Matteo Laporchio, 491283

Ex.1

$S = (2,5,6,11,14,16,20,38)$ $n=8$ $m=2^6$

2	000010
5	000101
6	000110
11	001011
14	001110
16	010000
20	010100
38	100110

$W = \lceil \log_2(m/n) \rceil = \lceil \log_2(2^6/2^3) \rceil = 3$

$Z = \log_2 n = \log_2 8 = 3$

concateno gli $8 \cdot W$ bit

$L = 010\,101\,110\,011\,110\,000\,100\,110$

Configuration	#occurrences
000	3
001	2
010	2
011	0
100	1
101	0
110	0
111	0

codifica in unario e concateno

$H = 1110110110010000$

```
BinSearch(x, left, right):
(*) → c = ⌈(right - left)/2⌉
    K = elemento centrale della sequenza, ottenuto concatenando il c-esimo gruppo
        di W bit in L con la codifica binaria su z bit di Select_1(H, c) - c.
    if (x == K) return c
    if (x > K) retum BinSearch(x, c+1, r)
    return BinSearch(x, l, c-1)
```

~~Posso costruire tutta la sequenza S applicando ripetutamente l'algoritmo precedente. Usando Select~~

(*) Se l'array ha un solo elemento, ovvero se right-left = 0 allora ricavo il primo elemento della sequenza S e lo confronto con x. Per ricavarlo, basta effettuare la concatenazione fra il primo gruppo di W bit in L e la codifica binaria su Z bit di $Select_1(H, 1) - 1$.

Ex. 2

La tecnica dello shingling può essere utilizzata per individuare i documenti che sono near-duplicates (quasi duplicati). Se ho due documenti D_1 e D_2 posso trasformarli in insiemi di shingle (o q-grams) ovvero gruppi di q parole consecutive ottenuti da ogni parola nel documento. Una volta ottenuti i due insiemi di shingle S_1 e S_2 da D_1 e D_2 posso usare una funzione di tipo rolling hash (come quella del metodo di Karp e Rabin) che trasformi gli shingle in valori hash (cioè interi). La similarità fra i due documenti è ora una similarità fra due insiemi di interi. Su tali insiemi è possibile usare la tecnica del MIN-HASHING che consente di ottenere una stima della similarità fra i due insiemi (ola, intendersi come Jaccard similarity):

H_1, H_2 = insiemi contenenti gli hash degli shingle di D_1 e D_2

↓ permuto gli elementi secondo una permutazione casuale Π

$\Pi(H_1), \Pi(H_2)$ sono i due insiemi permutati: detti m_1 e m_2 i loro minimi si ha che: $P(m_1 = m_2) = J(H_1, H_2)$

Ripetendo questo procedimento K volte ottengo K minimi per S_1 e K minimi per S_2. Assumo di utilizzare ogni volta una diversa permutazione casuale Π. Ottengo due vettori con i K minimi: $V_1 = (m_{11}, m_{12}, \dots, m_{1K})$ e $V_2 = (m_{21}, m_{22}, \dots, m_{2K})$. Confrontando il numero di componenti condivise su un totale di K ottengo una stima di quanto siano simili i documenti di partenza. Invece di confrontare tante componenti ~~(i documenti originali, se si considerano come vettori binari)~~ ne confronto solamente K. I documenti originali, se interpretati come vettori binari aventi tante componenti quanti i termini nel dizionario, possono avere anche milioni di componenti.

d1 = "the box is red and large"

S1 = {<the box is>, <box is red>, <is red and>, <red and large>}

d2 = "the large box is red"

S2 = {<the large box>, <large box is>, <box is red>}

d3 = "the fox is red"

S3 = {<the fox is>, <fox is red>}

Assumiamo che applicando le funzioni hash agli shingles si ottengano:

$H_1 = \{2, 3, 5, 7\}$ $H_2 = \{1, 4, 3\}$ $H_3 = \{6, 8\}$

Applico ora le due permutazioni $\Pi_1(x) = 2x+1 \bmod 11$ e $\Pi_2(x) = 3x+3 \bmod 11$ ottenendo:

1) con Π_1: $\Pi_1(H_1) = \{5, 7, 0, 4\}$ $\Pi_1(H_2) = \{3, 9, 7\}$ $\Pi_1(H_3) = \{2, 6\}$
2) con Π_2: $\Pi_2(H_1) = \{9, 1, 7, 2\}$ $\Pi_2(H_2) = \{6, 4, 1\}$ $\Pi_2(H_3) = \{10, 5\}$

Osservo che con Π_1 la similarità (seppur minima, poiché si ha un solo elemento comune) tra H_1 e H_2 non viene catturata, poiché $\min \Pi_1(H_1) \neq \min \Pi_1(H_2)$ nel secondo caso invece sì, poiché $\min \Pi_2(H_1) = 1 = \min \Pi_2(H_2)$.

