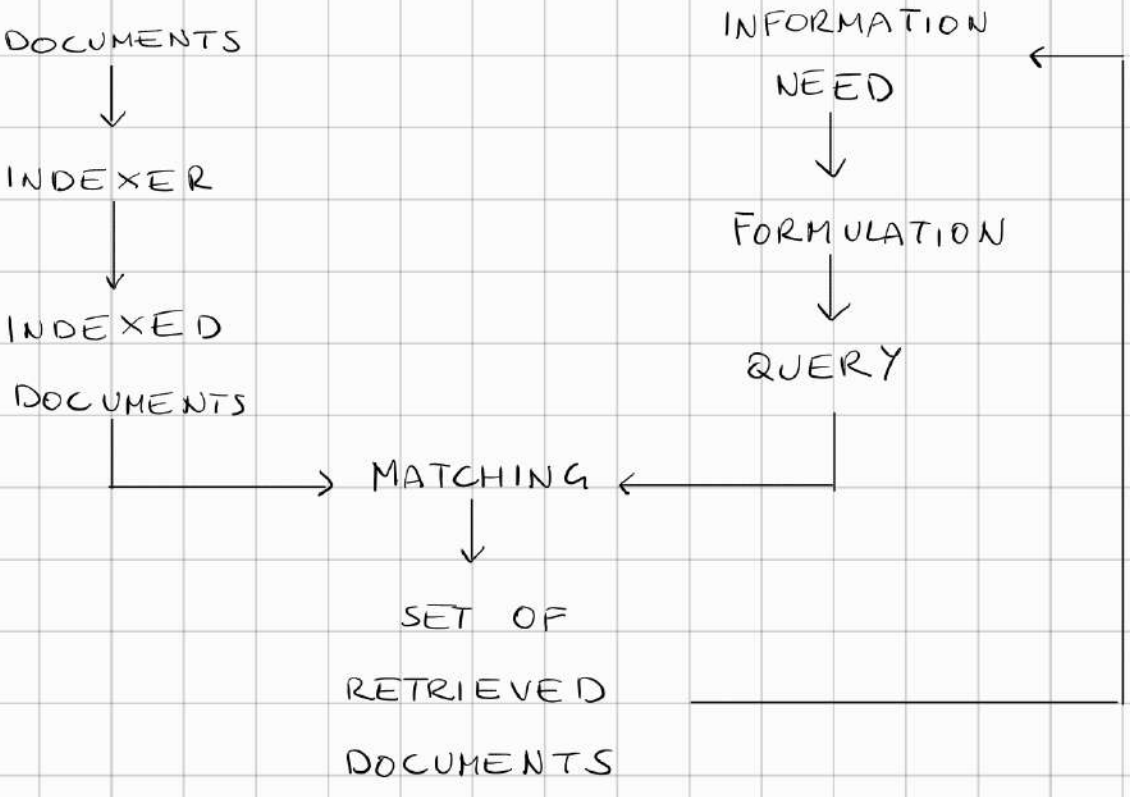
IR

Trovare documenti di natura non strutturata all'interno di una grande collezione e che soddisfano un bisogno informativo

Paradosso di Menon: se si conosce qualcosa, non serve cercarla. Se non si conosce qualcosa come si fa a cercarla, dato che non si saprebbe cosa cercare

IRS

Sistema che data una query recupera dei documenti ordinati per rilevanza da una repository



IR MODELS

Deve implementare tre aspetti:

- RETRIEVE
- RANK
- QUERY

INDICE

	K_1	K_2	K_3	K_4	K_t
d_1	1	0	0	0	1

D è l'insieme dei documenti di dimensione n

d_2	0	0	0	0	0
d_n	1	0	1	1	0

q	0	0	0	1	1

K è l'insieme di termini di dimensione t

$w_{ij} \in \{0, 1\}$ indica se $K_i \in d_j$

Questa implementazione crea un' indice sparso

MODELLO BOOLEANO

È basato sulla teoria degli insiemi e sull' algebra booleana

La query è formulata usando connettivi logici

PRO:

- semplice da implementare e trasparente
- decisione binaria

CONTRO:

- poco intuitivo
- non c'è un ranking dei documenti
- matching esatto

MODELLO VECTOR SPACE

Utilizza uno spazio t-dimensionale, dove ad ogni termine è associata una dimensione e ogni documento può essere rappresentato come un vettore.

