Motori di Ricerca

presente e futuro prossimo

Cosa è un crawler ?

Paolo Ferragina, Università di Pisa

Fase di Crawling

- Numerosi problemi di progettazione:
  - Copertura: Quali pagine occorre visitare ?
  - Aggiornamento: Quanto spesso occorre visitarle ?
  - Invadenza: Come minimizzare il carico dei siti visitati ?
  - Efficienza: Come parallelizzare/scalare il “crawling” ?

Paolo Ferragina, Università di Pisa

"Ciclo di vita" di un Crawler

Link Extractor

```
while(<ci sono pagine da esaminare nel repository>){
  <prendi una pagina p>
  <estrai i link contenuti in essa>
  <inserisci i link estratti in una coda, ciascuno
   con una priorità dipendente dalla politica scelta>
 <marca p come pagina da cui abbiamo estratto i link>
}
```

Downloader

```
while(<ci sono link assegnati dal Manager>){
  <estrai i link>
  <scarica le pagine p_i dalla rete>
  <invia le p_i al page repository>
}
```

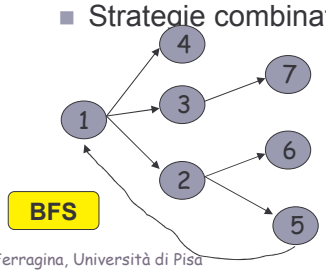
Crawler Manager

```
<estrai un gruppo di link dalla coda in ordine di priorità>
while(<ci sono link nel gruppo>){
   foreach link u {
        if ( (u ∉ "pagine già viste" )
             || ( u ∈ "pagine già viste"  && <sul Web server la pagina è più recente> )
             && ( <u è un link accettato dal robot.txt del sito>) ) {
               <risolvi u rispetto al DNS>
               <invia u alla coda dei downloaders>
        } }
```

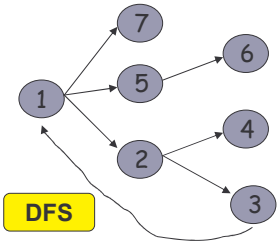

Da quali pagine si parte ?

Come assegnare la priorità di visita!!

- Data una pagina P, definire quanto sia "buona", così da assegnarle alta priorità e visitarla subito dal CM.
- Esistono molte metriche:
  - Guidate dal topic coperto dal motore
  - Guidate dalla popolarità
  - BFS, DFS, Random
  - Strategie combinate

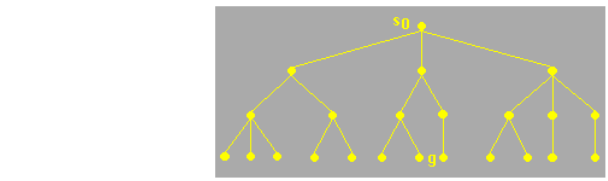


BFS



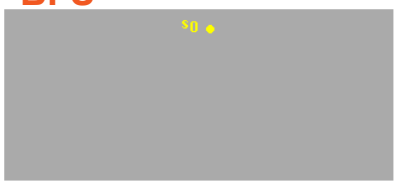
DFS

Raggiungimento di pagine interessanti



Grafo Web

BFS



DFS

Ma cosa accade se l'albero è molto profondo a sinistra ?

Focused Crawling

- Si scelgono selettivamente le pagine sulle quali continuare la visita, in accordo a un insieme di topic rilevanti definiti apriori.
  - I topic sono specificati mediante documenti campione
  - I topic sono specificati mediante indirizzi
  - I topic devono occorrere nei *metatag* delle pagine
- Risparmio di risorse di rete e di hardware.
- Esempi di crawler open-source
  - Nutch, also used by Yahoo
  - Hentrix, used by Archive.org

Motori di Ricerca

presente e futuro prossimo

Cosa è un Analizzatore Lessicale ?

Paolo Ferragina, Università di Pisa

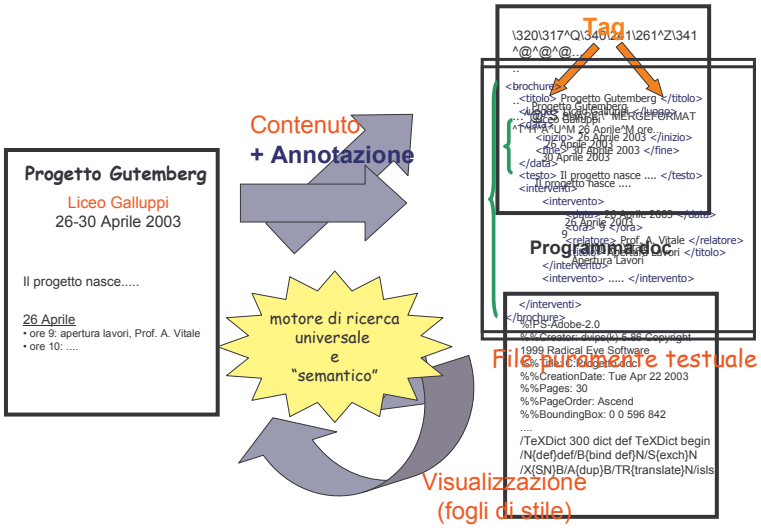
Codifica di un testo

- ASCII (1963)
  - 8 bit per carattere (7+1)
  - Sufficiente per l'inglese ma non italiano e tedesco.
- XML (W3C 1998, o SGML IBM 1986)
  - Distinguiamo struttura – contenuto – visualizzazione
  - Standard aperto e personalizzabile, deve essere *well-formed* o *valido*
  - Indipendente dalla piattaforma, in quanto puramente testuale

```
<email>
        <from> Paolo Ferragina </from>
        <to> Pinco Pallino </to>
        <subject> prova </subject>
        <bodymsg> bla bla... </bodymsg>
</email>
```

