Luigi Rizzi
04-05

Linguistica computazionale

Elementi di base sulle grammatiche formali e gli automi astratti

I. Grammatiche formali

(1) Linguaggio L = insieme di stringhe, sequenze finite di elementi del vocabolario

(2) "…I will consider a language to be a set (finite or infinite) of sentences, each finite in length and constructed out of a finite set of elements… All natural languages are languages in this sense… Similarly, the set of "sentences" of some formalized system of mathematics can be considered a language" Chomsky 1957

(3) Dato un vocabolario A, l'insieme di tutte le stringhe costruite su A (A*) più l'operazione di concatenazione (formazione di stringhe complesse a partire da stringhe semplici) costituisce un monoide libero su A.

(4) Normalmente, non tutte le stringhe del monoide costituiscono frasi ben formate di L: per es, dato un frammento del lessico italiano, non tutte le sequenze arbitrarie sono frasi dell'italiano:

Il bambino corre
Corre il bambino
*il corre bambino
* bambino corre il
…..

Quindi L con vocabolario A è in genere un sottoinsieme proprio di A*;.

(6) Compito della grammatica di L è di catturare il sottoinsieme in questione generando tutte e sole le stringhe che lo compongono (capacità generativa debole). E di esprimere la struttura delle frasi di L (capacità generativa forte).

[il bambino] [legge [il libro]] e non * [[[[il] bambino] legge] il] libro]

(7) Per lo studio delle proprietà formali, è utile considerare linguaggi semplificati, con lessici molto ridotti e sintassi semplice, per es, linguaggi come i seguenti:

ab, abab, ababab, abababab,…
ab, aab, aabbbb, aaaaabb,...
ab, aabb, aaabbb, aaaabbbb...
aa, bb, abba, baab, abaaba, aaabbaaa, ...
aa, abab, abbabb, abbababbab,...
abc, aabbcc, aaabbbccc, aaaabbbbcccc

(8) Una grammatica formale è un sistema deduttivo di assiomi e regole di inferenza, che genera le frasi della lingua come suoi teoremi (capacità generativa debole). Inoltre, assegna rappresentazioni strutturali alle frasi (capacità generativa forte).

(9) Una grammatica formale è definita da una quadrupla :

(i) Un vocabolario terminale V_T
(ii) Un vocabolario non terminale V_N
(iii) Un assioma, o simbolo iniziale $S \in V_N$
(iv) Un insieme di regole di riscrittura $\varphi \rightarrow \psi$

(10) Nella sua forma più generale, la parte sinistra e la parte destra della regola di riscrittura sono stringhe qualsiasi costruite sui due vocabolari, con la sola restrizione che la parte sinistra deve contenere almeno un simbolo di V_N.

(11) Una derivazione è una sequenza di stringhe che parte dall'assioma e arriva fino a generare una stringa del linguaggio tramite le regole, eliminando via via i simboli non terminali.

Esempio:

V_T : {a, b}

V_N : {S, A, B}

Assioma: S

Regole:

$S \rightarrow A\ B\ S$
$S \rightarrow \varepsilon$
$AB \rightarrow BA$
$BA \rightarrow AB$
$A \rightarrow a$
$B \rightarrow b$

Esempio di derivazione:

S
ABS
BAS
BAABS
BAAB
bAAB
baAB
baaB
baab

(12) "ε" è la stringa vuota, l'elementi di identità rispetto alla concatenazione: $\varepsilon\varphi = \varphi\varepsilon = \varphi$

II. La gerarchia di Chomsky

(20) Chomsky (1956, 59) ha osservato che ponendo restrizioni via via più forti sulla forma delle regole si può stabilire una gerarchia di grammatiche di potere generativo decrescente

Tipo 0: sistemi di riscrittura non ristretti: $\varphi \rightarrow \psi$, con $\varphi \neq \varepsilon$ (alternativamente, con φ contenente almeno un simbolo non terminale).

Tipo 1: sistemi di riscrittura contestuali (context-sensitive): $A \rightarrow \psi / \alpha\_\_\_\beta$, con $\psi \neq \varepsilon$.

