Soluzione dell'Esercitazione 1 del Corso di Elaborazione Linguaggio Naturale (seconda versione)

Luca Baronti*

25 marzo 2011

1 Espressioni Regolari

Le varie classi di stringhe riconosciute sono le seguenti:

[a-zA-Z]+ Tutte le parole all'interno di un testo separate da spazi. Escludendo i segni di punteggiatura, le lettere accentate e, in generale, qualsiasi simbolo che non sia le lettere upper-lower case

[A-Z][a-z]* Tutte le parole che cominciano con una lettera maiuscola, in quanto è necessario che sia presente innanzitutto un carattere nel range [A-Z] e successivamente zero o più caratteri in lowercase. Nel caso in cui ci sia una parola conntenente due lettere uppercase, questa espressione effettuerà un doppio matching suddividendo la parola in due che hanno come iniziale il carattere uppercase

\d+(\.\d+)? Tutte le cifre intere o decimali (la cui parte decimale è separata da un punto)

([bcdfghjklmnpqrstvwxyz][aeiou][bcdfghjklmnpqrstvwxyz])* Tutte le triple costituite da una vocale in mezzo a due consonanti (tutto in lowercase)

\w+|[^\w\s]+ Tutte le parole (esclusi i segni di punteggiatura e gli spazi) oppure uno o più segni di punteggiatura.

Le espressioni regolari per far corrispondere le seguenti classi di stringhe sono:

1. Un singolo articolo determinato ("a", "an", "the"):

   \b(a[n]?|the)\b

2. Un'espressione aritmetica che usa interi, somme e moltiplicazioni (2*3+8): Nel caso non siano previsti spazi tra le cifre e gli operandi usiamo

*lbaronti@gmail.com

$$[\backslash d+|\backslash *|\backslash +]+$$

Se invece vogliamo includere *anche* gli spazi nel riconoscimento possiamo usare

$$\backslash d+[\backslash +| |\backslash *|\backslash d+]*$$

2 T9

Un esempio di codice che prende in ingresso un corpus ed una sequenza di caratteri digitati da una tastiera, e restituisce le **5** parole più probabili è il seguente

```
def T9(corpusPath,seq):
  # definisco la tastiera
  tastiera={
    '1':('␣'),
    '2':('a','b','c'),
    '3':('d','e','f'),
    '4':('g','h','i'),
    '5':('j','k','l'),
    '6':('m','n','o'),
    '7':('p','q','r','s'),
    '8':('t','u','v'),
    '9':('w','x','y','z')}
  # recupero le informazioni del corpus
  d={}

  txtfile = open(corpusPath, "r")

  for line in txtfile:
    line=line.strip()
    if line in d:
      d[line]=d[line]+1
    else:
      d[line]=1.0
  txtfile.close()

  size=len(d)
  # elimino le parole di lunghezza sbagliata
  d=dict([(k,v) for k,v in d.items() if len(k)==len(seq)])
  # scansiono la sequenza di numeri eseguendo i controlli con il dizionario
  i=0
  for c in seq:
    if c=='1':
      break

    el=tastiera[c]
    elcandidate=[]

    d=dict([(k,v) for k,v in d.items() if k[i] in el])

    i=i+1

  # stampo il dizionario risultante
  import operator
  i=0
  for e in sorted(d.items(), key=operator.itemgetter(1), reverse=True):
    print e[0], e[1]/size
    i=i+1
    if i==5:
      break
```

Naturalmente è possibile creare una versione equivalente che utilizza le espressioni regolari. Il seguente esempio costruisce una lista di pattern ordinata, ed esegue i match sui termini del dizionario, lettera per lettera.

```
def T9(corpusPath,seq):
  import re
  # creo la lista di pattern corrispondenti alla tastiera
  tastiera=[
   re.compile('\s'),
   re.compile('[a-c]'),
   re.compile('[d-f]'),
   re.compile('[g-i]'),
   re.compile('[j-l]'),
   re.compile('[m-o]'),
   re.compile('[p-s]'),
   re.compile('[t-v]'),
   re.compile('[w-z]')]

  # recupero le informazioni del corpus
  d={}

  txtfile = open(corpusPath, "r")

  for line in txtfile:
    line=line.strip()
    if line in d:
      d[line]=d[line]+1
    else:
      d[line]=1.0
  txtfile.close()

  size=len(d)
  i=0
  # elimino le parole di lunghezza sbagliata
  d=dict([(k,v) for k,v in d.items() if len(k)==len(seq)])

  # seleziono il pattern corrispondente, ed eseguo il pruning sul dizionario
  for num in seq:
    currentPattern=tastiera[int(num)-1]
    # pruning
    d=dict([(k,v) for k,v in d.items() if currentPattern.match(k[i])])
    i+=1

  # stampo il dizionario risultante
  import operator
  i=0
  for e in sorted(d.items(), key=operator.itemgetter(1), reverse=True):
    print e[0], e[1]/size
    i+=1
    if i==5:
      break
```

Per entrambe le versioni, la ricerca della parola inglese "*home*", codificata in "4663" da luogo al seguente risultato

```
>>> from t9 import *
>>> T9("NPS_chat.txt","4663")
good 0.0222606975019
home 0.00173138758348
gone 0.00148404650012
goof 0.000247341083354
```

dal quale si evince che la parola "*good*", codificata allo stesso modo, è più frequente (di un ordine di grandezza) rispetto a quella voluta, all'interno del corpus scelto.

