Il dizionario colpisce ancora

Seconda puntata del discorso sulla "difficoltà" di costruire un Dizionario. Questa volta si tratta della costruzione del modello matematico. La sperimentazione si muove in tre direzioni, costruzione del modello; confronto delle previsioni con una simulazione di laboratorio; confronto delle previsioni con gli esperimenti sull'italiano. Come vedremo riusciremo ad ottenere un modello che non solo fornisce un fitting abbastanza accurato (per questo bastava fare un'approssimazione ai minimi quadrati di II grado) ma che prevede anche il comportamento terminale tipico di un dizionario finito (nella realtà quando si sono trovate tutte le parole, per quanto si cerchi non si trova più niente di nuovo). La trattazione di un argomento così complesso è come al solito effettuata in modo "sportivo" (mi scuso tutti i mesi ma ne sento proprio il bisogno). In ogni caso data la delicatezza degli argomenti trattati se ne consiglia la lettura ad un pubblico maturo

di Francesco Romani

Il nuovo campione

Il campione usato questo mese consiste in tutti i testi del mese scorso più altri cinque, anch'essi trovati su Liber Liber:
"La città del sole", Tommaso Campanella
"Pinocchio", Carlo Collodi
"L'esclusa", Luigi Pirandello
"Il turno", Luigi Pirandello
"Quando vendettero il Natale", Enrico Maria Ferrari.
Questa volta oltre alla lista che lega il numero di parole trovate in funzione delle parole cercate vogliamo calcolare anche la distribuzione di frequenza delle varie parole. Lavorando su file già tradotti nel formato Mac e ridotti a solo lettere minuscole il programma più efficiente che mi è riuscito di mettere insieme è il seguente.

In[1]:=

```
leggi1[s_String]:=Module[{ll},
   testo=ReadList[s,Word,
   WordSeparators-> separatori];
   Scan[(Conta[#]=Conta[#]+1)&,
        testo];
   fps=Drop[Transpose[
   Map[List@@#&,
    DownValues[Conta]]][[2]],-1];
   AppendTo[pl,
       {nn=Plus@@fps,ll=Length[fps]}];
   fps=Reverse[Sort[fps]]/nn;
   b=Fit[Map[N[Log[#]]&,fps],
       {1,Log[x]},x][[2,1]];
   Print[s," ",nn," ",ll," ",b]]

leggi[l_List]:=(
      Clear[Conta,testo];
      pl={};
      Conta[_]:=0;
      Scan[leggi1,l]);
```

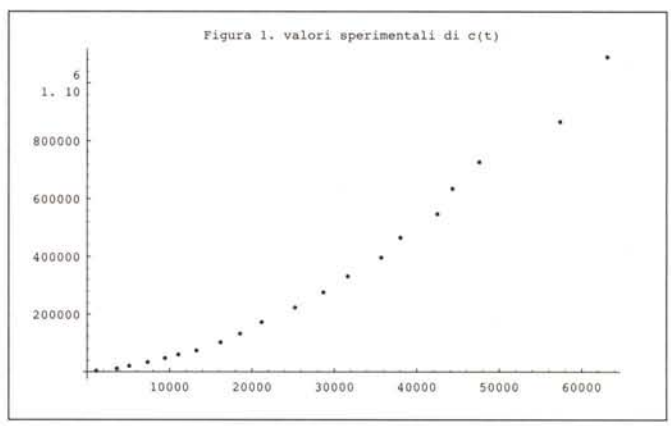
La lista **separatori** contiene il blank, i segni di interpunzione e tutti i caratteri strani presenti nei file (isolati in una prima passata). Ogni file viene letto e ogni parola usata come argomento per l'assegamento **Conta[#]= Conta[#]+1**. La funzione **DownValues** contiene la definizione di **Conta** sotto forma di lista di regole, applicandovi **List** si ottiene una matrice la cui seconda colonna è il vettore delle frequenze, la somma del vettore delle frequenze è il numero delle occorrenze delle parole cercate mentre la sua lunghezza è il numero delle parole trovate. La lista delle frequenze viene restituita in ordine decrescente nella variabile **fps**, la lista **pl** contiene invece il campionamento della funzione **c(t)** introdotta nella puntata precedente. Per ogni file la lista delle frequenze viene approssimata ai minimi quadrati con la funzione a x^{-b} e viene stampato il valore di **b.** Si nota che l'esponente cresce al crescere del campione.

