Analisi sintattica efficiente

Giacomo Baldi

Elaborazione del Linguaggio Naturale

Il problema

- Data una frase già segmentata in tokens si vuole costruire in modo efficiente la struttura della frase secondo una grammatica che descrive il linguaggio.
- Occorre quindi definire con accortezza la grammatica.

La grammatica

- La grammatica viene inizialmente data in BNF

S ::= NP VP | S Congiunzione S

NP ::= Pronome | Sostantivo | Articolo Sostantivo | NP PP | NP Relativa

VP ::= Verbo | VP NP | VP Aggettivo | VP PP | VP Avverbio

PP ::= Preposizione NP

Relativa ::= CongRel VP

Sostantivo ::= …

Verbo ::= … → Presenti nel dizionario

Aggettivo ::= …

La grammatica aumentata

- Sovragenerazione: “Me smells a flower”
- Una grammatica in BNF avrebbe un numero enorme di regole se si dovessero esplicitare tutte per garantire le concordanze.
- Si ricorre all’aumento delle categorie sintattiche

S ::= NP(nominativo) VP | …

NP(caso) ::= Pronome(caso) | Nominativo | Articolo Nominativo …

VP ::= VP NP(accusativo) | …

Sottocategorizzazione dei verbi

- Ad ogni verbo viene associata la lista dei complementi che possono accompagnarlo.

GIVE ME SOMETHING	[NP, NP]
GIVE IT TO ME	[NP, PP]

VP (cat) ::=
- | VP([NP, cat]) NP (accusativo)
- | VP([Aggettivo, cat]) Aggettivo
- | VP([PP, cat]) PP
- | Verbo (cat)

GIVE IT TO ME
Verbo([NP,PP])
VP([NP,PP])
VP([NP,PP]) NP
VP([PP])
VP([PP]) PP
VP([])

Sottocategorizzazione dei verbi (2)

Affinchè una frase sia riconosciuta almeno una lista di complementi del verbo deve essere soddisfatta

S ::= NP(Nominativo) VP([])

Per rendere espressiva la grammatica si aggiungono i **complementi esterni**

VP(cat) ::= VP(cat) PP | VP(cat) Avverbio

I walked Sunday
- VP([NP]) NP(accusativo)
- VP([]) PP

Parser nondeterministico bottom-up

Input: parole (vettore di simboli), grammatica (insieme di regole)

Controlla che la lunghezza di parole (n) sia 1 e che il simbolo iniziale sia S —si→ Ritorna parole[1]

↓ no

Scegli i tra {1..n}

Scegli regola da grammatica di lunghezza m

Se parole[i..i+m-1] combacia con la parte destra di regola —no→ fallisci

↓ si

Aggiorna parole[i..i+m-1] con il simbolo a sinistra di regola

Parsing Top-Down

Un parser top-down (o predittivo) cerca di interpretare una frase con una derivazione leftmost della grammatica associata al linguaggio.

I DRINK TEA

S → NP VP → Pronome VP → “I” VP → “I” Verbo NP →

→ “I DRINK” NP → “I DRINK TEA”

Parsing Bottom-Up

Un parser bottom-up cerca di interpretare una frase con una derivazione rightmost della grammatica associata al linguaggio.

I DRINK TEA

"I DRINK TEA" → Pronome "DRINK TEA" → NP "DRINK TEA"
→ NP Verbo "TEA" → NP Verbo NP → NP VP → S

Verso un'analisi efficiente

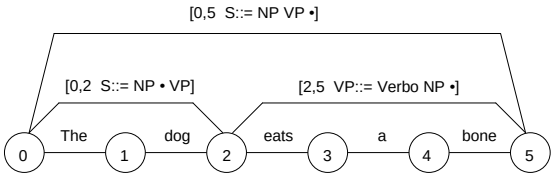
- Non fare due volte ciò che può essere fatto una sola volta
- Non fare ciò che si può evitare
- Non rappresentare ciò che non è necessario

Chart parsing

I risultati parziali vengono memorizzati in una struttura detta chart.

