

# Bloomfilter

Markus Winter, Homas Müller, Thierry Hundt

December 2019

## 1 Vor- und Nachteile

### 1.1 Vorteile

- Es lässt sich mit 100 prozentiger Sicherheit aussagen ob ein Wort nicht vorkommt. Aus dem zu prüfenden Wort, werden Hashwerte generiert, diese Hashwerte bezeichnen je eine Position im Bitarray. Wenn an einer dieser Positionen eine 0 steht, ist das Wort definitiv nicht im Filter vorhanden.
- Der Bloomfilter benötigt im Vergleich mit ähnlichen Algorithmen sehr wenig Speicherplatz.

### 1.2 Nachteile

- Je kleiner das Bitarray ist, desto grösser ist die Wahrscheinlichkeit, dass ein Wort als enthalten markiert wird, obwohl es nicht im Filter hinzugefügt wurde.
- Der Filter kann nur aussagen, ob ein Wort sicher nicht oder ob es vielleicht enthalten ist.

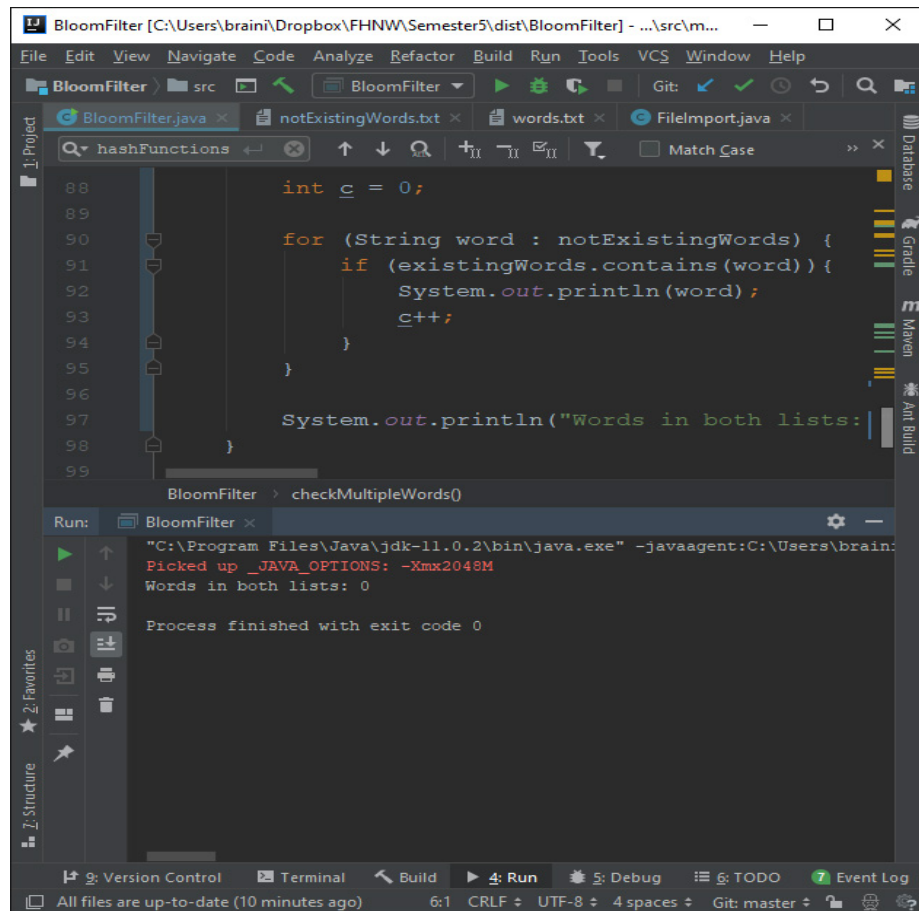
## 2 Praxisbeispiel

Beispiel Anwendung des Bloom Filters Google Chrome verwendet den Bloom Filter zu prüfen ob eine eingegebene URL “böswillig” ist. In einem ersten Schritt werden alle bekannten “bösenartigen” URLs in einer Hashtabelle gespeichert. Jedes mal wenn man an eine neue URL navigiert, wird diese mittels Bloom Filter in dieser Tabelle gesucht. Wenn der Bloom Filter positive retourniert, wird die URL genauer untersucht. Der Bloom Filter ist hier sehr gut geeignet, da er wenig Speicher benötigt und sehr schnell arbeitet. Er beschützt sozusagen das Google API für die Prüfung böswilliger Webseiten vor unnötigem Traffic und übertriebenem Arbeitsaufwand.

### 3 Testen Fehlerwahrscheinlichkeit

#### 3.1 1. Schritt – Wortliste erstellen

Als Grundlage haben wir die 10'000 meist verwendeten Wörter der deutschen Sprache verwendet. Die Wörter mit der Länge 1 und Wörter diejenigen welche einen Punkt enthalten wurden direkt herausgefiltert. Als nächstes haben wir diejenigen Wörter herausgefiltert, welche sowohl in der vorgegeben Wortliste als auch in der neu erstellten Wortliste vorkamen (Nur exakte Übereinstimmungen). Kurz gesagt, alle Duplikate wurden entfernt. Am Schluss beinhaltete die neue Wortliste dann 9636 Wörter, welche garantiert nicht in der vorgegebenen Wortliste vorkamen.



