Karlsruher Institut für Technologie

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 16 – Retrieval

Randomisierte Algorithmik

Aufgabe 1 – AMQ aus Retrieval

Sei $b \in \mathbb{N}$ und S eine Menge der Größe $n = |S|$. Verwende die Peeling-basierte Retrieval Datenstruktur der Vorlesung als Black-Box um einen statischen Filter (also eine Approximate-Membership-Query Datenstruktur) für S mit falsch-positiv Wahrscheinlichkeit $\varepsilon = 2^{-b}$ zu konstruieren.

Nenne Vor- und Nachteile der entstandenen Datenstruktur im Vergleich zu einem Bloom-Filter mit gleicher falsch-positiv Wahrscheinlichkeit. Du darfst annehmen, dass $b = O(\log n)$, sodass Bistrings der Länge b in Zeit $O(1)$ verarbeitet werden können.

Lösung 1

Sei $f \sim \mathcal{U}([2^b]^D)$ eine voll zufällige Hashfunktion. Gemäß der Simple Uniform Hashing Assumption können wir uns f merken ohne Speicher dafür aufzuwenden. Wir nennen $f(x)$ den *Fingerprint* von $x \in D$. Sei nun $f_S : S \to [2^b]$ die Einschränkung von f auf S („dieselbe Funktion“ aber kleinerer Definitionsbereich).

Wir konstruieren eine Retrieval Datenstruktur R für f_S. Wir fassen R als eine Filterdatenstruktur auf, die bei Anfrage $x \in D$ genau dann YES zurückgibt, wenn $\text{eval}(R, x) = f(x)$ gilt. Nach Konstruktion gibt es keine falsch-negativen Antworten. Nehmen wir nun an, dass $x \in D \setminus S$ gilt. Der Fingerprint $f(x)$ ist uniform zufällig in $[2^b]$ verteilt und unabhängig von allem was in der Konstruktion von R eine Rolle gespielt hat. Was auch immer $\text{eval}(R, x)$ ist, die Wahrscheinlichkeit, dass $f(x)$ damit übereinstimmt, und somit x ein falsch-positives Element ist, ist somit $2^{-b} = \varepsilon$ wie gewünscht.

Der Vergleich zu einem Bloom-Filter mit $\varepsilon = 2^{-b}$ fällt wie folgt aus:

- Geringerer Speicherbedarf: Rund $1.23kn$ statt rund $1.44kn$.
- Schnellere Konstruktion: $O(n)$ statt $O(nk)$.
- Schnellere Anfragen: $O(1)$ statt $O(k)$.
- Kein Unterstützung von `insert`.

Aufgabe 2 – Learned Data Structures

Sei S eine Menge von $n = |S|$ Namen mit eindeutig zuordenbaren Geschlecht $f : S \to \{\text{F}, \text{M}\}$. Eine findige Studentin bemerkt, dass die meisten $x \in S$ mit $f(x) = \text{F}$ auf einen Vokal enden und die meisten $x \in S$ mit $f(x) = \text{M}$ auf einen Konsonanten enden. Diese simple Regel funktioniert für alle außer δn der Namen, für ein kleines $\delta > 0$.

Konstruiere eine Datenstruktur mit erwartetem Speicherbedarf $O(\delta n \log(1/\delta))$, die für jedes $x \in S$ das korrekte Geschlecht $f(x)$ liefert.

Hinweis: Stöpsle dazu einen AMQ-Filter und eine Retrieval Datenstruktur geschickt zusammen.

Bemerkung: Unter *Learned Data Structures* versteht man eine Kombination aus klassischen Datenstrukturen und Machine Learning Techniken. Wie in dieser Aufgabe angedeutet, geht es darum, die Mustererkennungsfähigkeiten von Machine Learning Techniken mit den Verlässlichkeitsgarantien klassischer Datenstrukturen nutzbringend zu verheiraten.

