3nd AA Project — Bloom Filter

Rui Fernandes

***Abstract* –This report seeks to present how the author coded and tested a Bloom Filter and it's appliance to unique word counting. The chosen coding language was Python(3.7), and the algorithms will be shortly explained before diving into an exposition of results.**

***Keywords* –Bloom Filter, Hash Function, Counter**

I. Introduction

The problem presented for this project and that this report seeks to explain is the application of Bloom Filters to estimate the number of unique words in a given text file.

The corpus used for testing the Bloom Filter, consisted in the manuscript of "The Odyssey" by Homer, in English.

To test the Bloom Filter counting, it was ran trough the text file with varying values for the filter size, number of hash functions and the type of hash function, totalling 360 runs.

With the report comes the code files used and the result files obtained, in terms of results there's a file that keeps the exact count of unique words, and a single file for the 360 results, saved in json format, ordered by type of hash, then number of functions and finally size.

The code is split and comprised into various scripts:

- 'bloom_filter' - The bloom filter class provided by the professor, with small alterations.
- 'exact_counter' - Simple counter that runs trough every word in the file and count unique words.
- 'bloom_counter' - Applies the bloom filter to counting the unique words of the text file, returning the estimate count.
- 'build_stats' - Utility script that runs the counter trough the text file with all variables to test.
- 'build_graphs' - Utility script that creates graphs based on the result file.

To remake and display the results:

```
$ python build_stats.py
$ python build_graphs.py
```

II. The Problem

This problem is simple in nature, and the code was mostly provided, rather it's more of a collecting and comparing of data, to verify how varying certain values can affect the estimated count.

The estimation is made by going word by word and checking if the given word is already in the Bloom Filter, and if not, inserting it, this of course is where the Bloom Filter can be erroneous, for there is a small chance that a false positive happens, and so a word won't be added to the count.

Three variables were experimented with, firstly the type of hash, the hash functions used were the standard one provided by python builtins, then the MurmurHash3 trough the mmh3 library, and finally the Fowler–Noll–Vo hash trough the fnv library. The second variable was the number of hash functions used in the Bloom Filter, the values tested went from 5 to 10. Finally the size of the filter's array was changed increasingly in 50% increments with the base being the exact count of unique words previously obtained, so going from 100% of this value, to 1050% of it.

