo la frase

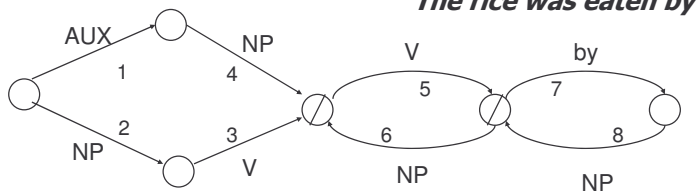
"The rice was eaten by the cat"



Siamo ora al nodo D.
'Eaten' è un verbo; il test sull'arco 5 indica di controllare che questo verbo si accordi con l'ausiliario.
Si compiono ora 4 azioni: il contenuto del registro SUBJ ('rice') viene trasferito nel registro OBJ. Il contenuto del registro VERB viene sostituito con 'was eaten'. Un flag indica che la frase è in forma passiva.
Il registro SUBJ viene settato al valore 'someone'.
By: si sceglie l'arco 7.
'The cat': un'altra 'noun phrase': si traversa l'arco 8 e si memorizza il frammento di frase in SUBJECT.
Il processo termina.

Riassumiamo il processo:

"The rice was eaten by the cat"



	test	*action*
•	T	SETR V*
		SETR TYPE'QUESTION
2	T	SETR SUBJ*
		SETR TYPE'DECLARATIVE
3	agrees SUBJ*	SETR V*
4	agrees* V	SETR SUBJ*
5	AND(GETF PPRT)(=V'BE)	SETR OBJ SUBJ
		SETR V*
		SETR AGFLAG TRUE
		SETR SUBJ'SOMEONE
6	TRANS V	SETR OBJ*
7	AGFLAG	SETR AGFLAG FALSE
8	T	SETR SUBJ*

The rice: **NP; arc 2 (SUBJ = the rice; DECLARATIVE)**

was: **V 3**

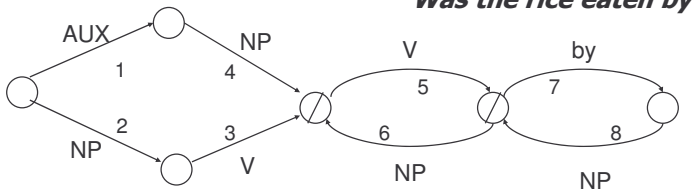
eaten: **V 5 (OBJ=rice; V=was eaten; AGFLAG=T; SUBJ=SOMEONE)**

by: **7 AGFLAG=F**

the cat: **NP 8 SUBJ=the cat**

Questa ATN permette di riconoscere anche frasi in forma interrogativa passiva, come:

"Was the rice eaten by the cat"

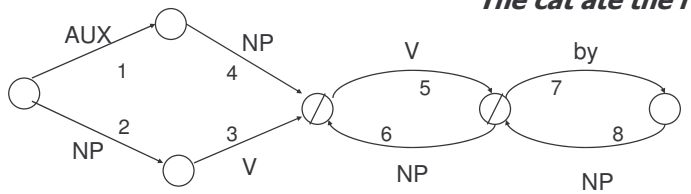


Percorrendo gli archi 1,4,5,7,8

Ma non permette di riconoscere frasi in forma attiva, come
"The cat ate the rice"
"The cat saw the TV"

Proviamo a riconoscere

"The cat ate the rice"



	test	*action*
•	T	SETR V*
		SETR TYPE'QUESTION
2	T	SETR SUBJ*
		SETR TYPE'DECLARATIVE
3	agrees SUBJ*	SETR V*
4	agrees* V	SETR SUBJ*
5	AND(GETF PPRT)(=V'BE)	SETR OBJ SUBJ
		SETR V*
		SETR AGFLAG TRUE
		SETR SUBJ'SOMEONE
6	TRANS V	SETR OBJ*
7	AGFLAG	SETR AGFLAG FALSE
8	T	SETR SUBJ*