$J(H_1, H_2) = \frac{1}{6}$ $J(H_2, H_3) = 0$ $J(H_1, H_3) = 0$

Ex. 4

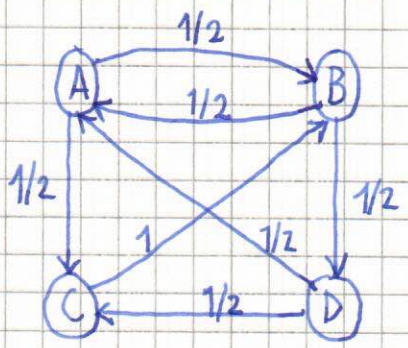
La champion list di ciascun termine contiene i top-k documenti per quel termine. In una AND query si vogliono recuperare i documenti che contengono entrambi i termini in AND. Avendo a disposizione le champion list per ognuno dei termini, si potrebbe effettuare un'intersezione delle due champion list, . In tal caso però, se volessimo restituire K risultati (i top-k) avremmo, almeno in linea teorica, bisogno di champion list per i due termini di partenza che contengano un numero $K' > k$ di top documents, poiché l'intersezione di due liste ha sempre cardinalità minore o uguale alla più piccola delle due liste di partenza.

FANCY HITS: questa tecnica permette di organizzare in modo più efficiente le liste contenenti i documenti relativi ad un dato termine. Si comincia assegnando i document ID (ovvero interi che identificano un documento) per page rank decrescente: più è basso il document ID, più è alto il pagerank di quel documento. In questo modo la lista può essere organizzata come segue: i primi K identificativi corrispondono a quelli dei top-k documenti e sono ordinati per page rank decrescente. Questo gruppo di ID costituisce le fancy hits per il termine corrispondente.

Per calcolare l'intersezione di due liste nel caso di una AND query è sufficiente scorrerle partendo dalla cima e aggiungendo al risultato gli elementi condivisi come nella procedura di merge dell'algoritmo mergesort.

Ex. 3

Matteo Loperchio, 491283



$$r(i) = \alpha \sum_{j \in B(i)} \frac{r(j)}{\#out(j)} + (1-\alpha)\frac{1}{N} \qquad N=4,\ 1-\alpha = \frac{1}{4} \Rightarrow \alpha = \frac{3}{4}$$

Uniform starting probability

1)
$$r(A) = \frac{3}{4}\left(\frac{1}{4}\cdot\frac{1}{2} + \frac{1}{4}\cdot\frac{1}{2}\right) + \frac{1}{4}\cdot\frac{1}{4} = \frac{3}{4}\cdot\frac{1}{4} + \frac{1}{16} = \frac{1}{4}$$

$$r(B) = \frac{3}{4}\left(\frac{1}{4}\cdot\frac{1}{2} + \frac{1}{4}\cdot 1\right) + \frac{1}{4}\cdot\frac{1}{4} = \frac{3}{4}\cdot\frac{3}{8} + \frac{1}{16} = \frac{11}{32}$$

$$r(C) = \frac{3}{4}\left(\frac{1}{4}\cdot\frac{1}{2} + \frac{1}{4}\cdot\frac{1}{2}\right) + \frac{1}{4}\cdot\frac{1}{4} = \frac{1}{4}$$

$$r(D) = \frac{3}{4}\left(\frac{1}{4}\cdot\frac{1}{2}\right) + \frac{1}{4}\cdot\frac{1}{4} = \frac{3}{4}\cdot\frac{1}{8} + \frac{1}{16} = \frac{5}{32}$$

2)
$$r(A) = \frac{3}{4}\cdot\left(\frac{11}{32}\cdot\frac{1}{2} + \frac{5}{32}\cdot\frac{1}{2}\right) + \frac{1}{16} = \frac{3}{4}\cdot\frac{1}{4} + \frac{1}{16} = \frac{1}{4}$$

$$r(B) = \frac{3}{4}\left(\frac{1}{4}\cdot\frac{1}{2} + \frac{1}{4}\cdot 1\right) + \frac{1}{16} = \frac{11}{32}$$

$$r(C) = \frac{3}{4}\left(\frac{1}{4}\cdot\frac{1}{2} + \frac{5}{32}\cdot\frac{1}{2}\right) + \frac{1}{16} = \frac{3}{4}\cdot\frac{13}{64} + \frac{1}{16} = \frac{55}{256}$$

$$r(D) = \frac{3}{4}\left(\frac{11}{32}\cdot\frac{1}{2}\right) + \frac{1}{16} = \frac{3}{4}\cdot\frac{11}{64} + \frac{1}{16} = \frac{43}{256}$$

Se ogni nodo contiene del testo, si potrebbe assegnare un peso diverso a ciascun arco a seconda di quanto sono simili i testi dei due nodi collegati. In altre parole, invece di assegnare $\frac{1}{\#out(j)}$ nella matrice stocastica nel caso in cui vi sia un arco fra i e j, assegno un valore che tiene conto di $sim(T_i, T_j)$ dove T_i e T_j sono i due testi di i e j.