Letter Frequency Algorithms

Vasco Regal

***Abstract* –This paper documents several algorithms run with text files as input with the aim to compute frequency metrics of present letters both with an exhaustive and approximation approach. Three procedures were developed with python in an attempt to analyze their performances depending on input, stopwords and algorithm parameters.**

***Resumo* –Este artigo documenta vários algoritmos corridos com ficheiros de texto como input, com o objetivo de calcular métricas de frequência de letras com métodos exaustivos e de aproximação. Três implementações foram desenvolvidas em python, numa tentavia de analizar os respetivos desenpenhos quando o input, stopwords e outros parametros do algoritmo são variados.**

***Keywords* –probabilistic, algorithms, text, letters, counting, frequency**

***Palavras chave* –probabilistico, algoritmos, texto, letras, contagem, frequencia**

I. Dataset

A. Input files

For this project, two text files were used containing George Orwell's **1984**, one in portuguese and one in english. Stopwords in each respective language were used during runs, obtained from publicly online sources.

II. Algorithms

A. Letter Counting

A.1 Exhaustive

An exhaustive algorithm means the output will be true solution of the problem. In this case, the algorithm will keep track of every single letter while passing through the file and keep a counter for the number of times it sees each character. Pseudocode 1 presents this very naive approach.

Algorithm 1 Exahustive Letter Counting

```
1:  procedure ExhaustiveCounter(stream)
2:      solution ← DICTIONARY
3:      while stream not empty do
4:          token ← stream.next
5:          if token in KEYS(solution) then
6:              solution[token]+ = 1
7:          else
8:              solution[token] ← 1
9:          end if
10:     end while
11:     return solution
12: end procedure
```

A.2 Probabilistic

Instead of keeping an exact counter for each letter, the probabilistic procedure increments the letter counter depending on a decreasing probability. This probability value is given by the formula $1/sqrt(2)^k$, being k the value of the counter. In pratical terms, this means that the higer the k, the less probability it has to be incremented. 2 presents this logic.

Algorithm 2 Decreasing Probability Letter Counting

```
1:  procedure DecreasingProbCounter(stream, k)
2:      solution ← DICTIONARY
3:      while stream not empty do
4:          token ← stream.next
5:          if token not in KEYS(solution) then
6:              solution[token] ← 0
7:          end if
8:          cnt ← solution[token]
9:          if random(0, 1) < 1/sqrt(2) ∗∗k then
10:             solution[token]+ = 1
11:         end if
12:     end while
13:     return solution
14: end procedure
```

B. Most Frequent Letters

To find an approximation of the most frequent letters in the texts, the Mistra Gries algorithm was implemented. This procedure keeps the **k** - 1 most probable letters to be the most frequent by simply analyzing the next letters in the stream. This means that, for every letter, if its frequency is greater than the total number of tokens divided by k, this token's counter will be a positive number. 3 holds the basis for the implementation.

III. Implementation

As for implementation, the code is structured in various classes with a CLI interface to run experiments.

DataStream class handles retrieving the next token and is responsible for text parsing (removing punctuation, stopwords, etc.)

The **Problem** class, which has a DataStream object, manages solvers, exporting data and measuring the

Algorithm 3 Misra & Gries algorithm

```
procedure MostFrequent(stream, k)
    solution ← DICTIONARY
    while stream not empty do
        token ← stream.next
        if token in KEYS(solution) then
            solution[token] += 1
        else
            if solution.length < (k − 1) then
                solution[token] = 1
            else
                for i in KEYS(solution) do
                    solution[token] −= 1
                    if solution[token] == 0 then
                        POP(solution[token])
                    end if
                end for
            end if
        end if
    end while
    return solution
end procedure
```

computation time.

Solver class has three subclasses which are the implementation presented above, **ExahustiveSolver**, **ProbabilisticSolver** and **MistraGriesSolver**

All runs can be exported to files, which are used, after running batch experiments, to process the outputs.