The rice: **NP; arc 2 (SUBJ = the cat; DECLARATIVE)**

ate: **V 3**

The rice: **NP**

Devo necessariamente fermarmi perché non ho archi in uscita etichettati come NP (stringa non riconosciuta)

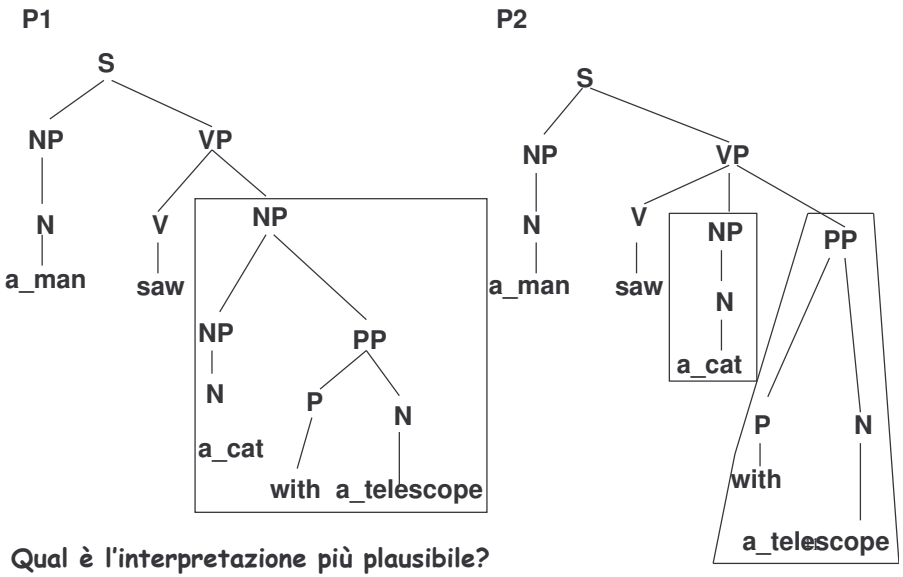
Parsing probabilistico

Non tutte le sequenze di parole si presentano, in un linguaggio, con la stessa frequenza.

Questa conoscenza può essere utilizzata per riconoscere e interpretare frasi in condizioni di incertezza.

In questo caso, il processo di riconoscimento produce, come risultato, un valore di probabilità per ciascuna delle interpretazioni possibili della stringa di input.

Esempio: "A man saw a cat with a telescope"



Qual è l'interpretazione più plausibile?

Quale grammatica

S → NP VP
VP → V NP
VP → V NP PP
NP → NP PP
NP → N
PP → P N

N → a_man
N → a_cat
N → a_telescope
P → with
V → saw

Ad es. il non terminale VP può essere riscritto in due modi differenti:

Quale tra i due è il più plausibile per 'The man saw a cat with a telescope'?

SOLUZIONE:

grammatiche probabilistiche

Grammatiche Probabilistiche

Si propongono di assegnare un valore di probabilità per ogni frase generabile con la grammatica.

Formalmente, la definizione di grammatica probabilistica differisce da quella delle grammatiche tradizionali soltanto nel senso che:

- *ad ogni produzione viene assegnato un valore di probabilità* e
- *la somma dei valori di probabilità associati alle produzioni che hanno, a sinistra, lo stesso non terminale è = 1.*

Esempio:

S → NP VP [.80]
S → AuxNP VP [.15]
S → VP [.05]
NP → Det Nome [.20]
NP → NomeProprio [.35]
NP → Nome [.05]
NP → ProNome [.40]

Nome → Noun [.75]
Nome → Noun Nome [.20]
Nome → NomeProprio Nome [.05]
VP → Verbo [.55]
VP → Verbo NP [.40]
VP → Verbo NP NP [.05]
Det → that [.05] the [.80] a [.15]
…

Nell'esercitazione sul parsing probabilistico affronteremo in dettaglio il problema (cruciale!) dell'**assegnazione delle probabilità** alle produzioni

Come usare l'incertezza nelle produzioni

La probabilità di un albero di parsing T per una frase S si può stimare come

$$P(T,S) = \Pi_{n \in T}\, p(r(n))$$

Se diverse interpretazioni sono possibili, l'algoritmo di parsing sceglierà, tra di esse, l'albero con probabilità massima.

