Laurea Specialistica in Informatica
a.a. 2006-2007

Interazione Uomo-Macchina II:

Interfacce Intelligenti

Nicole Novielli e Fiorella de Rosis

novielli_at_di.uniba.it

Prerequisiti

Per comprendere questa esercitazione è necessario fare riferimento alle slide che potete trovare al link
NLU Parte Prima

Esercitazione 2a

Costruzione di Markov Chain e Grammatiche probabilistiche per l'analisi e la comprensione della mossa dell'utente

Un esempio: riconoscimento di mosse di Self Introduction

Piacere mi chiamo Nicole
Ciao Valentina, io sono Nicole
Il mio nome è Nicole
…

Si nota facilmente come le mosse elencate abbiano una struttura più o meno costante

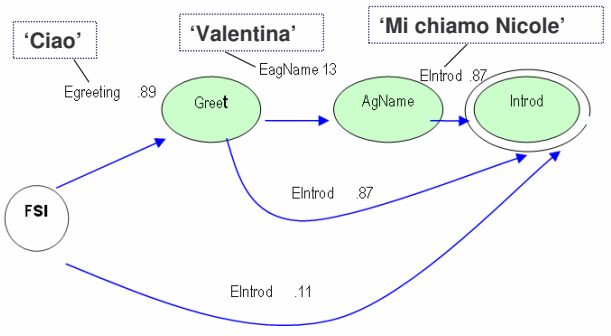
Una formula di presentazione sempre presente…

…Spesso preceduta da una forma di saluto, utilizzata dagli utenti più amichevoli…

.. Che a volte si rivolgono all'agente chiamandolo per nome, nel salutarlo

Proviamo ad ipotizzare una Markov Chain molto semplice con la grammatica corrispondente…

Una Markov Chain per la Self Introduction



Molto semplice poichè utilizza *Categorie Lessicali* come simboli terminali

Egreeting = {Ciao, piacere…}
espressioni di saluto

EagName = {Valentina, cara…}
appellativi per l'agente

EIntrod = {mi chiamo, mio nome è…}
formule di presentazione

FSI→ EGreeting Greet [.89]
FSI→ EIntrod Introd [.11]
Greet→ EAgName AgName [.13]
Greet→ EIntrod Introd [.87]
AgName→ EIntrod Introd [1]

EGreeting → *ciao | piacere | …*
EIntrod→ *mi chiamo | mio nome è | …*
EagName → *Valentina | cara| …*

Parser Probabilistico in Java[1]

Parser probabilistico open source che permette di definire grammatiche probabilistiche

Riceve in input una stringa in linguaggio naturale e fornisce in output la rappresentazione grafica di tutti i possibili alberi di parsing della stessa

Consente di definire grammatiche in CNF (Chomsky Normal Form)

Quindi regole del tipo

S => NP VP
NP => John

Ma consente di ampliarle con l'uso di *Null Rules* per agevolare la definizione di grammatiche a partire da automi…

Null rules: S=>

… oltre che di *Unary Rules*, per consentire la definizione del lessico

Unary rules: S => VP

[1] by Bob Carpenter, http://www.colloquial.com/carp/Projects/index.html

Un esempio molto semplice

Il sistema effettua un tentativo di persuasione per cercare di indurre l'utente a mangiare vegetali

S: '*Dovresti mangiare più frutta e verdura, ti aiutano a mantenere la pelle sana e luminosa*'

Definiamo una grammatica che consenta di riconoscere due tra i possibili tipi di reazione dell'utente a questo suggerimento

- *Claim* – U dice '*Io adoro i vegetali*'

- *Reject* – U dice '*Io odio i vegetali*'

Abbiamo bisogno di un set di produzioni che definiscano

a. la struttura della frase (tramite produzioni che coinvolgono l'uso di *simboli non terminali*)

b. il lessico con cui la frase viene espressa (*set di terminali*, il linguaggio usato per comporre la frase)