Tipo 2: sistemi di riscrittura acontestuali (context free): $A \rightarrow \psi$.

Tipo 3: sistemi regolari: $A \rightarrow xB$, $A \rightarrow x$

(21) Nelle regole dei linguaggi di tipo 0 qualunque stringa non nulla può essere riscritta come qualunque stringa (inclusa la stringa nulla). Questi sistemi di riscrittura caratterizzano i linguaggi ricorsivamente enumerabili (o computabili), quelli che possono essere generati da una macchina di Turing. Nei linguaggi ricorsivamente enumerabili, ogni frase grammaticale può essere generata (riconosciuta) in un numero finito di passi da una macchina di Turing; tuttavia, una frase agrammaticale può non essere riconosciuta in quanto tale in un numero finito di passi. I linguaggi di tipo 1 o inferiori sono ricorsivi: si può sempre decidere in un numero finito di passi se una frase è grammaticale o no.

(22) Un altro modo di concettualizzare le regole dei sistemi di tipo 1 consiste nel notare che riscrivono una sequenza di simboli in un'altra sequenza di simboli non decrescente: cioè l'output della regola ha almeno tanti simboli quanto l'input.

(23) I sistemi di tipo 2 riscrivono un simbolo non-terminale come qualunque stringa di terminali o non-terminali, inclusa la stringa nulla.

(24) I sistemi di tipo 3 hanno un simbolo non-terminale a sinistra della freccia, e lo riscrivono come una stringa di simboli con un solo non-terminale in fondo (regole lineari a destra), oppure all'inizio (regole lineari a sinistra).

(25) valgono le seguenti relazioni insiemistiche tra i linguaggi generati dai diversi tipi di grammatiche:

(i) i linguaggi di tipo 3 sono propriamente inclusi nei linguaggi di tipo 2;
(ii) i linguaggi di tipo 2 non contenenti ε sono propriamente inclusi nei linguaggi di tipo 1;
(iii) i linguaggi di tipo 1 sono propriamente inclusi nel linguaggi di tipo 0.

Alcuni esempi:

(26) $L = \{x \in \{a, b\}^* \mid x$ contiene un egual numero di a e di b, in qualsiasi ordine$\}$

aaababababbba

$G = < \{a, b\}, \{S, A, B\}, S, R >$

R =
$S \rightarrow aB$
$S \rightarrow \varepsilon$

$S \rightarrow bA$
$B \rightarrow b$
$B \rightarrow bS$
$A \rightarrow a$
$A \rightarrow aS$
$A \rightarrow bAA$
$B \rightarrow aBB$

---

(27) $L = \{x \in \{a, b\}^* \mid x = a^n b^n (n \geq 0)\}$

aaaaaabbbbbb

$G = < \{a, b\}, \{S\}, S, R >$

R =
$S \rightarrow aSb$
$S \rightarrow \varepsilon$

Linguaggio di tipo 2. Non può essere di tipo 3, intuitivamente, perché una volta generata la stringa di a la grammatica non ha modo di "ricordarsi" quante occorrenze di a ha prodotto per riprodurle con b.

---

(28) $L = \{x \in \{a, b\}^* \mid x = yz$, in cui z è l'immagine speculare di $y\}$

abbababbababba

$G = < \{a, b\}, \{S\}, S, R >$

R =
$S \rightarrow aSa$
$S \rightarrow bSb$
$S \rightarrow \varepsilon$

Anche questo linguaggio di tipo 2, ma non 3

---

(29) $L = \{x \in \{a, b\}^* \mid x = a^n b^m\}$

aaaabbbbbbbb

$G = < \{a, b\}, \{S, T\}, S, R >$

R =
$S \rightarrow aS$
$S \rightarrow bT$
$T \rightarrow bT$
$S \rightarrow a$
$S \rightarrow b$
$T \rightarrow b$

Questo è un linguaggio di tipo 3, visto che, una volta passato dalla generazione di a alla generazione di b, non ha il problema di "ricordarsi" il numero delle occorrenze.

