Motori di Ricerca

presente e futuro prossimo

Rilevanza dei Risultati:

Prima generazione

Paolo Ferragina, Università di Pisa

Rilevanza derivata dal contenuto

- Per ogni occorrenza di una parola si memorizzano:
  - Luogo
    - URL: www.pisa.comune.it
    - Titolo pagina
    - Testo hyperlink: "Città di Pisa"
    - Metatag: autore, data,...
  - Tipo
    - Dimensione e tipo di carattere
    - Maiuscolo o minuscolo
  - Informazioni sulla "frequenza"

Assegnamo il "peso" a ogni termine e sommiamo i contributi per ogni pagina

Paolo Ferragina, Università di Pisa

Infatti

- La frequenza nel singolo documento non aiuta…
  - 10 occorrenze di ***culla***
  - 10 occorrenze di ***e***
- Per ogni coppia <*termine*,*documento*> assegnamo un peso che riflette l'importanza del termine in quel documento
  - Il peso cresce con il "numero di occorrenze" del termine *entro* quel documento
  - Il peso cresce con la "rarità" del termine *fra tutti* i documenti della collezione

Un "peso" famoso: tf x idf

$$w_{t,d} = tf_{t,d} \times \log(n / n_t)$$

$tf_{t,d}$ = Frequenza del termine *t* nel documento *d*

$idf_t = \log\left(\frac{n}{n_t}\right)$ dove n_t = #documenti che contengono il termine *t*
n = #documenti della collezione

Termine t ha associato un vettore D-dim: [w_{t1}, $w_{t2,}$ ..., w_{tD}]

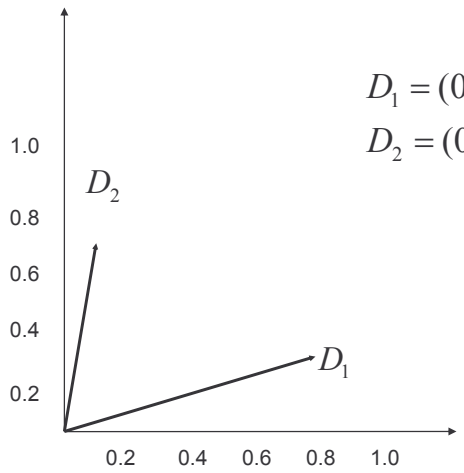
Documento d ha associato un vettore T-dim: [w_{1d}, $w_{2d,}$ ..., w_{Td}]

D = # docs in collezione, **T** = #termini del dizionario di tutta la collezione

Come usiamo questi pesi ?

- Data una interrogazione sui termini t_h e t_k potremmo:
  - Sommare w_{hj} e w_{kj} per ogni documento d_j che li contiene, o utilizzare un'altra funzione dei due valori
  - *Pesare* l'importanza di t_h e t_k all'interno della query e quindi calcolare una combinazione lineare di w_{hj} e w_{kj}.
  - Interpretare ogni documento e la query come vettori, e postulare la similarità tra doc-query in base alla loro *vicinanza euclidea* o tramite altra misura correlata.

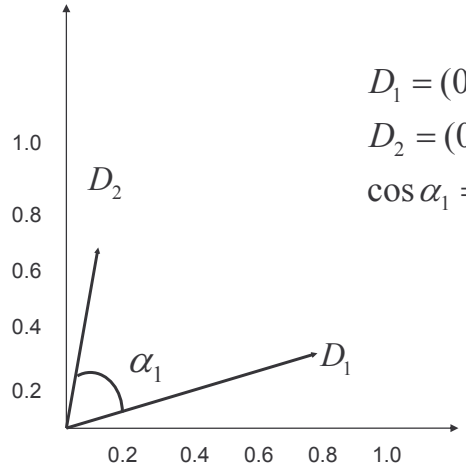
Documenti come vettori



$$D_1 = (0.8, 0.3)$$
$$D_2 = (0.2, 0.7)$$

Il dizionario ha 2 termini !!

