

TASSONOMIA DEL WEB

Per il web possiamo classificare 3 tipi di query:

- 1) **Navigational Query**: per raggiungere un sito specifico
- 2) **Informational Query**: per avere info su qualsiasi cosa
- 3) **Transactional Query**: per raggiungere un sito da dove vengono gestite nuove interazioni (shopping, download,...)

MODELLO A FARFALLA o A CUSCIDA

Andrei Broder descrisse la distribuzione delle pagine web come una farfalla, perché esistono pagine estremamente connesse col resto e altre più isolate.

PROBLEMI DEL WEB

- 1) DINAMICITÀ
- 2) DIVERSITÀ
- 3) VARIETÀ

PAGE RANK

Questo algoritmo è utile per fare un rank delle pagine web, sulla base della struttura del grafo della rete.

Non c'è una semplice analisi dei link di un grafo, perché c'è meno suscettibile ai link di spam.

Authoritiveness

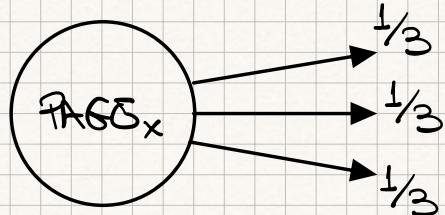
Introduce il concetto di autoritatività di una pagina web. Questa proprietà è dipendente dalla popolarità delle pagine durante una simulazione.

Simulazione

Con simulazione si intende tener traccia di utenti che casualmente attraversano il grafo del web, e che hanno ognuno una certa probabilità di attraversare un link piuttosto che un altro.

Random Walk

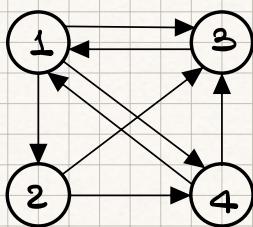
Immaginiamo di avere un utente che casualmente naviga nel grafo del web, partendo da una pagina a caso. A ogni step percorre uno degli out-link di quella pagina. La probabilità di attraversare un out-link è equiprobabile per ogni out-link della pagina:



Durante l'esecuzione della simulazione, con più di un utente, avremo per ogni pagina un **rate di visita**.

Come possiamo rappresentare il grafo del web? Usando la **matrice di adiacenza**:

Esempio:



$$A = \begin{bmatrix} \emptyset & 1 & 1 & 1 \\ \emptyset & \emptyset & 1 & 1 \\ 1 & \emptyset & \emptyset & \emptyset \\ 1 & \emptyset & 1 & \emptyset \end{bmatrix}$$

Il passo successivo è la **matrice delle transizioni** che tiene conto delle probabilità di attraversare un out-link:

$$T = \begin{bmatrix} \emptyset & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \emptyset & \emptyset & \frac{1}{2} & \frac{1}{2} \\ 1 & \emptyset & \emptyset & \emptyset \\ \frac{1}{2} & \emptyset & \frac{1}{2} & \emptyset \end{bmatrix}$$

La cui trasposta T^T è la **Markov Chain M**

$$M = T^T = \begin{bmatrix} \emptyset & \emptyset & 1 & \frac{1}{2} \\ \frac{1}{3} & \emptyset & \emptyset & \emptyset \\ \frac{1}{3} & \frac{1}{2} & \emptyset & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & \emptyset & \emptyset \end{bmatrix}$$

La matrice M ci dice, per ogni entrata M_{ij} , la probabilità di andare allo stato (la pagina) j data la condizione di essere nello stato corrente i .

Chiarimento:

$$\sum_{j=1}^n M_{ij} = 1 \quad \text{fissato } i$$

ERGODIC MARKOV CHAIN

Una catena di Markov è ergodica se è sia:

- 1) **Iriducibile**: tutti gli stati sono raggiungibili da tutti gli altri stati (Anche con più tap)
- 2) **Aperiodica**: Non esistono stati che si verificano con una certa periodicità.

ALGORITMO

Per calcolare il rank è sufficiente risolvere il sistema lineare:

