On the usage of approximate counters for the occurrences of words

Manuel Gomes, 88939

***Abstract* –This report present three counter algorithms to solve a problem consisting of counting the number of words inside different editions of the same book. The first one is an exact while the rest are based on probabilistic counters. After their implementation in Python 3, an experimental analysis of each algorithm is described.**

***Resumo* –Este relatório apresenta três algoritmos de contadores para resolver um problema consistente em contar o número de palavras dentro de diferentes edições do mesmo livro. O primeiro é um algoritmo exato, enquato que os restantes são baseados em contadores probabilísticos. Após a sua implementação em Python 3, uma análise experimental de cada algoritmo é descrita.**

I Introduction

The need for gathering statistics on a large number of events is a very common occurrence in data science applications, such as social media networks and search engines. However, the resources are finite and the totality of data to analyse is larger than the available memory. The largest challenge in this area is to store the data in a storage efficient way. One solution for this problem is the usage of probabilist counter algorithms. In these algorithms, an occurrence is not always accounted, being dependent on a probabilty. Although this reduces the memory used, the accuracy is reduced as well.

The objective behind this report is to explore the accuracy and efficiency behind some probabilistic counters in counting the occurrence of words inside a book. The report is divided in five section: the first (section I), where the problem is introduced; the second (section II), where the algorithms used are described; the third (section III), where the experiments are described and results for them are detailed; and the fourth (section IV), where conclusions are extracted.

II Algorithms Used

To count the number of occurrences of every word in a book, an exact counter and two different probabilistic counters were used. The two different probabilist counters were a fixed probability counter and a Csűrös' counter.

II-A Exact Counter Algorithm

An exact counter is a data structure that maintains an accurate count of unique items in a stream of data. In this specific case, the counter maintains an accurate count of words inside a book.

In this algorithm, an empty data structure is created. This data structure creates a subdivision for every new word accounted for. Inside that subdivision, an integer keeps count of every account of that specific word. This can be better visualised in Algorithm 1.

Algorithm 1 Exact counter algorithm

```
Inputs:
    book ← string of text to be analysed
Initialize:
    counter ← data structure used for
    counting the occurrences of every word
for word in book do
    if word in counter then
        counter[word]+ = 1
    else
        counter[word] ← data structure
    end if
end for
```

This algorithm allows for an exact count of unique items, with no error margin. The downside of using an exact counter is that it can use a lot of memory.

II-B Fixed probabilistic counter

A fixed probabilistic counter is a data structure that keeps count of the number of distinct elements in a stream of data, with a certain level of precision. It employs random sampling to determine the count and the degree of accuracy can be altered by altering the probability of sampling. This probabilty remains constant for the entire duration of the algorithm.

Similarly to Algorithm 1, an empty data structure is created. Every time a new word is analysed, a random number from 0 to 1 is generated. If this number is lower than the sampling probability, then the word is sampled. If a subdivision for the sampled word does not exist inside the data structure, this division is created. Inside that subdivision, an integer keeps count of every account of that specific sampled word. This algorithm is explained in Algorithm 2.

For a specific number of occurrences x with p being the sampling probability, to calculate the expected count $f(x)$, it is used:

$$f(x) = \frac{x}{p} \tag{1}$$

Algorithm 2 Fixed probability counter algorithm

```
Inputs:
    book ← string of text to be analysed
    p ← sampling probability
Initialize:
    counter ← data structure used for
    counting the occurrences of every word
    RandomNumber ← function that
    generates a number between 0 and 1
for word in book do r_number = RandomNumber
    if r_number < p then
        if word in counter then
            counter[word]+ = 1
        else
            counter[word] ← data structure
            counter[word] = 1
        end if
    end if
end for
```

II-C Csűrös' probabilty counter algorithm

Another type of probabilistic counter are the counters with decreasing probabilty. A decreasing probabilistic counter is, similarly to a fixed probabilistic counter, a data structure that keeps count of the number of distinct elements in a stream of data, with a certain level of precision. It also employs random sampling to determine the count. However, the sampling probabilty decreases as the algorithm evolves. This allows for an accurate count in the first occurrences of an event without using to much memory.