Anche la query è un vettore, di conseguenza si può usare la similarità del coseno:

$$sim(d_j, q) = \frac{\sum (w_{ij} \cdot w_{iq})}{\sqrt{\sum w_{ij}^2} \cdot \sqrt{\sum w_{iq}^2}} \qquad d_j = (w_{1j}, w_{2j}, \dots, w_{tj}) \qquad q = (w_{1q}, w_{2q}, \dots, w_{tq})$$

normalizza la lunghezza

Un documento può essere recuperato anche se non contiene tutti i termini della query

TF-IDF

Come scegliere w_{ij}?

- se un termine appare spesso nello stesso doc, il peso sarà maggiore

- se un termine appare in tanti doc, il peso sarà minore

TERM FREQUENCY: $tf(i,j) = \frac{n_{ij}}{N_j}$

INVERTED DOCUMENT FREQUENCY: $idf(i) = \log \frac{N}{n_i}$

$$w_{ij} = tf(i,j) \cdot idf(i)$$

PRO:

- pesi non binari permettono di rappresentare meglio gli argomenti
- matching parziale
- ranking
- migliore del booleano

CONTRO:

- il modello si basa sull'assunzione che i termini siano indipendenti, cosa non vera. Non si sa ancora se questo ha implicazioni pratiche
- curse of dimensionality: più aumentano le dimensioni più lo spazio aumenta esponenzialmente, la distanza non è quindi più informativa e i punti sempre più lontani tra loro

BIM - MODELLO PROBABILISTICO

Data una query il modello stima la probabilità di rilevanza per ogni documento della collezione utilizzando le evidenze disponibili

Assume che la rilevanza sia binaria.

$p(R)$: probabilità che un documento sia rilevante rispetto una query

$p(R|d_j)$: probabilità che d_j sia rilevante. Saperla $\forall$ documento permette il ranking

$p(d_j)$: probabilità di estrarre d_j dalla collezione

$p(d_j|R)$: probabilità di estrarre d_j dall'insieme dei rilevanti

Si può definire una misura di similarità:

$$sim(d_j, q) = \frac{p(R|d_j)}{p(\bar{R}|d_j)}$$

Non conosciamo questi termini ma si può usare Bayes

$p(Q|e) = \frac{p(Q) \cdot p(e|Q)}{p(e)} \longrightarrow sim(d_j, q) = \frac{p(R)}{p(\bar{R})} \frac{p(d_j|R)}{p(d_j|\bar{R})}$

$p(R)$ e $p(\bar{R})$ sono indipendenti dal singolo documento, date una query sono costanti, di conseguenza:

$$sim(d_j, q) \sim \frac{p(d_j|R)}{p(d_j|\bar{R})}$$

Un documento è un'insieme di feature, ad esempio i suoi termini:

$$p(d_j|R) = p(\{K_1, K_2, K_3, \ldots\}|R)$$

Si assume che le feature siano binarie e indipendenti:

$$K_i = \begin{cases} 0 & \text{if } K_i \notin d_j \\ 1 & \text{if } K_i \in d_j \end{cases}$$

L'indipendenza permette di dividere la probabilità nel prodotto delle probabilità

$$\begin{aligned} p(d_j|R) &= p(\{K_1, K_2, K_3, \ldots\}|R) \\ &= p(K_1|R) \cdot p(K_2|R) \cdot p(K_3|R) \ldots \\ &= \prod (K_i|R) \end{aligned}$$

probabilità che un documento estratto dai rilevanti abbia la feature K_i

$$sim(d_j, q) \sim \frac{p(d_j|R)}{p(d_j|\bar{R})}$$

$$= \prod \frac{p(K_i|R)}{p(K_i|\bar{R})}$$

$$= \prod_{K_i \in q} \frac{p(K_i|R)}{p(K_i|\bar{R})} \cdot \prod_{K_i \notin q} \frac{p(K_i|R)}{p(K_i|\bar{R})}$$ Distinguo tra termini in q e non

$$= \prod_{K_i \in q \wedge K_i \in d_j} \frac{p(K_i|R)}{p(K_i|\bar{R})} \cdot \prod_{K_i \in q \wedge K_i \notin d_j} \frac{p(K_i|R)}{p(K_i|\bar{R})}$$ Distinguo tra termini in d_j e non

doc ind

$$= \prod_{K_i \in q \wedge K_i \in d_j} \frac{p(K_i|R)}{p(K_i|\bar{R})} \cdot \prod_{K_i \in q \wedge K_i \notin d_j} \frac{p(\bar{K}_i|R)}{p(\bar{K}_i|\bar{R})}$$

$$= \prod \frac{p(K_i|R)}{p(K_i|\bar{R})} \cdot \prod \frac{p(\bar{K}_i|\bar{R})}{p(\bar{K}_i|R)}$$ → probabilità inverse