Un chart contiene nodi ed archi etichettati (completi e incompleti).

Un arco consiste dell'indicazione dei nodi iniziali e terminali e della regola che deve venir completata.



Metodi per il chart parsing

Fare chart parsing significa aggiungere nodi ed archi al chart fino a terminare la frase e ottenere come soluzioni tutti gli archi completi che terminano sull'ultimo nodo del chart.

Inizializzatore: aggiunge al chart il primo nodo e il primo arco.

Predittore: scelto un arco incompleto che cerca una categoria C, aggiunge al chart un nuovo arco che se completato darebbe C.

Estensore: prende due archi. Uno incompleto che termina al nodo j e che cerca una C. L'altro completo che inizia al nodo j e aggiunge un arco in cui la C è stata trovata.

Analizzatore: Stesso comportamento dell'estensore tranne per il fatto che invece di prendere un arco completo prende una parola dallo stream di input.

Versione nondeterministica

Inizializzazione:

Chart ← [0,0, S' ::= • S]

While (altri archi possono essere aggiunti) do

arco ← scegli [i, j, A ::= α • B β] dal chart

scegli uno fra i seguenti metodi che abbia successo:

Predittore:

scegli (B ::= γ) dalla grammatica

aggiungi [j,j, B ::= • γ] al chart

Estensore:

scegli [j, k, B ::= F •] dal chart

aggiungi [i, k, A ::= α B • β] al chart

Analizzatore:

se la parola corrente è di categoria B

allora aggiungi [j, j+1, A ::= α B • β] al chart

Esempio

$_0$ I $_1$ SMELL $_2$ FOOD $_3$

1) [0, 0, S' ::= • S]	→	Inizializzatore
2) [0, 0, S ::= • NP VP]	→	predittore applicato a 1
3) [0, 1, S ::= NP • VP]	→	analizzatore applicato a 2
4) [1, 1, VP ::= • Verbo NP]	→	predittore applicato a 3
5) [1, 2, VP ::= Verbo • NP]	→	analizzatore applicato a 4
6) [1, 3, VP ::= Verbo NP •]	→	analizzatore applicato a 5
7) [0, 3, S ::= NP Verbo NP •]	→	estensore applicato a 3 e 6

Versione deterministica

ChartParser(*stringa*, *grammatica*)
Aggiungi([0,0, S' ::= • S])
Per ogni parola x in *stringa* chiama Analizzatore(x, *stringa*[x])

Predittore([i, j, A ::= α • B β]):
per ogni (B ::= γ) in grammatica
Aggiungi([j,j, B ::= • γ])

Aggiungi(*arco*):
se *arco* è già presente in chart[fine(*arco*)] non fa nulla
altrimenti
inserisci arco in chart[fine(*arco*)]
se l'arco è completo allora Estensore(*arco*)
altrimenti Predittore(*arco*)

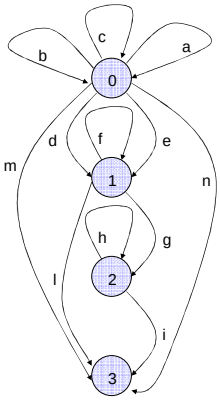
Estensore ([j, k, B ::= γ •]):
per ogni [i, j, A ::= α • B' β] in chart[j]
se B=B' Aggiungi([i, k, A ::= α B' • β])

Analizzatore(*j*, *parola*):
per ogni arco [i, indice, A::= α • B β] in chart[indice]
se *parola* è di categoria B allora Aggiungi([i,j+1, A::= α B • β])

