Terceiro Trabalho de Algoritmos Avançados - Letras mais frequentes

João Reis, 98474

***Resumo* - Este documento tem como objetivo documentar a implementação e a análise realizada sobre três abordagens diferentes de contagem: Contagem exata, Contagem com probabilidade decrescente e Lossy-Count. Ao longo deste documento é explicado o raciocínio de implementação destes algoritmos. No final é realizada uma análise e comparação dos resultados obtidos.**

***Abstract* - This paper aims to document the implementation and analysis performed on three different counting approaches: Exact-Count, Counting with decreasing probability, and Lossy-Count. Throughout this paper the reasoning for implementing these algorithms is explained. At the end an analysis and comparison of the results obtained is performed.**

***Key words* - Python, Exact Count, Decreasing Probability Counter ($\frac{1}{2^k}$), Lossy-Count, Most Frequent Letters, Erro Absoluto, Erro Relativo, Idioma**

I. INTRODUÇÃO

Como terceiro trabalho da unidade curricular Algoritmos Avançados é proposto a resolução de um pequeno problema envolvendo ficheiros de texto, utilizando três abordagens diferentes.

Para testar estas abordagens é utilizado ficheiros de texto da famosa obra literária "*Hamlet*", escrita por William Shakespeare, obtidos através do Projeto Gutenberg.

As três abordagens implementadas são: contadores exatos; contadores aproximados, mais propriamente, um contador de probabilidade decrescente ($\frac{1}{2^k}$); e, por fim, o algoritmo Lossy-Count.

Para executar o código, basta executar um dos comandos indicados abaixo.

```
$ python3 main.py

$ python3 main.py -t <threshold: float> -r
<repetitions: int>
```

As opções -t e -r são opcionais e servem para definir um threshold e o número de repetições, respectivamente.

Este relatório é acompanhado com todos os ficheiros python produzidos para implementar estas abordagens.

II. OBJETIVO

O objetivo do programa é identificar as letras mais frequentes em ficheiros de texto utilizando diferentes métodos e comparar a qualidade das estimativas desses métodos com o número exato.

Os métodos a ser implementados, como já foi referido anteriormente, são: contadores exatos; contadores aproximados, mais propriamente, um contador de probabilidade decrescente ($\frac{1}{2^k}$); e o algoritmo Lossy-Count.

Os ficheiros de texto utilizados são traduções da obra literária "*Hamlet*" nos seguintes idiomas: Português, Inglês e Alemão, para que as letras mais frequentes não fossem semelhantes nos ficheiros de texto da mesma obra literária.

Após a implementação e da análise dos resultados de cada método, foi realizada uma análise em termos da eficiência computacional e das limitações para cada abordagem.

III. PROCESSAMENTO DOS DADOS

Antes de implementar os métodos, é realizado um processamento dos ficheiros de texto. Primeiramente, o cabeçalho e o rodapé do *Project Gutenberg* são removidos, para ser considerada somente a obra literária.

De seguida, as stop words são removidas do texto, tal como qualquer tipo de pontuação. No final todas as letras passam para a sua forma maiúscula. A função *read_and_process* lê as obras literárias, aplica o processamento anteriormente explicado, e retorna uma única string com todas as letras que restaram do processamento.

IV. ABORDAGEM 1: CONTAGEM EXATA

Esta abordagem é bastante simples, apenas faz uma contagem exata do número de letras. Para tal, a função *exact_counters*, que recebe a string com todas as letras como argumento, conta a frequência de cada letra aplicando a classe *Counter* da livraria *collections*.

```python
def sort_dict(dic):
    """ Function that sort the dicionary by value and then, by keys """
    return dict(sorted(dic.items(), key = lambda x: (x[1], x[0]), reverse=True))


def exact_counters(content):
    """ Function that returns the exact frequency of each letter"""
    exact_freq = dict(Counter(content))
    return sort_dict(exact_freq)
```

Fig. 1 - Código em Python para aplicar a contagem exata

V. ABORDAGEM 2: CONTAGEM APROXIMADA

O método de contagem aproximada aplicado é **contador de probabilidade decrescente** ($\frac{1}{2^k}$)**.** Este algoritmo funciona da seguinte maneira: a contagem de uma letra é incrementada caso o número gerado aleatoriamente seja inferior ou igual à probabilidade calculada. Esta probabilidade segue a seguinte fórmula: $\frac{1}{2^k}$, onde k é valor da contagem dessa mesma letra. Se esse valor aleatório for maior que a probabilidade calculada, o valor da contagem dessa letra mantém-se.