$K_i \in q \wedge K_i \in d_j$ $p(K_i|R)$ $\quad$ $K_i \in q \wedge K_i \in d_j$ $p(\bar{K}_i|R)$

$$sim(d_j, q) \sim \log \prod_{\substack{K_i \in q \wedge \\ K_i \in d_j}} \frac{p(K_i|R)\,p(\bar{K}_i|\bar{R})}{p(K_i|\bar{R})\,p(\bar{K}_i|R)}$$

Si aggiunge il log perché è monotono e per scopi implementativi

$$\sim \sum_{\substack{K_i \in q \wedge \\ K_i \in d_j}} \log \frac{p(K_i|R)\,p(\bar{K}_i|\bar{R})}{p(K_i|\bar{R})\,p(\bar{K}_i|R)}$$

- La similarità aumenta se è probabile che i termini siano in doc rilevanti o che non siano in doc non rilevanti
- La similarità diminuisce se è probabile che i termini siano in doc non rilevanti o che non ci siano in doc rilevanti

Non è propriamente una misura di similarità ma può essere usata per rankare i documenti

Come calcoliamo le 4 probabilità (in realtà 2):

$$p(\bar{K}_i|R) = 1 - p(K_i|R)$$
$$p(\bar{K}_i|\bar{R}) = 1 - p(K_i|\bar{R})$$

2 opzioni:

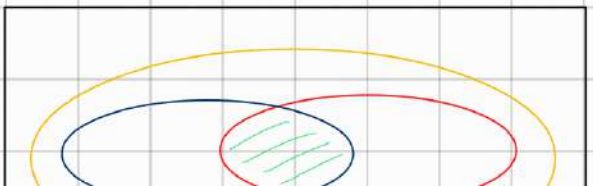
- all'inizio: subito dopo la specificazione della query. $p(K_i|R)$ è sconosciuta ma si può approssimare a 0.5. Anche $p(K_i|\bar{R})$ è sconosciuta ma si può approssimare $\frac{n_i}{N}$
- dopo il relevance feedback: l'utente comunica la rilevanza fornendo quindi nuove evidenze:

  V = # doc valutati

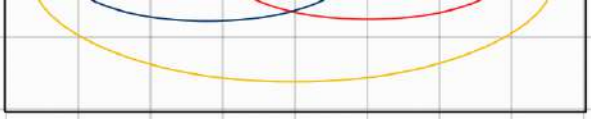
  VR = doc valutati rilevanti

  V_i = doc valutati che contengono K_i

  VR_i = doc valutati rilevanti che contengono K_i



$$p(K_i|R) = \frac{VR_i + 0.5}{VR + 1}$$



$$p(K_i|R) = \frac{V_i - VR_i + 0.5}{V - VR + 1}$$

Non è fondamentale un feedback reale, si può utilizzare uno pseudo relevance feedback assumendo che i primi n documenti siano rilevanti

PRO:

- forti basi matematiche
- interazione con l'utente
- molto efficace

CONTRO:

- stime iniziali complicate
- assunzione di indipendenza dei termini

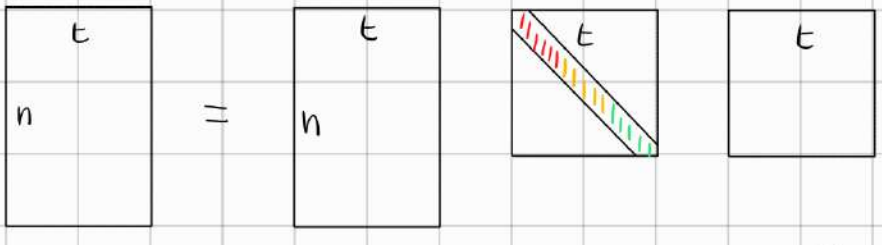
LATENT SEMANTIC INDEXING

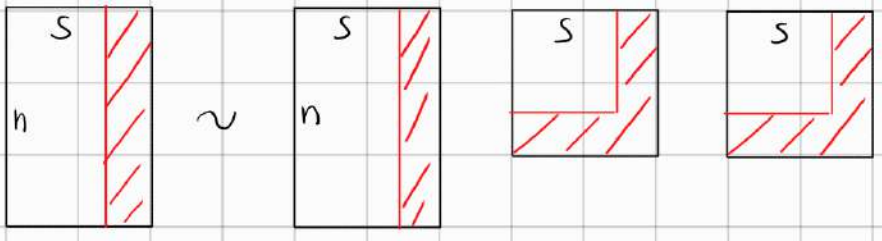
L'indice è una matrice $n \times t$, dove $n \sim 10^6/10^{10}$ e $t \sim 10^5$

SINGULAR VALUE DECOMPOSITION

$M = K \cdot S \cdot D^T$ dove:

