Most Frequent Words - Estratégias de Contagem

André Oliveira

***Resumo* – O desafio de contar palavras num texto é relativamente simples de abordar. Neste relatório, são exploradas três abordagens distintas para esse propósito: uma contagem exata, uma estimativa aproximada e o algoritmo de Misra & Gries, desenvolvido para identificar os itens mais frequentes em fluxos de dados com um uso eficiente de memória. Estas metodologias foram aplicadas a livros em vários idiomas e analisadas sob diferentes parâmetros, com o objetivo de identificar as condições ideais para a utilização de cada abordagem.**

***Abstract* – The challenge of counting words in a text is relatively straightforward to address. This report explores three distinct approaches for this purpose: exact counting, approximate estimation, and the Misra & Gries algorithm, designed to identify the most frequent items in data streams while optimizing memory usage. These methodologies were applied to books in various languages and analyzed under different parameters to determine the ideal conditions for using each approach.**

***Keywords* – Most Frequent Words, Text Processing, Exact Counter, Approximate Counter, Misra-Gries Algorithm, Frequent-Count**

I. INTRODUÇÃO

A análise da frequência de palavras num texto é uma prática amplamente utilizada em diversas áreas, como processamento de linguagem natural, análise de dados textuais e até em aplicações como motores de busca e sistemas de recomendação. Automatizar este processo é essencial para lidar com grandes volumes de dados, garantindo eficiência e precisão sem depender do esforço humano.

Existem várias abordagens para resolver este problema, desde métodos tradicionais que percorrem o texto contabilizando as ocorrências de cada palavra, até algoritmos mais avançados que utilizam técnicas probabilísticas ou estratégias de otimização de recursos. Estes métodos permitem não apenas identificar as palavras mais frequentes, mas também adaptar a análise a diferentes contextos, como fluxos contínuos de dados ou textos em múltiplos idiomas.

O desenvolvimento dessas soluções abre possibilidades para realizar testes em larga escala, comparar desempenhos e ajustar os métodos às exigências de cada aplicação. Este processo destaca-se como um passo essencial na compreensão e gestão de grandes volumes de informação textual.

A solução mais simples para contabilizar as palavras seria percorrer todo o texto e acumular as ocorrências de cada palavra, de forma semelhante a uma contagem de votos. Embora este método garanta uma precisão total na identificação da frequência e ordem das palavras, pode ser extremamente exigente em termos de memória e processamento, devido ao elevado número de operações de atualização e armazenamento necessárias durante a execução.

Para abordar este problema, foram desenvolvidas duas alternativas adicionais à abordagem já mencionada: uma baseada em probabilidades fixas e outra projetada para otimizar o uso de memória. Ambos os algoritmos foram implementados em Python 3.12, com o respetivo código disponibilizado, que acompanha este documento.

II. PRÉ-PROCESSAMENTO DE TEXTO

Para testar as três abordagens desenvolvidas, foram utilizadas obras literárias de diferentes idiomas disponíveis no Projeto Gutenberg. As obras selecionadas foram: "Don Quijote" de Miguel de Cervantes Saavedra (espanhol), "Os Lusíadas" de Luís de Camões (português) e "The Adventures of Sherlock Holmes" de Arthur Conan Doyle (inglês). Cada obra foi analisada apenas no idioma original, pois o objetivo é comparar os métodos de contagem e não as diferenças entre idiomas. Estes livros foram escolhidos porque, após o processamento, apresentaram menos partes irrelevantes ou ruídos que pudessem afetar a análise, garantindo assim resultados mais consistentes e confiáveis.