A. Analysis

A.1 Running experiments

The CLI interface was used, aided by the **experiments.py** script to automate runs.

A.2 Analysing runs

With the results stored in files, the analysis was conducted with the aid of **matplotlib** to generate plots, **pandas** module and other data science related python packages. The results of said study is presented below.

IV. RESULTS

For the experimentation, the runner script was used with following parameters, for both testing languages with varying values of **k** (3, 5, 10)

A. Runs

A.1 Exact Counter

The generated files from the exact computations contain the data on table I (portuguese) and II (english).

By looking at the tables, and as expected, different languages present a different distribution of letter frequency, beign the letter **r** the most common in total.

A.2 Probabilistic

As for the probabilistic approach, both languages' results can be seen in tables III and IV, with the varying values of K

Letter	Count
s	343
r	306
i	269
n	229
t	193
d	164
l	159
m	150
c	144
p	134

TABLE I
PORTUGUESE EXACT COUNT RESULTS

Letter	Count
e	324
r	184
l	169
n	167
c	124
u	114
d	84
h	81
o	79
g	76

TABLE II
ENGLISH EXACT COUNT RESULTS

K	Top 5 Counts	time(s)
3	$s:124, r:102, i:92, n:85, t:75$	0.00268
5	$r:60, s:56, i:53, t:36, d:35$	0.00266
10	$s:13, i:8, l:8, p:7, r:7$	0.00258

TABLE III
PORTUGUESE PROBABILISTIC COUNT RESULTS

B. Approximation Error

Given both the datasets presented above, the average absolute and relative counting errors were calculated, for each value of k and for each language. For the portuguese tests, table V and for the english version, table VI

In both language we witness an increase on the ap-

K	Top 5 Counts	time(s)
3	$e:114, r:59, n:53, l:52, u:50$	0.00250
5	$e:60, l:32, r:31, n:21, c:18$	0.00225
10	$e:13, c:9, r:7, n:7, u:6, l:5$	0.00224

TABLE IV
ENGLISH PROBABILISTIC COUNT RESULTS

K	Mean Absolute Error	Mean Relative Error
3	133.8	0.637
5	171.7	0.820
10	202.3	0.966

TABLE V
Portuguese Count Errors

K	Mean Absolute Error	Mean Relative Error
3	92.4	0.660
5	118.8	0.860
10	135.1	0.968

TABLE VI
English Count Errors

K	Most Frequent Letters	time(s)
3	s	0.00275
5	i, s	0.00279
10	s, r, i, n	0.00267

TABLE VII
Most Frequent Letters Portuguese

proximation error as K increases. The high error values are cause by the low probability of increasing the counter. However, and looking at the results, we can see that the order in which the letters appear is very similar for both study cases.

C. Frequent letter identification

In this section, the results for the MistraGriesSolver will be presented. Although this algorithm gives us no information regarding item count, it can predict the most frequent letters in the text successfully. The results obtained, in function of k are presented in table VII for Portuguese and VIII for English

From these results, and compairing with the exact counts, the algorithm effictevly identifies the most common letters without actually keeping counts. Also, the higher the k, the more accurate the results.

V. Conclusion

With the results at hand, we can conclude that the approximation algorithm, although not presenting good counts, works well to identify the most frequent letters. However, if our goal is to retrieve the most frequent letters in a text, an algorithm in the style of Mistra Gries should be used. Also, different languages, as expected, present different results when counting letters in texts.

K	Most Frequent Letters	time(s)
3	y	0.00234
5	y, r, e, g	0.00238
10	e, r, l, n, b, y, g, u, f	0.00217

TABLE VIII
Most Frequent Letters English

References

[1] Github User sebleier, NLTK English Stopwords https://gist.github.com/sebleier/554280, publicly online

[2] Github User alopes, Portuguese stopwords https://gist.github.com/alopes/5358189, publicly online

[3] Faded Page, 1984 [Nineteen Eighty-Four], https://www.fadedpage.com/showbook.php?pid=20120511, publicly online

[4]