



UNIVERSITÀ DEGLI STUDI DI UDINE

DIPARTIMENTO DI SCIENZE MATEMATICA, INFORMATICHE E FISICHE

TESI MAGISTRALE IN INFORMATICA

UNO STUDIO SUGLI ALGORITMI DI SKETCHING

PER LA STIMA DELLA CARDINALITÀ

RELATORE

PROF. GABRIELE PUPPIS

UNIVERSITÀ DEGLI STUDI DI UDINE

LAUREANDO MAGISTRALE

DANIELE FERROLI

MATRICOLA

137357

ANNO ACCADEMICO

2024-2025

“DA SCRIVERE”
— RENE DESCARTES

Contents

1	INTRODUZIONE	1
2	BACKGROUND	3
2.1	Modello di stream di dati	3
2.2	Algoritmi di streaming	5
2.3	Funzioni hash	5
2.4	Sketch	6
2.5	Stimatori	6
2.6	Metriche di errore	6
2.7	Famiglia di algoritmi per count-distinct	7
2.8	Spazio-accuratezza: ordini di grandezza	7
3	UN'ANALISI SULLO STATO DELL'ARTE	9
4	IMPLEMENTAZIONE	II
4.1	Algoritmi	II
4.2	Framework di benchmark	II
5	RISULTATI SPERIMENTALI	13
6	CONCLUSIONI	15
	REFERENCES	17
	ACKNOWLEDGMENTS	19

1

Introduzione

Random citation [?].

2

Background

In questa tesi, il modello che rappresenta i dati in input, a differenza di algoritmi più tradizionali, è chiamato *modello di stream di dati* (data stream model).

2.1 MODELLO DI STREAM DI DATI

Nel modello di stream i dati [1] arrivano in modo continuo e potenzialmente infinita. Rispetto l'utilizzo di un database tradizionale, non è possibile accumulare tutto in memoria o su disco e interrogare i dati. Gli elementi devono essere processati al volo oppure vengono persi.

Inoltre, la velocità con cui i dati arrivano non è controllata dal sistema (più stream possono arrivare a velocità e con formati diversi) e lo spazio di memoria disponibile è limitato. Eventuali archivi storici possono esistere, ma non sono pensati per rispondere a query online in tempi ragionevoli.

Iniziamo a definire formalmente gli elementi di una stream e come vengono processati. Possiamo considerare una stream come una sequenza di elementi ordinata che arriva uno alla volta, con possibili ripetizioni:

$$S = \langle x_1, x_2, \dots, x_s \rangle,$$

dove ogni x_i appartiene a un universo \mathcal{U} molto grande. Per ogni elemento $a \in \mathcal{U}$ definiamo la frequenza:

$$f(a) = |\{i \mid x_i = a\}|.$$

Uno degli obiettivi principale di questa tesi è stimare la cardinalità dello stream, ossia il numero di elementi distinti:

$$F_0 = |\{a \in \mathcal{U} \mid f(a) > 0\}|.$$

Più in generale, dato lo stream S si definiscono i *frequency moments*

$$F_k = \sum_{a \in \mathcal{U}} f(a)^k,$$

di cui F_0 è il numero di distinti. Queste quantità sono studiate nel modello di stream proprio per il loro valore statistico e per i vincoli di memoria sublineare [2].

Per formalizzare l'accuratezza di uno stimatore si usano due parametri (ε, δ) . Un algoritmo A è detto (ε, δ) -*approssimante* per F_0 se, per ogni stream, produce una stima \tilde{F}_0 tale che

$$\Pr(|\tilde{F}_0 - F_0| \leq \varepsilon F_0) \geq 1 - \delta,$$

dove la probabilità è rispetto alla randomizzazione interna dell'algoritmo [3].

Supponiamo inoltre che l'universo abbia dimensione $n = |\mathcal{U}|$ e che ogni elemento x_i richieda b bit per essere rappresentato.

In questo contesto, un algoritmo di streaming deve:

- processare ciascun elemento con costo $O(1)$ o quasi costante;
- usare memoria molto più piccola di n e di $|\mathcal{U}|$;
- produrre una stima \hat{F}_0 con errore controllato, utilizzando m bits, dove $m \ll n$.

Nel modello dello stream dei dati esistono diverse tipologie di modelli [4]. In questa tesi adottiamo il modello *insertion-only* (o *cash register*), in cui lo stream è una sequenza di aggiornamenti del tipo (a_t, Δ_t) , con $a_t \in \mathcal{U}$ e $\Delta_t \geq 0$. Indichiamo con $A_t[j]$ la frequenza dell'elemento j dopo i primi t aggiornamenti; allora

$$A_t[j] = \begin{cases} A_{t-1}[j] + \Delta_t & \text{se } a_t = j, \\ A_{t-1}[j] & \text{altrimenti.} \end{cases}$$

Esistono tuttavia modelli più generali, come il *turnstile*, che ammette anche aggiornamenti negativi (Δ_t può essere negativo), o il modello a *sliding window*, in cui si considera solo una finestra temporale degli ultimi W elementi. Questi casi esulano dallo scopo della tesi.

Per rispettare questi vincoli si ricorre a funzioni hash che approssimano una distribuzione uniforme sugli elementi e a strutture compatte, chiamate **sketch**, che riassumono le informazioni essenziali dello stream senza conservarlo esplicitamente.

Un esempio di istanza per la stima della cardinalità è lo stream:

$$S = \langle a, b, a, c, d, b, d \rangle.$$

In questo caso l'insieme dei distinti è $\{a, b, c, d\}$ e quindi $F_0 = 4$.