Para uniformizar os textos, os ficheiros passaram por um processo de normalização. Inicialmente, foram removidos os cabeçalhos e rodapés referentes ao Projeto Gutenberg, mantendo apenas o conteúdo principal de cada obra. Em seguida, eliminaram-se palavras comuns (stopwords) e sinais de pontuação, utilizando listas de stopwords específicas para cada idioma, disponíveis na pasta "stopwords" do projeto. Além disso, todas as palavras foram convertidas para letras minúsculas, e as ilustrações e quebras de linha foram eliminadas, criando uma sequência contínua de palavras. Estas etapas de normalização asseguraram que os textos estivessem preparados para a análise, minimizando diferenças indesejadas nos dados e maximizando a precisão da avaliação.

```
def load_stopwords(lang):
    with open(f"stopwords/stopwords_{lang}.txt",
        encoding="utf8") as file:
        return file.read().split("\n")


def process_files():
    books = [
        book.replace(".txt", "")
        for book in os.listdir("Project_Gutenberg
            ")
        if book != ".gitkeep"
    ]

    processed_books = {}

    stats = open("statistics/text_processing.txt"
        , "w", encoding="utf8")
```

```
    stats.write(f'{"Title":<40} {"Initial Length
        ":<25} {"Final Length"}\n')

    for book in books:
        lang = book[-2:]
        stopwords = load_stopwords(lang)

        with open("Project_Gutenberg/" + book + "
            .txt", encoding="utf8") as file:
            book = book[:-3]
            text = file.read()
            initial_length = len(text)
            header = f"*** START OF THE PROJECT
                GUTENBERG EBOOK {book.upper()}
                ***"
            start = text.find(header) + len(
                header)
            footer = f"*** END OF THE PROJECT
                GUTENBERG EBOOK {book.upper()}
                ***"
            end = text.find(footer)
            text = text[start:end]
            text = text.replace("[Illustration]",
                 "")
            punctuation = string.punctuation
            for char in punctuation:
                text = text.replace(char, " ")
            for char in text:
                if not char.isalpha() and char !=
                     " ":
                    text = text.replace(char, " "
                        )
            text = " ".join(
                [word for word in text.split() if
                     word.lower() not in
                    stopwords]
            )
            text = text.lower()
            final_length = len(text)
            processed_books[book] = text

        stats.write(f"{book:<40} {initial_length
            :<25} {final_length}\n")

    stats.close()

    return processed_books
```

III. CONTADOR EXATO

O contador exato é implementado para calcular com precisão a frequência de cada palavra num fluxo de texto. O funcionamento baseia-se na divisão do texto em palavras individuais e na contabilização das suas ocorrências, utilizando a classe *Counter* da biblioteca *collections*. Esta implementação garante resultados exatos e é particularmente útil para análises que requerem precisão absoluta.

O código responsável por esta funcionalidade é o seguinte:

```
def exact_counter(stream):
    start = time.time()
    words = stream.split()
    return Counter(words), time.time() - start
```

A função *exact_counter* recebe como entrada um fluxo de texto (*stream*). Inicialmente, regista o tempo de início do processamento com a função *time.time()*. Em seguida, divide o texto em palavras individuais utilizando o método *split*, que separa o texto com base nos espaços. A classe *Counter* é então utilizada para calcular a frequência de cada palavra, retornando um dicionário onde as chaves são palavras e os valores correspondem às suas frequências. Por fim, a função retorna o dicionário gerado e o tempo total de execução, calculado como a diferença entre o tempo atual e o momento de início do processamento.

Esta abordagem é eficaz para determinar as palavras mais frequentes num texto, fornecendo não apenas o resultado, mas também o tempo necessário para o cálculo, o que é útil para avaliar o desempenho do algoritmo em diferentes cenários. Embora seja preciso, é importante considerar que a utilização do *Counter* pode ser limitada por restrições de memória em fluxos de texto extremamente grandes, uma vez que todas as palavras precisam ser carregadas na memória.

IV. CONTADOR APROXIMADO

O contador aproximado é projetado para reduzir o consumo de memória ao custo de sacrificar a precisão dos resultados. Neste caso, foi implementado um algoritmo que utiliza uma probabilidade fixa de 1/16 para contabilizar palavras. A cada palavra do texto, é gerado um número aleatório entre 0 e 1. Caso esse número seja inferior ou igual a 1/16, a palavra é adicionada ou tem a sua contagem incrementada no dicionário responsável pela frequência.

