# Advanced Algorithms

## 3rd Project – Most Frequent Letters Counters

Simão Teles Arrais – 85132
Departamento de Eletronica, Telecomunicações e Informática
Universidade de Aveiro

***Resumo* – Este trabalho tem como objetivo documentar o desenvolvimento e análise sobre três tipos diferentes de contagem: um contador exato, um contador de aproximação e um frequent counter Misra-Gries. No final é realizado uma análise dos resultados obtidos.**

***Abstract* – This paper aims to document the development and analysis of three different types of counters: an exact counter, an approximation counter and a frequent counter Misra-Gries. At the end, it is performed an analysis of the results obtained.**

### I. Introduction

In this study, we investigate three methods for determining the most common letters in text files. The objective is to assess the performance of these methods in terms of efficiency and accuracy of letter count estimates. To accomplish this, we used the literary work "The Picture of Dorian Gray" by Oscar Wilde in English, Finnish, French, and Dutch, obtained from Project Gutenberg. Three strategies were then implemented - exact counters, approximate counters, and a data stream algorithm Misra-Gries to calculate the number of occurrences of each letter and estimate the top k most frequent letters for various values of k.

### II. Data Processing

Before doing any experiments it is necessary to process any text files in order for them to be considered valid. A text file is considered valid once it is removed all the Project Gutenberg headers, remove all stop-words and punctuation marks and finally, convert all letters to uppercase.

To ease the process of removing the Project Gutenberg headers, all of them were removed manually. This ensured a clean removal of all header information without hurting the literary work itself. The top header was super inconsistent on where it started and ended and neither the number of lines nor the text of the header itself were the same across all languages. Only the bottom header seemed to be somewhat consistent but since its removal manually was not time consuming and the beginning of the header was not always stated, it was best to just to apply the same treatment.

Next, it is needed to remove all stop-words, punctuation marks and capitalize all letters. To achieve this, an iteration on the book file is done and it is applied a regular expression (previously compiled) that only matches to one or more characters. This returns all the current words so it is possible to do a check if they match any of the stop-words and delete them if so. The only thing left to do is for every word, isolate all the letters and apply an uppercase.

```
    self.regex_words = re.compile(r"[a-zA-ZÀ-ÿ']+")
    self.regex_letters = re.compile(r"[a-zA-ZÀ-ÿ]")

def process_data(self, contents, stop_words):
    # Get all words and filter through stop-words
    pre_words = self.regex_words.findall(contents)
    words = [w for w in pre_words if not w in stop_words]

    # Get all letters
    letters_list = []
    for w in words:
        letters = self.regex_letters.findall(w)
        letters_list.extend(letters)

    # Apply upper case
    upper_letters = [lettter.upper() for lettter in letters_lis
    return upper_letters
```

*Figure 1: Python code that pre processes the data*

### III. Exact Counter

The exact counter approach is fairly simple, it does a normal exact count of the number of letters. To achieve that, it makes use of class *Counter* from *Collections* library. It is a matter of giving the letters as input and it returns an object similar to a dictionary with the bonus of having more information.

```
def exact_counter(self, contents):
    freq = collections_counter(contents)
    return freq
```

*Figure 2: Exact Counter code*

### IV. Approximate Counter

The assigned approach to use as an approximate counter is a fixed probability counter with a probability equal to 1/16.

The algorithm built receives the letters and the probability as input and randomly generates a number. For

each letter, if the generated number is lower or equal to 1/16, the letter will be incremented, otherwise, the counter will remain the same.

```
def approximate_counter(self, contents, prob = 1/16):
    freq_counter = collections_counter()

    for letter in contents:
        if random.random() <= prob:
            if letter in freq_counter:
                freq_counter[letter] += 1
            else:
                freq_counter[letter] = 1

    for letter in freq_counter:
        freq_counter[letter] /= prob

    return freq_counter
```

*Figure 3: Approximation Counter*

At the end, the algorithm multiplies all counters by the probability that it was used in order to have normalized results to compare to the rest of the counter algorithms developed.

