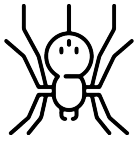
ALGORITMICA PER IL WEB

Luca Cappelletti
Prof. Sebastiano Vigna

6 CFU



2019
Informatica Magistrale
Università degli studi di Milano
Italia
2 febbraio 2020

Indice

1

Notazioni e definizioni di base

Definizione 1.1 | Prodotto cartesiano e proiezioni

Il prodotto cartesiano degli insiemi X e Y è l'insieme:

$$X \times Y = \{\langle x, y\rangle : x \in X \wedge y \in Y\}$$

delle coppie ordinate degli elementi di X e Y.

La definizione si estende per ricorsione a n insiemi: al prodotto cartesiano $X_1 \times X_2 \times \dots X_n$ sono naturalmente associate le proiezioni $\pi_1, \pi_2, \dots, \pi_n$ definite da:

$$\pi_i(\langle x_1, x_2, \dots, x_n\rangle) = x_i$$

Definizione 1.2 | Insieme con un solo elemento

La notazione per un insieme qualunque con un solo elemento è $X^0 = \{*\}$.

Definizione 1.3 | Potenza di insiemi

Definiamo la potenza n-esima di un insieme come:

$$X^n = \overbrace{X \times X \times \cdots \times X}^{n \text{ volte}}$$

Definizione 1.4 | Somma disgiunta

La **somma disgiunta** degli insiemi X e Y è un'unione d X e Y che però tiene separati gli insiemi comuni:

$$X + Y = X \times \{0\} \cup Y \times \{1\}$$

Definizione 1.5 | Relazione tra insiemi

Una **relazione** tra gli insiemi $X_1, X_2, \dots, X_n$ è un sottoinsieme R del prodotto cartesiano $X_1 \times X_2 \times \dots \times X_n$.

Definizione 1.6 | Relazione binaria

Una relazione tra due insiemi X e Y è detta binaria e si scrive come:

$$x\, R\, y \text{ per } \langle x, y\rangle \in R$$

X è detto il **dominio** di R ed è denotato come dom R mentre Y è detto **codominio** di R, ed è denotato da cod$R$.

Definizione 1.7 | Rango o Insieme di Definizione di una Relazione

Il **rango** o **insieme di definizione** di R è l'insieme ran$R = \{x \in X : \exists y \in Y, xRy\}$, e in generale può non coincidere col il dominio di R.

Definizione 1.8 | Immagine di una Relazione

L'**immagine** di R è l'insieme imm$R = \{y \in Y : \exists x \in X\, xRy\}$, e in generale può non coincidere con il codominio di R.

Definizione 1.9 | Relazione binaria monodroma

Una relazione binaria R tra X e Y è **monodroma** se per ogni $x \in X$ esiste al più un $y \in Y$ tale che xRy.

Definizione 1.10 | Relazione binaria totale

Una relazione binaria R tra X e Y è **totale** se per ogni $x \in X$ esiste un $y \in Y$ tale che xRy, cioè se ran$R$ == dom$R$.

Definizione 1.11 | Relazione binaria iniettiva

Una relazione binaria R tra X e Y è **iniettiva** se per ogni $y \in Y$ esiste al più un $x \in X$ tale che xRy.

Definizione 1.12 | Relazione binaria suriettiva

Una relazione binaria R tra X e Y è **suriettiva** se per ogni $y \in Y$ esiste un $x \in X$ tale che xRy, cioè se imm$R$ == cod$R$.

Definizione 1.13 | Relazione binaria biiettiva

Una relazione binaria R tra X e Y è **biiettiva** se risulta sia **iniettiva** che **suriettiva**.

Definizione 1.14 | Funzione tra insiemi

Una **funzione** da X e Y è una relazione **monodroma** e **totale** tra X e Y.

Definizione 1.15 | Monoide libero

Dato un insieme X, il **monoide libero** su X denotato da X^*, è l'insieme di tutte le sequenze finite di elementi di X, dette **parole** su X, dotate dell'operazione di concatenazione, di cui la parola vuota è l'elemento neutro.

Denoteremo con $|w|$ il numero di elementi di X della parola $w \in X$.

Definizione 1.16 | Funzione caratteristica

Dato un sotto-insieme A di X, possiamo associargli la sua **funzione caratteristica** $\chi_A : X \to 2$, definita da:

$$\chi_A(x) = \begin{cases} 0 & \text{se } x \notin A \\ 1 & \text{se } x \in A \end{cases}$$

Definizione 1.17 | Funzioni di ordine superiore e inferiore

Date due funzioni $f, g : \mathbb{N} \to \mathbb{R}$, diremo che f è di ordine non superiore a g e scriveremo $f \in O(g)$ se esiste una costante $\alpha \in \mathbb{R}$ tale che $|f(n)| \leqslant |\alpha g(n)|$ definitivamente. Diremo che f è di ordine non inferiore a g, e scriveremo $f \in \Omega(g)$ se $g \in O(f)$. Diremo che f è dello stesso ordine di g, e scriveremo $f \in \Theta(g)$, se $f \in O(g)$ e $g \in O(f)$.

Definizione 1.18 | Cricca (Clique)

Una **cricca** di un grafo G è un insieme di vertici $C \subseteq V_G$ mutuamente adiacenti. Il suo duale è un **insieme indipendente**.

Definizione 1.19 | Insieme indipendente

Un **insieme indipendente** di un grafo G è un insieme di vertici $I \subseteq V_G$ mutuamente non adiacenti.

Definizione 1.20 | Cammino

Un **cammino** in un grafo G è una sequenza di vertici $x_0, x_1, \ldots, x_n$ tale che x_i è adiacente a x_{i+1}. Diremo che i cammino va da x_0 a x_n.

Definizione 1.21 | Archi paralleli

Due archi a e b di un grafo G tali che $s_G(a) = s_G(b)$ e $t_G(a) = t_G(b)$ sono detti **paralleli.**

Definizione 1.22 | Grafo separato

Un grafo senza archi paralleli è detto **separato**.

Definizione 1.23 | Grado positivo (outdegree)

Il **grado positivo** $d^+(x)$ di un nodo x è il numero di archi che partono da x.

Definizione 1.24 | Grado negativo (indegree)

Il **grado negativo** $d^-(x)$ di un nodo x è il numero di archi che arrivano in x.

Osservazione 1.1 | Quali grafi consideriamo?

Consideriamo grafi diretti definiti da un insieme N di n nodi ed un insieme $A \subseteq N \times N$ di archi.

Definizione 1.25 | Grafo trasposto

Un grafo trasposto è ottenuto invertendo le direzioni di tutti gli archi.

Definizione 1.26 | Grafo simmetrico

Un grafo simmetrico è un grafo tale che se $x \to y$ allora $y \to x$: risulta invariante alla trasposizione e può essere identificato con un grafo indiretto.

Definizione 1.27 | Successore

Un successore di un nodo $x \in A$ è un nodo $y \in A$ tale che $x \to y$.

Definizione 1.28 | Predecessore

Un predecessore di un nodo $y \in A$ è un nodo $x \in A$ tale che $x \to y$.

Definizione 1.29 | Grado uscente (outdegree)

Il grado uscente $d^+(x)$ di un nodo $x \in A$ è il numero dei suoi successori.

Definizione 1.30 | Grado entrante (indegree)

Il grado entrante $d^-(y)$ di un nodo $y \in A$ è il numero dei suoi predecessori.

Definizione 1.31 | Percorso (path)

Un percorso di lunghezza k è una sequenza $x_0, x_1, \ldots, x_{k-1}$ dove $x_j \to x_{j+1}$

Definizione 1.32 | Cammino (walk)

Un cammino di lunghezza k è una sequenza $x_0, x_1, \ldots, x_{k-1}$ dove $x_j \to x_{j+1}$ o $x_{j+1} \to x_j$.

Definizione 1.33 | Componente connessa

Una componente connessa di un grafo è un sottoinsieme massimale di nodi in cui ogni coppia di nodi è connessa da un cammino. Le componenti formano una partizione dei nodi di un grafo.

Definizione 1.34 | Componente fortemente connessa

Una componente fortemente connessa di un grafo è un sottoinsieme massimale di nodi in cui ogni coppia di nodi è connessa da un percorso.

Definizione 1.35 | Componente fortemente connessa terminale

Una componente fortemente connessa è **terminale** se i suoi nodi non hanno archi verso altre componenti.

Definizione 1.36 | Grafo connesso

Un grafo è connesso se esiste una sola componente connessa, cioè per ogni possibile coppia di nodi $x, y \in A$ esiste un cammino da x ad y.

Definizione 1.37 | Grafo fortemente connesso

Un grafo è fortemente connesso se esiste una sola componente connessa, cioè per ogni possibile coppia di nodi $x, y \in A$ esiste un percorso da x ad y.

Definizione 1.38 | Distanza tra nodi

La distanza $d(x, y)$ da x a y è il percorso di lunghezza minima da x a y. Se non esiste un percorso del genere, la distanza è ∞.

Definizione 1.39 | Nodi raggiungibili

I nodi **raggiungibili** da un nodo x sono i nodi y tali che $d(x, y) < \infty$.

Definizione 1.40 | Nodi coraggiungibili

I nodi **coraggiungibili** da un nodo y sono i nodi x tali che $d(y, x) < \infty$.

Definizione 1.41 | Matrice associata stocastica

Data una matrice non-negativa associata a un grafo A, indichiamo con $\overline{A}$ la matrice normalizzata in norma ℓ_1 lungo le righe dividendo ogni elemento di una riga per la somma della riga. Se non vi sono righe nulle, la matrice $\overline{A}$ ottenuta è **stocastica**.

Definizione 1.42 | Notazione di Iverson

Se P è un predicato, $[P]$ vale 0 se P è falso e vale 1 se P è vero.

Definizione 1.43 | i-esimo numero armonico

Denotiamo con H_i l'i-esimo numero armonico:

$$H_i = \sum_{1 \leqslant k \leqslant i} \frac{1}{k}$$

Definizione 1.44 | Vettore caratteristico di i

Indichiamo il vettore caratteristico di i con $\chi_i(j) = [j = i]$.

1.1 Misure geometriche

Definizione 1.45 | Misure geometriche

Chiamiamo **misure geometriche** quelle misure che assumano che l'importanza sia una funzione della distanza e che una centralità geometrica dipenda unicamente da quanti nodi esistono ad ogni data distanza.

Definizione 1.46 | Grado entrante (indegree) come metrica

Il numero di archi entranti $d^-(x)$ può essere considerata una misura geometrica: si tratta semplicemente del numero di nodi a distanza uno.

Definizione 1.47 | Vicinanza (closeness) di Bavelas

La vicinanza è definita come:

$$\frac{1}{\sum_y d(y,x)}$$

L'idea è che i nodi più centrali hanno distanze minori, ma richiede che il grafo sia fortemente connesso. Senza questa condizione alcuni denominatori saranno ∞, ottenendo quindi un valore nullo per tutti i nodi che non coraggiungono tutto il grafo. Una possibile soluzione, che però introduce un forte bias verso l'insieme dei nodi raggiungibili, è ottenibile rimuovendo le distanze infinite.

$$\frac{1}{\sum_{d(y,x)<\infty} d(y,x)}$$

Definizione 1.48 | Indice di Lin

Si tratta di un tentativo per migliorare la definizione di vicinanza pesando la distanza corretta di Bavelas per il quadrato del numero dei nodi coraggiungibili.

$$\frac{|\{y|d(y,x)<\infty\}|^2}{\sum_{d(y,x)<\infty} d(y,x)}$$

Questa modifica normalizza la vicinanza per ogni nodo del grafo.