Una possibile grammatica

Il sistema effettua un tentativo di persuasione per cercare di indurre l'utente a mangiare vegetali

S: '*Dovresti mangiare più frutta e verdura, ti aiutano a mantenere la pelle sana e luminosa*'

```
Rules

  Nullary Rules:
  Unary Rules:
    s -> Claim : -0.5
    s -> Reject : -0.5
  Binary Rules:
    Like -> I Like : -1.0
    Hate -> I Hate : -1.0
    Claim -> Like Veg : -0.5
    Claim -> Hate Veg : -0.5
    Veg -> The Vegetables : -1.0
    ShouldDo -> Not Can : -1.0
    EatVeg -> Eat Vegetables : -1.0
    Reject -> ShouldDo EatVeg : -1.0
```

```
Lexicon

    Vegetables -> VEGETALI: -1.0
    The -> I: -1.0
    Hate -> ODIO : -1.0
    I -> IO: -1.0
    Eat -> MANGIARE   : -1.0
    Can ->POSSO : -1.0
    Not ->  NON: -1.0
    Like -> ADORO: -1.0
```

Attenzione! Il parser Java prevede che la sintassi delle `Lexicon` rules sia 'invertita'

ES.: `VEGETALI -> Vegetables: -1.0`

Come avviene il riconoscimento

Nel momento in cui la grammatica viene compilata, il parser crea una *Symbol Table* dei non terminali

```
Symbol Table

    0: I
    1: Like
    2: Hate
    3: The
    4: Vegetables
    5: Can
    6: Not
    7: Eat
    8: s
    9: Claim
    10: Reject
    11: Veg
    12: ShouldDo
    13: EatVeg
```

L'output del parser è grafico (rappresentazione grafica di tutti i possibili alberi di parsing)

Guardando nel codice e nella documentazione si nota che c'è una rappresentazione in forma di stringa (Standard Penn Treebank II notation **)**

Pertanto, all' input:

'Adoro i vegetali'

corrisponde quindi la stringa di parsing

```
8 (9 (1 ADORO) (11 (3 I) (4 VEGETALI))))
```

cui è associato un valore di probabilità pari al prodotto delle probabilità delle produzioni utilizzate per ottenerla

Mapping in linguaggio logico

Una volta ottenuta la stringa di parsing in output si può facilmente farne un mapping in linguaggio logico, sfruttando la Symbol Table

```
Symbol Table

    0: I
    1: Like
    2: Hate
    3: The
    4: Vegetables
    5: Can
    6: Not
    7: Eat
    8: s
    9: Claim
    10: Reject
    11: Veg
    12: ShouldDo
    13: EatVeg
```

```
8 (9 (1 ADORO) (11 (3 I) (4 VEGETALI))))
```

Azione	Risultato
L'8 corrisponde al *seme* della grammatica, indica che il riconoscimento è andato a buon fine. Non è utile ai fini del mapping	---
Il **9** si traduce in 'Claim' il cui argomento è sempre U perché si sta analizzando una move utente	Claim U …
1 = Like, denota il predicato oggetto del Claim.	Claim U, Like(U,
11 = Veg, denota l'argomento del Like	**Claim U, Like (U, Veg)**

Attenzione: l'implementazione della procedura di mapping è strettamente correlata al modo in cui avete definito la grammatica e quindi alla struttura ed ai livelli che appariranno nel parsing tree

Due possibilità di lavoro (a)

Arricchire la grammatica precedente in modo che sia possibile riconoscere e distinguere frasi che contengono elementi 'affettivi' nel linguaggio

Alcuni esempi
***'Mi piacciono i vegetali'* è diverso da**
Mi piacciono i vegetali!!
Adoro i vegetali
Mi faccio un bel piatto di meravigliosa insalata
***'Non mi piacciono/non amo i vegetali'* è diverso da**
Detesto i vegetali
Trovo immangiabile la verdura cotta
…

