3rd Project – Most Frequent Letters

Filipe Silveira - 97981

Resumo **–Este relatório aborda o desafio de identificar as letras mais frequentes em textos literários, uma tarefa essencial para diversas aplicações em análise de dados e processamento de linguagem natural. Reconhecendo as limitações dos métodos de contagem exatos em grandes conjuntos de dados, exploramos a eficácia de duas abordagens alternativas: Fixed Probability Counter, com probabilidade de 1/2 e o método Space-Saving Count. Avaliamos esses métodos com base na sua eficiência computacional, precisão em termos de erros absolutos e relativos e a sua eficácia na identificação consistente de letras frequentes em vários textos. Os resultados deste relatório pretendem fornecer informações sobre a viabilidade destas abordagens para análise de texto em grande escala, destacando as suas potenciais vantagens sobre os métodos tradicionais de contagem exata.**

Abstract **–This report addresses the challenge of identifying the most frequent letters in literary texts, a task essential for various applications in data analysis and natural language processing. Recognizing the limitations of exact counting methods in large datasets, we explore the efficacy of two alternative approaches: Fixed Probability Counter, with a probability of 1/2 and the Space-Saving Count method. We assess these methods based on their computational efficiency, accuracy in terms of absolute and relative errors, and their effectiveness in consistent identification of frequent letters across various texts. The results of this report aim to provide insights into the feasibility of these approaches for large-scale text analysis, highlighting their potential advantages over traditional exact counting methods.**

Keywords **–Frequent Letters, Fixed Probability Counter, Space-Saving Count**

I. Introduction

This reports' goal is to identify the most frequent letters in text files (literary works) using different methods, and to evaluate the quality of estimates regarding the exact counts. In order to accomplish that, three different approaches were made: Exact counters, Approximate counters, and Space-Saving Count algorithm (i.e. the Metwally et al. algorithm).

Firstly, we procure our text files of literary works from Project Gutenberg [1] in different languages as refereed in the assignment. For this report, the literary work chosen was *"Around the World in Eighty Days"* [2] by Jules Verne in English, French [3], Italian [4] and German [5]. The procedure to process the files will be explained in more detail in the next chapter.

After the texts are processed and ready to analyse, we start with the exact counters approach. An exact counter works by creating a frequency table (i.e. a dictionary) that counts the number of times each letter appears in the text file, providing a straightforward and simple reference point for our other algorithms.

Once we have established our starting point, we can then move on to the second approach: Approximate counters. This method involves using a probabilistic data structure to estimate the frequencies of the most frequent letters where, in this case, we used a fixed probability counter of 1 / 2 for each element in the stream. This approach allows for efficient memory usage, but the estimates may not be as accurate as the exact counts. For our third approach, we used the Metwally et al. algorithm, commonly known as Space-Saving Count. This algorithm is designed to efficiently identify frequent elements in data streams while using limited memory. It operates by maintaining 'k' counters to track the most frequent elements. This method ensures memory efficiency and provides a practical approach to handling large data streams.

Finally we will analyze the computational efficiency and limitations of the algorithms focusing on absolute and relative errors, average values, and the ability to identify the most frequent elements accurately as well as whether the most frequent letters identified by both algorithms are similar in the text files of the same literary work but in different languages. This study will give us insight into the generalization of the algorithms and whether they perform well in a variety of languages.

To generate the results and the charts folders follow these steps respectively:

```
> cd .\code\
> python main.py
> python charts.py
```

II. Text Processing

Before we could start the development of the algorithms and their analysis, firstly we had to obtain our data. We obtained our text files of *"Around the World in Eighty Days"* by Jules Verne from Project Gutenberg in English, French, Italian and German, as refereed in the Introduction.

After obtaining the files, we start processing them for analysis. The first step is to remove the Project Gutenberg file headers and footers, which we do by finding

the start and end makers of the book, normally refereed as "*** START/END OF THE PROJECT GUTENBERG EBOOK ... ***". With this we know where the content of our book is and therefore remove everything before and after it.

For our next step we process the text by removing all of the stopwords of each language and all punctuation marks. This step was done with ease by using the **nltk** library for python where we can get the stopwords of a specific language and check each word if it matches. For the punctuation marks, we compare if the words are not from the python **string.punctuation**.

Lastly we convert each word to uppercase and save in a new file in the *corrected_texts* folder.

III. Exact Counter

Exact counters, are normally used to provide a precise count of the number of occurrences of each item in a dataset. In our case, we use the exact counter as a control variable to compare and evaluate estimates obtained by the other approaches.