(30) $L = \{x \in \{a, b,c\}^* \mid x = a^n b^n c^n\ (n \geq 0)\}$

aaaaaabbbbbbccccccc

G = < {a, b,c}, {S, T}, S, R>

R =
$S \rightarrow$ aTSc
$S \rightarrow$ abc
$Ta \rightarrow$ aT
$Tb \rightarrow$ bb

Questo è un linguaggio di tipo 1, una grammatical context free non può generarlo perché, intuitivamente, una volta generato un egual numero di a e di b, non ha modo per "ricordarsi" di quanti c deve generare: per es, la seguente grammatica context free ha questo problema:

$S \rightarrow$ aTbPc
$T \rightarrow$ aTb
$P \rightarrow$ Pc

III. Pumping Lemma

(30) Come possiamo sapere se un dato linguaggio non è regolare?

(31) Pumping Lemma: Supponiamo che L sia un linguaggio regolare infinito. Allora qualunque stringa di L deve poter essere divisa in tre parti x, y, z, tali che la stringa $x\ y^n\ z$ (ottenuta "pompando" il segmento di mezzo) appartenga a L.

(32) Quindi, $a^n b^n$ $(n \geq 0)$ non è un linguaggio regolare: prendiamo una stringa, per es. aaaaabbbbb

1. se segmentiamo aaa – aa – bbbbb, "pompando" il segmento di mezzo perdiamo l'egual numero di a e b;

2. se segmentiamo aaaaa – bb – bbb, vale la stessa conclusione;

3. se segmentiamo aaaa – ab – bbbb, perdiamo il fatto che una sequenza continua di a sia seguita da una sequenza continua di b;
QED

(33) Invece, $a^n b^m$ è regolare: data una stringa qualunque aaaabbbbbbb, se la segmentiamo in aa – aa – bbbbbbb, possiamo "pompare" la stringa di mezzo ottenendo una stringa possibile nel linguaggio. Stessa conclusione se segmentiamo aaaa – bbbb – bbb.

IV. Le lingue naturali sono regolari?

(34) L'inglese non è una lingua a stati finiti (regolare) (Chomsky 1956, 57, 59)

(35)a If S1 then S2
b Either S3 or S4
c The man who said S5 is arriving today

(36) Queste opzioni danno luogo a strutture di tipo xx^R che non sono regolari. Più semplicemente, ci sono nelle lingua naturali delle strutture di tipo $a^n b^n$, che già sappiamo non essere regolari per il Pumping Lemma:

(37) Se è vero che, se la ditta è in crisi allora ci sarà una riduzione di personale, allora ci sarà uno sciopero.

(38) Se … se … se … allora … allora … allora

V. Le lingue naturali sono context-free?

(39) Lingue con due identiche stringhe concatenate (xx) non sono context free

(40) Le lingue umane presentano dipendenze attraverso serie di tipo

X1, X2, …Xn,… Y1, Y2, … Yn

Per esempio,

(41) Gianni, Maria, Francesca e Guido sono stati rispettivamente promosso, bocciata, promossa, bocciato

Con l'n-esimo participio accordantesi con l'n-esimo nome.

VI. Sulle capacità computazionali dei primati non umani

(42) T. Fitch – M. Hauser (to appear in Science) Computational constraints on syntactic processing in nonhuman primates. They compare:

A finite state grammar (FSG): $(AB)^n$
OK: AB, ABAB, ABABAB, ABABABAB,…
*: A, BA, ABA, AAB, ABB, AABB, ABABA,…

A phrase structure grammar (PSG): $A^n B^n$
OK: AB, AABB, AAABBB, AAAABBBB,…
*A, BA, ABA, ABB, AABBA,….

NB: A's and B's are arbitrary CV syllables made clearly distinguishable by having A pronounced by male voice and B by female voice.

(43) Humans, habituated to sentences of both languages, immediately figure out the pattern. ("same kind", "different kind" judgment.

(44) Tamarins figure out violations of FSG (they pay more attention to illegal sequences with a head turning paradigm. But do not figure out PSG violations.

(45) What would happen with more minimally different grammars, i.e. the following?