O código que implementa esta funcionalidade é o seguinte:

```
def approximate_counter(stream):
    start = time.time()
    words = stream.split()
    counter = {}
    for word in words:
        if word not in counter:
            counter[word] = 0

        prob = 1 / 16
        if random.random() <= prob:
            counter[word] += 1

    return counter, time.time() - start
```

A função *approximate_counter* inicia registando o tempo de execução com *time.time()*. O fluxo de texto é então dividido em palavras individuais usando o método *split()*. Um dicionário é inicializado para armazenar as contagens. Para cada palavra, se ela ainda não estiver no dicionário, é criada uma entrada com o valor inicial de 0. Com uma probabilidade de $\frac{1}{16}$, a contagem da palavra é incrementada. No final, a função retorna o dicionário contendo as contagens aproximadas e o tempo total de execução.

Este método tem como principal vantagem a economia de memória, uma vez que se espera armazenar apenas cerca de $\frac{1}{16}$ das ocorrências de cada palavra. Em termos de armazenamento, isto resulta numa redução significativa no número total de ocorrências registadas, tornando o algoritmo viável para grandes volumes de texto.

Contudo, é importante observar que a pseudoaleatoriedade utilizada pode impactar a precisão das frequências relativas. Em casos de palavras com frequências semelhantes, a ordem resultante pode não refletir a realidade devido à variabilidade introduzida pelo fator aleatório. Para avaliar o impacto dessa imprecisão, os resultados deste contador foram comparados com os do contador exato multiplicados por $\frac{1}{16}$, verificando a consistência das ordens geradas e a adequação deste método para diferentes cenários.

Don Quijote		**Os Lusíadas**		*The Adventures of Sherlock Holmes*	
Exato	**Approx.**	**Exato**	**Approx.**	**Exato**	**Approx.**
don	don	gente	mar	said	s
quijote	quijote	terra	céu	holmes	holmes
sancho	sancho	rei	gente	s	said
señor	merced	mar	terra	man	know
respondi	señor	co	rei	mr	mr
merced	caballero	mundo	co	little	think
caballero	respondi	reino	outro	think	little
dios	cosa	céu	armas	room	room
señora	dios	forte	ó	know	like
cosa	mundo	ó	mundo	shall	yes

TABELA I
10 PALAVRAS MAIS FREQUENTES DE CADA LIVRO PARA OS CONTADORES EXATO E APROXIMADO (1 EXECUÇÃO)

Don Quijote		**Os Lusíadas**		*The Adventures of Sherlock Holmes*	
Exato	**Approx.**	**Exato**	**Approx.**	**Exato**	**Approx.**
don	don	gente	gente	said	said
quijote	quijote	terra	terra	holmes	holmes
sancho	sancho	rei	rei	s	s
señor	respondió	mar	mar	man	man
respondi	señor	co	co	mr	mr
merced	merced	mundo	mundo	little	little
caballero	caballero	reino	reino	think	think
dios	dios	céu	céu	room	know
señora	señora	forte	forte	know	room
cosa	cosa	ó	ó	shall	shall

TABELA II
10 PALAVRAS MAIS FREQUENTES DE CADA LIVRO PARA OS CONTADORES EXATO E APROXIMADO (1000 EXECUÇÕES)

As tabelas apresentadas permitem uma análise detalhada do desempenho dos contadores exato e aproximado, considerando que as células destacadas a cinzento representam palavras cuja ordem ou existência nos resultados aproximados diverge do contador exato.

Na Tabela 1, com apenas uma execução, observa-se que o contador aproximado consegue capturar corretamente algumas palavras mais frequentes, mas apresenta várias discrepâncias em relação ao método exato. Essas discrepâncias, destacadas a cinzento, evidenciam que o fator aleatório do contador aproximado afeta a precisão, demonstrando que uma única execução do contador aproximado não é suficientemente confiável para capturar corretamente todas as frequências ou manter a ordem das palavras mais frequentes.