Portanto, à medida que o valor de contagem aumenta, menor é a probabilidade de esse valor ser incrementado.

```python
# Decreasing probability counter (1 / 2^k)
for letter in content:
    counter = 0

    if letter in estimate_freq.keys():
        counter = estimate_freq[letter]

    increment = decreasing_probability_counter( counter )

    counter += increment
    estimate_freq[letter] = counter
```

Fig. 2 - Código em Python para aplicar o contador de probabilidade decrescente (I)

```python
def calc_prob(counter):
    """ Function that returns value of the probability 1 / 2^k """
    return 1 / ( 2 ** counter )


def decreasing_probability_counter(current_count):
    """ Function that returns a count implementing decreasing probability counter """
    count = 0

    # As the counter value increases, it will be incremented with lesser probability
    if random.random() <= calc_prob(current_count):
        count += 1

    return count
```

Fig. 3 - Código em Python para aplicar o contador de probabilidade decrescente (II)

Este algoritmo é executado algumas vezes. Por defeito, o número de repetições é 10, mas o utilizador pode alterar este valor pelo comando de execução. Para cada repetição, é calculado o erro relativo da contagem de todas as letras. O resultado dessas repetições que obter menor erro relativo será considerado para as comparações com os outros métodos.

VI. ABORDAGEM 3: LOSSY-COUNT

O método de contagem aproximada aplicado é **Lossy-Count**. O algoritmo Lossy-Count é um algoritmo de contagem distribuída que permite aos usuários contar o número de ocorrências de letras em grandes conjuntos de dados de forma rápida e precisa. É necessário definir um threshold

O primeiro passo para implementá-lo é dividir o conjunto total de letras em $\frac{1}{thresold}$ partes. De seguida, incrementa-se a contagem de frequência de cada letra de cada parte. Após a contagem de um parte terminar, diminui-se todos os contadores uma (1) unidade. E volta-se a repetir todo este processo para todas as partes.

```python
error = 1

# Iterate through the content
for letter in content:
    # Increment the count of the letter
    if letter in estimate_freq:
        estimate_freq[letter] += 1
    else:
        estimate_freq[letter] = 1

    # Decrease the error bound by threshold
    error -= THRESHOLD

    # If the error bound becomes negative, divide all counts by 2 and reset the error bound to 1
    if error < 0:
        estimate_freq = { letter : estimate_freq[letter] - 1 for letter in estimate_freq.keys() }
        error = 1
```

Fig. 4 - Código em Python para aplicar o Lossy-Count

Analisando a figura anterior, a variável *error* é inicializada a zero (0). Para cada letra, incrementa-se a sua contagem, e decrementa-se a variável error pelo threshold definido.

Quando esse valor for negativo, significa que acabamos de analisar uma parte do conjunto todo, e decrementa-se todas as contagens uma (1) unidade.

Este processo é repetido até serem contabilizadas todas as letras.

Este algoritmo é aplicado para diferentes valores de k (3, 5 e 10), sendo k o número de letras mais frequentes a serem apresentadas.

VII. RESULTADOS

Neste tópico, os resultados serão apresentados por idioma analisado. O número de repetições usado é 100, e o valor do threshold é 0,0001.

A. Português

A próxima tabela indica as 10 letras mais frequentes em português, para cada um dos métodos implementados.

Contagem Exata			**Contagem com Probabilidade Decrescente**			**Lossy-Count** (threshold = 0,0001)	
A	10904		I	14		A	10896
E	10067		A	14		E	10059
O	8035		T	13		O	8027
S	6484		O	13		S	6476
R	5785		M	13		R	5777
I	5063		E	13		I	5055

M	4521		C	13		M	4513
N	4244		R	12		N	4236
T	4029		N	12		T	4021
U	3378		U	11		U	3370

Fig. 5 - Tabela com as 10 letras mais frequentes em português para cada método

Por análise da tabela, verifica-se que o algoritmo Lossy-Count tem uma maior precisão no que toca à ordem relativa das letras mais frequentes, enquanto que o algoritmo de contagem com probabilidade decrescente, apesar da contagem ser bastante desigual à contagem exata, o que é normal devido à estratégia de implementação, e apesar de a ordem relativa não estar correta, contém todas as letras que estão no top 10 da contagem exata, exceto a letra 'S'.