Osservazione 1.2 | Quale è il problema principale nel determinare la vicinanza?

Il problema principale nell'individuare una metrica di vicinanza è la presenza di coppie di nodi che non sono raggiungibili l'uno dall'altro.

Definizione 1.49 | Centralità armonica

Definiamo la **centralità armonica** di un nodo x come:

$$\sum_{y\neq x} \frac{1}{d(y,x)} = \sum_{d(y,x)<\infty, y\neq x} \frac{1}{d(y,x)}$$

La centralità armonica è fortemente correlata alla vicinanza in reti semplici, ma inoltre supporta anche le coppie di nodi non raggiungibili. Pertanto è possibile applicarla con successo anche a grafi che non sono fortemente connessi.

1.2 Misure spettrali

Definizione 1.50 | Misure spettrali

Le **misure spettrali** calcolano l'autovettore sinistro dominante di qualche matrice derivata dal grafo e in base a come la matrice è modificata prima del calcolo possiamo ottenere misure diverse.

Definizione 1.51 | Autovettore dominante sinistro

L'autovettore sinistro dominante può essere pensato come il punto fisso di una computazione iterata in cui ogni nodo inizia con lo stesso punteggio, e quindi aggiorna il proprio valore con la somma dei punteggi dei suoi predecessori. Il vettore è quindi normalizzato ed il processo si ripete sino alla convergenza.

Gli autovettori dominanti non si comportano come atteso su grafi che non sono fortemente connessi: in base all'autovettore dominante della componente fortemente connessa, l'autovettore dominante del grafo potrebbe (o non) essere non-zero sulle componenti non terminali.

Definizione 1.52 | Quale è l'idea dell'indice di Seeley?

L'idea alla base dell'autovettore dominante può essere leggermente corretta osservando che la regola di aggiornamento che abbiamo descritto può essere vista come se ogni nodo desse via il proprio punteggio, come una reputazione, ai propri successori.

Sembra più sensato dividere equamente la reputazione attraverso tutti i nodi successori: questo coincide con normalizzare ogni riga della matrice di incidenza in norma ℓ_1.

La matrice ottenuta è stocastica, quindi il punteggio può essere interpretato come lo stato stazionario di una catena di Markov.

Osservazione 1.3 | Indice di Katz

L'indice di Katz è una sommatoria pesata su tutti i cammini che giungono a un nodo, in modo tale che la sommatoria di ogni nodo risulti finita.

$$\boldsymbol{k} = \mathbf{1} \sum_{i=0}^{\infty} \beta^i A^i$$

Perché la sommatoria sia finita, il **fattore di attenuazione** β deve risultare più piccolo di $1/\lambda$, dove λ è un autovettore dominante di A.

Il limite normalizzato dell'indice di Katz per $\beta \to 1/\lambda$ è un autovettore dominante. Se quest'ultimo non fosse unico, il limite dipende dal vettore di preferenze $\underline{\boldsymbol{v}}$, che va a sostituire $\mathbf{1}$.

2

Modelli locali

Osservazione 2.1 | Cosa cambia tra modelli globali e locali?

Nei modelli globali ogni **inverted file entry** è compresso in un unico modello, e questo funziona bene assumendo una distribuzione uniforme degli intervalli (gap). La distribuzione degli intervalli, però, dipende da come viene ordinata la lista: se ordiniamo alfabeticamente una lista di URL quelli dello stesso sito saranno raggruppati e avremo una tendenza di clustering della distribuzione.

Per poter sfruttare questa proprietà vengono usati i modelli locali, dove ogni documento utilizza un proprio modello. In generale i modelli locali risultano migliori di quelli globali, ma sono più complessi da implementare.

Definizione 2.1 | Modello locale iperbolico

Il **modello locale iperbolico** assume che la probabilità di un intervallo di dimensione i è proporzionale a $1/i$:

$$\Pr(i) \sim \frac{1}{i}, \text{ for } i = 1,2,\ldots,m$$

Sia m il punto troncato della dimensione dell'intervallo:

$$\Pr(i) = \frac{1}{i \sum_{j=1}^{m} \frac{1}{j}} \approx \frac{1}{i \times \log_e^m}$$

Possiamo determinare il valore di m come:

$$\begin{aligned} E[\text{ gap size }] &= \sum i \times \Pr(i) \\ &\approx \sum i \times \frac{1}{i \times \log_e^m} \\ \frac{m}{\log_e^m} &= \frac{N}{f_t} \end{aligned}$$

dove N è il numero dei documenti mentre f_t è la frequenza dei termini.

3

Filtri di Bloom

Definizione 3.1 | Filtro di Bloom

Un filtro di Bloom è una struttura dati probabilistica che rappresenta un insieme: permette di aggiungervi elementi e chiedere se un elemento vi appartenga o no, con il **rischio di ottenere falsi positivi**.

Osservazione 3.1 | Cosa compone un filtro di Bloom?

Un filtro di Bloom per un insieme X è rappresentato da un vettore $\underline{\boldsymbol{b}}$ di m bit e da d funzioni di **hash** $f_0, f_1, \ldots, f_{d-1}$ da X in m.

Osservazione 3.2 | Come si aggiunge un elemento a un filtro di Bloom?

Per aggiungere un elemento di $x \in X$ a un filtro di Bloom, vengono posti a uno i bit $b_{f_k(x)}, (0 \leqslant k < d)$.

Osservazione 3.3 | Come si verifica se un elemento appartiene al filtro?

Per sapere se un elemento appartiene al filtro si controlla se tutti i bit $b_{f_k(x)}, (0 \leqslant k < d)$ sono a uno, e solo in questo caso si risponde positivamente.

Osservazione 3.4 | Come mai "filtri" di Bloom?

Il nome "filtro" deriva dall'idea che la struttura dovrebbe venire usata per filtrare le richieste a una secondaria soggiacente più lenta. Quando si prevede che la maggior parte delle richieste avrà risposta negativa, un filtro di Bloom può ridurre significativamente gli accessi alla struttura soggiacente.

Analisi 3.1 | Probabilità di un falso positivo in un filtro di Bloom dopo n inserimenti.

La probabilità che dopo n inserimenti uno specifico bit sia zero è:

$$\left(1 - \frac{1}{m}\right)^{dn}$$

dato che la probabilità che un singolo bit sia zero (se ne imposto uno a caso) è $1 - \frac{1}{m}$ e assumiamo che le funzioni di hash siano uniformemente distributive e perfettamente aleatorie. Per ottenere un positivo, dobbiamo trovare uno in tutti i punti del vettore che controlliamo e questo avviene con probabilità:

$$\varphi = \left(1 - \left(1 - \frac{1}{m}\right)^{dn}\right)^d \approx \left(1 - e^{-\frac{dn}{m}}\right)^d$$

dove abbiamo utilizzato il fatto che $(1 + \frac{\alpha}{n})^n \to e^\alpha$ per $n \to \infty$.

Poniamo ora $p = e^{-\frac{nd}{m}}$, sicché $d = -(\frac{m}{n}) \ln p$. Il nostro scopo è quindi quello di minimizzare:

$$(1-p)^{-(m/n)\ln p} = e^{-(m/n)\ln p \ln(1-p)}$$

La derivata prima è dunque:

$$-\frac{m}{n} e^{-(m/n)\ln p \ln(1-p)} \left(\frac{\ln(1-p)}{p} - \frac{\ln p}{1-p}\right)$$

e si azzera quando:

$$(1-p)\ln(1-p) = p \ln p$$

Dato che sia a sinistra che a destra abbiamo la stessa funzione $x \ln x$, una soluzione è certamente data da $1 - p = p$, cioè $p = \frac{1}{2}$.

Concludiamo che la probabilità di (falsi) positivi è minimizzata da $d \equiv \frac{m}{n} \ln 2$ e in tal caso la probabilità di un (falso) positivo è 2^{-d}. Vale a dire, possiamo migliorare esponenzialmente la probabilità di errore aumentando linearmente il numero di funzioni di **hash** e il numero di bit a $m \equiv \frac{dn}{\ln 2} \approx 1.44 dn$.

4

Tecniche di distribuzione del carico

Proprietà 4.1 | Proprietà di Controvarianza

La proprietà di Controvarianza afferma che aumentando l'insieme degli agenti, gli insiemi di URL associati agli agenti preesistenti si riducono. In particolare, se si aggiunge un agente agli agenti preesistenti non si vedono assegnare nessun nuovo URL.

$$\delta_B^{-1}(a) \subseteq \delta_A^{-1}(a) \text{ se } B \supseteq A, a \in A$$

Definizione 4.1 | Permutazioni aleatorie

Il primo approccio per realizzare una funzione bilanciata e controvariante è assumere che ci sia un universo P di possibili agenti. A ogni istante dato l'insieme di agenti effettivamente utili è un insieme $A \subseteq P$.

Dato un URL u, inizializziamo un generatore di numeri pseudocausali utilizzando u e utilizziamolo per generare una permutazione aleatoria di P scelta in maniera uniforme. A questo punto, l'agente scelto è il primo agente in A nell'ordine indotto dalla permutazione così calcolata.

È possibile generare una tale permutazione in tempo e spazio lineare in $|P|$: la tecnica è nota come **Knuth shuffle** o **Fisher-Yates shuffle**.

Definizione 4.2 | Knuth shuffle o Fisher-Yates shuffle

La tecnica consiste nell'inizializzare un vettore di $|P|$ elementi con i primi P numeri naturali. A questo punto si eseguono $|P|$ iterazioni: all'iterazione di indice i (a partire da zero) si scambia l'elemento del vettore di posto i con uno dei seguenti $|P| - i$ scelto a caso uniformemente (si noti che i stesso è una scelta possibile). Alla fine dell'algoritmo, il vettore contiene una permutazione aleatoria scelta in maniera uniforme.

Definizione 4.3 | Min Hashing

Questo approccio non richiede che l'universo sia predeterminato o ordinato: fissiamo una funzione di **hash** aleatoria h che prende due argomenti: un URL ed un agente. L'agente all'interno di A responsabile dell'URL u è quello che realizza il minimo tra tutti i valori $h(a,u), a \in A$. Nuovamente, assumendo che h sia aleatoria, il bilanciamento è banale, ma anche la proprietà di controvarianza è elementare: se $B \supseteq A$, gli unici URL che cambiano assegnamento sono quegli URL u per cui esiste un $b \in B \setminus A$ tale che $h(b,u) < h(a,b)$ per ogni $a \in A$.

Definizione 4.4 | Hashing coerente

Consideriamo idealmente una circonferenza di lunghezza unitaria e una funzione di **hash** aleatoria h che mappa gli URL nell'intervallo da zero (compreso) a uno. Fissiamo inoltre un generatore di numeri pseudocasuali.

Per ogni agente $a \in A$, utilizziamo a per inizializzare il generatore e scegliere C posizioni pseudoaleatorie sul cerchio: queste saranno le repliche dell'agente a. A questo punto per trovare l'agente responsabile di un URL u partiamo dalla posizione $h(u)$ e proseguiamo in senso orario finché non troviamo una replica: l'agente associato è quello responsabile per u.

