



Andrea Cacioli

Progetto Tecnologie per il Linguaggio Naturale (Parte 1 - Linguistica Computazionale Generale)

- Consegna CKY: tlhIngan maH!

Sviluppo

- Fase 1 - Implementazione struttura dati e Algoritmo CKY

Fase 2 - Grammatiche Inglesi e Klingon

Grammatica Inglese

- Never Ever
- Multiple adjectives

- Grammatica Klingon

Progetto Tecnologie per il Linguaggio Naturale (Parte 1 - Linguistica Computazionale Generale)

Consegna CKY: tlhIngan maH!

La consegna richiedeva di implementare l'algoritmo di riconoscimento CKY per l'identificazione di stringhe generate da una grammatica libera dal contesto.

Tale algoritmo è un riconoscitore e in quanto tale si preoccupa solo di dire se esiste un albero che genera la stringa senza però fornirlo.

Inoltre è richiesto di testare tale algoritmo su delle frasi inglesi a partire dalla grammatica L1 fornita nel Jurafsky.

Successivamente è richiesto che venga scritta una grammatica per la lingua Klingon per le sole frasi:

- tlhIngan Hol Dajatlh'a'? (Do you speak Klingon?)
- puq vllagh jIH (I see the child)
- pa'Daq jIHtaH (I'm in the room)
- tlhIngan maH! (We are Klingon!)

Sviluppo

Fase 1 - Implementazione struttura dati e Algoritmo CKY

Per questo progetto ho deciso di non limitarmi al solo riconoscitore CKY che restituisce vero se una stringa appartiene al linguaggio generato da una grammatica, ma invece di implementare il parser completo che restituisce l'albero (o gli alberi) che genera una tale stringa.

Per fare tutto questo si é implementata una classe GraphNode che rappresenta un nodo di un grafo con però delle peculiarità:

- Un nodo ha dei figli
- Un figlio può avere numerosi padri
- Figli diversi possono avere padri in comune

Tale struttura dati ci permetterà di trovare tutti gli alberi che un algoritmo genera a partire da un nodo start di una CFG.

Tale implementazione permette inoltre di avere una rappresentazione ad albero anche su terminale quando si stampa un nodo.

Successivamente si è implementato l'algoritmo vero e proprio utilizzando la matrice e costruendo utilizzando un approccio di programmazione dinamica tutti i sottoalberi delle sottostringhe della stringa di partenza.

Per rappresentare la grammatica ho usato le funzionalità di nltk, in particolare per rappresentare la grammatica inglese con un paio di mie aggiunte ho scritto:

```

grammar = CFG.fromstring(
    """
    S -> NP VP
    S -> X1 VP
    X1 -> Aux NP
    Aux -> does | do
    NP -> I | she | me | you
    NP -> Huston | NWA
    NP -> Det Nominal
    NP -> Det Noun
    Nominal -> morning | quick | cool | adventurous | mountain | cold
    Nominal -> Nominal Noun
    Nominal -> Nominal PP
    Nominal -> X3 Nominal
    X3 -> X3 Nominal
    X3 -> morning | quick | cool | adventurous | mountain
    VP -> book | include | prefer | love | like | drink
    VP -> Verb Nominal
    VP -> Verb NP
    VP -> Verb PP
    VP -> X2 PP
    X2 -> Verb NP
    VP -> VP PP
    VP -> Adverb VP
    PP -> Preposition NP
    Det -> that | this | the | a
    Noun -> book | flight | meal | money | day | water
    Verb -> book | include | prefer | love | like | drink
    Pronoun -> I | she | me | you
    Proper-Noun -> Huston | NWA
    Preposition -> from | to | on | near | through
    Adverb -> Adv1 Adv2
    Adverb -> definitely | often | never | really | rarely
    Adv1 -> never | rarely
    Adv2 -> ever
    """)

```

Questo oggetto é quindi passato alla funzione parse del mio modulo [CKY.py](#) insieme alla stringa che si vuole parsificare.

Ad esempio se si volesse ottenere l'albero della stringa "book that flight through Huston", basterebbe chiamare il parser nel seguente modo:

```
from CKY import parse

matrix = parse("book that flight through Huston", grammar)
```

Il valore di ritorno della funzione parse é la matrice su cui ha lavorato l'algoritmo CKY.