Il modulo implementato deve essere in grado di

a. **riconoscere** questo tipo di frasi (lessico più ampio)
b. **formalizzare** la semantica della frase in linguaggio logico
   Claim U Like(U, Veg)
c. **distinguere** una frase 'neutral' (*mi piacciono i vegetali*) da una con contenuto 'affettivo' ('*la frutta fresca è la mia passione', 'adoro i vegetall')*
   affective_Claim U Like(U, Veg)

Due possibilità di lavoro (b)…

Definire una grammatica che consenta di riconoscere gli atti comunicativi 'Inform', 'AskInfo' e 'Object'

Alcuni esempi di Inform e relativa formalizzazione in linguaggio logico

la frutta la mangio prima dei pasti	Inform U EatBeforeMeal(Fruit)
la frutta la mangio poco	Inform U not EatMuch(Fruit)
la frutta non sempre la mangio	Inform U not EatFrequently(Fruit)
la frutta non la mangio	Inform U not Eat(Fruit)
la frutta mi piace	Inform U not Like(Fruit)
la frutta mi piace poco	Inform U not LikeMuch(Fruit)
mi piace la frutta un sacco	Inform U LikeMuch(Fruit)
mangio la frutta con la buccia	Inform U Eat(EntireFruit)
mangio la frutta con la polpa gialla	Inform U Eat(YellowFruit)
mangio la frutta con la forchetta	Inform U EatWithFork(Fruit)

Il modulo implementato deve essere in grado

a. di riconoscere i tre atti comunicativi questo tipo di frasi (lessico più ampio)
b. formalizzare la semantica della frase in linguaggio logico, come illustrato negli esempi

…Due possibilità di lavoro (b) - esempi

Si, però non è molto buona. Preferirei altro. (**OBJECT**)

mmm… mi consigli di mangiare dell'insalata ogni giorno? (**ASKINFO**)

ehm ma a me non piace la frutta fresca: (**OBJECT**) come posso sostituirla? (**ASKINFO**)

eh, io non li mangio. (**OBJECT**) Cos'altro mi puoi consigliare? (**ASKINFO**)

va beh, ma mi sembra molto difficile seguire una dieta del genere con verdura cotta o cruda, (**OBJECT**) quindi non penso di riuscire ad allinearmi a questo consiglio (**INFORM**)

Spunto per una possibile grammatica

S -> AGR
S-> AGR OBJ
S-> THINK OBJ | THINK OBJ QUESTALT
AGR -> si | si, daccordo | ..
THINK -> eh| va beh | ehm | ...
OBJ -> CONTR PERS
CONTR -> ma | però | ...
PERS -> non è molto buona | io non li mangio | mi sembra molto difficile seguire una dieta del genere | a me non piace | ...
QUESTALT -> Cos'altro mi puoi consigliare? | come posso sostituirla?
S -> THINK ASKSUGG
ASKSUGG -> mi consigli di ?| ...

Riassumendo

Problema

Natural Language Understanding: vogliamo riconoscere e formalizzare (in linguaggio logico) la semantica di una move utente espressa in linguaggio naturale

In questa lezione abbiamo visto come è possibile definire un metodo di riconoscimento basato su grammatiche probablistiche context free (PCFG)

Esercitazione (2a)

'Riconoscimento di atti comunicativi tramite parsing probabilistico di move utente in linguaggio naturale'

Due possibilità:

- Arricchire la grammatica probabilistica per il Claim e il Reject definita a lezione in modo che sia possibile riconoscere e distinguere frasi che contengono elementi 'affettivi' nel linguaggio
- Definire una grammatica probablistica che consenta di riconoscere e formalizzare gli atti comunicativi 'Inform', 'AskInfo' e 'Object'

L'esercitazione prevede, oltre alla definizione delle grammatiche, lo sviluppo di un plugout per il parser Java fornito a lezione, che effettui la mappatura in linguaggio logico della semantica della mossa utente fornita in input, in accordo con il task che tra i due si intende svolgere