## V. Error Calculation

An algorithm was devised to calculate errors by running the Approximate Counter a specified number of times (default is 10).For each iteration, the minimum, maximum and average absolute and relative errors for each letter were calculated. Later in the paper we will be able to see what happens when we instead, run the algorithm 100 times.

## VI. Data Stream Algorithm: Misra-Gries

The Data Stream algorithm used in this work is a frequent-count algorithm called Misra & Gries algorithm. It is used to identify the k most frequent items in a stream, where k is a parameter of the algorithm that dictates the number of distinct elements that cab be stored in the algorithm's data structure. It maintains a count of the k most frequent items.

```
def misra_gries_counter(self, stream, k):
    # Initialization
    A = collections_counter()

    # Processing
    for j in stream:
        if j in A:
            A[j] += 1
        elif len(A) < k - 1:
            A[j] = 1
        else:
            for i in list(A.keys()):
                A[i] -= 1
                if A[i] == 0:
                    del A[i]

    return A
```

*Figure 4: Misra-Gries algorithm code*

When the number of distinct elements exceeds k, the algorithm discards the element with lowers count. For each element, If it is not in the dictionary, it is added with a count of 1. If the dictionary is full, the counter for the least frequent element is decremented by 1, and if it hits 0, the element is removed from the list. The algorithm will return the top k most frequent elements sorted by counter.

## VII. Results

In order to get these results, for each language file book the Exact Counter and Misra-Gries were executed a single time and the Approximate Counter ten times. The decision behind executing the Approximate Counter more times is to be able to calculate an average and other variables like absolute error, relative error, etc, between results that were randomly generated.

In this section we will expose all the results and plots that were obtained during the run and give comments about the different counting methods and frequency of the letters.

All information containing this results are saved in the results and plot folders.

### A. Exact and Approximate Counters

| N | Letter | Exact | Letter | Approximate |
|---|---|---|---|---|
| **1** | E | 21220 | E | 21173 |
| **2** | I | 13494 | I | 13570 |
| **3** | R | 12945 | R | 13059 |
| **4** | A | 12069 | A | 12074 |
| **5** | S | 11647 | S | 11578 |
| **6** | T | 11333 | T | 11210 |
| **7** | O | 10698 | O | 10763 |
| **8** | N | 10676 | N | 10598 |
| **9** | L | 9134 | L | 9125 |
| **10** | D | 8896 | D | 8915 |
| **11** | H | 6655 | H | 6368 |
| **12** | U | 5445 | U | 5269 |
| **13** | C | 5150 | C | 5069 |
| **14** | P | 4079 | Y | 4187 |
| **15** | G | 4065 | P | 4096 |
| **16** | Y | 4059 | G | 4032 |
| **17** | M | 3921 | M | 3971 |
| **18** | B | 2780 | B | 2790 |
| **19** | F | 2696 | F | 2640 |
| **20** | W | 2584 | W | 2622 |
| **21** | V | 1643 | V | 1632 |
| **22** | K | 1313 | K | 1302 |
| **23** | X | 395 | X | 397 |
| **24** | J | 260 | J | 259 |
| **25** | Q | 211 | Q | 206 |
| **26** | Z | 160 | Z | 158 |

*Table 1: Count comparison between Exact and Approximate Counter in English language*

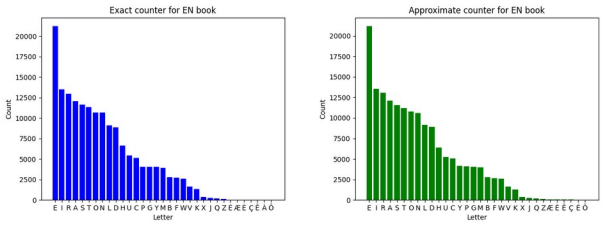



*Figure 5: Plot comparison between Exact and Approximate Counter in English language*