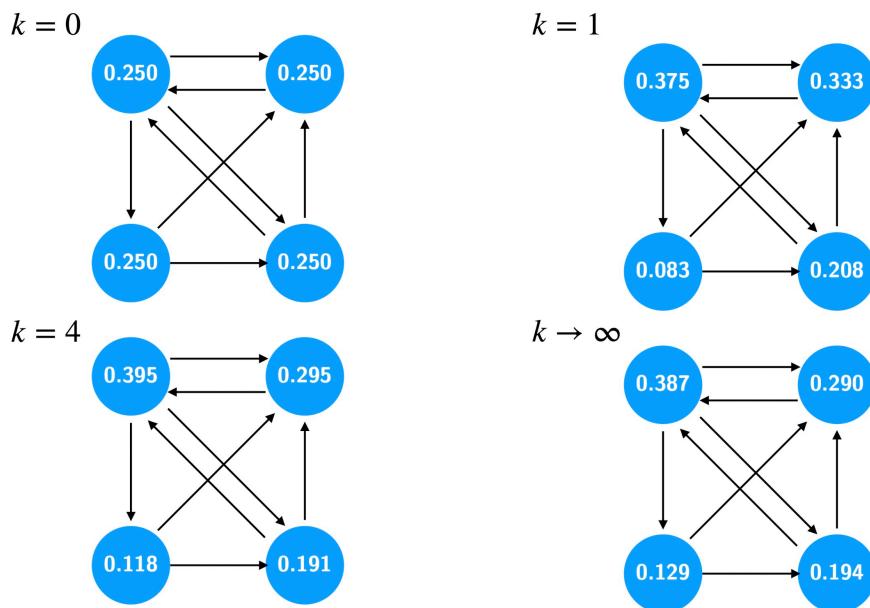
$$Mx = x$$

Se riprendiamo l'esempio di prima:

$$\begin{bmatrix} 0 & 0 & 1 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \Rightarrow \begin{cases} x_1 = x_3 + \frac{1}{2}x_4 \\ x_2 = \frac{1}{3}x_1 \\ x_3 = \frac{1}{3}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_4 \\ x_4 = \frac{1}{3}x_1 + \frac{1}{2}x_2 \end{cases}$$

Dove il vettore $\vec{x} = (x_1, x_2, x_3, x_4)$ darà un valore a ogni nodo del grafo, permettendo di fare un rank. Difatti abbiamo che:

$$\text{PageRank}(i) := x_i$$



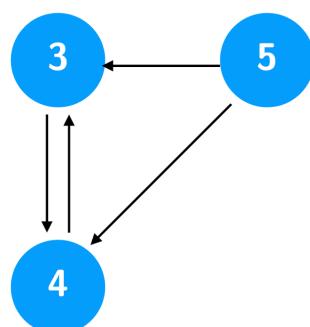
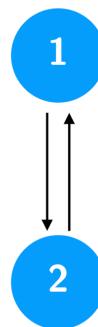
Analizziamo per le varie iterazioni K :

- 1) $K=0$: tutti i nodi hanno equi **authoritativeness** ($\frac{1}{4}$)
- 2) $K>0$: l'authoritativeness comincia a fluire verso i nodi che hanno più in-link. L'analogia è immagine c'authoritativeness come acqua che scorre nella rete e, i nodi che hanno più in-link ne raccolgono di più
- 3) $K \rightarrow \infty$: Ogni nodo arriva a un valore limite, che poi è lo score su cui fare il rank.

Tutto ciò presuppone che la catena di Markov sia ergonica. Ma il web, per sua natura, non è affatto un grafo ergotico. Per poter comunque usare il page rank, viene introdotta la matrice **G**, Google Matrix.

$$G = (1-\lambda)M + \lambda S \text{ con } S \in \mathbb{R}^{n \times n} : s_{ij} = \frac{1}{m}$$

Esempio:



$$M = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1/2 \\ 0 & 0 & 1 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$G = (1 - \lambda) \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1/2 \\ 0 & 0 & 1 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \lambda \begin{bmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{bmatrix}$$

Il parametro λ è detto **parametro di teletrasportazione** perché serve a teletrasportare il random walker in un modo a caso con probabilità λ . Con la probabilità inversa ($1-\lambda$) l'utente seguirà un out-link fra quelli che ha a disposizione.

Così facendo la matrice **G** è sempre ergonomica.

HITS - Hyperlink Induced Topic Search

Questo approccio porta avanti un'analisi dei link usando un grafo detto **query-induced graph**.

In risposta a una query, vengono calcolati due score:

- 1) **Hub Scores**, che è alto per quelle pagine contenenti un sacco di link verso pagine autoritative.
- 2) **Authority Score**, alto per pagine che presentano tanti in-link, che arrivano da buoni hub.

L'idea è che buoni hub puntino a buoni authority.

ALGORITMO

- 1) Da la query a un sistema IR per ricavare le top-K pagine, il **root set R**.
- 2) Estende il root set con pagine che presentano out-link verso pagine del root-set e pagine che hanno in-link che puntano da pagine che stanno nel root-set. Il nuovo set è detto **base set S**.