Na Tabela 2, com 1000 execuções, nota-se uma melhoria significativa. As discrepâncias são muito menos frequentes, indicando que o contador aproximado, com execuções repetidas, consegue convergir para resultados mais alinhados com o contador exato. Embora ainda haja algumas palavras a cinzento, como “room” e “know” no inglês, a maioria das palavras mais frequentes está corretamente posicionada, o que demonstra que o impacto da aleatoriedade diminui com múltiplas execuções. Esse comportamento confirma que a precisão do contador aproximado melhora consideravelmente quando a análise é realizada várias vezes e os resultados são acumulados.

O contador aproximado, apesar das discrepâncias numa única execução, ainda é útil para identificar tendências gerais nas palavras mais frequentes. Entretanto, as palavras destacadas a cinzento evidenciam que ele pode falhar em capturar corretamente a ordem ou mesmo incluir palavras erradas em cenários onde a precisão é essencial. Para mitigar este problema, múltiplas execuções, como mostrado na Tabela 2, são fundamentais para melhorar a confiabilidade dos resultados.

Em aplicações práticas, o contador aproximado é adequado para situações em que o objetivo é ter uma visão geral das palavras mais frequentes em grandes textos, sem a necessidade de precisão total. No entanto, para análises mais detalhadas ou que dependam de frequências exatas, como validações ou estudos linguísticos rigorosos, o método exato continua a ser indispensável. Assim, cada método deve ser utilizado de acordo com o contexto e os requisitos específicos, com a consciência de que o contador aproximado, embora eficiente em termos de memória e processamento, apresenta limitações evidentes quando utilizado de forma isolada.

Livro	*Don Quijote*
Valor esperado	9129.93
Variância esperada	4564.96
Desvio padrão esperado	67.56
Valor counter	9021
Erro absoluto	108.93
Erro relativo	1.19%
Precisão	98.81%
Precisão do Top 10	90%
Precisão do Top 10 (com ordem)	30%

TABELA III
RESULTADOS PARA O CONTADOR APROXIMADO COM 1 EXECUÇÃO NO LIVRO ’DON QUIJOTE’

Livro	*Don Quijote*
Valor esperado	9129.93
Variância esperada	4564.96
Desvio padrão esperado	67.56
Valor médio do counter	9130.82
Erro absoluto médio	76.34
Erro relativo médio	0.84%
Precisão média	99.16%
Menor valor do counter	8866
Maior valor do counter	9398
Desvio absoluto médio	76.34
Desvio padrão	94.69
Desvio máximo	267.17
Variância	8967.51
Precisão do Top 10	100%
Precisão do Top 10 (com ordem)	80%

TABELA IV
RESULTADOS PARA O CONTADOR APROXIMADO COM 1000 EXECUÇÕES NO LIVRO 'DON QUIJOTE'

Book	*Os Lusíadas*
Valor esperado	1724.37
Variância esperada	862.18
Desvio padrão esperado	29.36
Valor counter	1662
Erro absoluto	62.37
Erro relativo	3.62%
Precisão	96.38%
Precisão do Top 10	80%
Precisão do Top 10 (com ordem)	0%

TABELA V
RESULTADOS PARA O CONTADOR APROXIMADO COM 1 EXECUÇÃO NO LIVRO 'OS LUSÍADAS'

Book	*Os Lusíadas*
Valor esperado	1724.37
Variância esperada	862.18
Desvio padrão esperado	29.36
Valor médio do counter	1723.51
Erro absoluto médio	32.06
Erro relativo médio	1.86%
Precisão média	98.14%
Menor valor do counter	1581
Maior valor do counter	1866
Desvio absoluto médio	32.07
Desvio padrão	40.05
Desvio máximo	142.51
Variância	1604.32
Precisão do Top 10	100%
Precisão do Top 10 (com ordem)	100%