In pratica, il cerchio viene diviso in segmenti massimali che non contengono repliche. Associamo a ogni segmento la replica successiva in senso orario e tutti gli URL mappati sul segmento avranno come agente responsabile quello associato alla replica. La funzione così definita è ovviamente controvariante. L'effetto di aggiungere un nuovo agente è semplicemente quello di spezzare tramite le nuove repliche i segmenti preesistenti: la prima parte verrà mappata sul nuovo agente, mentre la seconda continuerà a essere mappata sull'agente precedente.

La parte delicata è in questo caso il bilanciamento: se C viene scelto sufficientemente grande i segmenti risultano così piccoli da poter dimostrare che la funzione è bilanciata con alta probabilità.

La struttura può essere realizzata in maniera interamente discreta memorizzando interi che rappresentano le posizioni delle repliche in un dizionario ordinato, come un albero binario bilanciato. A questo punto dato un URL u è sufficiente generare uno hash intero e trovare il minimo maggiorante presente nel dizionario (eventualmente debordando alla fine dell'insieme e restituendo quindi il primo elemento).

Questo metodo richiede quindi spazio lineare in $|A|$ e tempo logaritmico in $|A|$. Aggiungere o togliere un elemento ad A richiede, contrariamente ai casi precedenti, tempo logaritmico in $|A|$.

Definizione 4.5 | Collisioni

Sia il min-hashing sia lo hashing coerente possono presentare collisioni, situazioni in cui non sappiamo come scegliere l'output perché due minimi o due repliche coincidono. In tal caso è necessario disambiguare deterministicamente i risultato, per esempio imponendo un ordine arbitrario sugli agenti e restituendo il minimo.

5

Codici Istantanei

Definizione 5.1 | Codice

Un **codice** è un insieme $c \subseteq 2^*$, cioè un insieme di parole binarie.

Definizione 5.2 | Ordinamento per prefissi

L'**ordinamento per prefissi** delle sequenze in 2^* come segue:

$$x \preceq y \iff \exists z \quad y = xz$$

Vale a dire, $x \preceq y$ se e solo se un prefisso di y coincide con x.

Osservazione 5.1 | Elementi inconfrontabili

In un ordine parziale due elementi x e y sono **inconfrontabili** se nessuno dei due è minore o uguale all'altro.

Definizione 5.3 | Codice istantaneo

Un codice è **istantaneo** o **a decodifica istantanea** o **non ambiguo** o **privo di prefissi** se ogni coppia di parole distinte del codice è inconfrontabile.

Osservazione 5.2 | Codici istantanei nella pratica

L'effetto pratico di questa proprietà è che a fronte di una parola w formata da una concatenazione di parole nel codice non esiste una diversa concatenazione che dà w.

In particolare, leggendo uno a uno i bit di w è possibile ottenere in maniera istantanea le parole del codice che lo compongono.

Definizione 5.4 | Codice completo

Un codice **completo** o **non ridondante** se per ogni parola $w \in 2^*$ è confrontabile con qualche parola del codice.

Quando un codice istantaneo è completo, non è possibile più aggiungere parole al codice senza violare l'istantaneità. Inoltre, qualunque parola **infinita** è decomponibile in maniera unica come sequenza di parole del codice e qualunque parola **finita** è decomponibile in maniera unica come sequenza di parole del codice più prefisso di qualche parola del codice.

Definizione 5.5 | Diadico

Un **diadico** è un razionale della forma $k2^{-h}$.

5.1 Disuguaglianza di Kraft-McMillan

Teorema 5.1 | Disuguaglianza di Kraft-McMillan

Sia $C \subseteq 2^*$ un codice. Se C è istantaneo, allora:

$$\sum_{w \in C} 2^{-|w|} \leqslant 1$$

e C è completo se e solo se vale l'uguaglianza. Inoltre, data una sequenza (eventualmente infinita) $t_0, t_1, \ldots, t_{n-1}, \ldots$ che soddisfa:

$$\sum_{n} 2^{-t_n} \leqslant 1$$

esiste un codice istantaneo formato da parole $w_0, w_1, \ldots, w_{n-1}, \ldots$ tali che $|w_i| = t_i$.

Dimostrazione 5.1 | Disuguaglianza di Kraft-McMillan

A ogni parola $w \in 2^*$ associamo un intervallo con estremi diadici:

1. se w è la parola vuota, l'intervallo è $[0 \ldots 1)$
2. altrimenti, ricorsivamente, consideriamo l'intervallo della sottostringa $s = w_1 \ldots w_{|w|-1}$ pari a $[x \ldots y)$. Allora l'intervallo associato a w risulta pari a:

$$\begin{cases} [x \ldots \frac{x+y}{2}) & \text{se } w_{|w|} = 0 \\ [\frac{x+y}{2} \ldots y) & \text{se } w_{|w|} = 1 \end{cases}$$

In particolare vale che:

1. L'intervallo associato a una parola di lunghezza n ha lunghezza 2^{-n}.
2. Se s è una sotto-stringa di w l'intervallo associato a s contiene quello associato a w.
3. Due parole sono inconfrontabili se e solo se i corrispondenti intervalli sono disgiunti (e pertanto non sono sotto-stringhe l'una dell'altra).
4. Per ogni parola w esiste solo e soltanto un intervallo diadico tale che $\left[k2^{-|w|} \ldots (k+1)2^{-|w|}\right)$.

Sia ora C un codice istantaneo: gli intervalli associati alle parole (inconfrontabili) $w \in C$, sono disgiunti e per costruzione la sommatoria delle loro lunghezze è minore di uno.

Se la **sommatoria è strettamente minore di uno**, deve esserci un intervallo scoperto, diciamo $[x \ldots y)$. Questo intervallo contiene necessariamente un sotto-intervallo della forma $[k2^{-h} \ldots (k+1)2^{-h})$ per qualche h e k. Ma allora la parola associata a quest'ultimo potrebbe essere aggiunta al codice (essendo inconfrontabile con tutte le altre), che quindi risulta incompleto.

D'altra parte, se il codice è **incompleto** l'intervallo corrispondente ad una parola inconfrontabile con quelle del codice è necessariamente scoperto, e questo rende la somma strettamente minore di 1.

Assumendo senza perdite di generalità che la sequenza $t_0, t_1, \ldots, t_{n-1}, \ldots$ sia monotona non decrescente, possiamo sempre generare un codice istantaneo corrispondente: partendo dall'estremo sinistro d dell'intervallo, inizialmente $d = 0$.

A ogni iterazione i-esima generiamo la parola che ha associato l'intervallo di lunghezza t_i con estremo sinistro d: dato che l'intervallo associato è disgiunto dai precedenti le parole saranno inconfrontabili. Quindi aggiorniamo il valore di di d sommando 2^{-t_i}.

5.2 Codici istantanei per gli interi

Definizione 5.6 | Rappresentazione binaria ridotta

Con **rappresentazione binaria ridotta** di x si intende la rappresentazione binaria di $x+1$ privata del bit più significativo.

Definizione 5.7 | Codici binari minimali

I **codici binari minimali** sono codici istantanei per i primi k interi che utilizzano un numero variabile di bit.

Definizione 5.8 | Codice unario

Il **codice unario** rappresenta il naturale x tramite x zeri seguiti da un uno. Il codice unario è **istantaneo** e **completo**. Le prime parole del codice sono 1, 01, 001 e 0001.

Definizione 5.9 | Codice Gamma

Il codice Gamma, o γ, descrive un intero x scrivendo il numero di bit della rappresentazione binaria ridotta di x in unario, seguito dalla rappresentazione binaria ridotta di x. Il fatto che il codice sia istantaneo e completo si ottiene immediatamente dal fatto che lo è il codice unario. Le prime parole del codice sono 1, 010, 011, 00100, 00110 e 00111.

Definizione 5.10 | Codice Delta

Il codice Delta, o δ, descrive un intero x scrivendo il numero di bit della rappresentazione binaria ridotta di x in γ, seguito dalla rappresentazione binaria ridotta x. Il fatto che il codice sia istantaneo e completo deriva dal fatto che lo è il codice γ. Le prime parole del codice sono 1, 0100, 0101, 01100, 01110, 01111, 00100000 e 00100001.

Definizione 5.11 | Codice di Golonb di modulo b

Il **Codice di Golomb di modulo** b descrive un intero x scrivendo il quoziente q della divisione di x per b in unario, seguito dal resto r in binario.

$$q = \frac{x}{b}$$
$$r = x - q \cdot b$$

Il fatto che il codice sia istantaneo e completo si ottiene immediatamente dal fatto che lo sono il codice unario e il binario minimale. Per $b=3$ le prime parole del codice sono 10, 110, 111, 010, 0110, 0111.

Analisi 5.1 | Determinare b per il codice di Golomb

Per quanto il codice di Golomb è definito per qualsiasi intero b, i valori migliori risultano essere quelli che risultato in albero dei prefissi bilanciato. Procediamo ora a derivare il valore di b dalla probabilità p.

Dato un numero di estrazioni ℓ:

$$\begin{aligned}\sum_{x\geqslant\ell} \text{prob}[x] &= \sum_{x\geqslant\ell} p(1-p)^x - 1 \\ &= p(1-p)^{\ell-1} \sum_{x\geqslant 0} (1-p)^x \\ &= p(1-p)^{\ell-1}(1/p) \\ &= (1-p)^{\ell-1}\end{aligned}$$

Il parametro b è scelto in modo tale che la probabilità che meno di b estrazioni siano necessarie per essere il più vicino possibile all'evento tale per cui più di b estrazioni son necessarie.

Il parametro b deve essere l'intero che soddisfa quindi la seguente disuguaglianza:

$$(1-p)^b + (1-p)^{b+1} \leqslant 1 < (1-p)^{b-1} + (1-p)^b$$

Possiamo approssimare il valore come segue:

$$\begin{aligned}(1-p)^b &\approx \frac{1}{2} \\ b \log \frac{1}{1-p} &\approx \log 2 \\ b &\approx \frac{\log 2}{\log \frac{1}{1-p}}\end{aligned}$$

In particolare, per valori molto bassi di p, il parametro b può essere approssimato da $^{\log 2}/_{p}$

Definizione 5.12 | Codici a blocchi a lunghezza variabile (codice a nibble o a byte)

Nel codice variabile a nibble o a byte l'idea è che ogni parola è formata da un numero variabile di blocchi di k bit. Il primo bit, detto di **continuazione**, non fa parte dell'intero codificato, e serve a specificare (se diverso da zero) che il codice non termina in quel blocco. Un intero x viene scritto in notazione binaria, allineato a un multiplo di $k-1$ bit, diviso in blocchi di $k-1$ bit, e rappresentato tramite la sequenza dei suddetti blocchi, ciascuno preceduto da un bit di continuazione uguale 1, tranne nell'ultimo blocco. La lunghezza della parola di codice per x è quindi $\lceil (\log x + 1)/k \rceil (k+1)$. I codici a lunghezza variabile sono **istantanei** ma **non completi**, dato che, per esempio, una lista di blocchi di lunghezza maggiore di uno che contiene solo bit a zero (eccetto quelli di continuazione) risulta inconfrontabile con tutte le parole del codice.

Proprietà 5.1 | Codice universale

Un codice è detto **universale** se per qualunque distribuzione p sugli interi monotona non crescente il valore atteso della lunghezza di una parola rispetto a p è minore o uguale all'entropia di p a meno di costanti additive e moltiplicative (indipendenti da p).