| N | Letter | Exact | Letter | Approximate |
|---|---|---|---|---|
| **1** | A | 37545 | A | 37397 |
| **2** | I | 35548 | I | 35757 |
| **3** | T | 29847 | T | 29789 |
| **4** | N | 28985 | N | 29269 |
| **5** | E | 24027 | E | 24090 |
| **6** | S | 23809 | S | 23915 |
| **7** | L | 17878 | L | 17882 |
| **8** | U | 17062 | Ä | 17259 |
| **9** | Ä | 16937 | U | 16987 |
| **10** | K | 16580 | K | 16467 |
| **...** | ... | ... | ... | ... |
| **28** | Q | 10 | Ô | 16 |
| **29** | Ô | 2 | Q | 6 |

*Table 2: Count comparison between Exact and Approximate Counter in Finnish language*

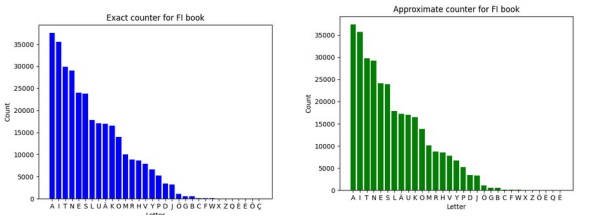



*Figure 6: Plot comparison between Exact and Approximate Counter in Finnish language*

| N | Letter | Exact | Letter | Approximate |
|---|---|---|---|---|
| **1** | E | 32181 | E | 32379 |
| **2** | R | 20912 | R | 20770 |
| **3** | I | 19528 | I | 19280 |
| **4** | A | 19206 | A | 18822 |
| **5** | S | 17552 | S | 17286 |
| **6** | T | 16413 | N | 16304 |
| **7** | N | 16371 | T | 16240 |
| **8** | O | 13816 | O | 13848 |
| **9** | U | 12356 | U | 12632 |
| **10** | L | 11547 | L | 11640 |
| **...** | ... | ... | ... | ... |
| **19** | G | 2952 | F | 2891 |
| **20** | F | 2933 | G | 2891 |
| **...** | ... | ... | ... | ... |
| **31** | W | 140 | Û | 150 |
| **32** | Ô | 133 | W | 136 |
| **33** | Û | 125 | Ô | 133 |
| **34** | K | 70 | K | 93 |
| **35** | À | 59 | À | 62 |

*Table 3: Count comparison between Exact and Approximate Counter in French language*

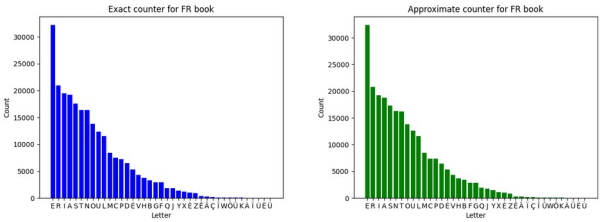



*Figure 7: Plot comparison between Exact and Approximate Counter in French language*

We can conclude from the data that the Approximate Counter with a fixed probability of 1/16 is a reliable method for identifying the frequency and ranking of the letters. Although the average approximate counter was only executed ten times, it provided a good insight into the frequency and ranking of the letters. The order of the most frequent letters is very similar, with only minor errors in some cases. For example, the incorrect ranking of a letter is usually not more than two levels, with most cases being incorrectly placed by only one level.

Additionally, even though there may be errors in the ranking of the top 10 most frequent letters, the letters that make up that top 10 remain consistent.