- S è una diagonale con valori decrescenti
- K e D sono autovettori di MM^T e M^TM





Mantenendo solo gli s elementi più grandi di S, si ottiene la migliore approssimazione di M

Il modello LSI usa M_s come indice, ottenendo:

- riduzione della dimensionalità, più efficienza, meno memoria...
- più astrazione: si passa da termini a concetti

- risolve i problemi dei sinonimi e polisemia

BM25

È un' estensione del BIM che include la term frequency e la document length normalization

$$sim(d_j, q) = \sum_{K_i \in q \wedge K_i \in d_j} \log \frac{p(K_i|R)\, p(\bar{K}_i|\bar{R})}{p(K_i|\bar{R})\, p(\bar{K}_i|R)} \frac{(\alpha+1)f_i}{\gamma + f_i} \frac{(\beta+1)qf_i}{\beta + qf_i}$$

dove:

- f_i è la frequenza di K_i in d_j
- qf_i è la frequenza di K_i in q
- α è un parametro che regola il contributo del TF
- β è un parametro che regola il contributo del TF della query
- γ (gamma) è un parametro che regola la DLN

$$\alpha \left((1-b) + b \frac{dl}{\overline{dl}} \right)$$

- $b = 0$, no DLN, TF aumenta con f_i
- $b > 0$, DLN, doc lunghi penalizzati

PREPROCESSING

Fase in cui si costruisce l' indice a partire dai documenti

PROPRIETÁ DEL LN

Il testo è una sequenza di caratteri generati da una distribuzione non uniforme, ogni lettera ha una determinata frequenza

Esistono diversi modelli:

- BINOMIALE: la probabilità dipende dalla frequenza
- MARKOVIANO: la probabilità dipende dalle lettere precedenti

LEGGE DI ZIPF

La distribuzione di frequenza delle parole segue una legge di potenza

$$F(r) = \frac{C}{r^{\alpha}}$$

dove:

- $C \sim 0.1$ e $\alpha \sim 1$
- r è il rank

La r-esima parola ha frequenza di $\sim 1/r$. Ci sono poche parole che appaiono molto spesso e molte parole che appaiono poco

LEGGE DI HEAP

Il vocabolario è l'insieme di termini unici in una collezione.

Questa legge modella l'aumento del numero di termini nel vocabolario di una collezione all'introduzione di nuovi documenti

$V = Kn^{\beta}$ dove:

- n è la lunghezza del testo
- $K \sim 10/100$

Il numero di nuovi termini in un testo di n parole cresce di $\sqrt{n}$

INDICE INVERTITO

È composto da due parti:

- vocabolario
- occorrenze

È chiamato invertito perché si passa da quali sono i termini in ogni documento a quali sono i documenti per ogni termine

VOCABOLARIO

- termine
- # di documenti in cui occorre
- # di occorrenze totali
- puntatore alla prima occorrenza

Spazio: $O(\sqrt{n})$

OCCORRENZE:

- id del documento
- # di occorrenze nel doc
- puntatore alla linked list delle posizioni delle occorrenze nel doc
- puntatore al documento successivo in cui il termine occorre

Spazio: $O(n)$ (almeno un'occorrenza $\forall$ termine)

COSTRUZIONE DELL'INDICE

Trie: struttura ad albero con search $O(m)$, m è la lunghezza

L'indice viene costruito con un trie e la lista delle occorrenze:

$\forall$ termine $\in$ collezione

∀ termine K ∈ collezione:

cerco K nel trie:

se K non è nel trie:

inserisco K nel trie con lista delle occorrenze vuota

altrimenti:

aggiungo la posizione dell'occorrenza alla lista

salvo trie (vocabolario) e lista delle occorrenze su due file

Costo $O(n)$: ∀ carattere $O(1)$ nel trie + inserire l'occorrenza $O(1)$

Se non c'è abbastanza memoria:

indexing parziale + merging $O(n \log(n/M))$

COMPRESSIONE DELL'INDICE

Tre cose possono essere compresse

- COLLEZIONE: decomprimere il documento solo in visualizzazione
- VOCABOLARIO: inutile, perché si dovrebbe mantenerli entrambi
- OCCORRENZE: salvare gli offset delle occorrenze