Vale a dire, se $\ell(x)$ è la lunghezza della parola di codice per x e $H(p)$ denota l'entropia (nel senso di Shannon) di una distribuzione p esistono costanti c e d tali che per ogni distribuzione p monotona non crescente si ha:

$$\sum_{x\in\mathbf{N}} \ell(x)p(x) \leqslant cH(p) + d$$

Il codice unario e i codici di Golomb **non sono universali**, mentre lo sono γ e δ.

Proprietà 5.2 | Codice asintoticamente ottimo

Il codice è detto **asintoticamente ottimo** quando a destra il limite superiore è della forma:

$$f(H(p)) \quad \text{con} \quad \lim_{x\to\infty} f(x) = 1$$

Il codice δ è asintoticamente ottimo, mentre γ non lo è.

Definizione 5.13 | Codici allineati ai byte

I codici **allineati ai byte** o **byte-aligned codes** sono codici che costringono le lunghezze delle parole a essere multiple di un numero di base (tipicamente otto). In alcuni casi si tratta di codici per brevi sequenze di interi, che vengono memorizzate in una singola parola.

Osservazione 5.3 | Perché si usano i codici allineati ai byte?

La ragione principale per l'utilizzo di codici allineati è che la decodifica dovrebbe essere molto più rapida. Il prezzo da pagare è una compressione decisamente peggiore, dato che i codici risultanti non sono completi.

Definizione 5.14 | Codifica interpolativa

La **codifica interpolativa** è una codifica non istantanea che ha risultati di compressione eccellenti. L'idea è che se devo codificare un intero nell'intervallo $[a\ldots b)$, mi basta specificare lo scarto nell'intervallo tramite un codice binario minimale. Data una sequenza di interi $n_0, n_1, \ldots, n_{s-1}$ e un limite superiore noto b agli n_k, la codifica interpolativa scrive lo scarto di $n_{\lfloor s/2\rfloor}$ in $[0,\ldots,b)$ e procede ricorsivamente scrivendo la codifica degli interi $n_0, n_1, \ldots, n_{\lfloor s/2\rfloor} - 1$ in $[0..n_{\lfloor s/2\rfloor})$ e degli interi $n_{\lfloor s/2+1\rfloor}, n_{\lfloor s/2+2\rfloor}, \ldots, n_{s-1}$ in $[n_{\lfloor s/2\rfloor} + 1..b)$.

Esempio 5.1 | Esempio di codifica interpolativa

Dovendo codificare la sequenza $0,2,3$ e conoscendo il limite 4, scriveremo:

$$2(=2-0) \in [0\ldots5) \rightsquigarrow 01$$
$$0(=0-0) \in [0\ldots2) \rightsquigarrow 0$$
$$0(=3-3) \in [3\ldots4) \rightsquigarrow \varepsilon$$

Esempio 5.2 | Codifica interpolativa di intervalli vuoti

Siccome l'intervallo $[3\ldots4)$ contiene un solo intero, il codice binario minimale associato contiene solo la stringa vuota: la scrittura di 3 non consuma neppure un bit.

Osservazione 5.4 | Limiti della codifica interpolativa

La **codifica interpolativa** si presta però alle stesse critiche di quella aritmetica: deve essere scritta in un flusso di bit continuo. È quindi adatta solo per le posizioni, ma presenta lo stesso inconveniente della Golomb quando viene utilizzata per le posizioni, cioè occorre avere a disposizione la dimensione del documento come limite superiore.

Definizione 5.15 | PFOR-DELTA

Si sceglie una dimensione di blocco B, per esempio 128 o 256, e per ogni blocco consecutivo di B scarti si trova l'intero b tale che il 90% degli scarti sono esprimibili in 2^b bit.

A questo punto viene scritto un vettore di B interi di b bit, che descrive correttamente il 90% degli scarti. Il rimanente 10% degli scarti, le **eccezioni**, viene scritto separatamente in un vettore di interi (che deve essere di lunghezza sufficiente per descrivere il massimo scarto). Le posizioni del primo vettore corrispondenti alle eccezioni vengono riempite con un intero che rappresenta la distanza dalla prossima eccezione.

Talvolta può essere necessario aggiungere delle eccezioni fittizie per far sì che la distanza dalla prossima eccezione sia sempre esprimibile in b bit.

In fase di decodifica, si copiano i primi B valori a b bit in un vettore di interi. Quest'operazione è molto veloce, in particolare se vengono creati cicli srotolati diversi per ogni possibile valore di b. A questo punto si passa attraverso la lista delle eccezioni, che vengono copiate dal secondo vettore nelle posizioni corrette.

Questo approccio fa sì che il processore esegua dei cicli estremamente predicibili, e riesce a decodificare un numero di scarti al secondo significativamente più alto delle tecniche basate su codici.

Definizione 5.16 | Distribuzione intesa

Ogni codice istantaneo completo per gli indici interi definisce implicitamente una **distribuzione intesa** sugli interi che assegna a w la probabilità $2^{-|w|}$. Il codice risulta **ottimo** per la distribuzione associata.

La scelta di un codice istantaneo può essere ricondotta a considerazioni sulla distribuzione statistica degli interi che si intendono rappresentare.

Definizione 5.17 | Distribuzione intesa del codice unario

Al codice unario è associata una distribuzione esponenziale negativa con decrescita estremamente rapida:

$$\frac{1}{2^{x+1}}$$

Definizione 5.18 | Distribuzione intesa del codice gamma

$$\frac{1}{2^{2\lfloor \log(x+1) \rfloor + 1}} \approx \frac{1}{2(x+1)^2}$$

Definizione 5.19 | Distribuzione intesa del codice delta

$$\frac{1}{2(x+1)(\log(x+1)+1)^2}$$

Definizione 5.20 | Distribuzione intesa del codice di Golomb

Nel caso del codice di Golomb la distribuzione è dipendente dal modulo:

$$\frac{1}{b(\sqrt[b]{2})^x}$$

Definizione 5.21 | Distribuzione intesa dei codici a lunghezza variabile

Sebbene i codici di lunghezza variabile non siano completi, a meno di un fattore di normalizzazione è comunque possibile osservarne la distribuzione intesa:

$$\frac{1}{(x+1)^{1+\frac{1}{k}}}$$

Definizione 5.22 | Hapax legomena

Con **hapax legomena** si intende delle parole che in un set appaiono solo una volta.

Osservazione 5.5 | Quale codice è utilizzato per frequenze e conteggi?

Nel caso di frequenze e conteggi, la presenza significativa degli **hapax legomena** rende il codice gamma quello usato più di frequente.

Osservazione 5.6 | Quale codice è utilizzato per scarti tra puntatori a documenti?

Il **modello Bernoulliano** di distribuzione prevede che un termine con frequenza f appaia in una collezione di N documenti con probabilità $p = \frac{f}{N}$ indipendente in ogni documento. L'assunzione di indipendenza ha l'effetto di semplificare enormemente la distribuzione degli scarti, che risulta una geometrica di ragione p: lo scarto $x > 0$ compare cioè con probabilità:

$$p(1-p)^{(x-1)}$$

Siccome il **codice di Golomb** ha **distribuzione intesa esponenziale negativa** e un risultato importante di teoria dei codici dice che il codice ottimo per una geometrica di ragione p è un Golomb di modulo:

$$b = \left\lceil \frac{\log(2-p)}{\log(1-p)} \right\rceil$$

Osservazione 5.7 | Quali controindicazioni esistono nell'usare i codici di Golomb per scarti tra puntatori a documenti?

Una controindicazione può essere la **correlazione** tra documenti adiacenti: se i documenti nella collezione provengono da un **crawl** e sono nell'ordine di visita, è molto probabile che documenti vicini contengano termini simili.

Questo sposta significativamente da una geometrica la distribuzione degli scarti e richiede quindi soluzioni diverse.

Osservazione 5.8 | Quali codice è utilizzato per le posizioni?

Per le posizioni non esistono modelli noti e affidabili degli scarti e si utilizza quindi un codice dalle buone prestazioni generali come il δ. È possibile utilizzare anche Golomb, assumendo un modello Bernoulliano anche per le posizioni ma per calcolarlo è necessario avere la lunghezza del documento che quindi deve essere disponibile in memoria centrale.

Osservazione 5.9 | Quali sono i limiti di una lista compressa?

Non è possibile ottenere in tempo costante un elemento arbitrario di una lista compressa per scarti: è necessario decodificare gli elementi precedenti. Più problematica è l'impossibilità di saltare rapidamente al primo elemento della lista maggiore o uguale a un limite inferiore b, operazione detta comunemente **salto** o **skip**.

Definizione 5.23 | Tabella di salto

Una **tabella di salto** o **skip table** memorizza, dato un **quanto** q, il valore degli elementi di indice $i \cdot q$, con $i \geqslant 0$, e la loro posizione (espressa in bit) nella lista di affissioni.

Quando si vuole effettuare un salto, e più precisamente quando si vuole trovare il minimo elemento maggiore o uguale a b, si cerca nella tabella il massimo elemento di posto $i \cdot q$ minore o uguale a b, e si comincia a decodificare la lista per cercare il minimo maggiorante di b. Al più q elementi dovranno essere decodificati, e q deve essere scelto sperimentalmente in modo da al tempo stesso non accrescere eccessivamente la dimensione dell'indice e fornire un significativo aumento di prestazioni.

6

Gestione della lista dei termini

Osservazione 6.1 | Qual'è il vantaggio di usare un order-preserving minimal perfect hash?

Data una lista di stringhe la tecnica dell'hash minimale perfetto che preserva l'ordine è in grado di restituire il numero ordinale di qualunque elemento della lista ma di **non memorizzare la lista stessa**. L'occupazione di memoria sarà di circa 5 byte per stringa, indipendentemente dalla lunghezza delle stringhe.

Definizione 6.1 | Hash

Una funzione di **hash** per un insieme di chiavi $X \subseteq U$, dove U è l'universo di tutte le chiavi possibili, è una funzione da $f : X \to m$.

Definizione 6.2 | Hash perfetto

Quando la funzione è iniettiva (cioè non esistono **collisioni**) l'**hash** è detto **perfetto**.

Definizione 6.3 | Hash minimale

Se $|X| = m$, l'**hash** è detto **minimale**

Definizione 6.4 | Funzione che preserva l'ordine

Se esiste un ordine lineare su X e si ha che $x \leqslant y$ se e solo se $f(x) \leqslant f(y)$ diremo che f **preserva l'ordine**.

Definizione 6.5 | Order-preserving minimal perfect hash

Si tratta di una funzione di hash che, per un insieme di stringhe arbitrario, sia **minimale**, **perfetta** e che preservi l'ordine.

Definizione 6.6 | Firma

Una firma è una funzione che consente di riconoscere se un elemento fa parte di un insieme X o no: essa viene definita come una funzione $s : U \to 2^r$, che associ a ogni chiave possibile una sequenza di r bit casuali, nel senso che la probabilità che $s(x) = s(y)$ se x e y sono presi uniformemente a caso da U è $\frac{1}{2}^2$. Consideriamo l'enumerazione ordinata $x_0 \leqslant x_1 \leqslant \ldots \leqslant x_{n-1}$ degli elementi di X e, oltre a f, memorizziamo ora una tabella di n firme a r bit $S = \langle s(x_0), s(x_1), \ldots, s(x_{n-1}) \rangle$.