TABELA VI
RESULTADOS PARA O CONTADOR APROXIMADO COM 1000 EXECUÇÕES NO LIVRO 'OS LUSÍADAS'

Book	*The Adventures of Sherlock Holmes*
Valor esperado	2523.75
Variância esperada	1261.87
Desvio padrão esperado	35.52
Valor counter	2595
Erro absoluto	71.25
Erro relativo	2.82%
Precisão	97.18%
Precisão do Top 10	80%
Precisão do Top 10 (com ordem)	30%

TABELA VII
RESULTADOS PARA O CONTADOR APROXIMADO COM 1 EXECUÇÃO NO LIVRO 'THE ADVENTURES OF SHERLOCK HOLMES'

Book	*The Adventures of Sherlock Holmes*
Valor esperado	2523.75
Variância esperada	1261.87
Desvio padrão esperado	35.52
Valor médio do counter	2524.18
Erro absoluto médio	39.37
Erro relativo médio	1.56%
Precisão média	98.44%
Menor valor do counter	2357
Maior valor do counter	2666
Desvio absoluto médio	39.38
Desvio padrão	49
Desvio máximo	167.18
Variância	2401.42
Precisão do Top 10	100%
Precisão do Top 10 (com ordem)	80%

TABELA VIII
RESULTADOS PARA O CONTADOR APROXIMADO COM 1000 EXECUÇÕES NO LIVRO 'THE ADVENTURES OF SHERLOCK HOLMES'

Com base nos resultados apresentados nas tabelas, que consideram o número total de palavras em cada livro, é possível observar um padrão consistente no desempenho do contador aproximado em diferentes obras, tanto com uma única execução quanto com múltiplas execuções. Resultados adicionais, como os obtidos para 10 e 100 iterações, encontram-se disponíveis na pasta *statistics/approximate_counter* para consulta e análise detalhada.

Numa única execução, o contador aproximado apresenta resultados razoáveis, mas com variações notáveis em relação aos valores esperados. O erro absoluto e o erro relativo mostram que a precisão é limitada, especialmente na ordem das palavras mais frequentes. A precisão do Top 10 é geralmente boa, variando entre 80% e 90%, mas a precisão do Top 10 (com ordem) é baixa, em torno de 30% e até mesmo 0%, indicando dificuldade em capturar a hierarquia correta das palavras. Este comportamento reforça que,

numa única execução, o contador aproximado é útil apenas para identificar tendências gerais, mas não para análises que exijam exatidão ou detalhes precisos.

Com 1000 execuções, o contador aproximado apresenta uma melhoria significativa na precisão e consistência dos resultados. O erro absoluto médio diminui consideravelmente, e o erro relativo médio é reduzido, aproximando-se dos valores esperados. A precisão do Top 10 atinge 100% nos 3 livros, enquanto a precisão do Top 10 (com ordem) também aumenta significativamente, variando entre 80% e 100%. Isso demonstra que múltiplas execuções são eficazes para minimizar o impacto da aleatoriedade do método, permitindo uma análise mais confiável.

Além disso, com múltiplas execuções, observa-se uma maior estabilidade no desempenho. O desvio padrão e a variância dos valores registados são menores, e os desvios absolutos e máximos indicam que os resultados convergem para valores consistentes. A pequena diferença entre os valores máximos e mínimos também reflete a pouca dispersão nos resultados, tornando o método mais confiável em identificar padrões frequentes.

De maneira geral, o contador aproximado é uma alternativa viável ao contador exato, especialmente em cenários onde há restrições de memória ou processamento. No entanto, o seu desempenho é fortemente dependente do número de execuções. Enquanto uma única execução pode fornecer uma visão geral das palavras mais frequentes, múltiplas execuções são indispensáveis para garantir maior precisão e consistência, aproximando-se dos resultados do contador exato. Assim, o método é ideal para análises exploratórias em grandes volumes de texto, desde que o número de execuções seja suficiente para compensar as limitações da abordagem aproximada.