L'algoritmo ad ogni iterazione svolge due fasi: la prima in cui riempie l'elemento sulla diagonale principale della matrice con nodi relativi alle produzioni che portano ai simboli terminali presenti nella stringa alla posizione che é sotto analisi, la seconda fase é quella in cui "guarda" come potrebbe essere possibile arrivare a tali nodi tramite altre regole.

Alla fine dell'esecuzione si possono considerare alberi validi per una tale stringa quelli che si trovano nella posizione in alto a destra della matrice ([0][len(matrix)]) che hanno come simbolo il simbolo di partenza della grammatica S.

Stabilito questo si possono stampare solamente gli alberi che si reputano validi in questo modo:

```
import CKY

table = CKY.parse(string, grammar)
for node in table[0][len(table)]:
    if node.name == "S" and node.getString() == string:
        node.printTreeString()
        print()
```

Output (La frase si legge dal basso verso l'alto):

Processing the string: you rarely ever drink cold water

```
S
|___VP
|   |___VP
|       |___Nominal
|           |___Noun
|               |___water
|                   |___Nominal
|                       |___cold
|                           |___Verb
|                               |___drink
|                                   |___Adverb
|                                       |___Adv2
|                                           |___ever
|                                               |___Adv1
|                                                   |___rarely
|___NP
|   |___you
```

Fase 2 - Grammatiche Inglesi e Klingon

In questa fase discuterò come ho deciso di scrivere le grammatiche.

Grammatica Inglese

Sono partito con la grammatica L1 scritta nel Jurafsky ma ho fatto un paio di aggiunte per vedere provare a vedere se riuscivo a descrivere delle feature della lingua inglese.

Never Ever

Una prima feature che ho provato a descrivere sono state le phrase in cui due parole non possono essere separate perché non avrebbero senso se prese da sole. Per esempio la frase "I would never ever try parachuting" che significa "Non proverei mai e poi mai il paracadutismo" non potrebbe essere parsificato dalla sola grammatica L1.

Per questo si sono aggiunte le seguenti regole di derivazione:

- VP -> Adverb VP
- Adverb -> Adv1 Adv2
- Adverb -> definitely | often | never | really | rarely
- Adv1 -> never | rarely
- Adv2 -> ever

In questo modo é possibile avere frasi come quella menzionata poco sopra e altre che contengono la phrase "rarely ever"

Multiple adjectives

In inglese a volte capita che si concatenino numerosi aggettivi uno dopo l'altro prima di un nome. Sebbene la grammatica L1 permettesse già di apporre degli aggettivi a dei nomi, si sono aggiunte le seguenti produzioni in modo tale da rendere questa pratica potenzialmente infinita.

- Nominal -> X3 Nominal
- X3 -> X3 Nominal
- X3 -> morning | quick | cool | adventurous | mountain

Grazie a queste produzioni si possono scrivere e parsificare frasi come: **"do you like a cool adventurous mountain day"**

Attenzione: in realtà in inglese non sarebbe possibile avere gli aggettivi nell'ordine che si vuole ma di solito devono seguire un ordine, tuttavia questo non é catturato dalla mia grammatica.

Grammatica Klingon

La grammatica per la lingua klingon che ho creato é la seguente:

```
klingonGrammar = CFG.fromstring(
    """
    S -> NP VP
    VP -> Verb NP
    VP -> Dajatlha | vIlegh | jIHtaH | maH
    NP -> Adj Noun
    NP -> tlhIngan | Hol | puq | paDaq | jIH

    Adj -> tlhIngan
    Noun -> tlhIngan | Hol | puq | paDaq | jIH
    Verb -> Dajatlha | vIlegh | jIHtaH | maH
    """
)
```

Tale grammatica prova a descrivere ciò che sono riuscito a capire del funzionamento della lingua klingon.

In particolare ci sono parole come "tlhIngan" (pronuncia Klingon) che sono sia aggettivi che sostantivi.

La forma di una frase é Oggetto Verbo Soggetto.