Most Frequent Words

Gil Lopes Teixeira

*Resumo* – Este relatório apresenta um método de estimar a frequência de palavras chamado LossyCounting. Este método recebe um parâmetro k e um texto de tamanho n que permite manter apenas as palavras que no total tenham no seu contador e que portanto aproxima os resultados por cima com um erro máximo de .

*Abstract* – This report presents a method to estimate the frequency of words called LossyCounting. This method receives a paramether k and a text of length n that allows to only keep words with in it’s counter and it never understimates the results, overestimating at most .

# I. Introduction

The Lossy Counters allows us to have an estimate of words frequency. The difference between this and exact counting is that in this method we discard words with under counts. All the tables shown here are excerpts from excel files available in this reports folders that contain statistics from 1000 runs of each counter for each language.

# II. LossyCounter

def LossyCount(k):

T = {}

n = 0

**Δ** = 0

for line in fp.readlines():

for word in line.split(' '):

n += 1

if word in index.keys():

T[word] += 1

else:

T[word] = 1 + **Δ**

if math.floor(n/k) != **Δ**:

delta = math.floor(n/k)

for word in T:

if T[word] < **Δ**:

T.pop(word)

return T

For a given file with n characters excluding spaces the algorithm does:

The delete happens every iteraction where delta, an implicit value, is different of !

This algorithm might overestimate de result by the but never underestimated! This means that for a large n the algorithm behaves badly for small k!

# III. Results

For testing two different books were used. Winters tails in English, with 26880 words, and French by Shakespear, and The Odissey by Homero in English, with 133590 words, and Swedish.

# III.1 Winters tails - Shakespear

For the first example, in English, running the lossy counter with k=1000 we know that at most the frequency will be overestimated by 26.88% as it’s confirmed in the table bellow order by the relative error!

|  |  |  |  |
| --- | --- | --- | --- |
| char | exact\_val |  | expected |
| E | 14765 |  | 14762 |
| T | 9214 |  | 9211 |
| O | 9121 |  | 9122 |
| A | 7508 |  | 7507 |
| H | 6886 |  | 6889 |
| S | 6864 |  | 6862 |
| I | 6680 |  | 6670 |
| N | 6455 |  | 6456 |
| R | 6171 |  | 6173 |
| L | 4625 |  | 4624 |

Top 10 characters in the english version of the book from Shakespears book Winter Tales.

The expected count was calculated by multiplying the average count of each character, in 1000 tests, by 4. For all the characters with more than 1000 occurrences in the text the relative error was always bellow 0.15%!

French version:

|  |  |  |
| --- | --- | --- |
| char | exact\_val | expected |
| E | 23186 | 23196 |
| S | 11874 | 11874 |
| I | 10429 | 10427 |
| N | 10390 | 10395 |
| U | 10053 | 10052 |
| A | 9958 | 9969 |
| R | 9904 | 9909 |
| O | 9465 | 9462 |
| T | 9438 | 9439 |
| L | 8093 | 8096 |

# III.2 Odissey - Homero

In this second example, in the English version, it’s easy to conclude that with n=133590 if we use the lossy counter with k=1000 the max relative error jumps to 133.59% wich might not be acceptable if we do count on using the lowest ranked counters. But as seen in the last example the top results have 0% relative error and with the increase of k, more top results will keep this property.