Per interrogare la struttura risultante su input $x \in U$, agiamo come segue:

1. calcoliamo $f(x)$
2. recuperiamo $S_{f(x)}$
3. se $S_{f(x)} = s(x)$ restituiamo $f(x)$, altrimenti restituiamo -1 per indicare che x non fa arte dell'insieme X.

Se $x \in X$, il valore $f(x)$ è il suo rango in X e quindi $S_{f(x)}$ conterrà per definizione $s(x)$. Se invece $x \notin X$, $S_{f(x)}$ sarà una firma arbitraria, e quindi restituiremo quasi sempre -1, tranne nel caso ci siano collisioni di firme, il che avviene con probabilità $\frac{1}{2}^r$. Tarando il numero r di bit delle firme è quindi possibile bilanciare lo spazio occupato e la precisione della struttura.

Analisi 6.1 | Costruzione di un Order-preserving minimal perfect hash

Scegliamo **due** funzioni h e g con uno spazio dei valori m un po' più ampio della cardinalità n di X, e di creare un vettore $\underline{\boldsymbol{w}}$ di interi di dimensione m. Cercheremo quindi di fare in modo che:

$$w_{h(x)} + w_{g(x)} \mod n$$

sia esattamente il valore che cerchiamo (cioè l'hash minimale perfetto che preserva l'ordine). Avendo due funzioni tali che $m > n$ e il vettore $\underline{\boldsymbol{w}}$ abbiamo un po' di margine.

Vogliamo un **hash** che preserva l'ordine, quindi denotiamo con $r(x)$ il **rango** di x in X, cioè la posizione ordinale di x nell'ordine di X. La condizione che ci interessa imporre è:

$$w_{h(x)} + w_{g(x)} \bmod n = r(x)$$

Si tratta quindi di un sistema di n equazioni modulari nelle variabili w_i, che potrebbe non avere soluzione. Possiamo però rappresentare i vincoli imposti dal sistema come segue: costruiamo un grafo G non orientato con m vertici e n lati in cui per cui per ogni $x \in X$ abbiamo che $h(x)$ è adiacente a $g(x)$ tramite un lato etichettato da $r(x)$.

Consideriamo una sequenza di **pelature** del grafo G. La prima pelatura, $F_0 \subseteq m$, è data dall'insieme dei vertici che sono una foglia (cioè che hanno grado 0 o 1) nel grafo $G_0 = G$. La seconda, F_1, dall'insieme dei vertici che sono una foglia nel grafo G_1 ottenuto cancellando i vertici in F_0 (e i lati loro adiacenti) dal grafo. Continuiamo così finché non arriviamo a una pelatura F_k in cui non ci sono più foglie: F_k è l'insieme vuoto se e solo se il grafo è aciclico.

Se F_k è vuoto, possiamo risolvere ora il sistema nel seguente modo: partiamo da F_{k-1}, e per tutti i vertici x che sono di grado 0 in G_{k-1} assegniamo $w_x = 0$. I vertici x di grado 1 sono invece adiacenti esattamente a un altro vertice y. In genere, w_y è già stato assegnato, dal momento y che fa parte di una pelatura successiva: fanno eccezione i vertici agli estremi di un lato isolato, che possono essere entrambi non assegnati, nel qual caso poniamo $w_y = 0$. Se v è il valore assegnato al lato che collega x e y, poniamo allora:

$$w_x = v - w_y \mod n$$

Possiamo sempre effettuare l'assegnamento perché gli F_i sono una partizione dei vertici del grafo, e quindi non incontreremo mai un vertice già assegnato.

Ora, assumendo che h e g siano casuali e indipendenti, il grafo che andiamo a costruire è un grafo casuale di m vertici con n archi, e un risultato importante di teoria dei grafi casuali dice che per n sufficientemente grande quando $m > 2.09 \cdot n$ il grafo è **quasi sempre** privo di cicli.

Scegliendo bene h e g quindi, e posto che il grafo non risulti degenere, quasi tutti i grafi che otterremo permetteranno di risolvere il sistema.

La teoria degli ipergrafi casuali ci dice che utilizzando 3-ipergrafi (cioè un insieme di sottoinsiemi di ordine 3 dell'insieme dei vertici) è possibile dare una nozione di aciclicità che permette di risolvere i sistemi nel caso di 3 funzioni di **hash**, ma in questo caso il limite che garantisce l'aciclicità è $m > 1.23 \cdot n$, che rappresenta un miglioramento netto rispetto al caso di ordine due.

Per memorizzare lo **hash** dovremo utilizzare solo $1.23 \log n$ bit.

7

Matrici non-negative e catene di Markov

Definizione 7.1 | Catena di Markov

Sia S un insieme finito di **stati**. Una successione di variabili aleatorie $(X_k)_{k\in\mathbb{N}}$ a valori in S è detta **catena di Markov** se e solo se per tutti i $k>0$ e $i_0, i_1, \ldots, i_k$ in S:

$$\begin{aligned}\Pr\{X_k = i_0 | X_{k-1} = i_1, X_{k-2} = i_2, \ldots, X_0 = i_k\} \\ = \Pr\{X_k = i_0 | X_{k-1} = i_1\}\end{aligned}$$

e il lato destro non dipende da k (ogni qual volta il lato sinistro è definito). Il vettore $\underline{\boldsymbol{p}}$ definito da $p_i = \mathbb{P}(X_0 = 1)$ è la **distribuzione iniziale** e la matrice P definita da $P_{ij} = \mathbb{P}\left(X_k = j \,\middle|\, X_{k-1} = i\right)$ è la **matrice di transizione** della catena di Markov. Per ogni $k \geqslant 0$, sia $\underline{\boldsymbol{p}}_{(k)}$ il vettore della distribuzione di probabilità marginale di X_k:

$$\left(\boldsymbol{p}_{(k)}\right)_i = \Pr\{X_k = i\}$$

e in particolare $\underline{\boldsymbol{p}}_{(0)} = \underline{\boldsymbol{p}}$. Vale che $\boldsymbol{p}_{(k)}^T = \boldsymbol{p}^T P^k$, per cui la catena dipende solo dalla distribuzione iniziale e dalla matrice di transizione.

Definizione 7.2 | Matrice Stocastica

In una matrice stocastica tutte le righe sono distribuzioni, cioè ogni riga ha somma unitaria. Una matrice stocastica definisce in maniera naturale un grafo con insieme di nodi S e con archi colorati su $(0\ldots1]$ che corrispondono a transizioni non nulle.

La matrice di transizione P di una catena di Markov è stocastica. Il grafo associato a una matrice stocastica è detto stocastico.

Definizione 7.3 | Matrice non-negativa

Ogni matrice non-negativa M definisce un grafo G colorato su $R_{>0}$. N_G è l'insieme degli indici di M, $A_G = \left\{\langle i,j\rangle | M_{ij} > 0\right\}, s(\langle i,j\rangle) = i, t(\langle i,j\rangle) = j$ e il colore di $\langle i,j\rangle$ è M_{ij}.

Analogamente, dato un grafo colorato su $R_{>0}$, si può costruire la matrice M che ha N_G come insieme di indici e $M_{ij} = \sum_{a\in G(i,j)} c_G(a)$

Definizione 7.4 | Periodo

Il **periodo** di un indice i è definito da:

$$\gcd\left\{k>0 | \left(M^k\right)_{ii} > 0\right\}$$

Un indice è detto **aperiodico** se ha periodo 1.

Definizione 7.5 | Matrice primitiva

Data una matrice non-negativa M, diciamo che M è **primitiva** se esiste $k>0$ tale che tutti gli elementi di M^k sono positivi, e che M è **irriducibile** se, per ogni i e j, esiste un intero $k>0$ tale che $\left(M^k\right)_{ij} > 0$.

In particolare, M è irriducibile se e solo se il grafo associato è fortemente connesso. Una matrice è primitiva se e solo se è irriducibile e aperiodica.

Osservazione 7.1 | Quando una componente fortemente connessa è aperiodica?

Una componente fortemente connessa è aperiodica se esiste un cappio su un nodo i contenuto nella componente, cioè $M_{ii} > 0$.

Definizione 7.6 | Distribuzione invariante

Una distribuzione $\underline{\boldsymbol{p}}$ è **invariante** per P se $\underline{\boldsymbol{p}}^T P = \underline{\boldsymbol{p}}^T$, cioè $\underline{\boldsymbol{p}}$ è un autovettore sinistro di P. Inoltre, data una distribuzione $\underline{\boldsymbol{p}}$, quando $\lim_{k\to\infty} \boldsymbol{p}^T P^k$ è definito, è detto **distribuzione limite** di $\underline{\boldsymbol{p}}$ sotto P.

Definizione 7.7 | Limite di Cesàro

Un modo di comprendere il comportamento al limite di una catena è considerare il suo **limite di Cesàro**:

$$P^* = \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} P^k$$

Esso è sempre definito e uguale a $\lim_{k\to\infty} P^k$ quando quest'ultimo è definito. In particolare, vale che:

$$P^* P = P P^* = \left(P^*\right)^2 = P^*$$

Proposizione 7.1 | Prima proposizione sulle matrici stocastiche

Sia P una matrice stocastica. Una distribuzione $\underline{\boldsymbol{p}}$ è invariante per P se e solo se:

$$p^T = \boldsymbol{q}^T P^*$$

per qualche $\underline{\boldsymbol{q}}$. Inoltre, $\underline{\boldsymbol{p}}$ è la dimostrazione limite di un dato $\underline{\boldsymbol{q}}$ sotto P se e solo se:

$$p^T = \boldsymbol{q}^T P^*$$

Infine il limite $\lim_{k\to\infty} \boldsymbol{q}^T P^k$ è definito per ogni distribuzione $\underline{\boldsymbol{q}}$ se e solo se anche $\lim_{k\to\infty} P^k$ è definito.

Osservazione 7.2 | Conseguenza della prima proposizione sulle matrici stocastiche

Una semplice conseguenza della prima proposizione sulle matrici stocastiche è che c'è sempre almeno una distribuzione invariante.

Proposizione 7.2 | Seconda proposizione sulle matrici stocastiche

Se P è una matrice stocastica unicatenaria i cui indici sono aperiodici, allora

$$\lim_{k\to\infty} P^k = \mathbf{1}p^T$$

dove $\underline{\boldsymbol{p}}$ è l'unica distribuzione invariante di P, e:

$$\lim_{k\to\infty} \boldsymbol{q}^T P^k = \boldsymbol{p}$$

per ogni distribuzione $\underline{\boldsymbol{q}}$.

8 Ranking

Definizione 8.1 | Sistema di reperimento di informazioni

Un sistema di reperimento di informazioni è dato da una collezione documentale D di dimensione N, da un insieme Q di interrogazioni, e da una funzione di **ranking** $r : Q \times D \to \mathbb{R}$ che assegna a ogni coppia data da un'interrogazione e un documento un numero reale.

I documenti sono tanto più rilevanti quanto il punteggio è alto.

Definizione 8.2 | Ranking endogeno

Un tipo di algoritmo di ranking che utilizza il contenuto del documento.

Definizione 8.3 | Ranking esogeno

Un tipo di algoritmo di ranking che utilizzano la struttura esterna.

Definizione 8.4 | Ranking statico

Un criterio di ranking statico è indipendente dall'interrogazione. Il punteggio assegnato a ciascun documento è fisso e indipendente da q.

Definizione 8.5 | Ranking dinamico

Un criterio di ranking dinamico è dipendente dall'interrogazione.