Paolo Ferragina, Università di Pisa

XML: un nuovo "e-alfabeto"



Passi principali dell'Analizzatore

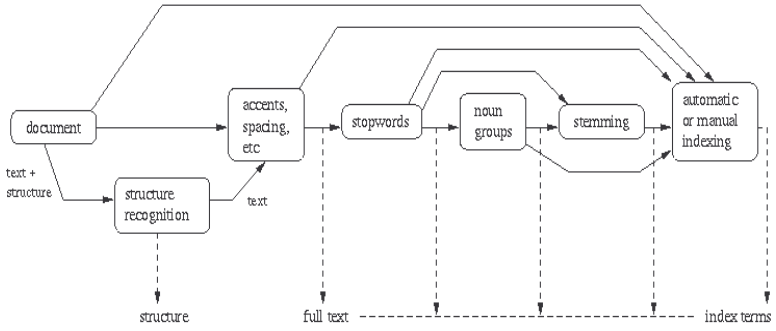


Figure from Baeza-Yates & Ribeiro-Neto

Fase di analisi delle pagine *(eterogenee)*

- Varie difficoltà per la "normalizzazione"
  - State-of-the-art, U.S.A. vs. USA, a.out
  - 3/12/91, Mar. 12, 1991, 55 B.C., B-52, 100.2.86.144
  - Cooper's *vs* Cooper *vs* Coopers
  - résumé vs resume
- Google: kid's toys, kids toys, kid's toy (anche singolare/plurale in italiano)
- *Stemming*: riduce le parole alle loro radici
  - Dipende dal linguaggio (inglese: Porter)
  - Errori: ***automate(s), automatic, automation → automat***

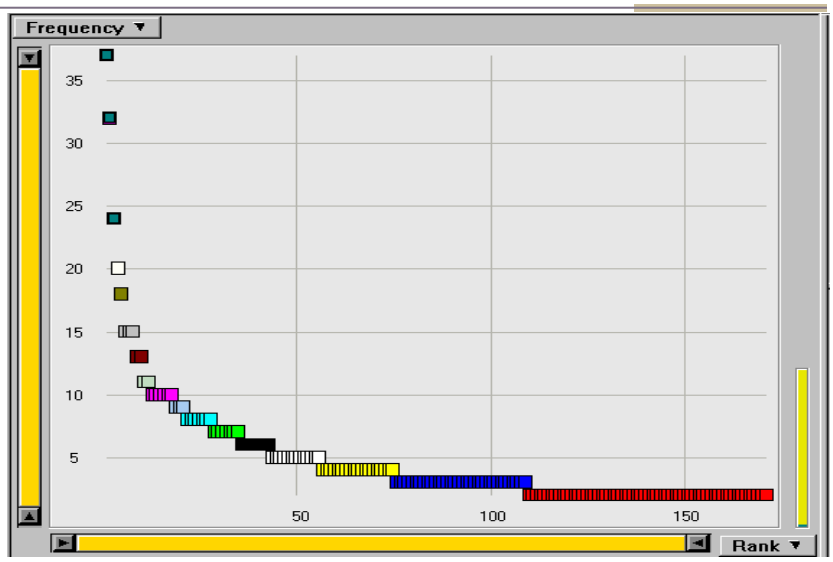
for example compressed and compression are both accepted as equivalent to compress

for exampl compres and compres are both accept as equival to compres

Proprietà statistiche dei testi

- I token **non** sono distribuiti uniformemente nel testo
  - Ma seguono la cosiddetta "legge di Zipf"
    - Pochi elementi sono molto frequenti
    - Un numero medio di essi ha frequenza media
    - Moltissimi sono infrequenti
- Il numero di token *distinti* non cresce linearmente
  - Ma secondo la "legge di Heaps" ($|T|^{\beta}$ con $\beta<1$)
- Le parole interessanti hanno una caratterizzazione
  - Sono quelle mediamente frequenti (Luhn)

Un esempio di “Curva di Zipf”



La legge di Zipf, nel dettaglio

- Il prodotto della frequenza (f) di un token per il suo rango (r) è approssimativamente constante

$$r * f = c\,N$$
$$f = c\,N / r$$

Legge di base

$$f = c\,N / r^{\alpha}$$
$$\alpha = 1.5 \div 2.2$$

Legge generale

- Un modo alternativo di vedere la cosa:
  - Il termine di rango 1 occorre C volte
  - Il secondo termine più frequente occorre C/2 volte
  - Il terzo termine occorre C/3 volte
  - …

Consequenze della Legge di Zipf

- Esistono pochi token molto frequenti che non fungono da buoni discriminatori. Le cosiddette "stop words" in IR
  - Inglese: *to, from, on, and, the, ...*
  - Italiano: *a, per, il, in, un,…*
- Esistono moltissimi token che occorrono una volta sola (o pochissime volte) nel testo e quindi sono poco utili (errore / corretto?).
  - Inglese: Calpurnia
  - Italiano: Precipitevolissimevolmente (o, paklo)

Parole mediamente frequenti ☺ Parole discriminanti

Frequenza vs. Potere discriminante (Luhn)