FSG: $A^n B^m$
PSG: $A^n B^n$

The latter admits: AB, AAB, ABB, AABB, AABB, AAAB, ABBB,….

VII. Automi: Macchine di Turing

(50) Macchina di Turing: macchina computante astratta:

- Unità di controllo (con numero finito di stati e istruzioni esplicite sulle azioni da compiere)
- Testina di lettura-scrittura (può spostarsi in entrambe le direzioni e leggere o scrivere un simbolo alla volta)
- Nastro a caselle (illimitato in entrambe le direzioni)

(51) La macchina può compiere tre operazioni elementari su una sequenza di simboli:
- cancellare il simbolo in lettura e sostituirlo con un altro simbolo;
- spostarsi di una casella a sinistra o a destra
- fermarsi

(52) Le istruzioni esplicite sono definite dalle seguenti possibili quadruple:

1. $S_i\ a_j\ a_k\ S_l$
2. $S_i\ a_j\ R\ S_l$
3. $S_i\ a_j\ L\ S_l$

(53) Il comportamento della macchina è definito ad ogni istante dal suo stato in interno e dal simbolo in lettura sul nastro. La macchina dispone di una memoria potenzialmente infinita perché può usare il nastro potenzialmente infinito per memorizzare informazioni. La macchina si forma quando non c'è nessuna quadrupla che cominci con il suo stato corrente e il simbolo corrente.

(54) La macchine di Turing sono equivalenti ai sistemi di riscrittura non ristretti

VIII. Automi a stati finiti.

(55) Un automa a stati finiti consta di:

- Unità di controllo (con numero finito di stati e istruzioni esplicite sulle azioni da compiere)
- Testina solo di lettura (o solo di scrittura): può spostarsi solo in una direzione e leggere (o scrivere) un simbolo alla volta.
- Nastro a caselle (illimitato in una direzione)

(56) La differenza fondamentale rispetto alla macchina di Turing è che la macchina a stati finiti non può usare il nastro come supporto di memoria, quindi le sue capacità di memoria sono limitate ai suoi stati interni.

(57) Le istruzioni hanno la forma

$(S_i , a_j) \rightarrow S_k$

"se la macchina è nello stato S_i e legge il simbolo a_j, allora passa allo stato S_k"

(58) Le macchine a stati finiti possono essere rappresentate come diagrammi (o grafi) di stati.

(59) Le macchine a stati finiti generano i linguaggi regolari

IX. Automi a Push-Down Storage.

(60) Sono automi a stati finiti cui si aggiunge un nastro di memoria.

(61) Le istruzioni hanno la forma:

$(p, S_i , q) \rightarrow (S_k, r)$

"se sei nello stato S_i e leggi p nella stringa di input e q nella memoria, allora scrivi r nella memoria, e passa allo stato S_k"

$(p, S_i , q) \rightarrow (S_k, sigma)$

"se sei nello stato S_i e leggi p nella stringa di input e q nella memoria, allora cancella il simbolo in lettura nella memoria e passa allo stato S_k"

(62) Gli automi a Push-Down Storage sono equivalenti alle grammatiche context-free.

X. Complessità nella competenza e nell'esecuzione

(39)Le lingue naturali ammettono certe strutture grammaticali che sono difficili da utilizzare. Un caso classico è l'autoincassamento:

(40) Il cane ha inseguito il gatto che ha catturato il topo che ha mangiato il formaggio

(41) # Il formaggio che il topo che il gatto ha catturato ha mangiato era sul tavolo

(42) ## Il formaggio che il topo che il gatto che il cane ha inseguito ha catturato ha mangiato era sul tavolo

(43) Gianni è consapevole del fatto che Mario tema la possibilità che Francesca sia malata

(44) (#) Il fatto che la possibilità che Francesca sia malata sia nota preoccupa Mario

(45) ## Il fatto che la possibilità che l'ipotesi che Francesca sia malata sia nota preoccupi Mario è evidente

(46) La nonna continuerà a dormire poiché il padre darà una mano se la madre si sveglia quando il bambino piange

(47) # Poiché se quando il bambino piange la madre si sveglia il padre darà una mano, la nonna continuerà a dormire