Em relação aos erros absolutos e relativos aos dois algoritmos, os resultados são os seguintes:

Contagem com probabilidade decrescente	
Menor Erro Absoluto	0
Erro Absoluto Médio	3077.1
Maior Erro Absoluto	10890
Menor Erro Relativo	0
Erro Relativo Médio	0.91
Maior Erro Relativo	0.99

Fig. 6 - Tabela com informações sobre os erros relativos e absolutos

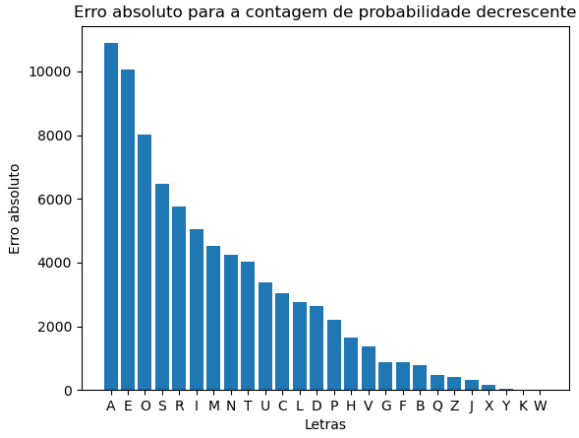

Fig. 7 - Erro absoluto para cada letra, na contagem com probabilidade decrescente

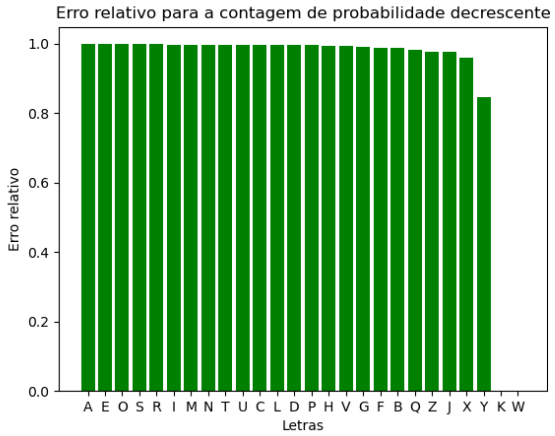

Fig. 8 - Erro relativo para cada letra, na contagem com probabilidade decrescente

Lossy-Count (threshold = 0,0001)	
Menor Erro Absoluto	8
Erro Absoluto Médio	8.0
Maior Erro Absoluto	8
Menor Erro Relativo	0.00073
Erro Relativo Médio	0.0015
Maior Erro Relativo	0.0024