ALGORITMO DI HUFFMAN

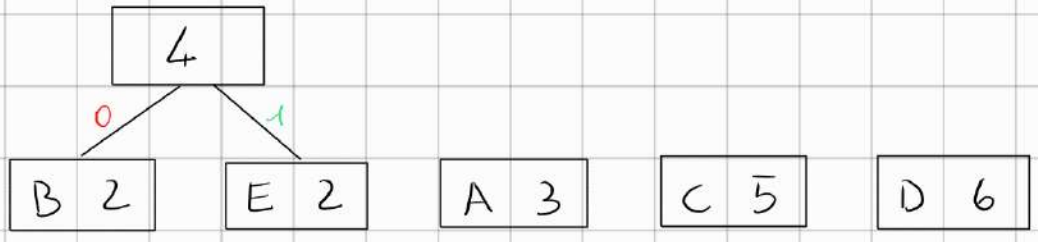
Si usa per comprimere collezione e vocabolario

- codifica più corta per i termini più frequenti
- codifica più lunga per i termini meno frequenti

1) Sulle foglie i termini ordinati per frequenza

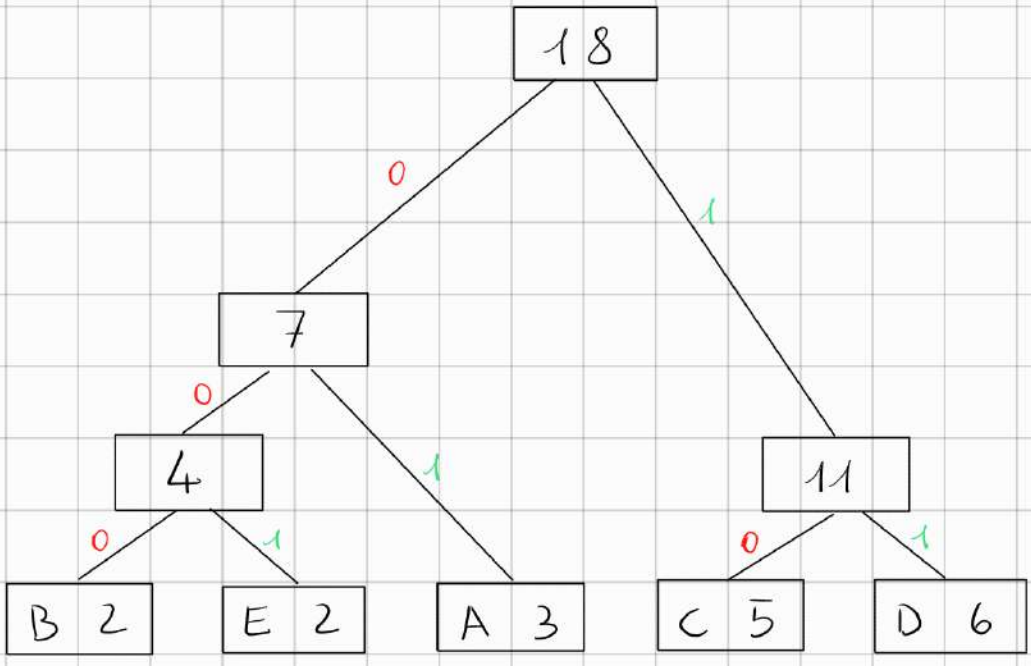
B	2		E	2		A	3		C	5		D	6

2) creo un nodo con la somma delle frequenze, che ha come figli due nodi con frequenza minore e etichetto gli archi con 0 e 1



3) continuo fino a raggiungere la radice. I bit sui cammini

indicano la codifica



A = 01
B = 000
C = 10
D = 11
E = 001

QUERY RIFORMULATION

Tecniche per modificare la query:

- RELEVANCE FEEDBACK: l'utente esamina l'insieme dei documenti recuperati e esprime un giudizio sulla loro rilevanza.
  Queste informazioni vengono impiegate per rimuovere termini dalla query, aggiustare i pesi...
  Si basa sulla CLUSTER HYPOTHESIS, secondo la quale documenti vicini sono rilevanti per le stesse query
  DERIVARE UNA NUOVA QUERY
  Muovere q verso il cluster dei documenti rilevanti e lontano da quelli non rilevanti usando i centroidi
- LOCAL ANALYSIS: MM^T si ottiene la matrice delle co-occorrenze che dice quante volte due termini appaiono nello stesso documento
  Nella local analysis i documenti recuperati vengono analizzati in query time per scegliere i termini per la query expansion
  - LOCAL CLUSTERING: costruire cluster di termini associati per ogni termine della query:
    - ASSOCIATION: usa il contesto locale e non tutta M
      data una query q definiamo:
      - D_q l'insieme dei documenti recuperati

· V_1 il vocabolario di D_1,
con i quali costruiamo una matrice $D_1 \times V_1$,
simile a M, ma che usa il contesto locale
· le colonne hanno solo $d_j \in D_1$
· le righe sono tutte > 0
si può quindi calcolare la matrice delle co-occorrenze,
normalizzarla, e per ogni $K_u \in q$ considerare i
termini che co-occorrono con K_u sopra un certo
treshold