Riprendiamo l'esempio precedente

associando ad ogni produzione un valore di probabilità

Qual è l'interpretazione più probabile, per 'A man saw a cat with a telescope'?

S → NP VP [1]
VP → V NP [.25]
VP → V NP PP [.75]
NP → NP PP [.5]
NP → N [.5]
PP → P N [1]

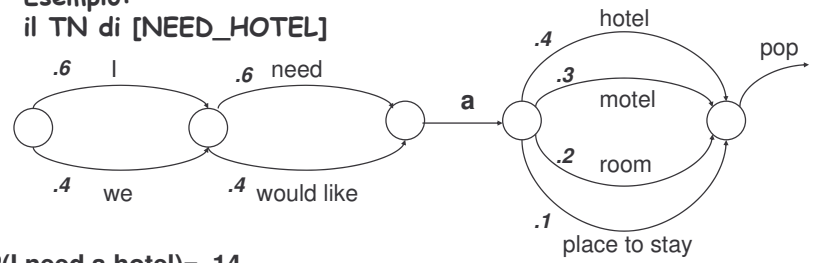
P1 **S ⇒ NP VP** [1] ⇒ **N** [.5] **V NP** [.25] ⇒ a_man saw **NP PP** [1] ⇒ a_man saw **N** [.50] **P N** [1] ⇒ a_man saw a cat with a_telescope
P(P1) = 1*.5*.25*.50*1= .0625

N → a_man
N → a_cat
N → a_telescope
P → with
V → saw

P2 **S ⇒ NP VP** [1] ⇒ **N** [.5] **V NP PP** [.75] ⇒ a_man saw **N** [.50] **P N** [1] ⇒ a_man saw a cat with a_telescope
P(P2) = 1*.5*.75*.50*1= .1875

E' possibile rappresentare l'incertezza associata a processi di interpretazione di frasi associando valori di probabilità agli archi dei diagrammi di transizione fra gli stati anziché alle produzioni

Esempio:
il TN di [NEED_HOTEL]



P(I need a hotel)= .14
P(I need a place to stay)= .04
P(we would like a room) = 03
…

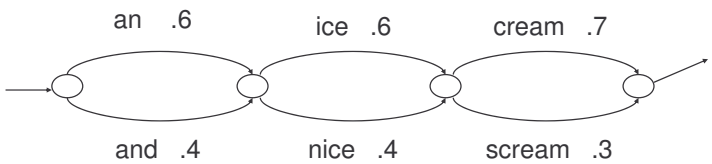
Associamo agli archi delle ATN un valore di probabilità, che indica *"qual è la probabilità di attraversare quell'arco, se ci si trova nello stato da cui l'arco inizia".*

La somma delle probabilità associate agli archi che escono da uno stato deve essere = 1.

Usi delle Probabilistic FSTN (o Markov Chain)

Nella interpretazione dello speech, i Probabilistic FSTN possono essere applicati al riconoscimento di stringhe 'poco chiare' (es. tool di speech recognition)

Esempio:

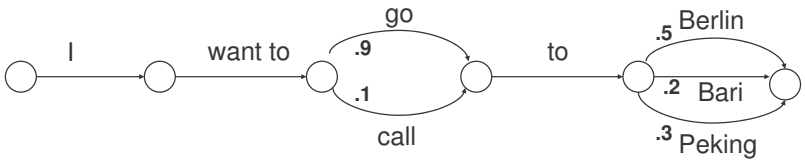


La probabilità di riconoscere una stringa è data dal prodotto delle probabilità associate agli archi attraversati.