- 3) Per ogni pagina $x \in S$ bisogna mantenere:

a) **Authority Score** $\alpha(x)$: $\sum_{x \in S} \alpha(x) = 1$

b) **Hub Score** $h(x)$: $\sum_{x \in S} h(x) = 1$

Possi

1) Inizializza, $\forall x \in S, \alpha(x) = 1 \wedge h(x) = 1$

2) $\text{for } (i=1 \dots K)$

a) $\forall x \in S: h(x) = \sum_{y \rightarrow x} \alpha(y)$

b) $\forall x \in S: \alpha(x) = \sum_{y \rightarrow x} h(y)$

c) $\forall x \in S: \alpha(x) = \frac{\alpha(x)}{c}$ dove $\sum_{x \in S} \left(\frac{\alpha(x)}{c} \right)^2 = 1$

d) $\forall x \in S: h(x) = \frac{h(x)}{c}$ dove $\sum_{x \in S} \left(\frac{h(x)}{c} \right)^2 = 1$

3) Gli score convergeranno dopo qualche iterazione. Nella pratica 5 iterazioni portano a una certa stabilità.

Dimostrazione della convergenza:

Sia A la matrice di adiacenza del base set S .

Possiamo calcolare:

1) Hub Score: $h = A \cdot \alpha$

2) Authority score: $\alpha = A^T h$

Sostituendo:

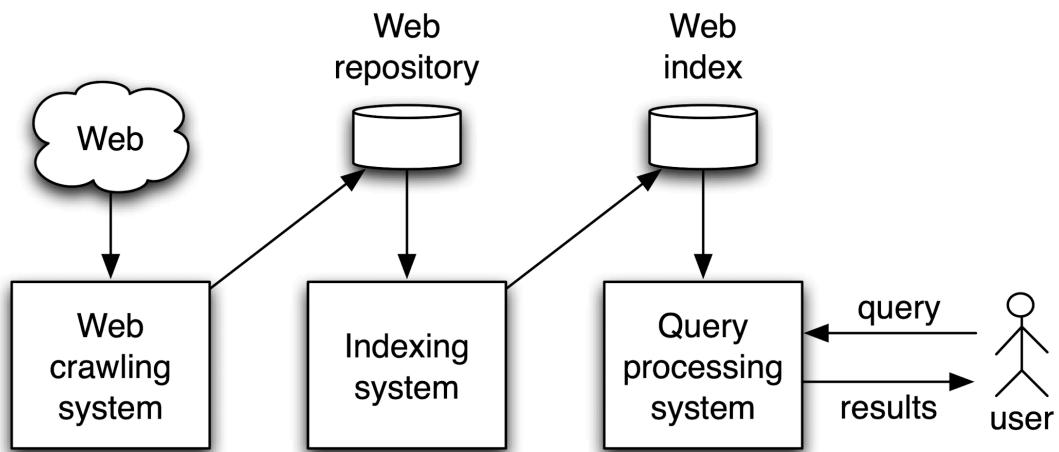
$$h = A A^T h \quad \text{dove } h \text{ è un autovettore di } A A^T$$

$$\alpha = A^T A \alpha \quad \text{dove } \alpha \text{ è un autovettore di } A^T A$$

Possiamo calcolarli usando il metodo delle potenze, che è un moto algoritmico che converge.

Note: l'algoritmo di HITS non viene usato perché dipendente dalle query. Questo comporta un overhead aggiuntivo per ogni query, ergo tempo perso.

ARCHITETTURA DEL WEB SEARCH



WEB CRAWLING

Questo è il processo usato dai motori di ricerca per creare il corpus su cui costruire l'indice. Il processo è diviso in più fasi:

1) Locating

Bisogna prima di tutto trovare le pagine web. Questo compito non è facile, per via della grandezza del web e anche perché il web non è un grafo连通的, ci possono essere nodi isolati.

2) Fetching

Una volta trovato l'URL, bisogna scaricare il contenuto delle pagine, ma non tutte le pagine offrono il loro contenuto facilmente e soprattutto gratuitamente.

Non possiamo nemmeno bombardare i vari server, altrimenti esso ci buterà fuori pensando ci sia un attacco DDoS.

3) Storing

Compito salvare tutto il web.

4) Maintaining

Dato la dinamicità del web, bisogna stare al passo con i nuovi update che arrivano ogni secondo. Tenere conto di informazioni forse troppo vecchie e inutili da tenere nel disco (dischi).