8.1 Pagerank

Definizione 8.6 | PageRank

PageRank è un criterio di ranking statico esogeno basato sulla passeggiata naturale del grafo del web G. Fissato un parametro α tra 0 e 1, a ogni passo con probabilità α si segue un arco uscente e con probabilità $1-\alpha$ si sceglie un qualunque altro nodo del grafo utilizzando una qualche distribuzione $\underline{v}$, detta **vettore di preferenza**. Assumendo che non esistano pozzi, la catena è quindi rappresentata dalla combinazione lineare:

$$\alpha G + (1-\alpha)\mathbf{1}v^T$$

dove G è la matrice della passeggiata naturale su G. Un valore tipico scelto per α è 0.85.

Il grafo associato è unicatenario: tutti i nodi hanno archi entranti da ogni altro nodo e sono pertanto tutti essenziali e siccome hanno anche un loop sono tutti aperiodici. Tutti i nodi essenziali stanno nella stessa unica componente terminale e la matrice associata risulta unicatenaria e aperiodica, e ha quindi sempre un'unica distribuzione limite, detta **PageRank**.

Definizione 8.7 | Fattore di attenuazione (damping factor)

In effetti è possibile pensare a PageRank come una funzione di α: per $\alpha \approx 0$ PageRank è pari al vettore di preferenza o per $\alpha \approx 1$ risulta concentrato nelle componenti terminali.

Esso è essenziale per rendere la matrice aperiodica e unicatenaria e riduce il numero di cifre significative necessario per il calcolo della distribuzione limite e rende inoltre il calcolo stesso molto più rapido.

Osservazione 8.1 | Come si risolvono i pozzi in PageRank?

Solitamente per risolvere i pozzi si aggiunge un arco verso ogni altro nodo, di fatto redistribuendo il ranking dei pozzi in maniera uniforme: giunti a un pozzo si salta in un altro nodo a caso. Altrimenti si può saltare a un altro nodo usando per esempio il vettore di preferenza.

8.2 HITS

Definizione 8.8 | HITS (Hyperlink-Induced Topic Distillation)

HITS è un algoritmo di Ranking sia esogeno che endogeno. Parte infatti da un sottografo del web ottenuto a partire dall'interrogazione, ma utilizza poi la decomposizione a valori singolari per assegnare un punteggio alle pagine.

La selezione del sottografo può essere fatta in vari modi: l'originale procede prendendo un insieme di risultati T ottenuto da un motore di base e generare un sottografo sulla base di R e delle pagine che puntano a quelle in R. Si ottiene una matrice rettangolare M che ha per righe i predecessori delle pagine in R e per colonne le pagine di R. Si procede quindi a calcolare il primo autovettore (cioè quello associato al massimo autovalore) di $M^T M$ e il primo autovettore di MM^T e di considerarli rispettivamente i punteggi di **autorevolezza** e di **hubbiness**. I due vettori esistono sempre dal teorema di Perron-Frobenius, ma l'unicità può essere problematica siccome non vi è nessuna garanzia di connessione.

Definizione 8.9 | Pagina autorevole per HITS

Una **pagina autorevole** nel contesto del ranking HITS è una pagina con contenuto pertinente a una data query ed interessante.

Definizione 8.10 | Hub per HITS

Un **hub** è una pagina contenente numerosi link a pagine autorevoli.

Osservazione 8.2 | In che modo i concetti di hub e pagina autorevole si rinforzano?

I due concetti si **rinforzano mutuamente** perché una pagina autorevole è puntata da molte pagine centrali e una buona pagina centrale punta a molte pagine autorevoli.

Osservazione 8.3 | La matrice ottenuta in HITS è una catena di Markov?

La matrice risultante MM^T contiene nella riga i e colonna j il numero di modi di passare da i a j andando da i verso una pagina del risultato e tornando indietro. La matrice però è solo **non-negativa**, ma non è **stocastica**. Le transizioni più probabili sarebbero quindi quelle tra **hub** fortemente collegati attraverso pagine del risultato.

Analisi 8.1 | Come autorevolezza e hubbiness si rinforzano

Per vedere meglio come il punteggio di **autorevolezza** $\underline{\boldsymbol{a}}$ e di **hubbiness** $\underline{\boldsymbol{h}}$ si rinforzano, notiamo che per quanto detto abbiamo:

$$a = M^T \boldsymbol{h} \quad \boldsymbol{h} = M\boldsymbol{a}$$

Andiamo ad assegnare un valore di autorevolezza costante 1 a ogni parola in R, ottenendo $\underline{\boldsymbol{a}}_0 = \underline{\mathbf{1}}$. Calcoliamo $M\underline{\boldsymbol{a}}_0$ e lo normalizziamo, ottenendo $\underline{\boldsymbol{h}}_0$. Calcolando quindi $M^T\underline{\boldsymbol{h}}_0$ e normalizzandolo otteniamo $\underline{\boldsymbol{a}}_1$.

Le sequenze degli $\underline{\boldsymbol{a}}_i$ e degli $\underline{\boldsymbol{h}}_i$ convergono precisamente ad $\underline{\boldsymbol{a}}$ e $\underline{\boldsymbol{h}}$ ed ad ogni step di aggiornamenti di $\underline{\boldsymbol{a}}_i$ l'autorevolezza di una pagina viene calcolata sommando i punteggi di **hubbiness** degli **hub** che puntano alla pagina stessa mentre a ogni step di aggiornamenti di $\underline{\boldsymbol{h}}_i$ la **hubbiness** di una pagina viene calcolata sommando l'autorevolezza delle pagine da essa puntata.

Se il vettore iniziale $\underline{\boldsymbol{a}}_0$ non è **ortogonale** al vettore singolare principale di M, il processo converge. Per garantire questa proprietà basta prendere un vettore interamente positivo.

8.3 BM25

Definizione 8.11 | Modello a due Poisson

Il modello delle due Poisson ipotizza che la distribuzione delle frequenze dei termini in un documento è il risultato di una combinazione di due distribuzioni Poisson, una per i termini frequenti e una per i termini rari.

Definizione 8.12 | BM25

Con BM25 si intende una funzione $w_j(d,C)$ di pesatura dei termini che utilizza informazioni quali la frequenza dei termini df_j, la lunghezza del documento dl, la lunghezza media dei documenti della collezione $avdl$ e la frequenza del termine considerato nel documento d_j:

$$w_j(d,C) = \frac{(k_1+1)\,d_j}{k_1\left((1-b)+b\frac{dl}{avdl}\right)+d_j}\underbrace{\log\frac{N-df_j+0.5}{df_j+0.5}}_{IDF}$$

I pesi k_1 e b sono parametri liberi, tipicamente rispettivamente $[1.2,2.0]$ e 0.75.

Il punteggio del documento per una data query si ottiene quindi sommando i termini dei pesi corrispondenti ai termini della query:

$$W(\overline{d},q,C) = \sum_j w_j(\overline{d},C)\cdot q_j$$

Più termini risultano comuni con la query più il peso sarà maggiore.

Può essere visto come un'approssimazione del modello a due Poisson.

BM25 tende a fallire quando i documenti sono molto lunghi.

Analisi 8.2 | Che significato hanno le varie parti di BM25?

Innanzitutto, il membro di destra funge da **inverse document frequency (IDF)**, cioè le parole comuni risultano meno importanti. Guardando invece il termine di sinistra, iniziamo dal nominatore: qui troviamo il termine d_j, che nei casi in cui viene sommato coincide con la frequenza del termine della query nel documento. Se esso aumenta di valore, allora lo score aumenta.

Guardando invece il denominatore, questo rende la frequenza del termine meno importante che non diverse parole della query, ma più importante (con la parte frazionaria $dl/avdl$) se il documento è particolarmente lungo rispetto alla media.

9 Metodi computazionali

9.1 Metodo della potenza

Osservazione 9.1 | Cosa è necessario per usare il metodo della potenza?

È necessario che il primo autovalore sia **separato** dal secondo cioè che l'autovalore dominante abbia molteplicità uno. Questo condizione è necessaria perché un qualunque vettore $\underline{\boldsymbol{v}}$ con componente non nulla lungo l'autovettore associato all'autovalore dominante, l'iterazione $M^k\underline{\boldsymbol{v}}$ produce un vettore in direzione sempre più vicina all'autovettore dominante.

Definizione 9.1 | Metodo della potenza

Si tratta di un metodo per identificare l'autovettore dominante di una matrice. Partiamo da un vettore $\underline{\boldsymbol{v}}_0$ e ad ogni passaggio otteniamo il vettore successivo $\underline{\boldsymbol{v}}_{k+1}$ moltiplicando il vettore corrente per M e normalizziamo il risultato: $\boldsymbol{v}_{k+1} = \frac{M\boldsymbol{v}_k}{\|M\boldsymbol{v}_k\|}$. Se $\underline{\boldsymbol{e}}_0, \underline{\boldsymbol{e}}_1, \dots, \underline{\boldsymbol{e}}_{n-1}$ è una base normalizzata di autovettori e $\boldsymbol{v}_0 = \sum \alpha_i \boldsymbol{e}_i$ abbiamo che:

$$\boldsymbol{v}_{k+1} = M^k\boldsymbol{v}_0 = \sum \alpha_i M^k \boldsymbol{e}_i = \alpha_0\lambda_0^k\left(e_0 + \sum_{i>0}\frac{\alpha_i}{\alpha_0}\left(\frac{\lambda_i}{\lambda_0}\right)^k \boldsymbol{e}_i\right)$$

Per le assunzioni di separazione, l'espressione tra parentesi equivale a $\underline{\boldsymbol{e}}_0$.

Osservazione 9.2 | Come converge il metodo della potenza?

Dato che dividiamo a ogni passo per la norma, $\underline{\boldsymbol{v}}_k$ converge proprio a $\underline{\boldsymbol{e}}_0$. Il resto è controllato da $(\lambda_1/\lambda_0)^k$ Il metodo della potenza converge **geometricamente** con ragione tanto più più piccola quanto più grande è la separazione tra il primo ed il secondo autovalore.

Osservazione 9.3 | Cosa cambia nel metodo della potenza applicato a PageRank?

1. Partendo da un vettore normalizzato da una matrice stocastica si ottengono sempre vettori normalizzati, quindi si itera semplicemente la matrice.
2. La matrice che descrive la transizione è la **trasposta** di quella di cui vogliamo calcolare gli autovalori, o equivalentemente ne vogliamo calcolare gli autovalori **sinistri** e non quelli destri.
3. Il calcolo effettivo non viene eseguito in maniera matriciale perché genererebbe un numero di moltiplicazioni ingestibile, dato che per via del **fattore di attenuazione** ogni nodo risulta connesso a ogni altro nodo.

Osservazione 9.4 | Come si risolve il problema delle moltiplicazioni nel metodo della potenza applicato a PageRank?