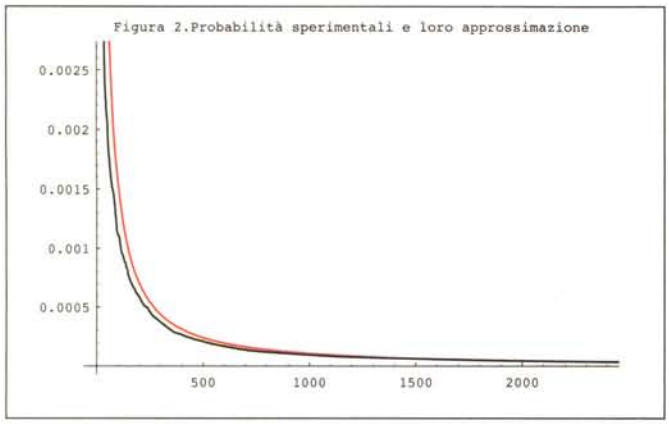
In[1]:=

```
leggi[FileNames[]]
```

Out[2]=

```
LAGIARA     3159   1108   - 0.7800
QUANDO-V   11018   3618   - 0.7415
COSTITUZ   20263   5116   - 0.8386
GIORNIDI   33494   7350   - 0.8824
CITTADEL   47325   9456   - 0.8969
SEIRA      60134  11078   - 0.9220
```






```
LETTORE     73912 13275  - 0.9200
IL_TURNO   101956 16197  - 0.9570
GIORNATA   132322 18565  - 0.9908
PINOCCHIO  172890 21191  - 1.0301
C4         223570 25245  - 1.0465
C3         276760 28672  - 1.0653
ESCLUSA    332589 31627  - 1.0894
C2         397083 35682  - 1.1010
SENILITA   464664 38010  - 1.1274
C1         547274 42497  - 1.1388
MALAV      635448 44319  - 1.1694
C5         727307 47568  - 1.1858
C6         866216 57378  - 1.1689
PROMESSI  1090175 63088  - 1.2039
```

In un campione di circa 6.5 Megabyte vengono trattate 1090175 parole e ne vengono trovate 63088 distinte. La lista **pl** è plottata in figura 1.

In[1]:=

```
listplot=ListPlot[pl, PlotRange->All,
  PlotLabel->"Figura 1."];];
```

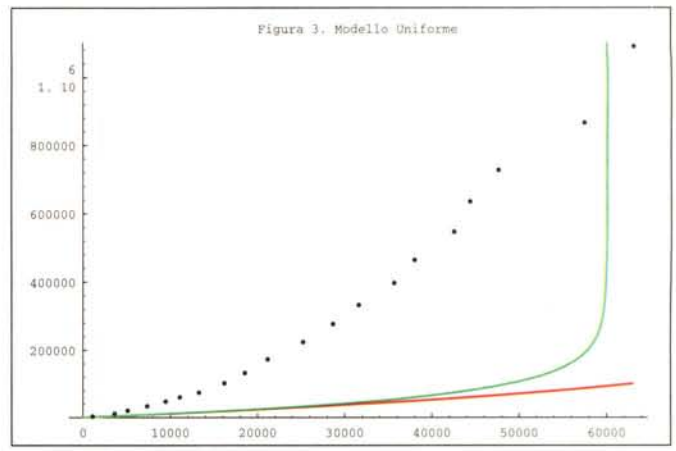
Stampare il grafico della lista delle frequenze non è una cosa banale perché la presenza di un gran numero di punti genera un file molto grosso, con un po' di trucchi si ottiene il grafico di figura 2, in cui la curva nera rappresenta la curva delle frequenze misurata mentre la curva rossa presenta la curva di approssimazione del tipo a x^{-b}.