Also it is interesting to note that the letters that appear in the top 10 don't seem to vary much between languages and all Exact and Approximations Counters appear to have a similar plot between them, we can look at Figure 5, 6, 7 or 8 as an example.

| N | Letter | Exact | Letter | Approximate |
|---|---|---|---|---|
| **1** | E | 36734 | E | 37133 |
| **2** | N | 17706 | N | 17442 |
| **3** | R | 13586 | R | 13664 |
| **4** | I | 13269 | I | 13379 |
| **5** | O | 13248 | A | 13285 |
| **6** | A | 13207 | O | 12994 |
| **7** | T | 11036 | T | 11168 |
| **8** | D | 10230 | D | 10379 |
| **9** | L | 9996 | L | 10029 |
| **10** | G | 8371 | G | 8416 |
| **...** | ... | ... | ... | ... |
| **17** | U | 3888 | Z | 3872 |
| **18** | W | 3837 | W | 3862 |
| **19** | Z | 3799 | U | 3850 |
| **...** | ... | ... | ... | ... |
| **26** | Ê | 231 | Q | 304 |
| **27** | É | 231 | È | 240 |
| **28** | X | 79 | Ê | 190 |
| **29** | Ó | 76 | É | 98 |
| **30** | Ë | 59 | X | 91 |
| **31** | Â | 44 | Ó | 69 |
| **32** | Q | 36 | Ë | 46 |
| **33** | È | 25 | Â | 43 |
| **34** | Ï | 25 | Ï | 30 |
| **35** | À | 21 | À | 29 |

*Table 4: Count comparison between Exact and Approximate Counter in Dutch language*

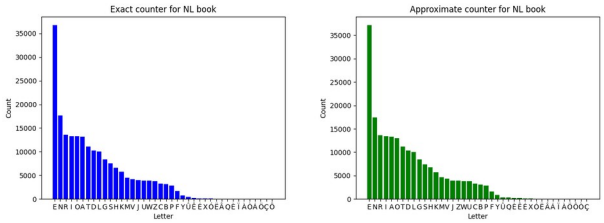
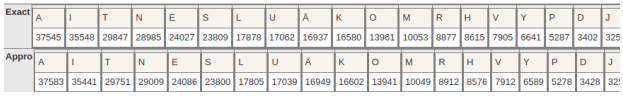



*Figure 8: Plot comparison between Exact and Approximate Counter in Dutch language*

The limitation of using the average of ten iterations for the approximate counter is that it leaves room for error. Since the results are based on probabilities, the more we execute the counter, the more accurate the results will

be. In most cases, the results will have negligible error, but there may be instances of more extreme errors, such as shown in Table 4. The Dutch “Dorian Gray” book, for example, presented a total of 12 misplaced ranks in 35 letters, with the letter Q being placed six ranks above its true ranking.

It is clear that running the approximate counter more times can decrease the error rate. However, this comes at the cost of increased time and space complexity, which goes against the purpose of using an approximate counter. Therefore, a balance must be found between the number of iterations and the desired level of accuracy.



| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Exact | A | I | T | N | E | S | L | U | Ä | K | O | M | R | H | V | Y | P | D | J |
| | 37545 | 35548 | 29847 | 28985 | 24027 | 23809 | 17878 | 17062 | 16937 | 16580 | 13961 | 10053 | 8877 | 8615 | 7905 | 6641 | 5287 | 3402 | 325 |
| Appro | A | I | T | N | E | S | L | U | Ä | K | O | M | R | H | V | Y | P | D | J |
| | 37583 | 35441 | 29751 | 29009 | 24086 | 23800 | 17805 | 17039 | 16949 | 16602 | 13941 | 10049 | 8912 | 8576 | 7912 | 6589 | 5278 | 3428 | 32 |

*Figure 9: French top 20 for 100 iterations on Approximate Counter*

B. Misra-Gries Algorithm

The Misra-Gries results show that this method does not yield good results with the given K values. Across all languages besides always getting the first ranking letter right to all K values, we don’t get much value until setting K equal to 10.

| K | N | Letter | Count |
|---|---|---|---|
| **3** | **1** | E | 1 |
| | | | |
| **5** | **1** | E | 2 |
| | **2** | D | 1 |
| | | | |
| **10** | **1** | E | 4961 |
| | **2** | I | 3 |
| | **3** | N | 2 |
| | **4** | D | 2 |
| | **5** | R | 1 |
| | **6** | G | 1 |
| | **7** | Z | 1 |
| | **8** | T | 1 |
| | **9** | H | 1 |