One example of this type of counter is the Csűrös' counter. This is a floating-point counter defined with the aid of a design parameter $M = 2^d$, where d is a nonnegative integer. This counter counts with deterministic updates for the first M occurrence, similarly to Algorithm 1. The next updates are probabilistic, with the next M updates having a probability of $\frac{1}{2}$, followed by M updates with probabilty of $\frac{1}{4}$, etc... This algorithm is described in Figure 1 and Algorithm 3.

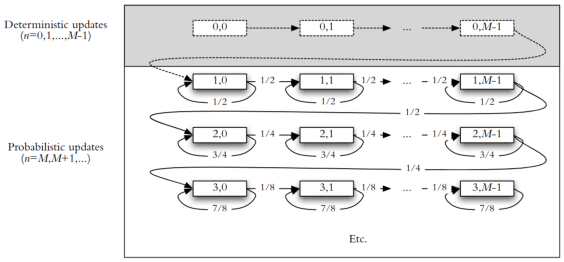

Fig. 1
CSŰRÖS' COUNTER EXEMPLIFIED IN GRAPHICAL FORM.

For a specific number of occurrences x with p being the design parameter and $M = 2^p$, to calculate the expected count $f(x)$, it is used:

$$x = (M + u)2^t - M \tag{2}$$

Algorithm 3 Csűrös' counter algorithm

```
Inputs:
    book ← string of text to be analysed
    d ← design parameter
Initialize:
    counter ← data structure used for
    counting the occurrences of every word
    RandomNumber ← function that
    generates a number between 0 and 1
    t ← 0
    u ← 0
for word in book do
    r_number = RandomNumber
    if r_number < (1/2)^t then
        if word in counter then
            counter[word]+ = 1
            u+ = 1
        else
            counter[word] ← data structure
            counter[word] = 1
            u+ = 1
        end iffor a specific number of occurrences x,
with p being the sampling probability:
    end if
    if u == 2^d then
        t+ = 1
        u = 0
    end if
end for
```

$$f(x) = Mt + u \tag{3}$$

with u and t representing the variables in Algorithm 3.

However, these equations are not enough because we have three unknown variables $f(x)$, t, and u, and only two different equations. Nevertheless, it is known that u is not larger than 8 and t is the number of times u hits the number 8. Therefore, the following algorithm was developed:

Algorithm 4 Csűrös' estimation algorithm

```
Inputs:
    x ← specific number of occurrences
    d ← design parameter
Initialize:
    t ← 0
    u ← 0
    f(x) ← 0
while x > 0 do
    f(x)+ = (1/2)^t
    u+ = 1
    x− = 1
    if u == 2^d then
        t+ = 1
        u = 0
    end if
end while
```

III Results

After implementing the algorithms described in section II, experimental results were retrieved. These results range from top-20 most frequent words to statistics for the top-3 words. The results were taken in a machine with the AMD Ryzen 5 5600X processor and implemented in Python 3. The experiments were taken using the book *Manifesto of the Communist Party*, by Karl Marx and Friedrich Engels, due to its historical importance, various number of editions and availability of various editions for free-use. The three editions used will be described from now on as *2004 edition*, *2005 edition*, and *2010 edition*. Every edition had every punctuation mark removed, as well the most common stop-words. Each edition had one thousand runs of the approximate counters algorithms, in order to retrieve a vast number of results. The fixed probability counter used a probability of $\frac{1}{4}$, while the Csűrös' counter used a d of 3.

III-A 2004 edition

For the 2004 edition, the top-20 words calculated by the exact counter, an average of one thousand fixed probability counters, and an average of one thousand Csűrös' probability counters are shown in Figure 2, Figure 3, and Figure 4, respectively. From these graphs we can observe that the top-6 words are the same in every case. The first discrepancy seen is on the sixth position of Figure 3, where *production* overtakes *conditions*. Both approximate counters present only four different relative positions in the graphs. Regarding the same words present in the top-20, the Csűrös' counter graph present the word *development* in favour of the word *communists*.

Table I presents the statistics of the top-3 words for the 2004 edition. In this statistics we can observe that the difference between the expected and the mean counter values is slim. It is also verifiable that the fixed probability counter present a larger standard deviation than the Csűrös's counter. The mean relative error is smaller in the Csűrös' counter, while the mean accuracy ratio stands closer to the number one on the fixed probability counter.

TABLE I
Statistics of the top-3 words for the 2004 edition, according to the fixed probability counter (Fixed) and the Csűrös' Counter (Csűrös).