- METRIC: utilizza una misura della distanza (in parole) per definire la matrice delle co-occorrenze.
Il valore di co-occorrenza è inversamente proporzionale alla distanza
- LOCAL CONTEXT ANALYSIS: lavora su concetti non termini
  - i documenti recuperati sono suddivisi in passaggi di lunghezza fissa
  - i passaggi vengono rankati e i primi n selezionati
  - $\forall$ concetto c viene calcolato $sim(q, c)$
  - i primi m concetti vengono aggiunti alla query con peso decrescente e minore dei termini iniziali

CLASSIFICAZIONE

Task di apprendimento supervisionato

NAIVE BAYES CLASSIFIER

È naive perché assume indipendenza, anche se spesso non è vero

$$p(c|d) = \frac{p(d|c)\,p(c)}{\cancel{p(d)}} = p(c)\prod p(K_i|c)$$

↳ è uguale $\forall c$

- $p(c) = \frac{\#\text{ docs in } c}{\#\text{ docs}}$
- $p(K_i|c) = \frac{\#\text{ occ of } K_i \text{ in } c}{\#\text{ occ in } c}$ smoothing $\frac{\#\text{ occ of } K_i \text{ in } c + 1}{\#\text{ occ in } c + |V|}$

Problemi:

- documenti con termini misti
- documenti con termini nuovi, risolvibile con lo smoothing

CLUSTERING

Task di apprendimento non supervisionato

Si usa in IR per ottimizzare l'efficienza, matchare la query con un centroide anziché diversi documenti

CLUSTERING GERARCHICO

Usa un approccio bottom-up. Diversi algoritmi:

- SINGLE LINK: similarità massima, distanza minima
- COMPLETE LINK: similarità minima, distanza massima
- GROUP AVERAGE: similarità media → # termini in comune tra d_i e d_j

Si usa la matrice delle similarità $M^T M$, costo $O(n^2)$, costoso quindi si usano algoritmi basati su euristiche con costo $O(n)$

- ONE PASS: prendo il primo documento e lo metto nel primo cluster, il secondo se ha similarità con il primo minore di un certo threshold lo metto nel primo cluster, altrimenti ne creo uno nuovo
- ROCCHIO: se un doc passa il test della densità, ovvero che ha un determinato numero di vicini, e non appartiene a nessun cluster, lui e i suoi vicini vengono clusterizzati
- K-MEANS: vengono scelti K centroidi casualmente. I documenti vengono processati sequenzialmente e ogni documento viene clusterizzato con il centroide più simile. I centroidi vengono poi ricalcolati

VALUTAZIONE

PRECISION AND RECALL

Sono interessato a trovare tutti i doc rilevanti (recall) o solo documenti rilevanti (precision)

$$P = \frac{|rel \wedge retr|}{|retr|} \qquad R = \frac{|rel \wedge retr|}{|rel|}$$

	RETR	NOT RETR
REL	A	B
NOT REL	C	D

$$P = \frac{A}{A+C}$$

NOT REL	C	D	$R = \frac{A}{A+B}$

Si utilizza il grafico P/R per visualizzare la precisione a diversi livelli di recall

Altre metriche:

- FALLOUT: proporzione di documenti non rilevanti recuperati rispetto a tutti i documenti non rilevanti

$$F = \frac{C}{C+D}$$

- GENERALITY FACTOR: proporzione di documenti rilevanti (dipende dalla q)

$$G = \frac{A+B}{A+B+C+D}$$

Non c'è certezza del numero di documenti rilevanti, di conseguenza, il recall non può essere calcolato con certezza

AVERGE PRECISION: calcola la precisione media ad ogni livello di recall rispetto ad una query e corrisponde all'area sotto la curva P/R

MAP: è l'AP media per diverse query

P@n è la frazione di documenti rilevanti considerando solo i primi n

WIR

Alcune differenze:

- dimensioni della collezione $50 \cdot 10^9$
- dinamicità
- multimedialità
- lingue
- spam

WEB SEARCH ENGINE

- CRAWLER: estrae dalla collezione le cose da indicizzare. Esegue un browsing del web ricorsivo seguendo gli URL
- INDEXER: processa i dati raccolti dal crawler e costruisce l'indice
- QUERY PROCESSOR: processa le query e restituisce la risposta

THE WEB GRAPH

$G = (\text{web pages}, \text{HTML links})$

4 - (web pages, HTML links)

|N|? Varie tecniche:

- DATI PUBBLICI: forniti dai SEs
- CAPTURE & RECAPTURE: con l'assunzione del campionamento casuale
- RANDOM QUERIES: fare query casuali e fare intersezione e unione delle pagine ottenute
- RANDOM HTTP REQUESTS
- RANDOM WALK: andare da una pagina all'altra scegliendo casualmente i link in essa

Qual'è la topologia del grafo del web?