Fig. 9 - Tabela com informações sobre os erros relativos e absolutos

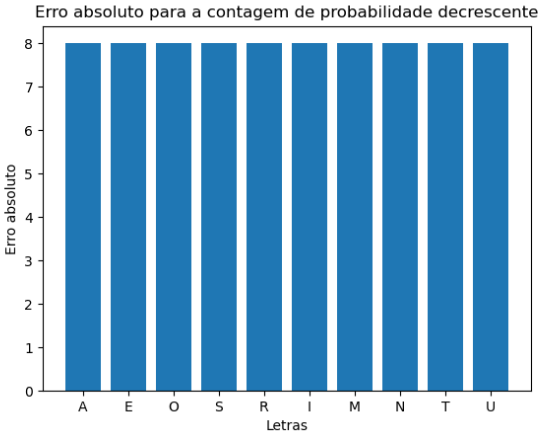

Fig.10 - Erro absoluto para para as top 10 letras, no Lossy-Count

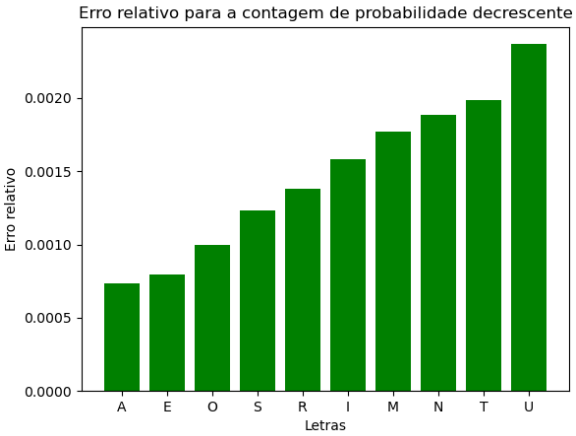

Fig. 11 - Erro relativo para cada letra, no Lossy-Count

B. Alemão

A próxima tabela indica as 10 letras mais frequentes em alemão, para cada um dos métodos implementados.

Contagem Exata			**Contagem com Probabilidade Decrescente**			**Lossy-Count** (threshold = 0,0001)	
E	18103		I	14		E	18093
N	11727		E	14		N	11717
I	8961		A	14		I	8951
S	7540		U	13		S	7530
R	6994		T	13		R	6984
H	6508		S	13		H	6498
T	5913		R	13		T	5903
U	5386		N	13		U	5376
A	5257		H	13		A	5247
D	5002		D	13		D	4992

Fig. 12 - Tabela com as 10 letras mais frequentes em alemão para cada método

Por análise da tabela, verifica-se que o algoritmo Lossy-Count tem uma maior precisão no que toca à ordem relativa das letras mais frequentes, enquanto que o algoritmo de contagem com probabilidade decrescente, apesar da contagem ser bastante desigual à contagem exata, o que é normal devido à estratégia de implementação, e apesar de a ordem relativa não estar correta, contém todas as letras que estão no top 10 da contagem exata.

Em relação aos erros absolutos e relativos aos dois algoritmos, os resultados são os seguintes:

Contagem com probabilidade decrescente	
Menor Erro Absoluto	4
Erro Absoluto Médio	4130.2
Maior Erro Absoluto	18089
Menor Erro Relativo	0.5
Erro Relativo Médio	0.96
Maior Erro Relativo	0.99 (aprox., 1)

Fig. 13 - Tabela com informações sobre os erros relativos e absolutos

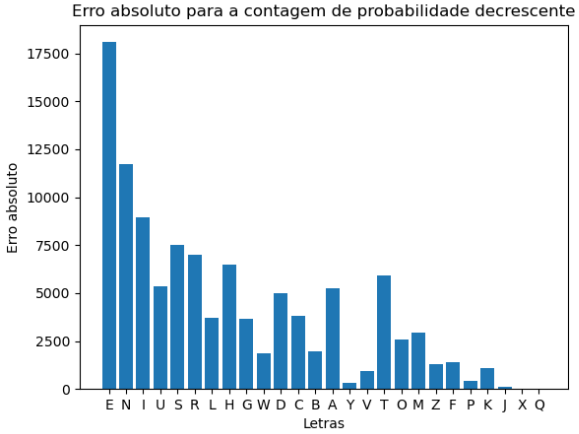

Fig. 14 - Erro absoluto para cada letra, na contagem com probabilidade decrescente

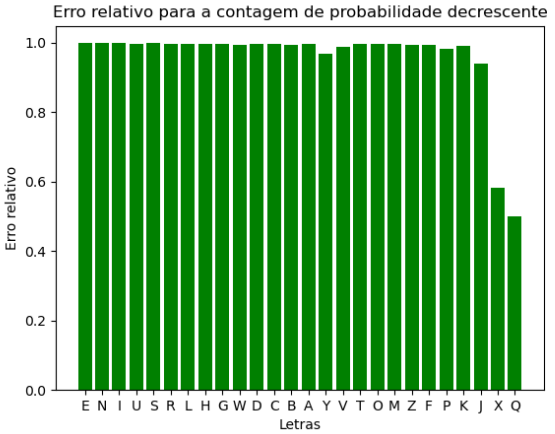

Fig. 15 - Erro relativo para cada letra, na contagem com probabilidade decrescente