Word	Class		Bourgeois		Bourgeoisie	
Counter Type	Fixed	Csűrös	Fixed	Csűrös	Fixed	Csűrös
Real Value	137		110		99	
Expected Counter Value	34.25	33.06	27.5	30.75	24.75	29.38
Mean Counter Value	34.19	32.78	27.68	30.41	24.72	29.2
Standard Deviation	4.96	2.24	4.52	2.26	4.25	2.44
Maximum Absolute Error	18.75	9.06	15.5	6.75	12.25	8.62
Mean Absolute Error	3.95	1.72	3.675	1.88	3.41	2.03
Mean Relative Error	0.115	0.051	0.134	0.061	0.138	0.069
Mean Accuracy Ratio	0.998	0.991	1.006	0.989	0.999	0.994
Smallest Counter Value	19	24	15	24	14	24
Largest Counter Value	53	40	43	37	37	38

Additional data retrieved during the experience by using the `sys.getsizeof()` function states that the exact counter takes 147568 bytes of memory, while the

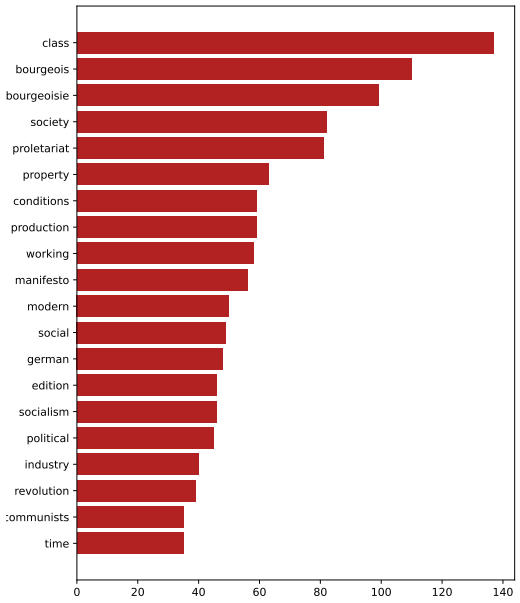

Fig. 2
2004 edition's top-20 words according to the exact counter.

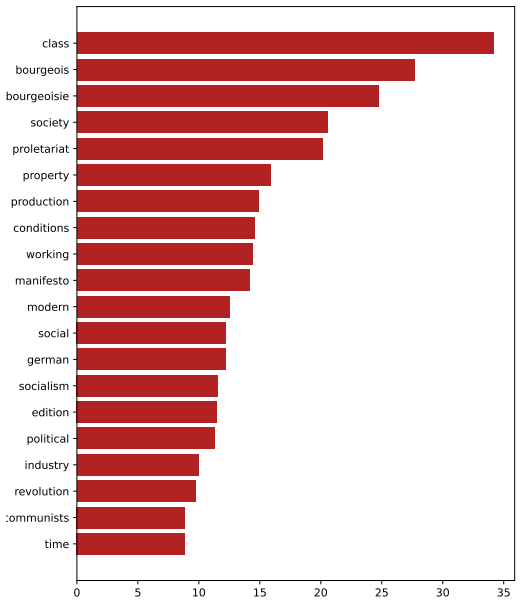

Fig. 3
2004 edition's top-20 words according to the average of one thousand fixed probability counters.

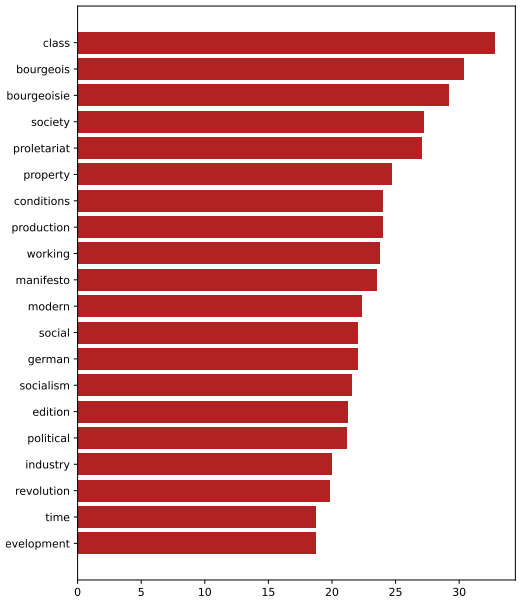

Fig. 4
2004 EDITION'S TOP-20 WORDS ACCORDING TO THE AVERAGE OF ONE THOUSAND CSŰRÖS' COUNTERS.

approximate counters only take on average 40 bytes.