Di nuovo alla ricerca del modello

Sia **u** il numero di tutte le parole della lingua in esame. Rappresentiamole con i numeri da 1 a u in ordine di probabilità decrescente e sia p(t) la probabilità della parola t-esima. Per semplificarci la vita consideriamo p(t) una funzione continua. Supponendoche le parole escano in ordine di probabilità decrescente se ho trovato **z** parole dopo averne scandite **c** la probabilità di trovarne di nuove è

$$q(z) = 1 - \int_1^{z+1} p(x)dx$$

e si può scrivere allora l'equazione differenziale dt/dc=q(t). Invertendo entrambi i termini si ottiene dc/dt = 1/q(t), da cui si vede che la soluzione cercata è l'integrale di 1/q(t), utilizzando la condizione iniziale c(0)=0 si deduce che la soluzione è



$$c(t) = \int_0^t \frac{dx}{q(x)}$$

Purtroppo in generale non è vero che le parole vengono trovate in ordine di probabilità decrescente e quello che si ottiene è solo una limitazione superiore al valore di c(t).
Adesso proviamo alcuni tipi di distribuzioni e confrontiamo i risultati previsti con quelli sperimentali di figura 1.

Distribuzione uniforme

Se le **u** parole hanno tutte la stessa probabilità p(t) =1/u allora la funzione c(t) si trova facilmente

In[1]:=

```
q=Together[1-Integrate[1/u,{x,1,z+1}]]
```

Out[1]=

```
u - z
-----
  u
```

In[2]:=

```
c=Integrate[1/q,{z,0,t}];
```

Out[2]=

```
-(u*Log[t - u]) + u*Log[-u]
```

Il risultato si legge meglio nella forma

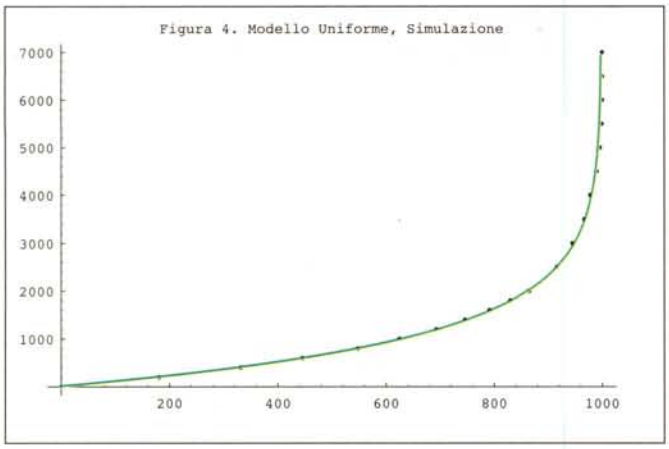
```
          u
u Log[-------]
      u - t
```

Confrontando la funzione trovata con quella sperimentale si vede che il risultato ottenuto è troppo ottimistico. La stima con u=100000 (in rosso) è troppo bassa e quella con u=60000 (in verde) resta ancora bassa prima di andare all'infinito troppo presto. Le parole reali sono ben lungi da essere equiprobabili e trovare quelle rare è molto più difficile *(Figura 3).*

In[3]:=

```
Plot[Evaluate[
 {c/.u->100000,c/.u->60000}],
 {t,0,65000},PlotStyle->{Red,Green}];
Show[%,listplot,PlotRange->All,
  PlotLabel->"Figura 3."];];
```

Per vedere se comunque la soluzione del modello è verosimile effettuiamo una simulazione nel caso u=1000. Estraiamo numeri tra 1 e 1000 con probabilità uniforme e studiamo l'andamento di c(t) confrontandolo con quello teorico. La funzione **sim[{{a1,b1},{a2,b2},...}]** estrae dapprima **a1**



gruppi di **a2** parole poi **b1** gruppi di **b2** parole ecc.

In[4]:=

```
sim[v_List]:=Module[{i,j,pc,pt,l},
  all={};
  pl={};
  pc=0;
  pt=0;
  Do[inc=v[[j,2]];
     Do[l=Union[Table[rand,{inc}]];
        pc=pc+inc;
        all=Union[all,l];
        pt=Length[all];
        AppendTo[pl, {pt,pc}],
     {j,Length[v]}]];

rand:=Random[Integer,{1,1000}];
```

Generiamo 7000 parole e confrontiamo il risultato teorico in rosso con quello sperimentale. Come si vede in figura 4 l'accordo è perfetto. In questa sperimentazione si produce anche l'asintoto verticale, fenomeno che nella creazione di un vero formario si trova quando si sono trovate tutte le forme attestate della lingua in esame.