Chart

I EAT PIZZA

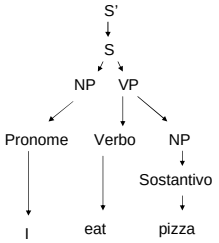
Arco	Metodo	Chart
a	Inizializzazione	[0,0, S' ::= • S]
b	Predittore(a)	[0,0, S ::= • NP VP]
c	Predittore(b)	[0,0, NP ::= • Pronome]
d	Analizzatore(1, I)	[0,1, NP ::= Pronome •]
e	Estensore(d) (b)	[0,1, S ::= NP • VP]
f	Predittore(e)	[1,1, VP ::= • Verbo NP]
g	Analizzatore(2, eat)	[1,2, VP ::= Verbo • NP]
h	Predittore(g)	[2,2, NP ::= • Sostantivo]
i	Analizzatore(3, pizza)	[2,3, NP ::= Sostantivo •]
l	Estensore(i) (g)	[1,3, VP ::= Verbo NP •]
m	Estensore(l) (e)	[0,3, S ::= NP VP •]
n	Estensore(m) (a)	[0,3, S' ::= S •]



Estrazione delle analisi

- Modificando opportunamente Estensore in modo che associ all'arco che produce, i due archi che fonde si possono estrarre gli alberi di analisi dal chart.

Arco	Metodo	Chart
a	Inizializzazione	[0,0, S' ::= • S]
b	Predittore(a)	[0,0, S ::= • NP VP]
c	Predittore(b)	[0,0, NP ::= • Pronome]
d	Analizzatore(1, I)	[0,1, NP ::= Pronome •]
e	Estensore(d) (b)	[0,1, S ::= NP • VP]
f	Predittore(e)	[1,1, VP ::= • Verbo NP]
g	Analizzatore(2, eat)	[1,2, VP ::= Verbo • NP]
h	Predittore(g)	[2,2, NP ::= • Sostantivo]
i	Analizzatore(3, pizza)	[2,3, NP ::= Sostantivo •]
l	Estensore(i) (g)	[1,3, VP ::= Verbo NP •]
m	Estensore(l) (e)	[0,3, S ::= NP VP •]
n	Estensore(m) (a)	[0,3, S' ::= S •]



Ambiguità

Fall leaves fall and spring leaves spring

Le foglie autunnali cadono e le foglie primaverili nascono

Le foglie autunnali cadono e la primavera lascia la primavera

L'autunno lascia l'autunno e le foglie primaverili nascono

L'autunno lascia l'autunno e la primavera lascia la primavera

Ambiguità derivante dalle interpretazioni molteplici di fall, leaves e spring.

Ogni sottofrase è ambigua in due modi per cui 4 interpretazioni.

N sottofrasi = 2^N interpretazioni

Impaccamento

- Si possono modificare Estensore e Aggiungi per ottenere una foresta impaccata di rappresentazioni.

[S [S { [NP fall leaves] [VP fall] / [NP fall] [VP leaves fall] }] and [{ [NP spring leaves] [VP spring] / [NP spring] [VP leaves spring] }]]

2N invece di 2^N = ottimizzazione esponenziale dello spazio

Il parser più efficiente

- Il chart parser più efficiente è stato prodotto da Robert Moore per Microsoft

Preprocessing con fattorizzazione sinistra della grammatica:

A ::= BC | BD → A ::= BA' A' ::= C | D

Implementazione del chart come hash table

Rappresentazione solo della parte a destra del punto negli archi

$A :: = \alpha X \bullet \beta \rightarrow A ::= \bullet \beta$

Bottom-up filtering anticipato rispetto al top-down

	CT Grammar	ATIS Grammar	PT Grammar
Rules	24,456	4,592	15,039
Nonterminals	3,946	192	38
Terminals	1,032	357	47
# Test Sentences	162	98	30
Average Length	8.3	11.4	5.7
# Grammatical	150	70	30
Average # Parses	5.4	940	7.2×10^{27}

Table 1: Grammars and test sets for parser evaluations

	CT Grammar	ATIS Grammar	PT Grammar
LC_1	4.3	15.6	45.0
LC_2	3.4	11.9	43.0
LC_3	3.1	11.6	41.8
LC_4	2.7	11.8	42.3

