Diskrete Stochastik - Programmieraufgabe Bloom-Filter

Nguyen, My-Nhien und Ravichandran, Pradip

November 27, 2018

1 Einleitung

1.1 Idee des Bloom-Filters

Ein Bloom-Filter ist eine probabilistische Datenstruktur, die ausgeben kann, ob ein gesuchtes Element in einem bestimmten Set vorhanden ist. Dabei ist das Ergebnis entweder ein *definitives Nein* oder ein *höchstwahrscheinliches Ja*. Dies geschieht indem sie eine Anzahl an geeigneten Hashfunktionen und eine geeignete Filtergrösse berechnet anhand der Fehlerquote und der Anzahl an erwarteten Elementen. Des weiteren ist ein Bit-Array der berechneten Filtergrösse besonders wichtig; jedes einzelne Set-Element wird dabei zu einem Index-Wert gehashed und im Array wird der Wert an diesem Index zu *1* gesetzt. (Das Daten-Array wurde zu Beginn mit Nullen aufgefüllt.) Das gesuchte Element wird nun ebenfalls gehashed und der entstandene Wert mit dem Daten-Array verglichen. Ist an diesem Index eine 0, ist der Wert sicherlich nicht vorhanden - ist dort eine 1, ist es vorhanden oder dieser Index stamm von einem anderen Element. Unsere eigene Implementation hat dabei einen boolean-Array verwendent und die Werte werden jeweils auf *true* gesetzt.

1.2 Vor- und Nachteile

Vorteile dieser Datenstruktur ist ihre Schnelligkeit und Speichereffizienz - mehrere Elemente besetzen einen Slot im Array, was viel Speicher sparen kann und sehr schnell aufzeigt, ob ein Element nicht vorhanden ist. Dagegen muss unbedingt beachtet werden, dass der Filter *nur* probabilistisch ist. Das heisst, er sagt aus, dass ein Element **wahrscheinlich** vorhanden ist, aber dies nicht mit 100-prozentiger Genauigkeit.

2 Beispiel aus der Praxis

Bloom-Filter können beispielsweise zur vorläufigen Erkennung von schädlichen URL's verwendet werden, wie es beim Webbrowser Google Chrome der Fall ist. Alle eingebenen URL's werden gehashed und mit der internen Liste verglichen. Diese URL ist nun entweder garantiert sicher (nicht in der List vorhanden) oder mit einer bestimmten Wahrscheinlichkeit schädlich. Im zweiten Fall wird die URL mit einem Extra-Request nochmals auf Server-Seite überprüft. Der Bloom-Filter braucht in diesem Fall so wenig Speicher, dass er problemlos auf dem Computer auf Client-Seite gespeichert werden kann.
Quelle: http://wufawei.com/2013/11/Bloom-Filter/

3 Testen der Fehlerwahrscheinlichkeit und Resultate

Wir haben unseren Bloom-Filter getestet, indem wir zuerst die words.txt-Datei dem Bloom-Filter als Basis übergeben haben. Bei einer weiteren .txt-Datei, die zum Beispiel nur aus deutschen Wörtern besteht und diese daher nicht im Bloom-Filter vorhanden sein können, haben wir auf jedes Wort die contains()-Methode aufgerufen und gezählt, wie viele Male **true** (Anzahl der *false positives*) zurückgegeben wird. Daraus haben wir auch die erreichte Fehlerquote berechnet. Zum Zeitpunkt des Schreibens dieses Dokuments wurden die untenstehenden Resultate erzielt.

3.1 Test-Resultate

Fehlerquote	Anzahl getesteter Wörter	false positives	erreichte Fehlerquote
0.1	189'951	39123	0.206
0.01	189'591	8376	0.044
0.001	189'591	2142	0.011