Bloom Filter

Cloe Hüsser, Andreas Gassmann, Jonas Frehner

Idee

Beim Bloom Filter wird mithilfe von Hash-Funktionen analysiert, ob gewisse Daten in einem Datenstrom bereits einmal vorgekommen sind. Hierfür werden die Daten durch eine feste Anzahl verschiedener Hashfunktionen betrachtet und bei den erhaltenen Werten wird in einer Datenstruktur eine 1 gesetzt (alle Werte der Datenstruktur werden mit 0 initialisiert). Wenn ein neuer Datensatz eingelesen wird, wird mit den Hashfunktionen überprüft, ob an allen entsprechenden Stellen in der Datenstruktur bereits eine 1 steht. Ist dies nicht der Fall, ist der Datensatz garantiert neu. Andernfalls liegt die Wahrscheinlichkeit, dass die Daten bereits betrachtet wurden, nahe an einem vordefinierten Wahrscheinlichkeitswert p.

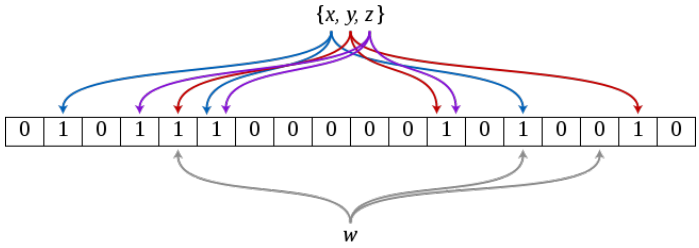

Abbildung 1: Bloom Filter, Quelle: David Eppstein, Wikipedia

Vorteile

- Der Algorithmus ist Speichereffizient
- Beim Überprüfen muss nicht die gesamte Datenstruktur durchsucht werden
- Falsch negative Werte sind nicht möglich

Nachteile

- Eingefügte Einträge können nicht mehr entfernt werden
- Falsch positive Werte sind möglich

Anwendung

Bloom Filter werden in der heutigen Zeit aufgrund der Speichereffizienz oft für grosse Datenbanken oder Netzwerke eingesetzt. Ein weiteres Einsatzgebiet sind Newsseiten, um anzuzeigen, welche Artikel eine Person bereits gelesen hat. So verwendet beispielsweise die Seite medium.com den Bloom Filter, um zu entscheiden, welche Artikel einer Person vorgeschlagen werden sollen. Hat eine Person einen Artikel betrachtet, so wird er mit dem Bloom Filter gespeichert und in den personalisierten Vorschlägen nicht mehr erscheinen. Mit diesem Vorgehen stellt medium.com sicher, dass eine Person nie einen bereits gelesenen Artikel vorgeschlagen bekommt. Allerdings besteht das Risiko, dass ein noch nicht gelesener Artikel nicht vorgeschlagen wird.

Unsere Implementation – Hash zu Index

Nachdem ein Hash eines Wertes erzeugt wurde, wird eine Funktion *getIntFromHash(HashCode c)* aufgerufen. Darin wird der Hashwert zuerst Modulo m (die Länge der Datenstruktur welche die Einträge des Bloom Filters speichert) gerechnet und anschliessend mit *Math.abs()* sichergestellt, dass der erhaltene Wert positiv ist.

Unsere Implementation – Test Fehlerwahrscheinlichkeit

Um zu überprüfen, wie hoch die Fehlerwahrscheinlichkeit ist, werden die Wörter aus dem File words.txt einzeln eingelesen und in den Bloom Filter eingefügt. Dann wird das File deutsch.txt eingelesen, welche ausschliesslich andere Wörter als words.txt enthält. Es wird überprüft, ob diese Wörter vom Bloom Filter als vorhanden markiert wurden. Anschliessend teilt der Test die Anzahl dieser „falsch positiven“ Werte durch die gesamte Anzahl Werte und berechnet so die Fehlerwahrscheinlichkeit. Ein Test gemäss diesem Schema zeigte, dass die tatsächliche Auftretenswahrscheinlichkeit „falsch positiver“ Werte nur sehr minimal von der erwarteten Wahrscheinlichkeit abwich.

Quellen:
https://medium.com/the-story/what-are-bloom-filters-1ec2a50c68ff#.8i6aka5se
https://de.wikipedia.org/wiki/Bloomfilter
https://en.wikipedia.org/wiki/Bloom_filter