Denotiamo con $\underline{\boldsymbol{d}}$ il vettore caratteristico dei pozzi, e assumiamo di spostarci da un pozzo a un altro nodo scelto mediante il vettore di preferenza $\underline{\boldsymbol{v}}$. La matrice che stiamo effettivamente usando è ora:

$$\alpha G + \alpha\underline{\boldsymbol{d}}\boldsymbol{v}^T + (1-\alpha)\underline{\mathbf{1}}\boldsymbol{v}^T$$

Se assumiamo che l'approssimazione corrente sia $\underline{\boldsymbol{x}}$, la prossima sarà:

$$\alpha\boldsymbol{x}^T G + \alpha\boldsymbol{x}^T\boldsymbol{d}\boldsymbol{v}^T + (1-\alpha)\boldsymbol{v}^T = \alpha\boldsymbol{x}^T G + \alpha\kappa\boldsymbol{v}^T + (1-\alpha)\boldsymbol{v}^T$$

dove κ è la somma dell'approssimazione corrente è la somma dell'approssimazione corrente su tutti i pozzi. La coordinata i-esima del vettore precedente dipende però dalle singole coordinate di $\underline{\boldsymbol{x}}$ per mezzo dell'addendo $\boldsymbol{x}^T G$: il valore κ può essere calcolato a partire da $\underline{\boldsymbol{x}}$ separatamente.

La coordinata i-esima della nuova approssimazione è data da una certa quantità di **ranking** che proviene da altri nodi che non sono pozzi, distribuita secondo G, e da due addendi addizionali, uno legato al teletrasporto e uno dovuto al ranking diffuso dai pozzi. Questi ultimi dipendono solo da $\underline{\boldsymbol{v}}$ e κ. Questo fa sì che il metodo della potenza possa essere implementato **iterando unicamente sui lati di** G.

Osservazione 9.5 | Quando termina la convergenza per il metodo della potenza applicato a PageRank?

Le coordinate della nuova approssimazione sono definite sulla base di quella vecchia e questo fa sì che il metodo della potenza richiesta due vettori: uno contenente l'approssimazione corrente e uno la successiva. Quando la differenza tra i due vettori è sufficientemente piccola in una norma scelta opportunamente, l'iterazione viene terminata. Il secondo autovalore della matrice che utilizziamo è α o meno e quindi la convergenza è garantita e risulta particolarmente veloce se α non è vicino a 1.

9.2 Metodo di Gauss-Seidel

Definizione 9.2 | Metodo di Gauss-Seidel

L'algoritmo di Gauss-Seidel fornisce approssimazioni per le soluzioni di un'equazione della forma $M\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}$ partendo da un vettore $\underline{\boldsymbol{x}}^{(0)}$ e calcolando:

$$x_i^{(t+1)} = \left(b_i - \sum_{j<i} m_{ij} x_j^{(t+1)} - \sum_{j>i} m_{ij} x_j^{(t)} \right) / m_{ii}$$

Osservazione 9.6 | Come converge il metodo di Gauss-Seidel?

L'algoritmo di Gauss-Seidel utilizza sempre il valore più recente calcolato per una certa componente, il che ci permette di utilizzare un solo vettore. L'algoritmo utilizzando in maniera aggressiva valori più precisi rende la sua convergenza molto più rapida.

Osservazione 9.7 | Applicare il metodo di Gauss-Seidel a PageRank

Le sommatorie non sono indicizzate dalla prima ma dalla seconda componente della matrice. Nel caso di PageRank abbiamo bisogno dei **predecessori** e non dei successori di un nodo per aggiornare il suo valore e quindi dobbiamo scandire il grafo trasposto ma al tempo stesso conoscere il grafo positivo di ogni nodo per poter calcolare correttamente il suo contributo ai suoi successori. La conoscenza dei gradi positivi richiede quindi un'ulteriore vettore di interi.

10 HyperBall

Osservazione 10.1 | Quando è necessario HyperBall?

HyperBall è utile quando si ha a che fare con un grafo molto grande (social o web per esempio) e si vuole comprendere qualcosa della sua struttura globale. Si vuole inoltre comprendere quali nodi svolgono un qualche ruolo importante.

Osservazione 10.2 | A cosa serve HyperBall?

HyperBall consente di poter utilizzare la centralità armonica su grafi molto grandi.

Definizione 10.1 | Step intermedio di HyperBall

1. Per ogni nodo calcoliamo in sequenza il numero dei nodi a distanza esattamente pari a t.
2. Sommando tutti i nodi otteniamo la distribuzione della distanza (normalizzandola).
3. Le definizione di centralità, per esempio armonica, possono essere riscritte come:

$$\sum_{t>0} \frac{1}{t} |\{y | d(y,x) = t\}|$$

Osservazione 10.3 | Come si procede a calcolare lo step intermedio di HyperBall?

Esistono 3 approcci per calcolarlo:

1. Multiple visite breadth-first: queste hanno complessità $O(mn)$ e richiedono accesso diretto.
2. Approcci tramite sampling di visite breadth-first non danno buone approssimazioni su grafi che non sono fortemente connessi e comunque richiedono accesso diretto.
3. Si può utilizzare il framework per stimare la dimensione di Edith Cohen: è molto potente ma non scala o parallelizza molto bene e richiede accesso diretto.
4. Infine è possibile utilizzare la diffusione.

Definizione 10.2 | Diffusione

Sia $B_t(x)$ la sfera (ball) di raggio t attorno a x, l'insieme di nodi a distanza al massimo t da x. Allo step iniziale $B_0(x) = \{x\}$ e costruiamo la sfera iterativamente enumeando gli archi $x \to y$ ed eseguendo unioni tra insiemi:

$$B_{t+1}(x) = \bigcup_{x \to y} B_t(y) \bigcup \{x\}$$

Osservazione 10.4 | Quali sono le problematiche della diffusione?

Ogni insieme usa spazio lineare, e nel complesso è quadratico, quindi non sarebbe gestibile.

Osservazione 10.5 | Come si possono risolvere le problematiche della diffusione?

Si possono risolvere utilizzando degli insiemi approssimati, cioè utilizzando contatori probabilistici che rappresentano insiemi ma che consentono solo di identificarne la dimensione.

Definizione 10.3 | Contatori HyperLogLog

Al posto di contare effettivamente un insieme di elementi andiamo a **osservarne** una caratteristica statistica: questa è il numero degli zero alla fine del valore di una funzione di hash **molto buona**. Di questi teniamo traccia del valore massimo m, da cui lo spazio è $\log\log n$ bits.

Il numero degli elementi distinti è proporzionale a 2^m.

Definizione 10.4 | Diffusione con set approssimati

Si procede scegliendo un insieme approssimato su cui sia possibile eseguire delle unioni velocemente. Un esempio può essere ANF, che utilizza dei contatori di Martin-Flajolet (MF) che occupano spazio $\log n + c$. Nel contesto di HyperBall vengono utilizzati contatori HyperLogLog, che occupano spazio $\log\log n$.

I contatori MF possono essere combinati con degli OR.

Per combinare i contatori HyperLogLog rapidamente viene utilizzato il **broadword programming**.

Osservazione 10.6 | Come si può aumentare la confidenza?

Per aumentare la confidenza utilizzeremo molti contatori, tipicamente 2^b con $b \geq 4$ e calcolarne la media armonica. Ogni insieme quindi è rappresentato da una lista di piccoli contatori, comunemente di 5 bit.

Per calcolare l'unione tra due set si procede per massimizzazione uno a uno. Purtroppo estrarre, massimizzare e inserire tramite shifts è estremamente lento.

Nel caso dei contatori Martin-Flajolet si procede a fare un OR delle feature.

Osservazione 10.7 | Quali possibili miglioramenti esistono?

1. Teniamo traccia delle modifiche e non massimizziamo con contatori non modificati.
2. Utilizziamo la computazione sistolica, dove ogni insieme modificato segnala i predecessori che qualcosa dovrà succedere.
3. Possiamo migliorare le performance tramite parallelizzazione con decomposizione: una task aggiorna solo un sotto-insieme di contatori il cui outdegree generale è predetto utilizzando una rappresentazione Elias-Fano della distribuzione di outdegree cumulativa.

11

Boolean Retrieval

Definizione 11.1 | Dati non strutturati

Con dati non strutturati ci si riferisce a documenti che non possiedono una struttura semanticamente esplicita, semplice da comprendere per un computer.

Definizione 11.2 | Information retrieval

Information retrieval (IR) consiste nel trovare materiale, tipicamente documenti, di tipo non strutturato, tipicamente testi, che soddisfi una informazione richiesta da una collezione più grande.

Definizione 11.3 | Modello di boolean retrieval

Il modello di **boolean retrieval** è un modello di IR cui possiamo porre qualsiasi query sotto forma di espressione booleana dei termini, dove i termini sono combinati tramite le espressioni booleane and, or e not.

Questi modelli vedono ogni documento semplicemente come insiemi di parole.

Definizione 11.4 | Precisione

Quale frazione dei risultati identificati dal sistema IR sono rilevanti rispetto alle informazioni richieste?

Definizione 11.5 | Recall

Quale frazione dei documenti rilevanti nella collezione sono stati identificati dal sistema IR?

Definizione 11.6 | Efficacia di un sistema IR

Per valutare l'efficacia di un sistema IR, cioè la qualità dei suoi risultati di ricerca, si utilizzano due metriche statistiche sui risultati che il sistema ritorna per una data query: la **precisione** e **recall**.

Definizione 11.7 | Indice inverso

Per realizzare un **indice inverso** andiamo a realizzare un dizionario di termini e ad ogni termine associamo una lista (posting list o lista inversa) dei record (posting) dei documenti dove appare.

Si tratta di una struttura dati per rappresentare la matrice sparsa dei termini e documenti.

Osservazione 11.1 | Come viene costruito un indice inverso?

Ad alto livello, per costruire un indice inverso:

1. Si raccolgono i documenti che devono essere indicizzati.
2. Si tokenizza il testo.
3. Si procede ad eseguire una fase di preprocessing linguistico dei token realizzati.
4. Si indicizzano i documenti in cui appare ogni termine.

Definizione 11.8 | Proximity operator

Un **proximity operator** è un modo per specificare che due termini in una query devono occorrere vicino uno all'altro in un documento.

Definizione 11.9 | Stemming

Il processo di **stemming** tipicamente si riferisce a un'euristica piuttosto semplice in cui si va a tagliare la fine delle parole in modo tale da cercare di normalizzarle.

Definizione 11.10 | Lemmatization

Il processo di **lemmatization** tipicamente si riferisce ad un processo di stemming più preciso, in cui le parole vengono normalizzate utilizzando un dizionario con una forma base delle parole, chiamata **lemma**.

11.1 Rappresentazione di Elias-Fano

Definizione 11.11 | Rappresentazione di Elias-Fano

Si tratta di una struttura quasi-succinta per sequenze monotone. Dati due termini n e u, consideriamo la sequenza $0 \leqslant x_0, x_1, x_2, \ldots, x_{n-1} \leqslant u$. Andiamo a salvare gli $\ell = \log(u/n)$ bit minori esplicitamente. Salviamo i bit maggiori come una sequenza di intervalli unari codificati (dove 0^k1 rappresenta k).

Utilizziamo al più $2 + \log\left(\frac{u}{n}\right)$ bit per elemento. Siccome è prossimo al limite di una struttura succinta, per cui la struttura è detta quasi-succinta.

Osservazione 11.2 | Perché la rappresentazione di Elias-Fano è utile?

Lo spazio utilizzato è quasi ottimale e non assume nessuna particolare distribuzione dei dati. Inoltre, per essere letta in modo sequenziale richiede pochi operatori logici e restringe il problema di rank/selection ad un array di circa $2n$ bits con metà zero e metà uno.

Definizione 11.12 | Selezione su una rappresentazione Elias-Fano

Supponiamo di voler selezionare il k-esimo elemento rapidamente. Leggiamo i bit superiori (quelli in unario) uno alla volta, contando gli elementi a uno. Quando si arriva alla parola corretta, ne completiamo la sequenza e prendiamo i bit inferiori.