*Table 5: Misra-Gries English results for K = 3, 5, 10*

| K | N | Letter | Count |
|---|---|---|---|
| **3** | **1** | I | 1 |
| | **2** | N | **1** |
| | | | |
| **5** | **1** | F | 1 |
| | **2** | I | 1 |
| | **3** | N | 1 |
| | | | |
| **10** | **1** | E | 8870 |
| | **2** | R | 4 |
| | **3** | N | 2 |
| | **4** | F | 1 |
| | **5** | I | 1 |

*Table 6: Misra-Gries French results for K = 3, 5, 10*

Even with K set to 10, by analysing Table 5 which is using the English version of the book, we can see that the letter “E” has the highest frequency of 4961 and the rest has single digit counter. When K is increased, the Misra-Gries begins to return a more accurate result when compared to the Exact Counter.

| K | N | Letter | Count | Letter | Exact |
|---|---|---|---|---|---|
| **20** | **1** | E | 17233 | E | 36734 |
| | **2** | I | 9507 | N | 17706 |
| | **3** | R | 8958 | R | 13586 |
| | **4** | A | 8082 | I | 13269 |
| | **5** | S | 7660 | O | 13248 |
| | **6** | T | 7346 | A | 13207 |
| | **7** | O | 6711 | T | 11036 |
| | **8** | N | 6689 | D | 10230 |
| | **9** | L | 5147 | L | 9996 |
| | **10** | D | 4913 | G | 8371 |

*Table 7: Misra-Gries French results for K = 20*

As we increase the value of k, the Misra-Gries method starts to perform better. With k = 20, the count or ranking may not be as accurate as the Exact Counter but in the top 10 letters, we only get one letter that doesn't belong there.

In terms of cost per solution, Misra-Gries counters are generally less expensive than Exact Counters, as they require less memory and computational resources. However, the trade-off is the potential for errors which should be considered in the specific use case.

The value of K in Misra-Gries counter affects the error rate, with a larger value of k resulting in a lower error rate, but also a higher space complexity, so a balance must be found between the error rate and the space complexity.

C. Error

There is always error associated with these algorithms and approximations which rely on probability. We can make use of the Exact Counter results to calculate different type of errors:

- Minimun Relative Error
- Max Relative Error
- Min Absolute Error
- Max Absolute Error
- Average Relative Error
- Average Absolute Error

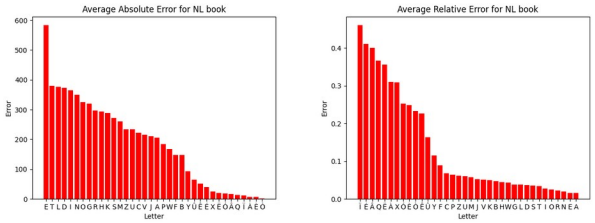



*Figure 10: Absolute and Relative error for Dutch*

For each language book, the metrics mentioned above are calculated for every letter in every execution of the Approximation Counter. These results are saved for each letter, allowing for a detailed analysis, specifically to understand the performance of each letter in the Approximation Counter.

```
fr:
Min Relative Error: -0.6390977443609023 -> Ô
Max Relative Error: 0.004557885141294439 -> S
Min Absolute Error: -1382.0 -> A
Max Absolute Error: 2.0 -> Ù
Average Relative Error: 0.09583019109013055
Max Absolute Error: 181.18378378378378


en:
Min Relative Error: -0.5692307692307692 -> J
Max Relative Error: -0.011870773854244 93 -> H
Min Absolute Error: -1103.0 -> S
Max Absolute Error: -79.0 -> H
Average Relative Error: 0.07501875629269827
Max Absolute Error: 200.78518518518518


nl:
Min Relative Error: -0.7288135593220338 -> Ë
Max Relative Error: 0.013893595391392748 -> N
Min Absolute Error: -1084.0 -> L
Max Absolute Error: 7.0 -> Ò
Average Relative Error: 0.13470495961381765
Max Absolute Error: 189.49722222222218


fi:
Min Relative Error: -0.6307692307692307 -> W
Max Relative Error: 0.027460380876281797 -> A
Min Absolute Error: -1305.0 -> A
Max Absolute Error: 6.0 -> Q
Average Relative Error: 0.0917991724150309
Max Absolute Error: 290.6571428571429
```