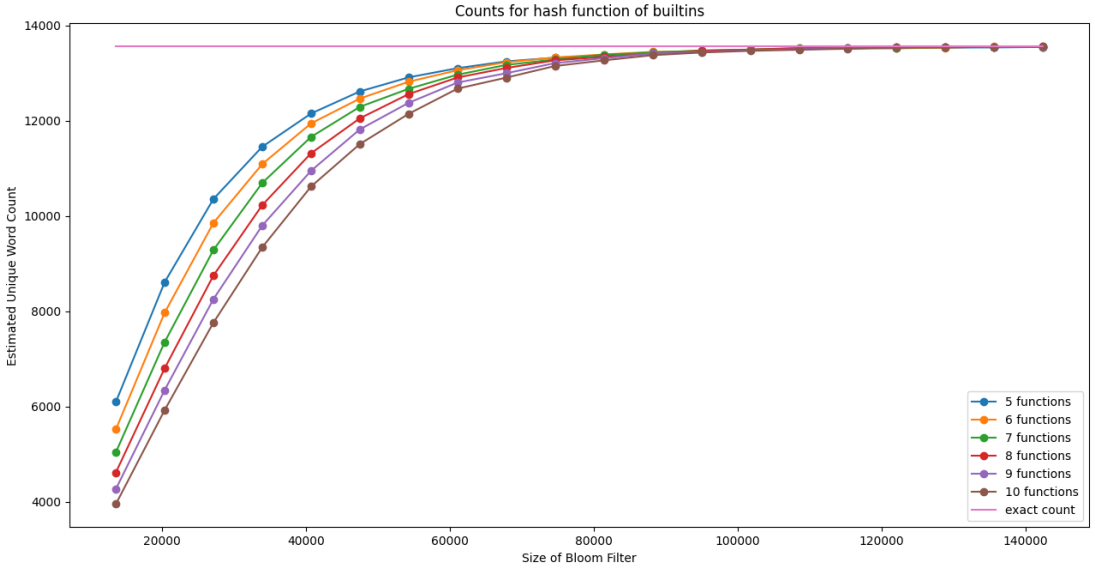
III. Results Discussion

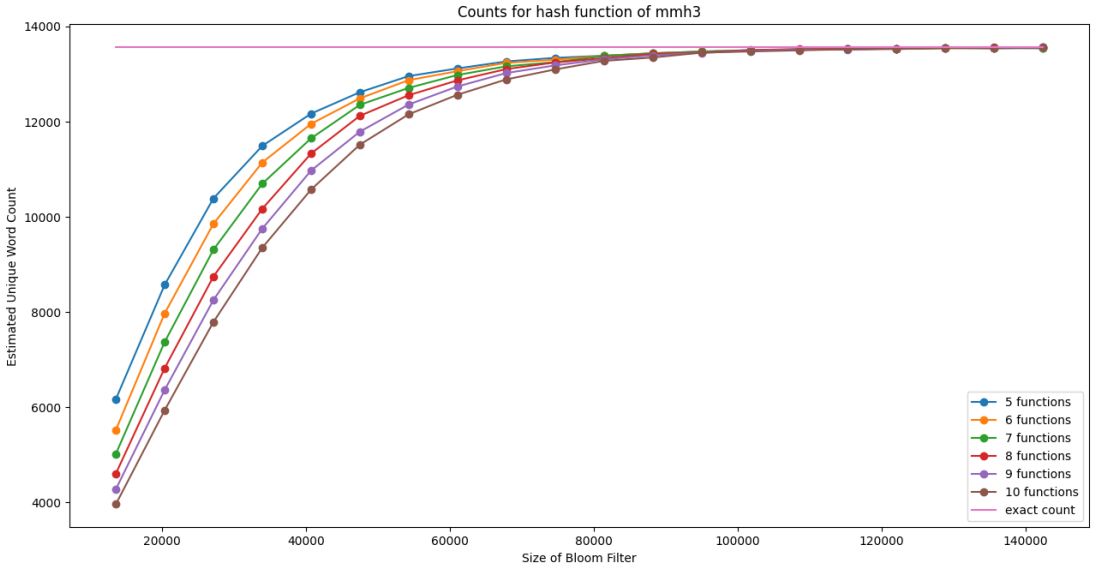
The results are quite easy to ascertain and analyse, three graphs were made, one per hash type, each line represents a number of functions used, and each point the size used and corresponding estimated count.

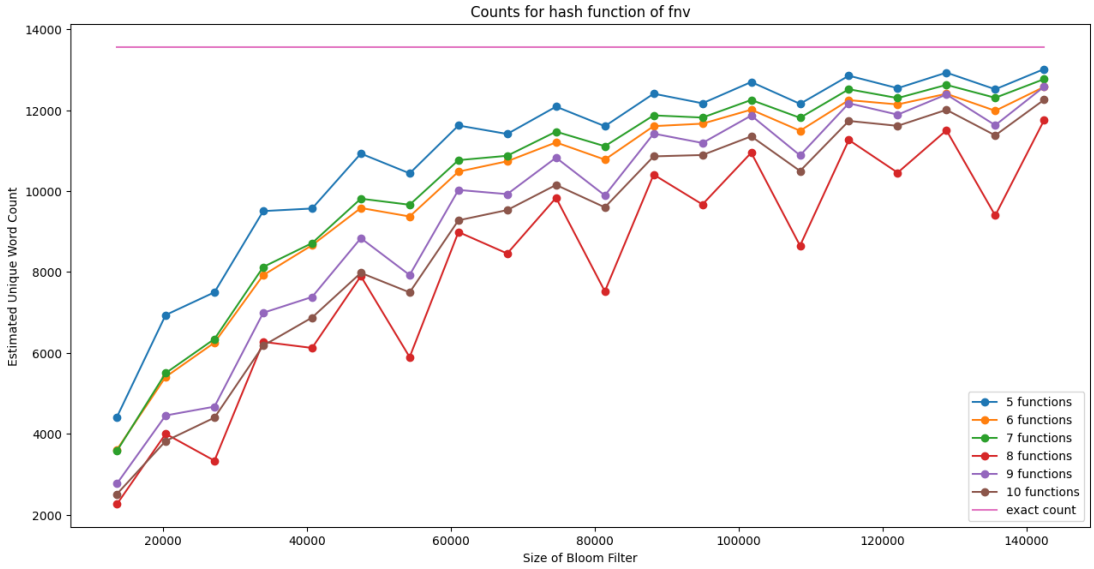
We can see that in general increasing the size of the filter's array gives us better results, to be expected since there are more spaces to be filled and so much less overlap, leading to less false positives. The inverse can be said of the number of hash functions, since increasing this number makes it so more bytes are set to 1 per word inserted.

Finally, the most interesting observation is that the fnv seems to be the less viable hash function option, it's much more volatile than the others and it doesn't even reach stabilization with the exact count within these experiments. On the contrary, both the others have extremely similar results and reach a stable point at around 100k size.

IV. Graphs


Counts for hash function of builtins
Estimated Unique Word Count
Size of Bloom Filter
14000
12000
10000
8000
6000
4000
20000
40000
60000
80000
100000
120000
140000
5 functions
6 functions
7 functions
8 functions
9 functions
10 functions
exact count


Counts for hash function of mmh3
Estimated Unique Word Count
Size of Bloom Filter
14000
12000
10000
8000
6000
4000
20000
40000
60000
80000
100000
120000
140000
5 functions
6 functions
7 functions
8 functions
9 functions
10 functions
exact count


Counts for hash function of fnv
Estimated Unique Word Count
Size of Bloom Filter
14000
12000
10000
8000
6000
4000
2000
20000
40000
60000
80000
100000
120000
140000
5 functions
6 functions
7 functions
8 functions
9 functions
10 functions
exact count