Sicuramente non è regolare

Non è casuale, perché i nodi sono influenzati da entità reali e i link sono influenzati da relazioni sociali, link già esistenti...

È una rete del piccolo mondo:

- la probabilità di link tra due nodi è tra 0 e 1
- $n >> K >> \ln(n)$
- diametro piccolo
- alto coefficiente di clustering, forte connessione con i vicini

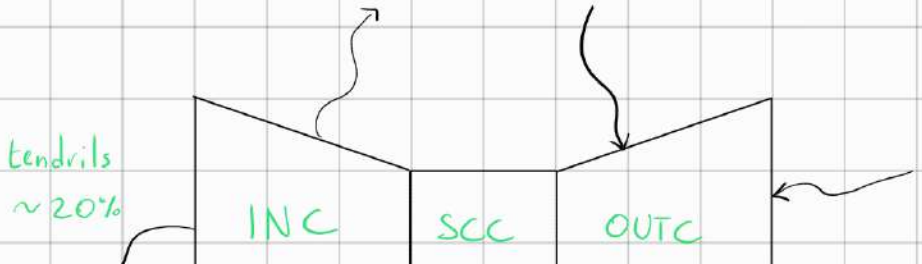
Distribuzione dei gradi?

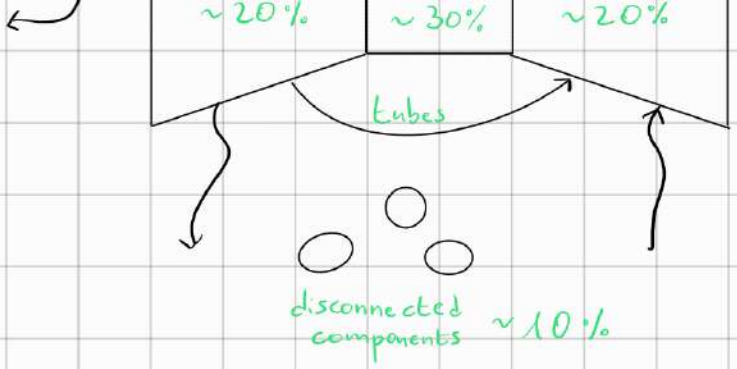
- 1° STUDIO - Barabasi:

  hanno fatto un crawl su un sottoinsieme del web per simulare la rete
  $P_{out}(k)$ e $P_{in}(k)$ seguono entrambe seguono una power law
  Inoltre hanno scoperto che il diametro cresce con $\log(n)$

- 2° STUDIO - Broder:

  hanno fatto crawl multipli ($\sim 1/2 \cdot 10^9$ pagine) confermando la distribuzione a power law
  Per quanto riguarda la forma, viene identificata una struttura complessa detta bow tie





- 3° STUDIO - Bhorat:

  confermano le ipotesi di Broder

LINK ANALYSIS FOR RANKING

NAIVE LAR

Usato nei primi SE. Calcola la:

- popolarità diretta: # in links
- popolarità indiretta: # in links + # out links

Le pagine sono recuperate in base all'argomento e ordinate per popolarità

Problemi:

- facilmente spammabile

PAGE RANK

Il valore di PageRank di una pagina è alto se le pagine che la linkano hanno alto PageRank

È indipendente dalla query, ogni pagina ha il suo score

$$r(v) \sim c \cdot \sum_{u \in I(v)} \frac{r(u)}{|O(u)|}$$

- $I(v)$ è l'insieme delle pagine che linkano v
- $O(u)$ è l'insieme delle pagine linkate da u

La pagina v riceve del PageRank dalle pagine che la linkano, che a loro volta ricevono una porzione di PageRank dalle pagine a loro collegate

Si usa un termine di normalizzazione c

CALCOLO DEL PAGERANK

Può essere memorizzato in un vettore riga: $r = [u_1, u_2, \dots, u_n]$

e viene calcolato iterativamente: $r_{i+1} = r_i P$

Si usa random walk: per il limite $t \to \infty$ (se $\exists$, $\exists$ uno stato stabile), il PageRank di una pagina è la probabilità di trovarsi in essa

P è una matrice $n \times n$, detta matrice di transizione. VE la catena di Markov si trova in uno degli n stati. $\forall$ stato i, P contiene la probabilità che il prossimo stato sia j

Il passo è quindi $x_{i+1} = x_i P$. Stiamo cercando una distribuzione stabile $x_{i+1} = x_i$, $x = x P$, dove x è l'autovettore sinistro di P

Una catena di Markov ergodica, ovvero che per $t \to \infty$ è possibile raggiungere tutti gli stati, la proporzione converge.