*Figure 11: Error results for 10 iterations of Approximate Counter*

With exception to the averages, the errors were not normalized in order to have a better understanding of how the error fluctuated for each approach and letter but, all plot results were normalized.

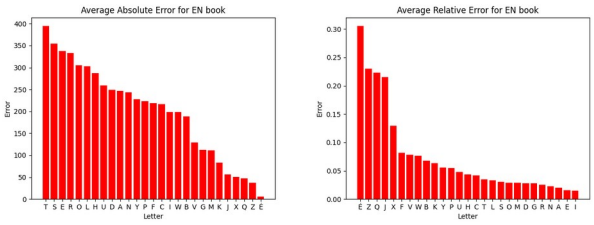



*Figure 12: Absolute and Relative error for English*

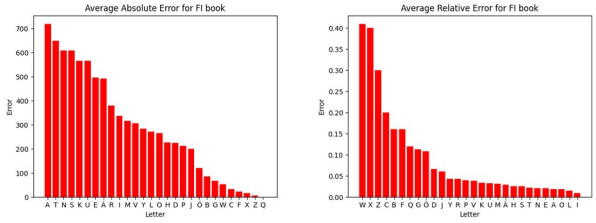



*Figure 13: Absolute and Relative error for Finnish*

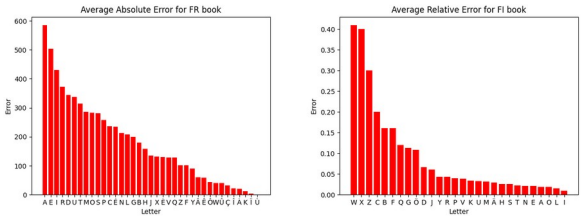



*Figure 14: Absolute and Relative error for French*

It is clear that when using a fixed probability of 1/16, the approximate counter has high average percentages of relative errors, which also results in high absolute errors. This is especially true when the approximate counter is executed a low number of times.

As it was previously discussed, when we increase the number of iterations for the approximate counter, we can see that the error rate decreases as we expected.

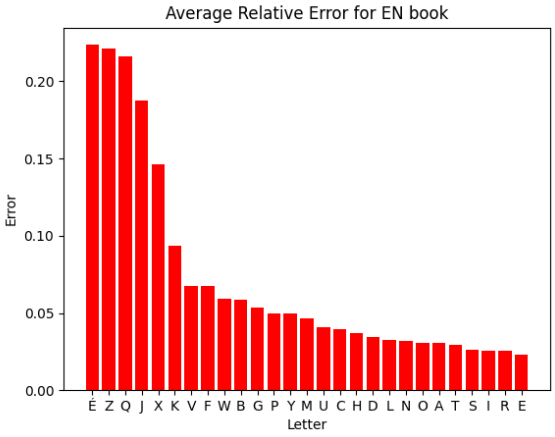



*Figure 15: Relative error for English with 100 iterations on Approximate Counter*

It is noteworthy that the letters appear to have a reversed ranking when it comes to errors. For example, by examining Figure 15, we can see that the bottom 10 letters are the same as the top 10 letters when compared to the Exact Counter.

## VIII. Conclusion

In conclusion, after analyzing the results, we can conclude that with both Approximate Counter and Misra-Gries Algorithm, errors are inherent. The decision is choosing between having a more precise result with a higher computational complexity and space usage or a faster solution that might have a certain error percentage associated with it.

## IX. References

[1] Stop words in various languages
https://github.com/Alir3z4/stop-words

## Links

https://github.com/simaoarrais/Advanced-Algorithms