III-B 2005 edition

For the 2005 edition, the top-20 words calculated by the exact counter, an average of one thousand fixed probability counters, and an average of one thousand Csűrös' probability counters are shown in Figure 5, Figure 6, and Figure 7, respectively. From these graphs we can observe that the top-9 words are the same in every case. The first discrepancy seen is on the sixth position of Figure 6 and Figure 7, where *production* overtakes *conditions*. Both approximate counters present only four different relative positions in the graphs. Every graph shares the same top-20 words.

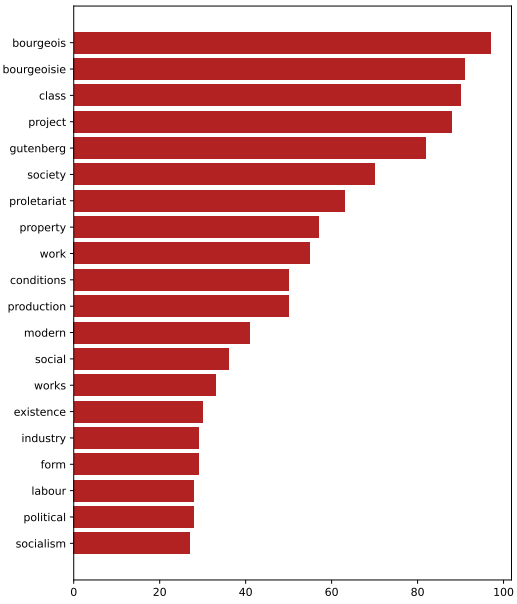

Fig. 5
2005 EDITION'S TOP-20 WORDS ACCORDING TO THE EXACT COUNTER.

Table II presents the statistics of the top-3 words for the 2010 edition. In this statistics we can observe that the difference between the expected and the mean counter values is slim. Similarly to Table I, the fixed probability counter present a larger standard deviation than the Csűrös's counter and the mean relative error is smaller in the Csűrös' counter. The mean accuracy ratio is similar between counters.

Similarly to the previous edition, additional data retrieved during the experience by using the `sys.getsizeof()` function states that the exact counter takes 147568 bytes of memory, while the approximate counters only take on average 40 bytes.

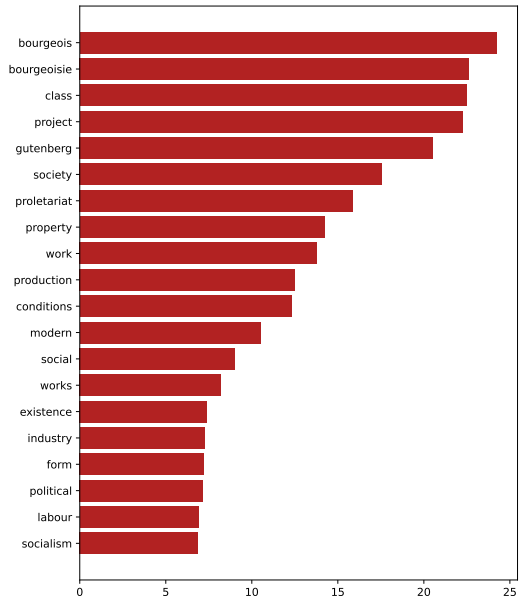

Fig. 6
2005 edition's top-20 words according to the average of one thousand fixed probability counters.

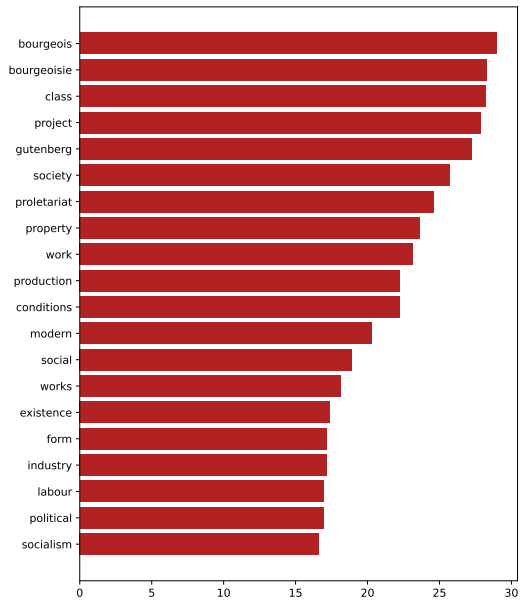

Fig. 7
2005 edition's top-20 words according to the average of one thousand Csűrös' counters.