3 Legge di Zipf

Come testo di riferimento si è usata la fonte di wikipedia su un argomento molto comune (l'acqua), dalla quale sono stati rimosse eventuali parole *outlier* come i

riferimenti o i pulsanti di modifica.
Si è utilizzata la seguente funzione per calcolarci le frequenze:

```
def p4(text):
  txtfile = open(text, "r")

  d={}
  # estrapolo le parole con la loro frequenza
  for line in txtfile.readlines():
    line=line.strip()
    for word in line.split():
      if word in d:
        d[word]=d[word]+1
      else:
        d[word]=1.0
  size=len(d)
  # normalizzo le frequenze
  d=dict([(k,v/size) for k,v in d.items()])

  import operator
  ordered=sorted(d.items(), key=operator.itemgetter(1), reverse=True)
  i=0
  outfile = open("acquaFreq.dat", "w")
  outfile2=open("costante.dat","w")
  outfile.write("id parola frequenza\n")
  outfile2.write("id parola k\n")
  for e in ordered:
    i=i+1
    outfile.write(str(i)+" "+str(e[0])+" "+str(e[1])+"\n")
    outfile2.write(str(i)+" "+str(e[0])+" "+str(e[1]*i)+"\n")
    if i==500: break

  outfile.close()
  outfile2.close()
```

Come era facile aspettarsi le parole più comuni sono state gli articoli e le preposizioni:

id	Parola	Frequenza (normalizzata)
1	di	0.126698262243
2	e	0.0581358609795
3	la	0.0429699842022
4	in	0.0382306477093
5	è	0.0379146919431
6	che	0.0366508688784
7	il	0.0309636650869
8	a	0.0296998420221
9	un	0.0284360189573
10	della	0.0249605055292
11	le	0.0243285939968
12	per	0.0233807266983

Una delle parole più significative del corpus analizzato, ovvero “acqua” si posiziona invece 22-esima, con una frequenza pari a ≈ 0.013902.

Dall’andamento generale delle 500 parole più frequenti, osservabile in Fig.1, emerge come il valore della frequenza delle parole tenda a diminuire molto nelle prime posizioni, mentre per quanto riguarda l’eventuale riscontro della legge di Zipf, come si evince in Fig.2 le parole non sembrano seguire un andamento perfettamente costante rispetto al loro rank, tuttavia è possibile notare che il

loro prodotto oscilla attorno a 0.25 e a 0.5, e questo intervallo potrebbe rientrare entro i parametri di incertezza previsti dalla suddetta legge.

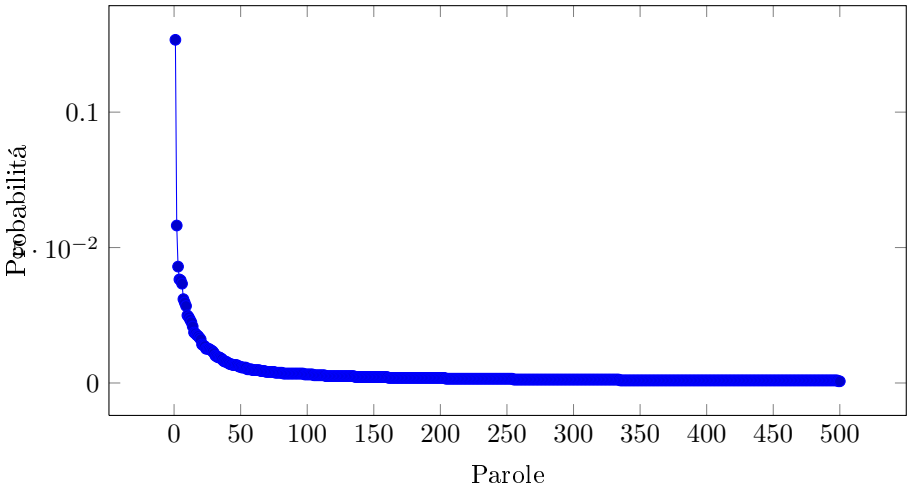

Figura 1: Frequenza delle parole in relazione al loro rank

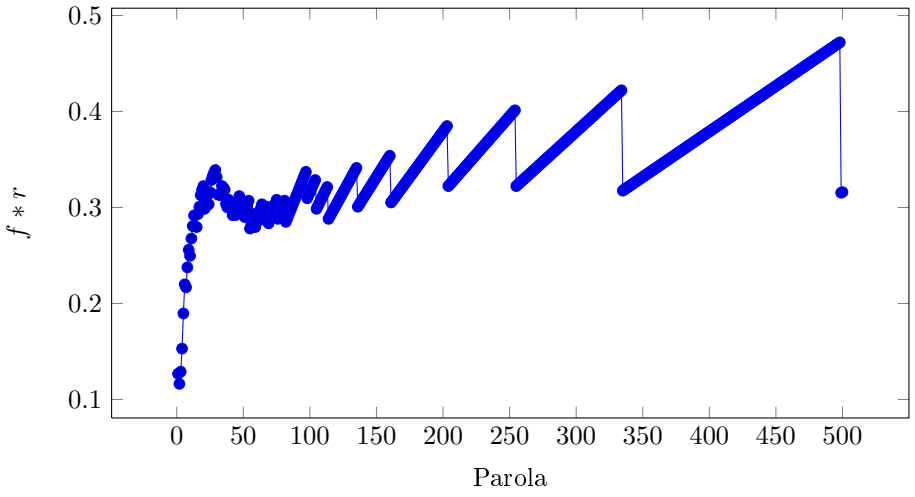

Figura 2: Prodotto delle frequenze delle parole con il loro rank