Similarità tra Doc e Interrogazione

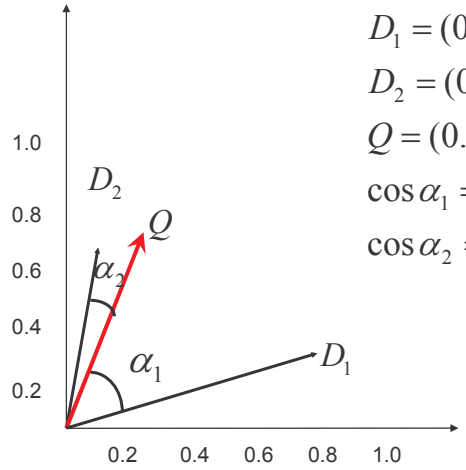


$$D_1 = (0.8, 0.3)$$
$$D_2 = (0.2, 0.7)$$
$$\cos\alpha_1 = 0,....$$

Similarità tra Doc

- sim(d1,d2) = Coseno dell'angolo compreso tra d1 e d2
- Nozione di **prodotto scalare**
  - Calcolare i prodotti di coppie di componenti in d1 e d2
  - Sommare i risultati dei prodotti
  - Esempio <0,1,2> * < 1,1,5> = 0*1 + 1*1 + 2*5 = 11
- Adottiamo come misura di similarità il prodotto scalare tra d1 e d2

Similarità tra Doc e Interrogazione



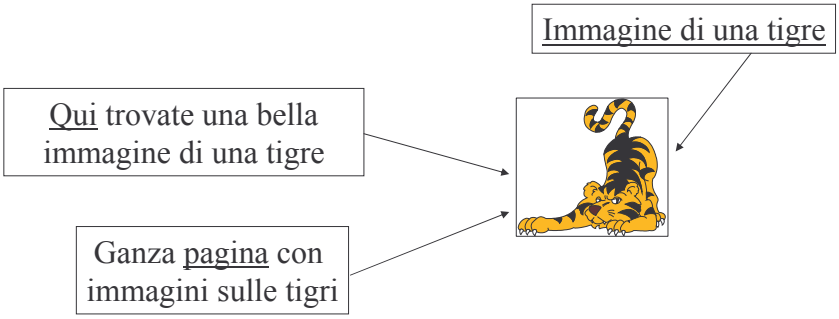
$$D_1 = (0.8, 0.3)$$
$$D_2 = (0.2, 0.7)$$
$$Q = (0.4, 0.8)$$
$$\cos\alpha_1 = 0.74$$
$$\cos\alpha_2 = 0.98$$

Matrice termini x documenti

- La memorizziamo tutta ?
- Ricordate il caso della matrice binaria sulla collezione di 6Gb, che richiedeva circa 500Gb?
- Immettiamo queste info nella lista invertita. Come?
- Siccome Q consiste di pochi termini t_i, non confrontiamo Q con tutti i docs, ma piuttosto:
  - Consideriamo Q come se fosse un piccolo doc
  - Lista invertita per prendere docs D_j che contengono i termini
  - Estraiamo da ogni D_j il peso w_{ij} , relativo ai t_i che contiene
  - Ricostruiamo i vettori e calcoliamo sim(Q, D_j)

Un altro peso: *Anchor text*

Indicizziamo i virtual doc costruiti concatenando gli anchor text dei link che puntano a una determinata pagina



NOTA: Il testo nella vicinanza di un hyperlink è molto descrittivo del contenuto della pagina a cui esso fa riferimento !

Ricapitolando

- Per ogni occorrenza di una parola si memorizzano:
  - Luogo
  - Tipo
  - TF x Idf
- I motori di prima generazione usavano questi pesi per inferire la *similarità* dei documenti con la query
- Poi ordinavano le risposte (docs) in accordo a questa

Motori di Ricerca

presente e futuro prossimo

Rilevanza dei Risultati:

Seconda generazione

Paolo Ferragina, Università di Pisa

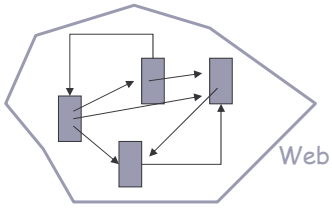
Sfruttare gli hyperlink

- Problema:

Molte pagine contengono le parole in Q ma sono “non rilevanti” oppure includono parole “diverse” dal loro contenuto (*spamming*).

Altre pagine sono sì rilevanti ma non contengono le parole di Q.