Lösung 2

Sei $F \subseteq S$ die Menge der δn Namen, für die die Heuristik versagt. Sei B ein AMQ-Filter für F mit falsch-positiv Wahrscheinlichkeit δ. Sei nun S^+ die Menge derjenigen Namen aus S, für die der Filter eine positive Antwort gibt. Neben F enthält S^+ auch alle $x \in S$, die falsch-positive Elemente von B sind. Die erwartete Größe von S^+ beträgt höchstens $|F| + |S \setminus F| \cdot \delta = \delta n + (1-\delta)n \cdot \delta \leq 2\delta n$. Sei $f_{S^+} : S^+ \to \{\text{F}, \text{M}\}$ die Einschränkung von f auf S^+. Wir konstruieren eine Retrievaldatenstruktur R für f_{S^+}.

Die Intuition ist nun: Der AMQ-Filter identifiziert die Namen, für die die Heuristik fehlschlägt, sowie ein paar falsch-positive. Die Retrievaldatenstruktur hat dann zur Aufgabe echt-positive von falsch-positiven Elementen zu trennen. Formal können wir für unsere Heuristik-gestützte Retrievaldatenstruktur R_H definieren:

Algorithm eval(R_H, x)**:**
- **if** query$(B, x) =$ NO **then**
  - **return** Heuristik(x)
- **else**
  - **return** eval(R, x)

Der Gesamtspeicherbedarf beträgt $O(\delta n \log(1/\delta))$ Bits für B sowie erwartet höchstens $O(\delta n)$ Bits für R.

Aufgabe 3 – Retrieval mit Variabler Bitlänge

Gemäß Vorlesung können wir für jedes Universum D, jede Menge $S \subseteq D$ und jede Funktion $f : S \to \{0, 1\}$ eine Retrieval-Datenstruktur für f mit Speicherbedarf $1.23|S|$ konstruieren. Dies soll hier als Black-Box verwendet werden.

Konstruiere eine Retrieval Datenstruktur für den Fall in dem der Wertebereich der Funktion f Bitstrings variabler Länge enthält.

Genauergesagt, sei $C \subseteq \{0,1\}^*$ ein präfixfreier Code, D' ein Universum, $T \subseteq D'$ und $g : T \to C$ eine Funktion. Konstruiere eine Datenstruktur R mit Speicherbedarf $1.23 \cdot \sum_{x \in T} |g(x)|$ und einen zugehörigen Algorithmus eval sodass für jedes $x \in T$ gilt $\text{eval}(R, x) = g(x)$.
Hinweis: Führe für jedes $x \in T$ soviele Schlüssel ein, wie die Länge $|g(x)|$ von $g(x)$ beträgt.

Lösung 3

Die Idee ist es, für jedes $x \in T$ mit $g(x) = (b_1, \ldots, b_k)$ die Schlüssel $\{(x,1), (x,2), \ldots, (x,k)\} \subseteq D' \times \mathbb{N}$ einzuführen, wobei (x,i) der Wert b_i zugeordnet ist. Es ergibt sich somit eine Funktion $f : S \to \{0,1\}$ wobei $S \subseteq D' \times \mathbb{N}$ eine Menge von Paaren ist mit $|S| = \sum_{x \in T} |g(x)|$.

Wir können also gemäß Vorlesung für das Universum $D = D' \times \mathbb{N}$ sowie S und f wie beschrieben eine Retrieval Datenstruktur R mit Speicherbedarf $1.23|S|$ konstruieren. Um dieser Datenstruktur dann für $x \in T$ den Wert $g(x)$ zu entlocken, betrachten wir der Reihe nach $\text{eval}(R, (x,1)), \text{eval}(R, (x,2)), \text{eval}(R, (x,3)), \ldots$ bis die gesehene Folge einen Code aus C ergibt. Weil C präfixfrei ist, wissen wir, dass keine Fortsetzung der Folge in C liegt, also haben wir $g(x)$ vollständig bestimmt.

Bei einer Anfrage für $x \in D' \setminus T$ ist egal, welches Element von C zurückgegeben wird, wir müssen lediglich sicherstellen, dass eval nicht abstürzt oder in eine Endlosschleife gerät. Das ist leicht (und im Grunde fast automatisch der Fall).