Il web non è ergodico → Teleporting, una costante

PRO:

- non spammabile
- indice di qualità per tutto il web
- molto utilizzato

CONTRO:

- non è query specific
- computazione complessa

HITS

- HUB: pagine che linkano pagine autorevoli
- AUTORITY: pagine linkate da buoni hub

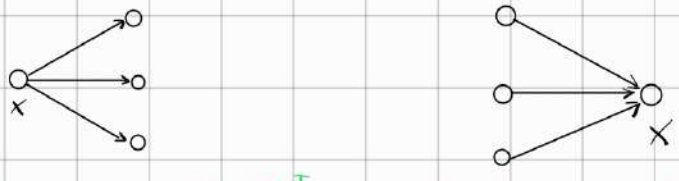
L'algoritmo non viene usato su tutta la collezione del web, ma sull'insieme delle pagine recuperate chiamato ROOT SET

Il RS viene espanso con:

- $I(v)$: in pages $\forall u, \exists v \in RS$ t.c. $u \to v$
- $O(v)$: out pages $\forall u, \exists v \in RS$ t.c $v \to u$

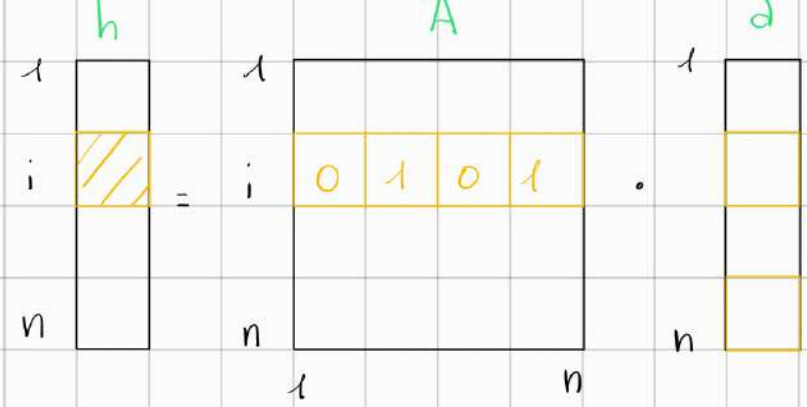
Si costruisce così il BASE SET sul quale l'algoritmo opera. Per ogni pagina $x \in BS$ viene calcolato l'hub e authority score $h(x)$ e $a(x)$

$$h(x) = \sum_{x \to y} a(y) \qquad a(x) = \sum_{y \to x} h(y)$$



$$h = [h(u_1) \dots h(u_n)]^T \qquad a = [a(u_1) \dots a(u_n)]^T$$

Per calcolare h_i si sommano tutti gli a_i corrispondenti agli outlink di u_i



Sostituendo: $h = Aa = AA^T h$ $\quad a = A^T h = A^T A a$

dove a e h sono rispettivamente gli autovettori delle matrice:

- $A^T A$ è la matrice delle co-citazioni, indica i predecessori in comune
- AA^T è la matrice dei co-link, indica i successori comuni

PRO:

- query specific
- funziona su grafi piccoli

CONTRO:

- spammabile sugli hub
- computazione complessa

TECNICHE DI SPAM E ANTI-SPAM

- TERM SPAM
- CLOACKING: pagine false create per ingannare i SE
- DOORWAY PAGES: pagine ottimizzate su alcuni termini che rimandano alla pagina vera
- MUTUAL LINK
- HIDDEN LINKS
- FAKE CLICKS
- FAKE QUERIES

Per l'anti-spam si possono implementare LAR più robusti, classificatori e test anti robot

CRAWLING

Naviga il web per raccogliere i dati da indicizzare

In un SE è fondamentale per avere una collezione di qualità e aggiornata

Due tipologie:

- BATCH (periodici): continuano fino al raggiungimento di un criterio

- INCREMENTAL (continuo): anche se il criterio è raggiunto, continua la navigazione sulla collezione aggiornandola, rimuovendo i duplicati...

Come fare il crawl? Quanto? Quanto spesso?

Molteplici fattori:

- PAGE: fare prima il crawl delle pagine cambiate e che cambiano più spesso
- CRAWLER LOAD: considerare prima l'età dell'indice, le pagine più piccole e che si trovano sui server più veloci
- SE: aggiornare prima le pagine più recuperate e con alto PageRank
- USER: aggiornare le pagine più visitate
- SERVER LOAD: evitare di sovraccaricare i server e di visitare lo stesso troppo spesso