In[5]:=

```
sim[{{10,200},{10,500}}]
ListPlot[pl, PlotRange->All];
Plot[Evaluate[c/.u->1000],
  {t,0,999},PlotStyle->{Green}];
```

Distribuzione x^{-b}

Per approssimare meglio la distribuzione sperimentale prendiamo in considerazione una distribuzione decrescente del tipo x^{-b}. Il fitting effettuato mano a mano che si leggono i file ha mostrato che l'esponente stimato b passava gradualmente da valori minori di 1 a valori maggiori, fino al valore 1.20 dopo un milione di parole. La funzione $(b-1)x^{-b}$ integrata tra 1 e ∞ da come risultato 1 e quindi $(b-1)x^{-b}$ è una distribuzione di probabilità, usiamola nel nostro modello

In[1]:=

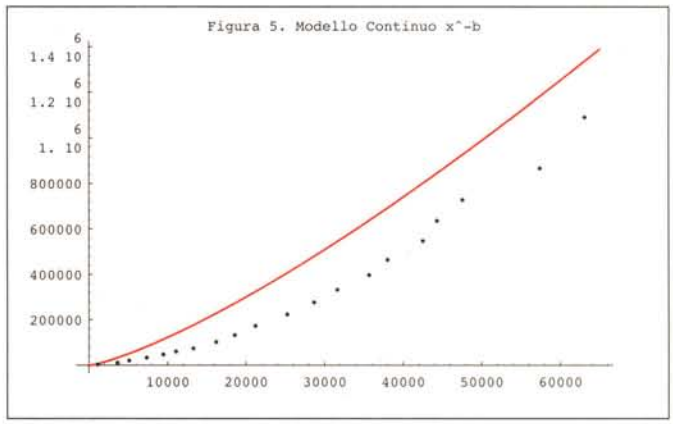
```
q=1-Integrate[(b-1)x^-b,{x,1,t}]
```

Out[1]=

t^{1-b}

In[2]:=

```
c=Integrate[1/q,t]
```



Out[2]=

$$\frac{t^b}{b}$$

E confrontiamo i risultati con la dura realtà.

In[3]:=

```
Plot[Evaluate[c/.b->1.3],
  {t,0,65000},PlotStyle->{Red}];
Show[%,listplot,PlotRange->All,
 PlotLabel->"Figura 5."];
```

Si ottiene una buona approssimazione dei dati sperimentali *(Figura 5)* ma resta il problema che questo modello è continuo e prevede un numero infinito di parole; viene quindi a mancare l'asintoto finale e si approssima male la presenza di insiemi molto grandi di probabilità pressoché uguale.

Distribuzione uniforme a tratti

L'ultimo modello che proviamo è quello di una distribuzione discreta uniforme a tratti. Consideriamo per esempio il seguente caso

In[1]:=

```
pairs={{100,1},{2000,1/10},
       {7900,1/100}};
tpairs=Transpose[pairs];
sum=N[Dot@@tpairs]
```