V. ALGORITMO DE MISRA-GRIES

O algoritmo de Misra-Gries, também conhecido como Frequent-Count, é uma técnica amplamente conhecida para identificar os itens mais frequentes em fluxos de dados. A sua principal característica é ser um algoritmo de passagem única (*one-pass*), o que significa que ele precisa de processar o fluxo de dados apenas uma vez. Esta abordagem torna-o particularmente eficiente e ideal para aplicações em tempo real, como processamento de linguagem natural, indexação de bases de dados e análise de grandes volumes de dados.

Uma variação comum deste algoritmo é o Misra-Gries com parâmetro k, que permite a deteção dos $k-1$ itens mais frequentes em um fluxos de dados. Para alcançar isso, o algoritmo mantém $k-1$ contadores, cada um associado a um item potencialmente frequente. Sempre que um item é processado, o contador correspondente é incrementado. Se o item não tiver um contador associado e todos os contadores estiverem ocupados, todos os contadores são decrementados simultaneamente. Este mecanismo assegura que o algoritmo rastreie apenas os $k-1$ itens mais frequentes, garantindo que nenhum item com frequência igual ou superior a $\frac{m}{k}$, onde m é o tamanho total do fluxo, seja ignorado. O parâmetro k, portanto, controla o equilíbrio entre eficiência e qualidade dos resultados.

O grande destaque deste algoritmo é a sua eficiência computacional. Ele consegue processar fluxos de dados de grandes dimensões em questão de milissegundos, tornando-o altamente aplicável em sistemas que exigem processamento em tempo real. Além disso, a sua implementação é relativamente simples, exigindo apenas um número limitado de contadores e operações aritméticas básicas. O código responsável por esta funcionalidade é o seguinte:

```
def frequent_counter(stream, k):
    start = time.time()
    words = stream.split()
    counter = {}

    for word in words:
        if word in counter:
            counter[word] += 1
        elif len(counter) < k - 1:
            counter[word] = 1
        else:
            for key in list(counter.keys()):
                counter[key] -= 1
                if counter[key] == 0:
                    del counter[key]

    return counter, time.time() - start
```

Esta simplicidade e eficiência tornam o algoritmo amplamente utilizado em diversas áreas, como processamento de grandes conjuntos de dados e aplicações de alta frequência, onde o tempo de resposta é crítico.

No entanto, como um algoritmo aproximado, o Misra-Gries tem as suas limitações. Ele fornece estimativas das frequências dos itens, em vez de contagens exatas. Essa característica é um compromisso necessário para alcançar alta eficiência, mas pode não ser adequado para aplicações que exigem precisão absoluta nas contagens. Em particular, ele não consegue identificar itens cuja frequência seja muito próxima do limite de deteção $\frac{m}{k}$, o que pode levar a erros em cenários onde há pouca diferença nas frequências dos itens analisados.

O uso de $k-1$ contadores torna o algoritmo altamente eficiente em termos de memória, uma vez que o número de contadores cresce linearmente com o parâmetro k. Isto é especialmente útil em contextos onde a memória disponível é restrita, como dispositivos com recursos limitados ou sistemas distribuídos. Além disso, o algoritmo é robusto, adaptando-se bem a fluxos de dados dinâmicos e lidando com grandes dimensões sem a necessidade de armazenar todo o fluxo em memória.

k	Correct Words	Accuracy
100	15/99	15.15%
200	40/199	20.1%
500	185/499	37.07%

TABELA IX
PRECISÃO A DETERMINAR AS PALAVRAS MAIS FREQUENTES PARA O LIVRO 'DON QUIJOTE'

k	Correct Words	Accuracy
100	11/99	11.11%
200	16/199	8.04%
500	138/499	27.66%

TABELA X
PRECISÃO A DETERMINAR AS PALAVRAS MAIS FREQUENTES PARA O LIVRO 'OS LUSÍADAS'

k	Correct Words	Accuracy
100	19/99	19.19%
200	74/199	37.19%
500	213/499	42.69%

TABELA XI
PRECISÃO A DETERMINAR AS PALAVRAS MAIS FREQUENTES PARA O LIVRO 'THE ADVENTURES OF SHERLOCK HOLMES'