Lossy-Count (threshold = 0,0001)	
Menor Erro Absoluto	10
Erro Absoluto Médio	10.0
Maior Erro Absoluto	10
Menor Erro Relativo	0.00055

Erro Relativo Médio	0.0014
Maior Erro Relativo	0.0020

Fig. 16 - Tabela com informações sobre os erros relativos e absolutos

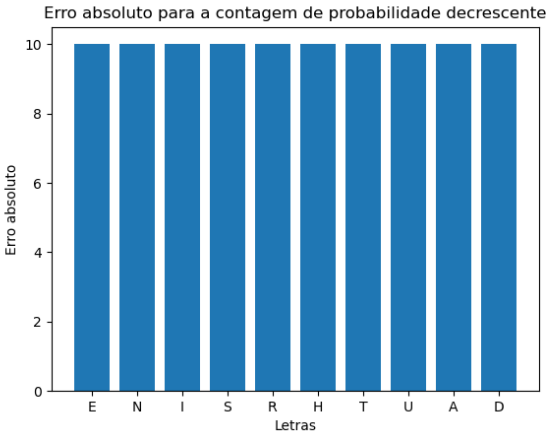

Fig. 17 - Erro absoluto para para as top 10 letras, no Lossy-Count

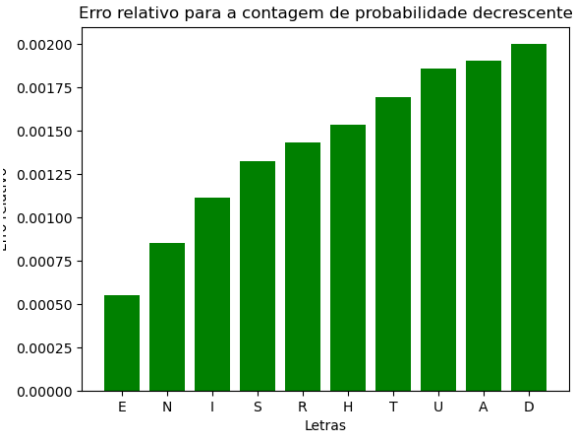

Fig. 18 - Erro relativo para cada letra, no Lossy-Count

C. Inglês

A próxima tabela indica as 10 letras mais frequentes em inglês, para cada um dos métodos implementados.

Contagem Exata			**Contagem com Probabilidade Decrescente**			**Lossy-Count** (threshold = 0,0001)	
E	4661		E	13		E	4658
T	2703		T	12		T	2700
S	2699		R	11		S	2696
R	2663		N	11		R	2660
A	2651		I	11		A	2648
N	2313		H	11		N	2310
O	2186		C	11		O	2183
I	2102		A	11		I	2099
L	1742		U	10		L	1739
D	1453		S	10		D	1450

Fig. 19 - Tabela com as 10 letras mais frequentes em inglês para cada método

Por análise da tabela, verifica-se novamente o que foi concluído da análise das tabelas anteriores.

Em relação aos erros absolutos e relativos aos dois algoritmos, os resultados são os seguintes:

Contagem com probabilidade decrescente	
Menor Erro Absoluto	40
Erro Absoluto Médio	1335.0
Maior Erro Absoluto	4648
Menor Erro Relativo	0.85
Erro Relativo Médio	0.97
Maior Erro Relativo	0.99 (aprox., 1)

Fig. 13 - Tabela com informações sobre os erros relativos e absolutos

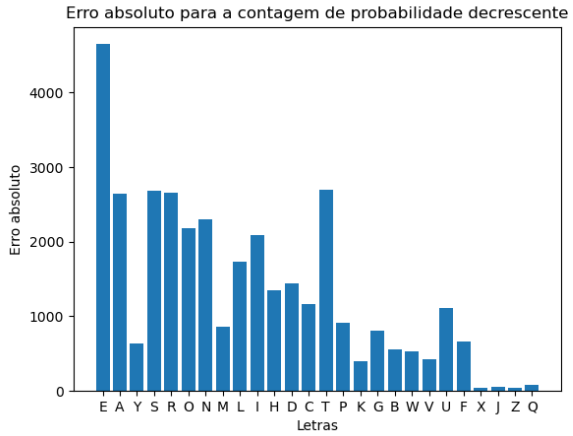

Fig. 14 - Erro absoluto para cada letra, na contagem com probabilidade decrescente