- Hyperlink ≡ Citazione



Paolo Ferragina, Università di Pisa

Analisi degli hyperlink

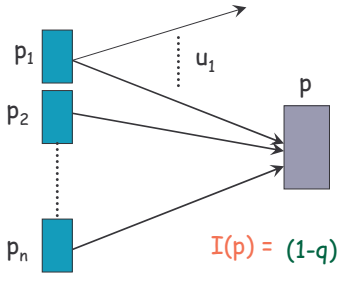
- Due approcci fondamentali
  - Indipendente dalla interrogazione
    - Se due pagine contengono le parole di Q, una sarà sempre migliore dell'altra indipendentemente da Q

      (Pagerank di Google)
  - Dipendente dalla interrogazione
    - Se due pagine contengono le parole di Q, una sarà migliore dell'altra a seconda del contenuto di Q

      (HITS di IBM e Teoma)

PageRank (Google)

- Pagina rilevante se:
  - Molte pagine puntanto a essa (popolare)
  - Alcune pagine "rilevanti" puntano a essa (élite)



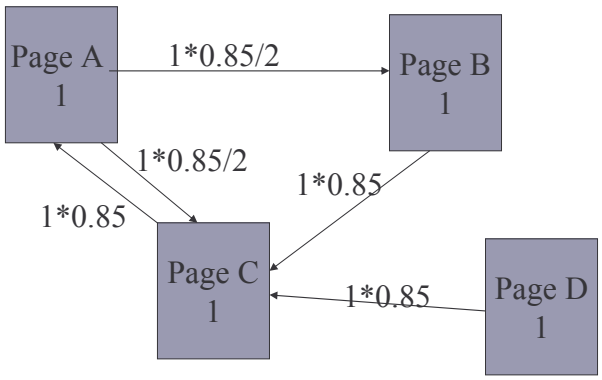
$$I(p) = (1-q)\left\{\frac{I(p_1)}{u_1} + \frac{I(p_2)}{u_2} + \ldots + \frac{I(p_n)}{u_n}\right\} + q$$

- Calcolato su tutte le pagine e in modo iterativo (~100)

Attenti ai Blog !

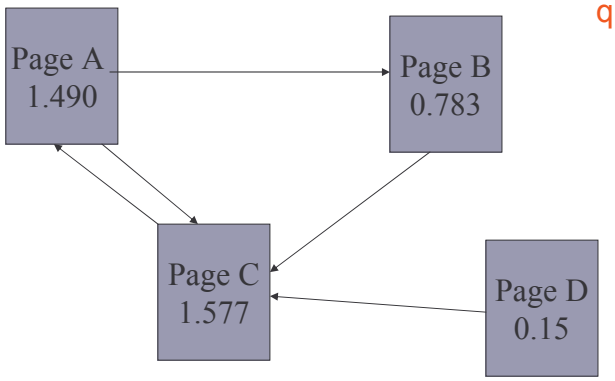
Un esempio: passo iniziale

q = 0.15



Esempio: dopo 20 iterazioni

q = 0.15



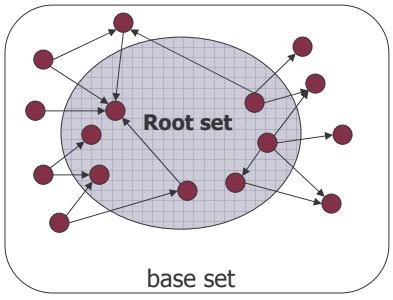
Sarebbe necessario, in verità, cambiare +q in +(q/#pagine)
questo garantisce che il vettore dei pesi uscenti abbia somma 1,
e quindi (Teorema) il PageRank è una distribuzione di probabilità

HITS (IBM)

- A seguito di una interrogazione si cercano *due insiemi* "correlati" di pagine:
  - *Pagine Hub* = pagine che contengono una buona lista di link sul soggetto della interrogazione.
  - *Pagine Authority* = pagine che occorrono ripetutamente nelle liste contenute dei buoni Hubs.