Table 2: LC parsing algorithm performance comparisons

		CT Grammar	ATIS Grammar	PT Grammar
LC_1	UTF	4.3	15.6	45.0
LC_1	FLF	7.4	63.5	timed out
LC_1	PLF	6.2	66.2	timed out
LC_1	BUPM	3.6	11.7	34.1
LC_2	UTF	3.4	11.9	43.0
LC_2	FLF	5.1	38.2	timed out
LC_2	PLF	4.2	37.7	timed out
LC_2	BUPM	3.1	7.0	27.0
LC_3	UTF	3.1	11.6	41.8
LC_3	FLF	4.2	12.3	45.4
LC_3	PLF	3.8	12.1	43.6
LC_3	BUPM	5.0	17.1	64.6
LC_4	UTF	2.7	11.8	42.3
LC_4	FLF	3.6	11.9	46.6
LC_4	PLF	3.2	11.7	44.4
LC_4	BUPM	3.2	14.7	63.6

Table 3: LC parsing grammar transformation performance comparisons

Parsing probabilistico

- Un parser probabilistico sceglie le regole da espandere in base a probabilità calcolate con numerosi meriti per arrivare il prima possibile ad un'analisi e restituirla come "più probabile"

Si cerca di massimizzare $p(N^i_{j,k}|t_{0,n})$ dove $t_{0,n}$ è la sequenza dei primi n+1 tokens e $N^i_{j,k}$ è la categoria i-esima che copre la frase da j a k.

Conoscendo queste probabilità si possono stabilire criteri di scelta delle categorie da espandere per raggiungere prima l'analisi voluta.

Parsing probabilistico

La probabilità che la categoria N^i copra da j a k data la stringa di caratteri da 0 a n è esprimibile in termini di α e β. α è dato dalla probabilità esterna mentre β è la probabilità interna.

Poiché α e il denominatore rappresentano l'influenza delle parole che circondano N^i , possiamo eliminarlle e usare solo β. Questo metodo è detto del β diretto. Può portare a parser inefficienti perché eliminando il denominatore non viene presa in considerazione l'influenza della lunghezza della sequenza, preferendo così sempre sequenze corte.

$$p(N^i_{j,k}|t_{0,n}) = \frac{p(N^i_{j,k}, t_{0,n})}{p(t_{0,n})} = \frac{p(N^i j,k,t_{0,j},tj,k,tk,n)}{p(t_{0,n})} = \frac{p(t_{0,j},N^i_{j,k},t_{k,n})p(t_{j,k}|t_{0,j},N^i_{j,k},t_{k,n})}{p(t_{0,n})}.$$

$$p(N^i_{j,k}|t_{0,n}) \approx \frac{p(t_{0,j},N^i_{j,k},t_{k,n})p(t_{j,k}|N^i_{j,k})}{p(t_{0,n})} = \frac{p(t_{0,j},N^i_{j,k},t_{k,n})\beta(N^i_{j,k})}{p(t_{0,n})}.$$

$$p(N^i_{j,k}|t_{0,n}) \approx \frac{\alpha(N^i_{j,k})\beta(N^i_{j,k})}{p(t_{0,n})}.$$

Altri metodi

β normalizzato: per tenere conto della lunghezza delle sequenze si utilizza come parametro per la scelta $\sqrt[k-j]{\beta(N^i_{j,k})}$.

$\alpha_L\beta$ normalizzato: per tenere conto della lunghezza delle sequenze e delle parole già parsate si usa $\sqrt[k]{\alpha_L(N^i_{j,k})\beta(N^i_{j,k})}$.

Trigram: Si fanno stime sulla frequenza di gruppi di tre parole e non di tutto il contesto precedente $\frac{p(N^i)\beta(N^i_{j,k})}{p(t_{j,k}|t_{j-2,j})}$.