Com base nas tabelas apresentadas, é evidente que o desempenho do algoritmo melhora significativamente à medida que o valor de k aumenta, refletindo uma maior precisão na identificação das palavras mais frequentes em cada livro. Para $k = 100$, a precisão é baixa em todos os casos, variando de 11.11% a 19.19%, indicando que o número de contadores é insuficiente para capturar um conjunto representativo das palavras mais frequentes. À medida que k aumenta para 200 e 500, observa-se uma melhoria clara, especialmente em livros maiores como *The Adventures of Sherlock Holmes*, onde a precisão alcança 42.69% para $k = 500$.

A escolha dos valores de k reflete a necessidade de ajustar o número de contadores à complexidade e ao tamanho do texto analisado. Livros contêm um grande número de palavras únicas, mas a maioria delas ocorre com baixa frequência. Valores baixos de k podem ser insuficientes para representar adequadamente as palavras mais frequentes, especialmente em textos mais extensos. Por outro lado, valores mais altos de k permitem que o algoritmo rastreie um número maior de palavras, capturando com maior precisão aquelas que aparecem frequentemente.

Em resumo, os valores de k foram escolhidos para balancear eficiência e precisão. Enquanto valores menores são mais econômicos em termos de memória, valores maiores como $k = 500$ são essenciais para alcançar resultados mais confiáveis, especialmente em livros com maior volume de palavras. Estes resultados confirmam a importância de calibrar k em função do tamanho e da diversidade do texto analisado.

VI. TEMPOS DE EXECUÇÃO

Em termos de tempos de execução, os resultados são os seguintes:

Contador	Tempo de Execução
Exato	0.01330113410949707
Aproximado (1 iteração)	0.015590906143188477
Aproximado (10 iterações)	0.01916794776916504
Aproximado (100 iterações)	0.016012599468231203
Aproximado (1000 iterações)	0.015456143379211426
Misra-Gries ($k = 100$)	0.025033950805664062
Misra-Gries ($k = 200$)	0.025233983993530273
Misra-Gries ($k = 500$)	0.026027679443359375

TABELA XII
TEMPOS DE EXECUÇÃO PARA OS 3 ALGORITMOS DE CONTAGEM NO LIVRO "DON QUIJOTE"

Contador	Tempo de Execução
Exato	0.0023529529571533203
Aproximado (1 iteração)	0.0030939579010009766
Aproximado (10 iterações)	0.00301663875579834
Aproximado (100 iterações)	0.0033066272735595703
Aproximado (1000 iterações)	0.0032349305152893065
Misra-Gries ($k = 100$)	0.0051021575927734375
Misra-Gries ($k = 200$)	0.00465703010559082
Misra-Gries ($k = 500$)	0.0050411224365234375

TABELA XIII
TEMPOS DE EXECUÇÃO PARA OS 3 ALGORITMOS DE CONTAGEM NO LIVRO "OS LUSÍADAS"

Contador	Tempo de Execução
Exato	0.0031981468200683594
Aproximado (1 iteração)	0.0042269229888916016
Aproximado (10 iterações)	0.0039055347442626953
Aproximado (100 iterações)	0.003934671878814697
Aproximado (1000 iterações)	0.003959594249725341
Misra-Gries ($k = 100$)	0.0070798397064208984
Misra-Gries ($k = 200$)	0.007016897201538086
Misra-Gries ($k = 500$)	0.007129192352294922

TABELA XIV
TEMPOS DE EXECUÇÃO PARA OS 3 ALGORITMOS DE CONTAGEM NO LIVRO "THE ADVENTURES OF SHERLOCK HOLMES"

Com base nos tempos de execução apresentados nas tabelas, observa-se que as diferenças entre os algoritmos são relativamente pequenas, independentemente do contador utilizado ou do valor de k. Isto ocorre devido ao tamanho relativamente reduzido dos livros analisados. Em dados de menor escala, a sobrecarga computacional dos contadores aproximados e do algoritmo Misra-Gries é marginal, resultando em tempos semelhantes ao do contador exato.