Si tratta di una definizione circolare che quindi richiede una computazione iterativa

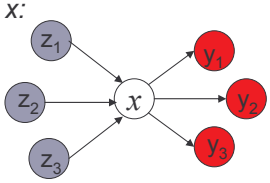
HITS: Primo passo per risolvere Q



- Data una interrogazione Q={ ***browser*** }, si forma il base set:
  1. Le pagine che contengono ***browser*** (root set)
  2. Le pagine collegate *da* o *per* quelle del *root set*

HITS: Secondo passo per risolvere Q

- Calcoliamo, per ogni pagina *x* del *base set*:
  - un hub score *h(x),* inizializzato a 1
  - un authority score *a(x),* inizializzato a 1
- Per poche iterazioni, ricalcoliamo di ogni nodo *x:*
  - $a(x) = \sum h(z_i)$, $h(x) = \sum a(y_i)$
  - Scaliamo i valori, e iteriamo

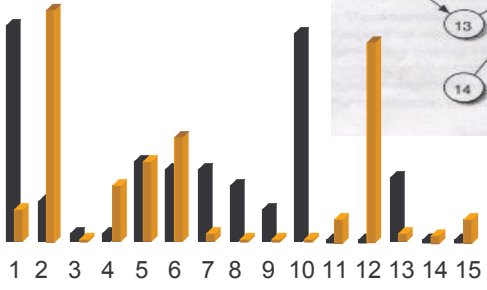


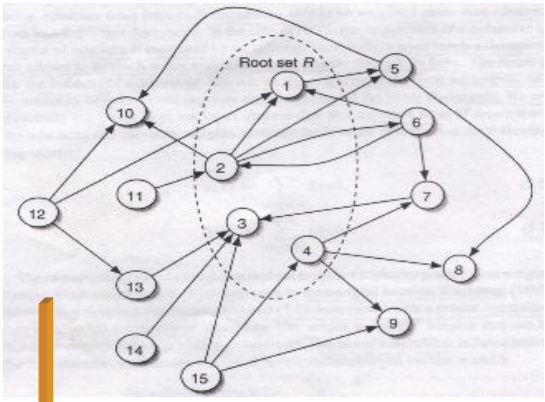
- Alla fine, restituiamo le pagine con più alto valore di *h()* come *hubs,* e di *a()* come *authorities*

Costoso: Accumulo del *base set* e calcolo iterativo !!

Controindicazioni: Facilmente soggetto a *SPAM* !!

Un esempio





Motori di Ricerca

presente e futuro prossimo

Rilevanza dei Risultati:

Terza generazione

Paolo Ferragina, Università di Pisa

Nuovi obiettivi

- Obiettivo: Integrare dati provenienti dalle sorgenti più disparate – quali, preferenze, click, affinità tra utenti, transazioni– al fine di soddisfare meglio l'interrogazione posta da un utente
- Esempio: Su una interrogazione come "San Francisco" il sistema dovrebbe trovare anche gli hotel o i musei, siti per le previsioni del tempo o mappe stradali, intuendo anche quali di questi è più rilevante per l'utente
- Tools: Ciò richiede analisi semantica, determinazione del contesto, selezione dinamica di archivi utili, confronto tra sessioni …

Nuove nozioni di Rilevanza !!!

Paolo Ferragina, Università di Pisa

Rilevanza per "affinità"

Precedenti transazioni: [Collaborative Filtering]

- Quali documenti/pagine sono state visitate, anche da altri utenti
- Quali prodotti sono stati acquistati, anche da altri utenti
- Pagine nei bookmarks dell'utente

Contesto corrente: [User behavior]

- Storia della presente navigazione
- Ricerche già formulate dallo stesso utente

Profilo: [Personalization]

- Professione dell'utente e informazione demografica
- Interessi dell'utente

➢ Esistono dei problemi di privacy !!!

Ricapitolando...

- Data una interrogazione Q su più parole
  - Troviamo le pagine dove occorrono quelle parole
  - Per ogni pagina determiniamo:
    - Peso testuale: font, luogo, posizione, vicinanza,…
    - Peso degli hyperlinks: grafo e anchor-text
    - Peso dato da altri fattori: preferenze, comportamento,…
  - Sommiamo "in qualche modo" i pesi
  - Ordiniamo le pagine in funzione di essi ⇨ Risultati !!
  - Offriamo possibilmente dei suggerimenti, anche semantici

Questo è un motore di ricerca moderno !!
(siamo alla terza generazione)

Motore di Ricerca: struttura