In this report, an exact counter was implemented by creating a counter for each letter of the alphabet and increment the corresponding counter whenever we encounter that letter in the text.

However, it's worth noting that exact counters can be slower and require more memory to store the count of each item. It is a limitation that becomes more significant if the dataset is large or if the processing speed is a concern.

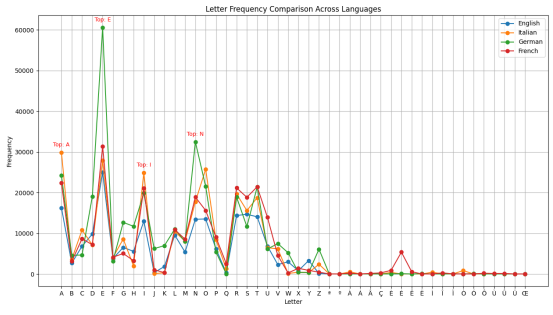

Fig. 1 - Exact count of letters in all languages.

As seen in Fig. 1 the top 4 letters for all languages were "A", "E", "I", "N". For the next chapters we will use the letter "E" to provide more analytics.

IV. Approximate Counter

Approximate counters are data structures that provide an estimate of the frequency of items in a dataset. These counters trade off some accuracy for efficiency, as they can often process large amounts of data more quickly than exact counters as well as requiring and less memory than them.

For the project, we implemented an approximate counter with a fixed probability of 1/2. The counter works by iterating through each letter in the text and, with a 50% chance, incrementing the count for that letter. This probabilistic approach effectively means that each letter has a 50% chance of being counted in each of its occurrences. To compensate for the undercounting due to the 50% probability, the final count for each letter is then scaled by a factor of 2 (the inverse of the 0.5 probability). This scaling aims to approximate the actual frequency of each letter in the text, acknowledging that some counts may be missed due to the probabilistic nature of the method.

We obtain the average results for a total number of 10 000 trials.

Stats \ Language	En	It
Mean	6434.472	6923.085
Variance	39677076.701	79941192.319
Standard Deviation	6298.974	8940.983
Smallest Counter Value	0.990	0.986
Largest Counter Value	24960.6396	29882.5086
Mean Absolute Error	0.502	0.449
Mean Relative Error	0.001	0.001
Mean Accuracy Ratio	0.999	0.999

TABLE I
STATISTICS OF THE APPROXIMATE COUNTER FOR ENGLISH AND ITALIAN.

Stats \ Language	Fr	De
Mean	6602.508	9173.974
Variance	69199661.705	146457460.314
Standard Deviation	8318.633	12101.961
Smallest Counter Value	0.996	0.998
Largest Counter Value	31347.985	60595.998
Mean Absolute Error	0.601	0.534
Mean Relative Error	0.00069	0.001
Mean Accuracy Ratio	1.00028	1.00016

TABLE II
STATISTICS OF THE APPROXIMATE COUNTER FOR FRENCH AND GERMAN.

Stats Let. "E" \ Language	En	It
Mean	24960.6396	27896.111
Variance	0.0	0.0
Standard Deviation	0.0	0.0
Smallest Counter Value	0.9904	0.986
Largest Counter Value	24960.6396	29882.5086
Mean Absolute Error	1.360	0.889
Mean Relative Error	5.449883e-05	3.18672e-05
Mean Accuracy Ratio	0.999945	0.999968

TABLE III
STATISTICS OF THE LETTER "E" — APPROXIMATE COUNTER FOR ENGLISH AND ITALIAN.

Overall, the results suggest that the approximate counters aren't as accurate as the exact counters for the four languages, as indicated by the high mean absolute error and mean relative error values. This approach performed poorly when identifying the most frequent orders of letters; however, it obtained the most fre-

Stats Let. "E" \ Language	Fr	De
Mean	31347.985	60595.9984
Variance	0.0	0.0
Standard Deviation	0.0	0.0
Smallest Counter Value	0.9966	0.998
Largest Counter Value	31347.985	60595.9984
Mean Absolute Error	1.985	0.9983
Mean Relative Error	6.33254e-05	1.64766e-05
Mean Accuracy Ratio	1.00006	1.000016

TABLE IV
Statistics of the Letter "E" — Approximate counter for French and German.

quent letter for all languages (letter "E") with relatively high accuracy.

V. Metwally algorithm

The Metwally algorithm, also known as the Space-Saving Count, is a well-regarded approach for identifying the frequency of items in a data stream. Utilizing a memory-efficient method, it maintains a fixed number of counters (denoted as **k**) to keep track of the most frequent elements. The accuracy of the algorithm improves with a higher value of **k**, as it allows tracking a greater number of elements.