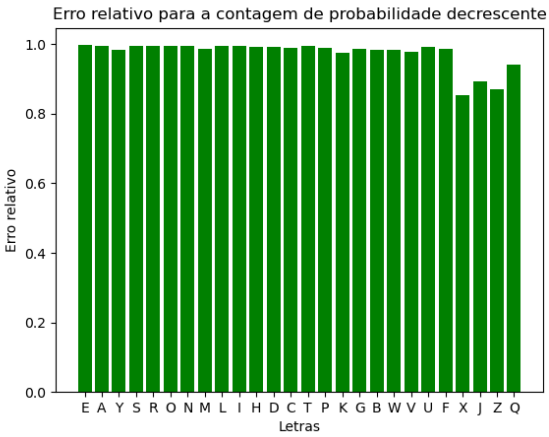

Fig. 15 - Erro relativo para cada letra, na contagem com probabilidade decrescente

Lossy-Count (threshold = 0,0001)	
Menor Erro Absoluto	3
Erro Absoluto Médio	3.0
Maior Erro Absoluto	3
Menor Erro Relativo	0.00064
Erro Relativo Médio	0.0013
Maior Erro Relativo	0.0021

Fig. 16 - Tabela com informações sobre os erros relativos e absolutos

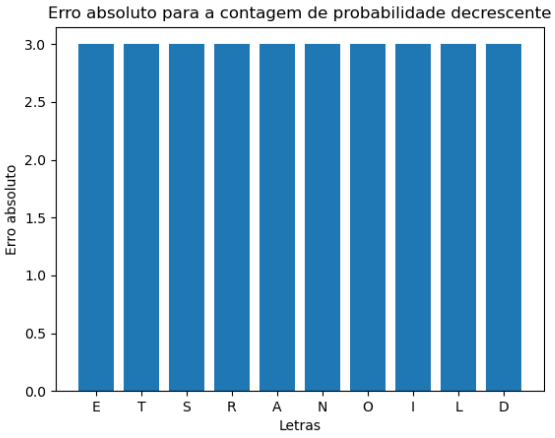

Fig. 17 - Erro absoluto para as top 10 letras, no Lossy-Count

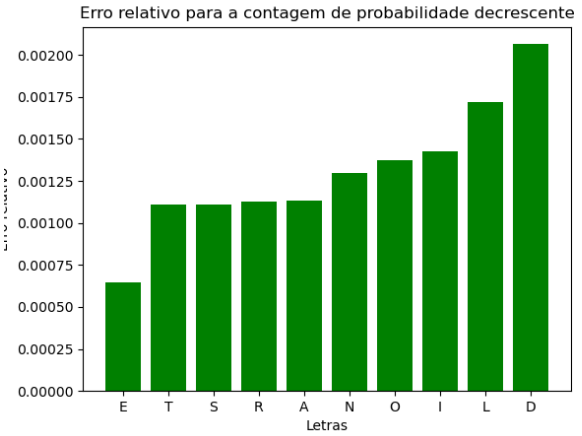

Fig. 18 - Erro relativo para cada letra, no Lossy-Count

VII. CONCLUSÃO

Em suma, após uma análise geral dos resultados, concluímos que, em termos de erros relativos e absolutos, o algoritmo Lossy-Count é significantemente mais preciso que o algoritmo de contagem com probabilidade decrescente.

Contudo, apesar da precisão deste algoritmo não ser tão alta, para um número elevado de eventos, o espaço ocupado é muito menor que o Lossy-Count.

VII. Referências

[1] Approximate Frequency Counts over Data Streams
https://www.vldb.org/conf/2002/S10P03.pdf

[2] Morris Algorithm for Counting
https://iq.opengenus.org/morris-algorithm-for-counting/

[3] Frequency Counting Algorithms over Data Streams | Michael Vogiatzis
https://micvog.com/2015/07/18/frequency-counting-algorithms-over-data-streams/

[4] List of common stop words in various languages.
https://github.com/Alir3z4/stop-words