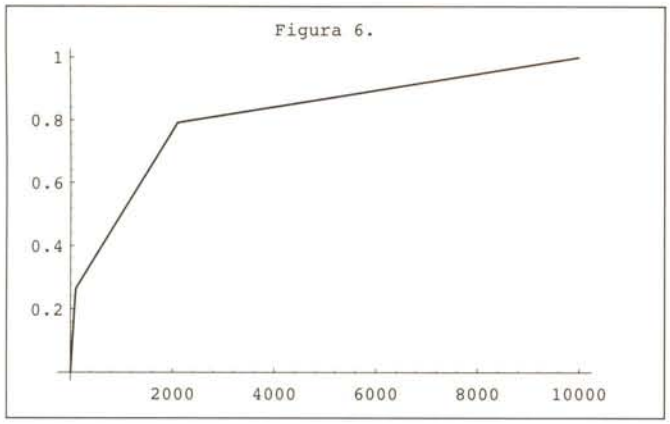
Out[1]=

379

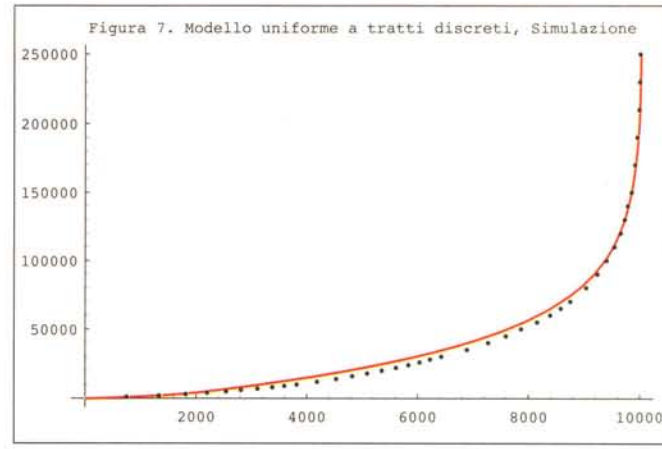
Abbiamo in tutto 10000 di cui 100 con probabilità 1/379, 2000 con probabilità 1/3790, 7900 con probabilità 1/37900. La funzione **FoldList** permette di accumulare le probabilità dei vari gruppi, e con un po' di trucchi si riesce a costruire l'integrale della funzione di probabilità che è ovviamente una funzione continua a tratti. Quello che segue è un esempio (abbastanza involuto) di come si può usare *Mathematica* per scrivere un programma in *Mathematica*. **PP[x]** usa il costrutto **Which** (equivalente al COND del LISP) che calcola una particolare funzione lineare a seconda del valore di **x**.

Per inciso si noti che la forma della funzione **PP** è la stessa di quella che associa le tasse da pagare al reddito in un sistema di aliquote (sigh!).

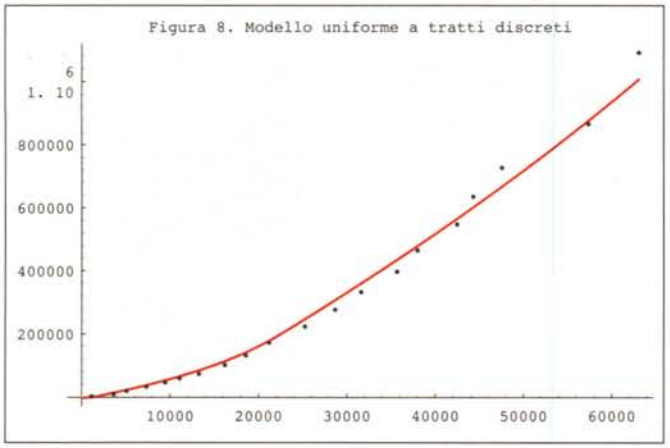
In[2]:=




```
l1=FoldList[Plus,0,tpairs[[1]]]
```
Out[2]=
```
{0, 100, 2100, 10000}
```
In[3]:=
```
l12=FoldList[Plus,0,Times@@tpairs]
```
Out[3]=
```
{0, 100, 300, 379}
```
In[4]:=
```
l2=Drop[l12,-1]
```
Out[4]=
```
{0, 100, 300}
```
In[5]:=
```
PP[x_]:=Evaluate[
 Which[Evaluate[Sequence@@
  Flatten[Transpose[
   {Map[(x<#)&,Rest[l1]],
     MapThread[(#2+(x-#1)#3)&,
             {Drop[l1,-1],
              l2/sum,
              tpairs[[2]]/sum}]}]]]]]
PP[x]
```
Out[5]=
```
Which[
  x<100,      0.00263852 x,
  x<2100,     0.263852+0.000263852(x-100),
  x<10000,    0.791557+0.0000263852(x-2100)]
```
La funzione **PP** è plottata in figura 6.
L'integrale di **1/(1-PP[x])** è ancora una funzione a tratti, per semplificarci la vita usiamo **NIntegrate** (rinunciamo cioè al valore chiuso dell'integrale e ci accontentiamo di una sua approssimazione numerica). Dapprima integriamo sui vari tratti.
In[6]:=
```
l4=Drop[FoldList[Plus,0,MapThread[
  NIntegrate[1/(1-#3-(x-#1)#4),
             {x,#1,#2}]&,
  {Drop[l1,-1],
   Rest[l1],
   l2/sum,
   tpairs[[2]]/sum}]],-1]
```
Out[6]=
```
{0, 116.096, 4898.18}
```
Quindi costruiamo con il metodo visto sopra la funzione