Prefix: Si fanno stime sulla frequenza del gruppo di parole precedente normalizzato $\frac{\alpha_L(N^i_{j,k})\beta(N^i_{j,k})}{p(t_{0,k})}$

Risultati

Figure of Merit	%E	%non-0 E	%popped
straight β	97.6	97.5	93.8
normalized β	34.7	31.6	61.5
normalized $\alpha_L\beta$	39.7	36.4	57.3
trigram estimate	25.2	21.7	44.3
prefix estimate	21.8	17.4	38.3

Figure of Merit	CPU time
straight β	3966
normalized β	1631
normalized $\alpha_L\beta$	68660
trigram estimate	1547
prefix estimate	26520

Le tabelle riassumono i risultati dei vari metodi.

%E indica la percentuale di archi inseriti rispetto ad un parser esaustivo.

%non-0 E sono le produzioni non vuote

%popped è la percentuale degli archi utilizzati per arrivare ad una soluzione.

I modelli migliori (prefix e $\alpha_L\beta$) risultano costosi poiché il calcolo dei parametri è impegnativo

Parsing incrementale

1) Possibilità di eseguire operazioni su parti d stringa già parsate in modo da garantire una riorganizzazione efficiente degli alberi di analisi.

2) L'input viene fornito al parser man mano che l'utente lo inserisce

3) Il parser, monitorando il processo di input, guida l'utente nella composizione del testo dando suggerimenti o chiedendo chiarificazioni.

Operazioni disponibili

Si definisce una relazione di dipendenza D fra archi del chart tale che un arco a è in relazione D con un arco b (a D b) se a è stato inserito nel chart o tramite predizione da b o tramite estensione con b.

Inserzione:

se avviene all'estrema destra dello stream di input viene trattata come una semplice chiamata ad Analizzatore .

se avviene in posizione x precedente allora vengono

1) rimossi gli archi dipendenti da quello in posizione x

2) aggiornati tutti gli archi con posizioni maggiori di x ai nuovi valori shiftati

3) rieseguito Analizzatore e aggiornato il chart

Operazioni disponibili (2)

Delezione:

se avviene all'estrema destra dello stream di input vengono rimossi tutti gli archi in relazione D* con l'ultimo aggiunto.

se avviene all'interno della stringa già parsata in posizione x:

1) Si rimuovono tutti gli archi in relazione D* con l'arco preterminale in posizione x

2) si shiftano a sinistra tutti gli archi con posizioni maggiori di x

Sostituzione:

si può realizzare attraverso combinazione di delezione e inserzione ma viene spesso implementata separatamente per motivi di efficienza.

Limiti e Benefici

1) Il parser incrementale così costruito non tiene conto delle relazioni fra periodi diversi, trattando ogni frase come indipendente. Questa limitazione può essere superata perdendo però notevolmente in efficienza.
2) Mantenere le dipendenza causa un aumento considerevole della memoria occupata, per questo si è pensato a varianti che utilizzino meno informazione peggiorando lievemente il tempo d'esecuzione
3) Un parser incrementale è un tool utilissimo in contesti come linguaggi di programmazione (individuazione di bug prima della compilazione), e linguaggio naturale (composizione testi assistita).

Conclusioni

In definitiva il problema dell'analisi sintattica non ha una soluzione unica ma viene adattato di volta in volta al problema che si vuole affrontare.

In ambito statistico sono infatti nati i parser probabilistici, in uno più applicativo i parser incrementali.

I passi da fare sono quelli di unificare gli sforzi verso un unico modello di parser che sia efficiente, probabilistico e incrementale.

In più si dovrebbe riuscire a compiere durante l'analisi sintattica una parte importante di analisi semantica per guidare soprattutto la disambiguazione e la scelta dell'analisi nel chart

Bibliografia

- Russel-Norving: **Intelligenza Artificiale**
- Robert Moore: **Improved Left-Corner Chart Parsing for Large Context-Free Grammars**
- A. Caraballo, E. Charniak: **Figures of Merit for Best-First Probabilistic Chart Parsing**
- Mats Wirèn:**Interactive Incremental Chart Parsing**