SPAZIO SUBLINARE. Nel modello streaming è fondamentale che la memoria utilizzata sia *sublineare* rispetto alla dimensione dell'universo o alla lunghezza dello stream. Diremo che un algoritmo usa spazio sublineare se la memoria m cresce asintoticamente meno di n , cioè $m = o(n)$, e in pratica si richiede che m dipenda in modo

polilogaritmico da n e polinomiale da $1/\varepsilon$ e $\log(1/\delta)$. Questa proprietà è cruciale perché memorizzare tutti i distinti richiede $\Omega(n)$ bit, mentre nel problema dei distinti esistono algoritmi che ottengono una $(1 \pm \varepsilon)$ -approssimazione con $O(\varepsilon^{-2} + \log n)$ bit, e questo ordine è ottimale [5]. In altre parole, lo sketching rende possibile stimare quantità globali con memoria molto più piccola di quella necessaria per conservare i dati grezzi.

2.2 ALGORITMI DI STREAMING

In questa sezione si definisce cosa si intende per algoritmo di streaming e quali vincoli lo distinguono dagli algoritmi tradizionali. Da completare:

- Definizione formale di algoritmo di streaming.
- Stato compatto M e operazioni:

$$M \leftarrow \text{Update}(M, x), \quad \hat{F}_0 \leftarrow \text{Query}(M).$$

- Modello di costo: un passaggio, update $O(1)$, query su stato compatto.
- Limiti di memoria e trade-off accuratezza/spazio.
- Differenza tra stime in tempo reale e analisi offline.
- Randomizzazione e garanzie probabilistiche.
- (Definizione) mergeabilità/composability degli sketch.

2.3 FUNZIONI HASH

Qui si introduce il ruolo delle funzioni hash nella riduzione dello spazio e nella randomizzazione delle stime. Da completare:

- Definizione di hash e proprietà desiderate.
- Modello ideale: $h : \mathcal{U} \rightarrow [0, 1]$ o $h : \mathcal{U} \rightarrow \{0, 1\}^w$.
- Assunzioni tipiche: uniformità, indipendenza (o hash quasi-uniformi).
- Impatto di scelte non ideali sull'errore degli estimatori.
- Eventuale gestione del seed per riproducibilità.

2.4 SKETCH

Si definisce lo sketch come struttura compatta che riassume lo stream. Da completare:

- Definizione generale di sketch e operazioni supportate (update, query).
- Dimensione dello sketch e relazione con i parametri (k, m, L).
- Esempi intuitivi (registri, bitmap, contatori).
- Due forme usate in tesi:
 - bitmap / pattern di bit (Probabilistic Counting).
 - registri e leading zeros (LogLog, HLL, HLL++).
- (Cenno) serializzabilità e footprint in byte.

2.5 STIMATORI

Si descrive cosa si intende per stimatore e in che senso la stima è corretta. Da completare:

- Definizione di stimatore \hat{F}_0 e proprietà desiderate.
- Distinzione tra stimatore corretto (unbiased) e stimatore biased.
- Consistenza e varianza dello stimatore.
- Cenno a tecniche di bias correction.

2.6 METRICHE DI ERRORE

Questa sezione introduce le metriche di qualità della stima. Da completare:

- Errore assoluto e relativo.
- Bias e unbiased estimator.
- Varianza e standard error.
- RMSE e MAE come metriche aggregate.
- RSE teorica vs osservata (quando applicabile).
- Definizioni allineate alle colonne CSV del framework.

2.7 FAMIGLIA DI ALGORITMI PER COUNT-DISTINCT

Sezione ponte (senza dettagli implementativi) per motivare il Capitolo 3. Da completare:

- Baseline esatta vs sketch probabilistici.
- Linea FM/Probabilistic Counting → LogLog → HLL → HLL++.
- Differenze qualitative: riduzione varianza, correzioni di range, uso di registri.

2.8 SPAZIO-ACCURATEZZA: ORDINI DI GRANDEZZA

Sezione opzionale ma utile per giustificare l'uso degli sketch. Da completare:

- Dipendenza dello spazio da (ε, δ) .
- Interpretazione di “ottimalità” a livello di ordine di grandezza.

3

Un'analisi sullo stato dell'arte

TODO: scrivere l'analisi dello stato dell'arte.

4

Implementazione

4.1 ALGORITMI

4.2 FRAMEWORK DI BENCHMARK

Example of Algorithm 4.1 reference.

Algorithm 4.1 Pseudocode

```
i ← 10
if  $i \geq 5$ 
     $i \leftarrow i - 1$ 
else
    if  $i \leq 3$ 
         $i \leftarrow i + 2$ 
    end if
end if
```

5

Risultati sperimentali

Example of Table 5.1, made using <https://www.tablesgenerator.com/>.

A	B
1	2
3	4

Table 5.1: Interesting results.

6

Conclusioni

References

- [1] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining Data Streams*. Cambridge University Press, 2014, p. 123–153.
- [2] N. Alon, Y. Matias, and M. Szegedy, “The space complexity of approximating the frequency moments,” *Journal of Computer and System Sciences*, vol. 58, no. 1, pp. 137–147, 2002.
- [3] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan, “Counting distinct elements in a data stream,” in *Randomization and Approximation Techniques in Computer Science (RANDOM 2002)*, ser. Lecture Notes in Computer Science, vol. 2483. Springer, 2002, pp. 1–10.
- [4] S. Muthukrishnan, “Data streams: Algorithms and applications,” Rutgers University, Tech. Rep., 2005.
- [5] D. M. Kane, J. Nelson, and D. P. Woodruff, “An optimal algorithm for the distinct elements problem,” in *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2010)*. ACM, 2010.

Acknowledgments