TABLE II
Statistics of the top-3 words for the 2005 edition, according to the fixed probability counter and the Csűrös' Counter.

Word	Bourgeois		Bourgeoisie		Class	
Counter Type	Fixed	Csűrös	Fixed	Csűrös	Fixed	Csűrös
Real Value	97		91		90	
Expected Counter Value	24.25	29.12	22.75	28.38	22.50	28.25
Mean Counter Value	24.23	28.98	22.621	28.281	22.49	28.2
Standard Deviation	4.34	2.36	3.93	2.27	4.14	2.23
Maximum Absolute Error	13.75	6.88	13.25	6.62	15.50	6.25
Mean Absolute Error	3.466	1.96	3.14	1.87	3.31	1.81
Mean Relative Error	0.1429	0.067	0.138	0.066	0.147	0.641
Mean Accuracy Ratio	0.999	0.995	0.994	0.997	1.000	0.998
Smallest Counter Value	12	23	11	23	11	22
Largest Counter Value	38	36	36	35	38	34

III-C 2010 edition

For the 2010 edition, the top-20 words calculated by the exact counter, an average of one thousand fixed probability counters, and an average of one thousand Csűrös' probability counters are shown in Figure 8, Figure 9, and Figure 10, respectively. From these graphs we can verify that they all share the same words in the same relative position.

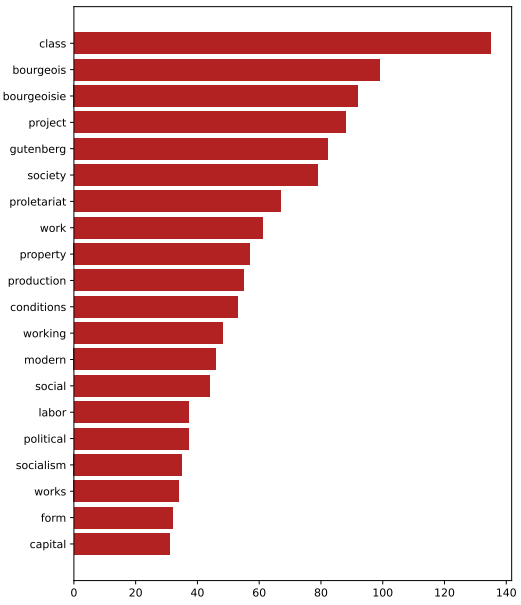

Fig. 8
2010 edition's top-20 words according to the exact counter.

Table III presents the statistics of the top-3 words for the 2010 edition. In this statistics we can observe that the difference between the expected and the mean counter values is slim, as seen in previous counters. The observations taken for Table II can be taken here as well, from the larger standard deviation on the fixed probability counter to the smaller mean relative error in the Csűrös' counter.

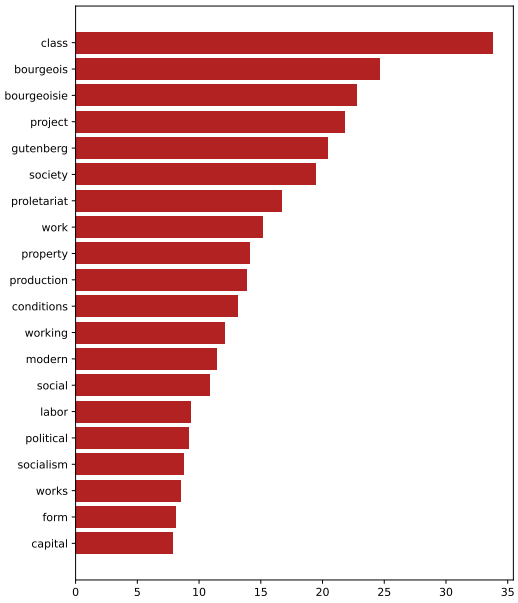

Fig. 9
2010 EDITION'S TOP-20 WORDS ACCORDING TO THE AVERAGE OF ONE THOUSAND FIXED PROBABILITY COUNTERS.