In practice, if a letter is already being counted, its counter is incremented. If it's new and there is capacity (i.e., fewer than **k** different letters are being monitored), a new counter is initiated. On the contrary, if there is no capacity, the counter of the least frequent letter is replaced with the new one and incremented.

Particularly suited for vast data streams where monitoring each unique item isn't viable due to memory limitations, the algorithm offers an estimated frequency count, mainly focusing on the most common elements. This is typically adequate for purposes like data analysis or identifying trends.

Nonetheless, the Metwally algorithm has its limitations. Its capability to accurately represent less frequent elements in expansive datasets is constrained, as it operates with a fixed number of counters (**k**). This may result in either inaccuracies or complete omission of infrequent items, thereby rendering the algorithm less effective in situations that demand precise frequency counts of all elements.

In our algorithm, we applied it for k = 3, k = 5, and k = 10. The results for the accuracy of the letter "E" are seen below.

K \ Lang.	En	It	Fr	De
3	0.0	0.0	2.808	1.8167
5	0.0	0.0	1.685	1.090
10	1.00008	1.00003	1.0	1.0

TABLE V
Mean accuracy ratio for Letter "E" — Metwally algorithm

As we can see from the results, in the cases of the English and Italian the accuracy was always 0 for the values 3 and 5 of **k**, however this was due to the letter not being among them. For **k** value of 10, the letter was in the counter and presented a high accuracy.

In the cases of the French and German language the letter was always present in the counter, although of smaller values of **k**, the counter was higher than the exact count, a known setback of using the Metwally algorithm with small **k** values. When the value of **k** increased, the accuracy became better eventually reaching the exact count as seen in Table V.

VI. Processing Time

Alg \ Language	En	It
Exact Counter	0.02093983	0.03320575
Approx Counter (Avg)	0.02860744	0.03646380
Metwally k = 3	0.11341214	0.13502287
Metwally k = 5	0.155676842	0.14390993
Metwally k = 10	0.09884763	0.10383558

TABLE VI
Processing time between the algorithms through English and Italian.

Alg \ Language	Fr	De
Exact Counter	0.03160691	0.03891945
Approx Counter (Avg)	0.03539243	0.04540751
Metwally k = 3	0.16832757	0.18148875
Metwally k = 5	0.15544367	0.16576219
Metwally k = 10	0.13679647	0.14424086

TABLE VII
Processing time between the algorithms through French and German.

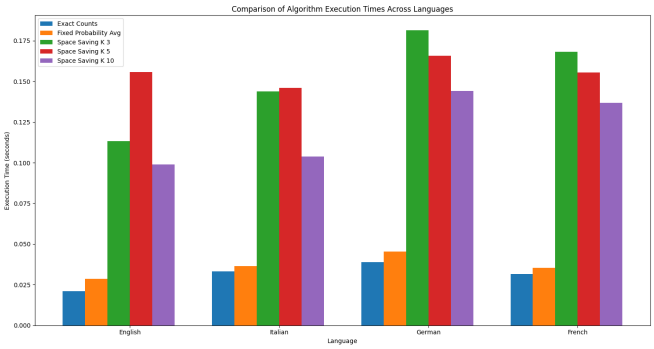

Fig. 2 - Comparison of Algorithm Execution Times Across Languages.

By analysing the tables and the chart above, we come to the conclusion that for execution time, the exact counter and the Approximate Counter have better time performance relative to the Metwally algorithm, since both never passed the 0.05 second mark.

VII. Conclusion

In general, the choice between using an exact counter, an approximate counter, or the Metwally algorithm de-

pends on the specific requirements of the task at hand. If precise counts are necessary, exact counters are the better choice. However, if memory efficiency is more important, approximate counters or the Metwally algorithm may be the better choice.

While the approximate counter performed better in terms of execution time, relative to the Metwally algorithm, it had worst accuracy for the same letter. However this is only ttrue for a high value of **k** counter in the Metwally algorithm.

References

[1] Anon, "Project gutenberg", 2019.
URL: https://www.gutenberg.org/

[2] Jules Verne, *Around the World in Eighty Days*, Jan 1994.
URL: https://www.gutenberg.org/ebooks/103

[3] Jules Verne, *Le tour du monde en quatre-vingts jours*, Jan 1997.
URL: https://www.gutenberg.org/ebooks/800

[4] Jules Verne, *Il giro del mondo in ottanta giorni*, Jun 2021.
URL: https://www.gutenberg.org/ebooks/65736

[5] Jules Verne, *De reis om de wereld in tachtig dagen*, Feb 2004.
URL: https://www.gutenberg.org/ebooks/11318