The screenshot shows an IDE window titled "BloomFilter [C:\Users\braini\Dropbox\FHNW\Semester5\dist\BloomFilter] - ...src\m...". The editor displays the following Java code in `BloomFilter.java`:

```
88     int c = 0;
89
90     for (String word : notExistingWords) {
91         if (existingWords.contains(word)) {
92             System.out.println(word);
93             c++;
94         }
95     }
96
97     System.out.println("Words in both lists: ");
98 }
99
```

The code is part of a method `checkMultipleWords()`. The IDE also shows a "Run" configuration for `BloomFilter` with the command:

```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" -javaagent:C:\Users\braini\...
```

The output of the run is displayed in the console:

```
Picked up _JAVA_OPTIONS: -Xmx2048M
Words in both lists: 0
Process finished with exit code 0
```

The IDE interface includes a sidebar with "Project", "Database", "Gradle", "Maven", and "Ant Build" views. The bottom status bar shows "All files are up-to-date (10 minutes ago)", "6:1", "CRLF", "UTF-8", "4 spaces", and "Git: master".

### 3.2 2. Schritt – Richtig-Positiv-Test

Um die Richtig-Positiven Wörter zu testen wurde ganz einfach die vorgegebene Wortliste mit dem BloomFilter getestet. Im BloomFilter eingelesen wurde ebenfalls die vorgegebene Wortliste. Es müsste also jedes einzelne Wort positiv getestet werden. Dies war dann auch so. 58109 von 58109 Wörter wurden positiv getestet.

```
Parameter BloomFilter
n (Anzahl Wörter in Wortliste):      58109
m (Länge des Bitarrays):              640812
k (Anzahl Hashfunktionen):           8
p (Gegebene Fehlerwahrscheinlichkeit): 0.5%
-----
Anzahl enthaltene Wörter:             58109
Anzahl Treffer in BloomFilter:        58109
Trefferquote:                        100%
```

### 3.3 3. Schritt – Falsch-Positiv-

Nun wurde die neu erstellte Wortliste, welche ausschliesslich Wörter enthält, welche in der vorgegeben Liste garantiert nicht vorkommen, getestet. Würde der Filter auch bei nicht enthaltenen Wörter zu 100% für übergebene Fehlerwahrscheinlichkeiten unter 0.0001 (0.01

```
Anzahl nicht enthaltene Wörter:      9636
Anzahl Treffer in BloomFilter:        937
Übergebene Fehlerwahrscheinlichkeit:  10.0%
Experimentelle Fehlerwahrscheinlichkeit: 9.72%
-----
Anzahl nicht enthaltene Wörter:      9636
Anzahl Treffer in BloomFilter:        111
Übergebene Fehlerwahrscheinlichkeit:  1.0%
Experimentelle Fehlerwahrscheinlichkeit: 1.15%
-----
Anzahl nicht enthaltene Wörter:      9636
Anzahl Treffer in BloomFilter:         11
Übergebene Fehlerwahrscheinlichkeit:  0.1%
Experimentelle Fehlerwahrscheinlichkeit: 0.11%
-----
Anzahl nicht enthaltene Wörter:      9636
Anzahl Treffer in BloomFilter:         0
Übergebene Fehlerwahrscheinlichkeit:  0.01%
Experimentelle Fehlerwahrscheinlichkeit: 0%
-----
Anzahl nicht enthaltene Wörter:      9636
Anzahl Treffer in BloomFilter:         0
Übergebene Fehlerwahrscheinlichkeit:  0.001%
Experimentelle Fehlerwahrscheinlichkeit: 0%
```

### 3.4 Fazit

Je nach Anwendung des Filters, kann eine sinnvolle Fehlerwahrscheinlichkeit übergeben werden. Wenn die Berechnung des neuen Wertes bei einem Falsch-Positiven Resultat sehr schnell geht, kann die Fehlerwahrscheinlichkeit höher gewählt werden. D.h. es werden mehr Werte als bereits vorhanden gewertet und es muss ein neuer Wert berechnet werden. Dauert die Berechnung des neuen Wertes allerdings sehr lange, ist es sinnvoll mit einer sehr kleinen Fehlerwahrscheinlichkeit zu arbeiten, was den Filter allerdings auch langsamer macht. In unserem Fall haben wir uns für eine Fehlerwahrscheinlichkeit von etwa einem Prozent entschieden. Nachfolgend die endgültige Ausgabe.

```
Parameter BloomFilter
n (Anzahl Wörter in Wortliste):      58109
m (Länge des Bitarrays):              835468
k (Anzahl Hashfunktionen):           10
p (Gegebene Fehlerwahrscheinlichkeit): 0.1%
-----
Anzahl nicht enthaltene Wörter:       9636
Anzahl Treffer in BloomFilter:        11
Experimentelle Fehlerwahrscheinlichkeit: 0.11%
```