No entanto, em volumes de dados significativamente maiores, essas diferenças seriam mais evidentes. O contador exato teria um aumento expressivo no tempo de execução devido ao armazenamento e processamento de todas as palavras, enquanto os contadores aproximados e o algoritmo Misra-Gries, que utilizam abordagens otimizadas com menor uso de memória, manteriam maior eficiência. Assim, a

escolha do algoritmo deve levar em consideração o volume de dados a ser processado, sendo os contadores aproximados e o Misra-Gries preferidos para fluxos de dados extensos.

VII. CONCLUSÃO

Com base nas análises realizadas ao longo deste relatório, conclui-se que as três abordagens estudadas – contador exato, contador aproximado e algoritmo de Misra-Gries – possuem características complementares e podem ser aplicadas de forma estratégica, dependendo das exigências específicas do problema e das limitações de recursos computacionais.

O contador exato é a abordagem mais precisa e ideal para cenários em que a exatidão total é indispensável, como em análises linguísticas rigorosas ou validações de frequência. No entanto, o elevado consumo de memória e o aumento significativo do tempo de execução em fluxos de dados extensos limitam a sua aplicabilidade em grandes volumes de texto.

Por outro lado, o contador aproximado demonstra ser uma alternativa eficiente em termos de memória e processamento, especialmente para análises exploratórias onde a precisão absoluta não é crucial. Contudo, a precisão da abordagem aproximada depende diretamente do número de iterações realizadas. Enquanto uma única execução apresenta resultados inconsistentes, múltiplas execuções aumentam significativamente a confiabilidade e a estabilidade dos resultados, aproximando-se dos valores do contador exato.

O algoritmo de Misra-Gries destaca-se como uma solução intermediária, combinando eficiência e simplicidade. A sua capacidade de rastrear os itens mais frequentes com um número limitado de contadores torna-o particularmente útil para grandes fluxos de dados ou dispositivos com restrições de memória. Apesar de ser uma abordagem aproximada, o controlo por meio do parâmetro k permite ajustar a qualidade dos resultados às necessidades do problema em questão.

A análise dos tempos de execução evidencia que as diferenças entre os algoritmos são mínimas em livros de menor escala, mas tornam-se significativas em volumes de dados maiores. O contador exato apresenta uma sobrecarga computacional considerável em cenários de alta escala, enquanto o Misra-Gries e o contador aproximado mantêm alta eficiência, mesmo sob cargas mais exigentes.

Em resumo, a escolha do algoritmo deve ser guiada pelos requisitos da aplicação. Para cenários que exigem precisão absoluta, o contador exato é indispensável. No entanto, para análises exploratórias ou contextos com limitações de recursos, o contador aproximado e o Misra-Gries são alternativas robustas e eficientes. Este estudo destaca a importância de calibrar os parâmetros dos algoritmos e adaptar a abordagem ao tamanho e à complexidade dos dados, maximizando a eficiência e a qualidade das análises realizadas.

Por fim, é importante mencionar que não foi possível identificar claramente as palavras menos frequentes, uma vez que os valores apresentam pouca variação, tornando difícil distingui-las. Além disso, optei por não considerar as palavras mais ou menos frequentes em livros de diferentes idiomas, pois, sendo uma questão de tradução, não julguei relevante para a análise em questão.

BIBLIOGRAFIA

[1] `AA_2425_Trab_3`

[2] `AA09ProbabilisticCountersFicheiro`

[3] `AA11DataStreamAlgorithmsIFicheiro`