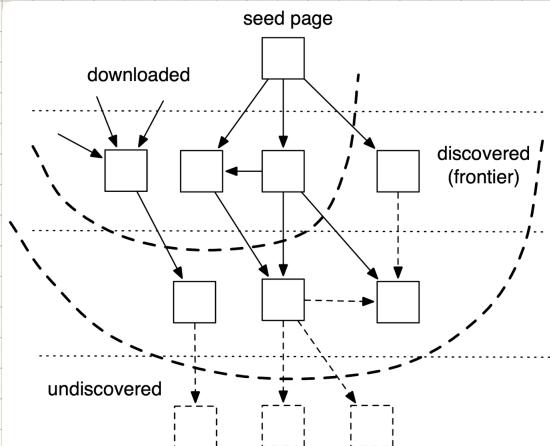
PASSI DI UN WEB CRAWLER

Inizializza una coda di URL da scaricare, attraverso qualche URL seed. Un buon URL di partenza deve avere tanti out-link.

```
while (TRUE)
{
    FETCH (URL)
    STORE (URL)
    EXTRACT_HYPOLINKS (URL)
    ADD (HYPER-LINKS) in coda
}
```

Questo processo divide l'intero web in tre subset:

- 1) Downloaded
- 2) Discovered
- 3) Undiscovered



La coda di download è gigantesca e non sta in una sola macchina. Fra l'altro deve essere una coda con priorità.

Proprietà dei crawler:

1) **Robustezza**: deve essere immune a vari atti maligni come le spider trap, ovvero pagine che generano URL spamm.

2) **Politica**: deve rispettare le condizioni esplicite e implicite delle pagine.

Non deve essere troppo veloce e fare crawl solo su pagine consentite.

Deve rispettare il contenuto del **robots.txt**.

Soluzione alle spider-trap

1) Controllare la lunghezza dell'URL, e evitare URL con una formattazione errata

2) Implementare le **trap guards**

ROBOTS.TXT

```
# robots.txt file

User-agent: googlebot          # all services
Disallow: /private/             # disallow this directory

User-agent: googlebot-news      # only the news service
Disallow: /                      # on everything

User-agent: *                    # all robots
Disallow: /something/           # on this directory

User-agent: *                    # all robots
Crawl-delay: 10                  # wait at least 10 seconds

Disallow: /dir1/                 # disallow this directory
Allow: /dir1/myfile.html         # allow a subdirectory

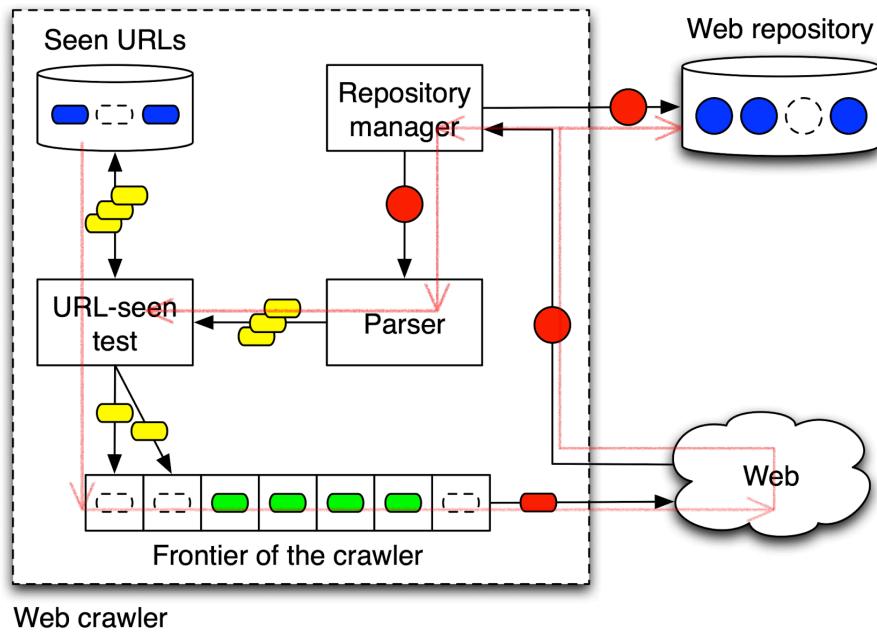
Host: www.example.com           # use this mirror
```

Dati info al crawler sulle parti da poter crawler e quelle proibite.

Pagine web specchio fatta per i crawler

Discovery Queue

- Stored pages
- Target page
- Visited URLs
- Target URL
- Extracted URLs
- Queued URLs



TRIARIA' DEGLI URL

- 1) **Randomica**
- 2) **Breadth-first**: Primo elemento scoperto nel grafo web
- 3) **In-degree**: Firma chi ha più in-link
- 4) **Page Rank**: Chi ha il PR maggiore

REFRESHING

- 1) Random
- 2) Age: il più vecchio viene aggiornato
- 3) Link-Quality
- 4) Search Impact
- 5) Longevity