cc[x] che è l'approssimazione cercata.
In[7]:=
```
cc[x_]:=Evaluate[
 Which[Evaluate[Sequence@@
  Flatten[Transpose[
   {Map[(x<#)&,Rest[l1]],
    MapThread[#5+
      NIntegrate[1/(1-#3-(z-#1)#4),
                 {z,#1,x}]&,
     {Drop[l1,-1],
      Rest[l1],
      l2/sum,
      tpairs[[2]]/sum,14}]}]]]]]
cc[x]
```
Out[7]=
```
Which[
x<100,  NIntegrate[
          1/(1-z 0.002638),{z,0,x}],
x<2100, 116.096 + NIntegrate[
          1/(1-0.26385-(z-100)0.0002638),
            {z, 100, x}],
x<10000, 4898.18+NIntegrate[
          1/(1-0.79155- (z-2100)*0.00002638),
            {z, 2100, x}]]
```
Per vedere se l'approssimazione è buona costruiamo anche un generatore di parole con la probabilità data. Si effettua una doppia estrazione. Un numero casuale tra (0,1) viene usato per scegliere in che intervallo ci troviamo e poi viene estratto un numero con probabilità uniforme nello stesso intervallo.
In[8]:=
```
l20=Rest[l12];
QQ[x_]:=Evaluate[
  Which[Evaluate[Sequence@@
   Flatten[Transpose[
   {Map[(x<#)&,N[l20/sum]],
    Map[r,Range[Length[l20]]]}]]]]]
QQ[x]
```
Out[8]=
```
Which[x < 0.2638,  r[1],
      x < 0.79155, r[2],
      x < 1.,      r[3]]
```
In[9]:=
```
r[i_]:=
  Random[Integer,{l1[[i]]+1,l1[[i+1]]}]
```



rand:=QQ[Random[]]

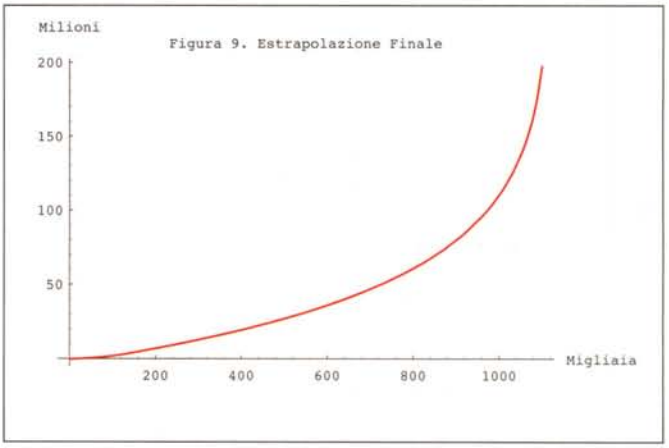
Generiamo 250000 parole e confrontiamo il risultato teorico in rosso con quello sperimentale

In[10]:=

```
sim[{{10,1000},{10,2000},
     {8,5000},{8,10000},{5,20000}}];
lp=ListPlot[pl, PlotRange->All];
Show[plot,lp];
```

Come si vede in figura 7 l'accordo è perfetto, con una leggera sovrastima anch'essa prevista dalla teoria (il nostro modello fornisce una limitazione superiore a c(t)).

Per finire abbiamo rifatto tutti i conti cercando di ritrovare la curva ottenuta per l'italiano. Saltando tutti i passaggi intermedi, partendo dalla distribuzione discreta



In[10]:=

```
pairs={{100,1},{2000,1/50},
       {20000,1/1000},
       {100000,1/15000},
       {1000000,1/300000}}
```

che prevede 1122100 parole si ottiene la approssimazione di figura 8. Estrapolando si vede che per trovare 800.000 forme attestate il nostro modello prevederebbe la ricerca in testi con almeno 50 milioni di parole (figura 9).

MC

Francesco Romani è raggiungibile tramite Internet all'indirizzo romani@di.unipi.it

TRASFORMA IL TUO COMPUTER IN UNA MACCHINA DA SOLDI

una grande inchiesta sul numero di aprile

SPECIALE INFORMATICA

Millionaire

INTRAPRENDERE