P(an ice cream) = .25
P(and nice scream) = .048
P(an nice cream) = .17
…

Un altro esempio

Sistema automatico di prenotazioni di viaggi aerei: le probabilità in questo caso dipendono anche dal contesto in cui la frase viene pronunciata



"I want to go to Berlin": P=.45
"I want to go to Bari": P=.18
"I want to go to Peking": P=.27

"I want to call to Berlin": P=.05
"I want to call to Bari": P=.02
"I want to call to Peking": P=.03

Le Catene Markoviane aiutano a individuare l'interpretazione più probabile di frasi 'incomplete'

Esempio: un estratto di dialogo speech-based In un Call Center.

U: "I want to … to …!"
S: "Do you want to go to Berlin?"
U: "No, to …!"
S: "Oh sorry, it's Peking where you want to go, right?"

Ma possono essere usate anche per risolvere problemi dovuti a frasi parziali o a modi diversi di esprimere lo stesso concetto:

U: on June sixth from San Jose to Denver, United
U: return June seventh

Modi diversi di specificare date (di partenza o di arrivo)

U: I believe there's one leaving at eleven o'clock in the morning
U: I believe there's one at five sixteen

Modi diversi di suggerire un orario

U: ok
U: right
U: that's fine
U: ok thank you very much

Modi diversi di esprimere 'accettazione'

Semantica

L'**obiettivo** del riconoscimento in un sistema di dialogo è quello di **formalizzare le frasi in linguaggio naturale tramite un linguaggio ad hoc** (ad es. linguaggio logico), allo scopo di 'ragionare' su di esse.

The dog ate.
Ate(dog)

A man saw a cat with a telescope
Saw(man, CatWithTelescope) gatto dotato di telescopio ☺
Oppure: Saw(man, Cat, UsingTelescope) usando un telescopio
nelle due interpretazioni della frase che abbiamo visto.

Jack downloaded the computer game
Downoaded(Jack, ComputerGame)

Stato dell'arte

Attualmente non esistono sistemi in grado di applicare tecniche di NLU a domini illimitati per via

- della quantità di informazioni sul dominio *(domain knowledge)* che dovrebbe essere codificata per realizzare il processo di NLU in maniera efficiente
- della complessità dei problemi da affrontare quando si tratta di risolvere l'*ambiguità* del linguaggio naturale (polisemia, sinonimia, riconoscimento di entità, etc.)

Si cerca di semplificare il problema

- agendo su *domini* limitati
- agendo su *task* limitati

Altri esempi di task di NLU

Text Categorisation: assegnazione di una label categorica ad un testo fornito in input, a scelta in un set predefinito (es. news, articolo scientifico etc.)

Information Extraction: estrazione di informazioni a partire da testi in linguaggio naturale con lo scopo di aggiornare un db o formulare una query

Summarization: produrre una versione sintetica di un documento in input

Text mining: task analogo al data mining classico, riguarda l'estrazione di nuova conoscenza (fatti o trend) a partire da un corpus (molto vasto!) di documenti in input.

La formalizzazione dell'output avviene in accordo con il goal del task che sto realizzando

Quale rappresentazione utilizzare

Non solo le tecniche di natural language processing ma anche la conversione in forma simbolica (logica, template, elementi di un'ontologia etc) è problematica poichè

- non esiste un accordo generale su quale rappresentazione sia la migliore -> è task-dependent!
- di nuovo, si ripresentano i problemi legati all'ambiguità del linguaggio naturale e quindi alla disambiguazione del significato delle parole

NLU nei sistemi di dialogo

Noi siamo interessati ai sistemi di dialogo quindi per noi, *comprendere una mossa dell'utente* può tradursi in:

Formalizzazione logica della mossa stessa, in termini di atto comunicativo e contenuto della mossa

(approfondiremo meglio nell'esercitazione con il parser probablistico)

Aggiornamento di modelli utente probabilistici o logici

Aggiornamento dello stato del dialogo

(vedrete cosa significa quando affronterete la lezione sui sistemi di dialogo)

Riferimenti

A chi è particolarmente interessato all'argomento, è fortemente consigliata la lettura del Tutorial di Allen.

http://www.di.uniba.it/intint/people/ArticoliDid/ATN-Allen.htm