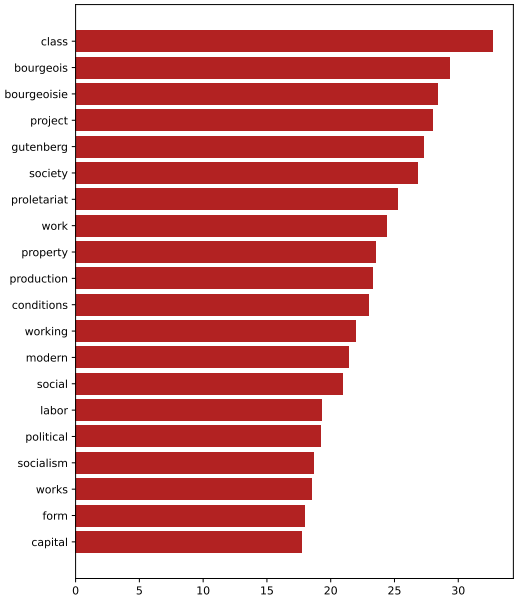

Fig. 10
2010 EDITION'S TOP-20 WORDS ACCORDING TO THE AVERAGE OF ONE THOUSAND CSŰRÖS' COUNTERS.

Similarly to the previous subsections, additional

TABLE III
STATISTICS OF THE TOP-3 WORDS FOR THE 2010 EDITION, ACCORDING TO THE FIXED PROBABILITY COUNTER AND THE CSŰRÖS' COUNTER.

Word	Class		Bourgeois		Bourgeoisie	
Counter Type	Fixed	Csűrös	Fixed	Csűrös	Fixed	Csűrös
Real Value	135		99		92	
Expected Counter Value	33.75	32.94	24.75	29.38	23.00	28.50
Mean Counter Value	33.77	32.69	24.67	29.33	22.79	28.38
Standard Deviation	4.97	2.28	4.37	2.32	4.14	2.32
Maximum Absolute Error	17.25	9.06	15.25	7.62	13.00	7.50
Mean Absolute Error	3.95	1.70	3.45	1.91	3.31	1.90
Mean Relative Error	0.117	0.052	0.140	0.065	0.144	0.067
Mean Accuracy Ratio	1.000	0.992	0.997	0.999	0.991	0.996
Smallest Counter Value	18	24	12	22	11	21
Largest Counter Value	51	42	40	37	36	36

data retrieved during the experience by using the `sys.getsizeof()` function states that the exact counter takes 147568 bytes of memory, while the approximate counters only take on average 40 bytes.

IV Conclusions

In this report, three different algorithm were presented to count the number of words inside a book. The first one is an exact algorithm, counting every occurrence of every word, prioritizing accuracy to efficient memory usage. The second is a fixed probability counter, which reduces accuracy in order to occupy less memory. This can however lead to large inaccuracies. To tackle this, the third counter, a Csűrös' counter was implemented. In this counter, the probability of counting a word decreases with the number of equal words counted.

Results show that, on average, both probabilistic counters are accurate. However, the fixed probability counter presents itself with a larger standard deviation. This is due to the chaotic nature of the sampling, following only a fixed probability. The Csűrös' counter has a lower standard deviation, which is to be expected, due to the probability of counting a word decreases with number of counts, leading to more severe probabilities for the top words. This severe probability creates an environment where augmenting your counter does not happen very often, leading to counter values being less spread out. This effect can also be visualized in the mean relative error, which is considerably smaller in the Csűrös' counter. Some can argue that the mean accuracy ratio is worse than the one presented by the fixed probability counter, which is true, but only shows that the average value of the fixed counter stays closer to the expected one, which does hold significant value when the variance is high. It also needs to be taken into account that due to reasons previously explained the Csűrös' counter is *low-biased*, achieving results that, on average, are smaller than the ones expected. This can be visualized on the mean accuracy error, never surpassing 1. Nevertheless, both of the probabilistic counters occupy much less memory than the exact one.

Results also show a difference between different editions of the same book, especially the ones retrieved from Project Gutenberg, where the words *project* and *gutenberg* appear in the top-10.

Possible work on this subject would be the implementation of more types of counters, such as the Morris' counter, and comparing them with the results obtain in here. Another possible line of work would be to invest in larger computational experiments to better validate these results.

References