Definizione 11.13 | Ranking su una rappresentazione Elias-Fano

Si procede analogamente alla selezione ma si contano gli zero, numero che indica quanto i bit superiori stanno aumentando. Saltiamo i primi $b >> \ell$ zero superiori e completiamo la sequenza.

12

Index compression

Definizione 12.1 | Bit di proseguimento (continuation bit)

Il bit di continuazione è tipicamente il primo bit di un byte e viene settato a 1 se è parte dell'ultimo byte, altrimenti a 0.

Definizione 12.2 | Variable byte codes

I **variable byte (VB) codes** usano un numero integrale di byte per codificare un gap. Gli ultimi 7 byte sono *payload* e codificano parte del gap. Il primo bit dei byte funge da *continuation bit*, pertanto estraiamo a concateniamo i segmenti da 7 bit rimanenti.

13

Scoring, term weighting, and the vector space model

Definizione 13.1 | Term Frequency

Con **term frequency**, la frequenza di un termine, si intende uno degli approcci più semplici per assegnare un punteggio tra un termine t ed un documento d. Viene denotato come $\text{tf}_{t,d}$.

Definizione 13.2 | Bag of words

Rappresentazione di un documento in cui viene considerata unicamente per ogni termine il numero di volte in cui appare, senza considerare l'ordine di apparizione.

Spesso, da questo insieme di parole vengono eliminate le stop-words.

Osservazione 13.1 | Di quale problema soffrono l'approccio a term frequency?

Nell'approccio a **term frequency** tutti i termini sono considerati equamente importanti quando si procede ad assegnare la rilevanza di essi in una query.

Definizione 13.3 | Collection frequency

Un approccio immediato per scalare l'importanza di un termine t è di normalizzarla per il numero di volte che appare nella collezione: essa viene rappresentata con cf_t e rappresenta il numero di volte che un termine appare nella collezione.

Definizione 13.4 | Document Frequency

La **document frequency** df_t è definita come il numero di documenti nella collezione che contiene un termine t.

Definizione 13.5 | Inverse Document Frequency

Chiamando il numero totale di documenti nella collezione come N, definiamo la **inverse document frequency** (idf) di un termine t come:

$$\text{idf}_t = \log {}^{N}/_{\text{df}_t}$$

Il valore di *idf* di un termine raro è alto, mentre di un termine frequente è basso.

Definizione 13.6 | TF-IDF

Combinando la **term frequency** e la **inverse document frequency** otteniamo lo schema di pesatura **tf-idf**, che assegna a un termine t in un documento d il valore:

$$\text{tf}-\text{idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

In altre parole, $\text{tf}-\text{idf}_{t,d}$ assegna ad un termine d un peso in un documento d che è:

1. Massimo quando t occorre spesso all'interno di pochi documenti.
2. Più basso quando il termine appare poche volte in un documento, o appare in molti documenti.
3. Minimo quando appare quasi ovunque.

Definizione 13.7 | Overlap score measure

Il punteggio di un documento d è la somma, su tutti i termini della query, del numero di volte che ogni termine appare in d. Possiamo raffinare questo punteggio utilizzando invece il peso $tf-idf$ di ogni termine della query nel documento:

$$\text{Score}(q,d) = \sum_{t \in q} \text{tf}-\text{idf}_{t,d}$$

Osservazione 13.2 | Cosine similarity applicata al TF-IDF

Un modo standard per quantificare la similarità di due documenti d_1 e d_2 è il calcolo della **cosine similarity** dei vettori di rappresentazione dei documenti:

$$\text{sim}(d_1,d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{\left|\vec{V}(d_1)\right| \left|\vec{V}(d_2)\right|'}$$

Definizione 13.8 | Sublinear tf scaling

Siccome è improbabile che 20 o più occorrenze di un termine in un documento possano davvero avere 20 o più volte l'importanza di una singola occorrenza, una modifica tipica è di utilizzare un logaritmo della term frequency:

$$\text{wf}_{t,d} = \begin{cases} 1 + \log \text{tf}_{t,d} & \text{if } \text{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

e quindi successivamente:

$$\text{wf}-\text{idf}_{t,d} = \text{wf}_{t,d} \times \text{idf}_t$$

Definizione 13.9 | Maximum tf normalization

Un altro approccio è quello di normalizzare i pesi di term frequency di tutti i termini che appaiono in un documento per il termine massimo del documento (normalizzazione max-min). Viene inoltro svolto uno **smoothing** con un termine a, generalmente pari a 0.4.

$$\text{ntf}_{t,d} = a + (1-a)\frac{\text{tf}_{t,d}}{\text{tf}_{\max}(d)}$$

Osservazione 13.3 | Di quali problemi soffre la maximum normalization?

La **maximum tf normalization** soffre dei seguenti problemi:

1. Il metodo risulta instabile perché un cambio nella lista delle stop word può cambiare drasticamente i risultati, e pertanto risulta difficile da regolare.
2. Un documento potrebbe contenere un termine outlier con una frequenza sorprendentemente alta, non rappresentativa del contenuto di quel documento.
3. Più generalmente, un documento in cui il termine più frequente appare circa quanto gli altri termini dovrebbe essere trattato diversamente da quelli con una distribuzione più particolare.

14

Matrix decompositions and latent semantic indexing

Definizione 14.1 | Decomposizione di matrici

Una matrice quadrata può essere fattorizzata nel prodotto di matrici derivate dai suoi auto-vettori.

Teorema 14.1 | Diagonalizzazione di matrici

Sia S una matrice $M \times M$ a valori reali, con M autovettori indipendenti. Allora esiste una decomposizione spettrale $S = U\Lambda U^{-1}$ dove i valori delle colonne di U sono gli autovettori di S e Λ è una matrice diagonale i cui valori sono gli autovalori di S in valore decrescente.

$$\begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \dots & \\ & & & \lambda_M \end{pmatrix}, \lambda_i \geqslant \lambda_{i+1}$$

In particolare, se gli autovalori sono distinti, la decomposizione risulta unica.

Teorema 14.2 | Diagonalizzazione simmetrica

Sia S una matrice simmetrica quadrata $M \times M$ a valori reali con M autovettori linearmente indipendenti. Allora esiste una decomposizione simmetrica diagonale

$$S = Q\Lambda Q^T$$

dove le colonne di Q sono autovettori di S ortogonali e normalizzati e Λ è la matrice diagonale i cui valori sono gli autovalori di S. Inoltre, tutti i valori di Q sono reali e vale che $Q^{-1} = Q^T$.

Osservazione 14.1 | Ma come sono fatte le matrici reali?

Le matrici C a cui siamo interessati son generalmente $M \times N$ dove molto raramente $M = N$ e ancora meno simmetriche.

Definizione 14.2 | Decomposizione a valori singolari (Singular value decomposition)

Sia r il rango della matrice C di dimensione $M \times N$. Esiste una **decomposizione a valori singolari** (SVD) di C della forma $C = U\Sigma V^T$ dove:

1. Gli autovalori $\lambda_1, \dots, \lambda_r$ di CC^T sono gli stessi degli autovalori di C^TC.
2. Per $1 \leqslant i \leqslant r$, sia $\sigma_i = \sqrt{\lambda_i}$, con $\lambda_i \geqslant \lambda_{i+1}$. Allora la matrice Σ $M \times N$ è composta da $\Sigma_{ii} = \sigma_i$ per $1 \leqslant i \leqslant r$ e zero altrimenti.

I valori σ_i sono chiamati **valori singolari** di C.

Definizione 14.3 | SVD ridotta o troncata

Convenzionalmente per scrivere i valori della SVD, si scrive Σ come una matrice $r \times r$ con i valori singolari sulla diagonale dato che tutti gli altri valori sono zero.

È convenzione omettere le $M - r$ colonne più a destra di U corrispondenti alle righe omesse di Σ. Analogamente si omettono le $N - r$ colonne più a destra di V.

Definizione 14.4 | Norma di Frobenius

Data una matrice X di dimensioni $M \times N$, la norma di Frobenius:

$$\|X\|_F = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N} X_{ij}^2}$$

Problema 14.1 | Low-Rank approximation

Data una matrice C di dimensioni $M \times N$ ed un intero positivo k, il problema consiste nel trovare la matrice $M \times N$ di rango al più C_k tale da minimizzare la norma di Frobenius della differenza delle matrici $X = C - C_k$. Tipicamente, k è molto più piccolo del rango r della matrice C.

Analisi 14.1 | Risolvere il problema di Low-Rank approximation usando le SVD

Le **singular values decompositions** (SVD) possono essere usate per risolvere il problema delle approssimazioni di Low-Rank:

1. Data C, determiniamo la sua SVD: $C = U\Sigma V^T$.
2. Determiniamo da Σ la matrice Σ formata sostituendo con zero gli $r - k$ valori singolari più piccoli sulla diagonale di Σ.
3. Calcoliamo $C_k = U\Sigma_k V^T$ come l'approssimazione a rango k di C.

Teorema 14.3 | Teorema di Eckart and Young

La matrice C_k di rango k ottenuta ha il valore di norma di Frobenius più basso possibile.

$$\min_{Z|rank(Z)=k} \|C - Z\|_F = \|C - C_k\|_F = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2}$$

14.1 Indicizzamento semantico latente

Definizione 14.5 | Indicizzamento semantico latente (Latent semantic indexing)

Si tratta di un processo per generare un'approssimazione di matrice dei termini C di rango minore rispetto all'originale utilizzando le SVD.

Osservazione 14.2 | Come viene eseguita una query su una rappresentazione LSI?

Un vettore query $\underline{\boldsymbol{q}}$ deve essere mappato nella sua rappresentazione LSI utilizzando la trasformazione:

$$\underline{\boldsymbol{q}}_k = \Sigma_k^{-1} U_k^T \underline{\boldsymbol{q}}$$

Analogamente si possono trasformare ed inserire nuovi documenti.

Osservazione 14.3 | LSI come Soft clustering

LSI può essere visto come una forma di **soft clustering** interpretando ogni dimensione dello spazio ridosso come un cluster e il valore che un documento ha in quella dimensione come un'appartenenza parziale a quel cluster.

15

Vettori dei suffissi

Definizione 15.1 | Albero dei suffissi

Un albero dei suffissi di una stringa è il trie compatto di tutti i suo suffissi.

È una struttura molto utile con molte applicazioni per esempio nel campo della biologia computazionale e può essere costruito in tempo lineare sulla lunghezza della stringa.

Definizione 15.2 | Vettore dei suffissi

Un vettore dei suffissi è un vettore dei suffissi di una stringa ordinato lessicograficamente. Contiene indicativamente la stessa informazione di un albero di suffissi, ma in un modo più implicito e risulta più semplice e compatto per molte applicazioni.

A

Domande d'esame

Le seguenti sono domande d'esame raccolte da studenti che lo hanno sostenuto.

1. Filtri di Bloom
2. Gestione duplicati incontrati durante crawling.
3. Misure di centralità.
4. Metodo potenza (a cosa serve, cosa si calcola con esso)
5. Codici istantanei.
6. Disuguaglianza di MC Millan
7. Esempi di codici istantanei (unario, gamma, delta)
8. Ranking endogeno
9. TF-IDF
10. BM25 (saper descrivere il comportamento dei vari pesi)
11. Elias-Fano