Contagem Aproximada de Ocorrências – Palavras em Ficheiros de Texto

Pedro Xavier Leite Cavadas, Nº Mec. 85090, MEI

***Resumo* – Este relatório é realizado no âmbito da cadeira Algoritmos Avançados do Mestrado em Engenharia Informática da Universidade de Aveiro.**

Neste relatório pretende-se analisar o desempenho de um contador aproximado com probabilidade fixa e de um contador aproximado com probabilidade decrescente logarítmica e compará-los entre si bem como a um contador exato.

Para isso será apresentado, detalhadamente, o problema, a estratégia de resolução utilizada, a solução obtida, bem como testes de *performance* e a análise aos mesmos. Isto é feito aplicando os contadores a palavras de uma dada obra.

***Abstract* – This report is written under the Advanced Algorithms course of the Master in Software Engineering, Universidade de Aveiro.**

This is repots aims to analyse the performance of an approximate fixed-probability counter and a logarithmic decreasing-approximate counter and to compare them with an exact counter.

With this in mind, a detailed explanation of the problem will be given, as well as, the resolution strategy, the solution obtained, the results of the performance testing and their respective analysis. This is done applying the counters to words of a given book.

I. INTRODUÇÃO

Este relatório é realizado no âmbito da cadeira Algoritmos Avançados onde é nos proposto um problema junstamente com um método de resolução. O objetivo do trabalho proposto é três contadores: um exato, um aproximado com probabilidade fixa e um aproximado com probabilidade decrescente logarítmica. O de probabilidade fixa deverá ter uma probabilidade igual a 1/64 e o de probabilidade decrescente logarítmica deverá ser de base 2. Além disto é necessário realizar-se testes de *performance* que comparem os vários contadores em termos de tempo gasto, memória gasta e exatidão dos valores obtidos. Além disto é necessário também comparar o *top-n* de palavras mais recorrentes para traduções diferentes do livro.

Tendo isto em conta foi escolhido o livro "Alice's Adventures in Wonderland" de Lewis Carroll. Além da versão em inglês, foram utilizadas as versões em italiano, francês e alemão.

Este projeto divide-se então em duas partes: um s*cript python* com a implementação dos contadores e outro que realiza os vários testes sobre os contadores.

Este relatório descrebe então alguns conceitos técnicos necessários para a resolução do problema, uma análise detalhada ao problema, à estratégia, ao algoritmo, à implementação, aos testes e à sua respetiva análise.

No final serão retiradas algumas conclusões acerca dos contadores.

II. CONCEITOS TÉCNICOS

A. CONTADOR EXATO

O tipo de contador mais simples existente. Sempre que se encontra uma ocorrência que se deseja contar incrementa-se o contador uma unidade.

B. CONTADOR APROXIMADO COM PROBABILIDADE FIXA

Um contador que é também ainda relativamente simples. O seu funcionamento é idêntico ao contador exato com a exceção que se incrementa apenas com base numa dada probabilidade. No final para obter o número de ocorrências multiplica-se o valor do contador pelo inverso da probabilidade.

Caso se queira obter um valor mais aproximado do valor real é necessário realizar-se várias contagens e no final o número de eventos é a média do número de ocorrências obtido em cada uma das contagens.

C. CONTADR APROXIMADO COM PROBABILIDADE DECRESCENTE LOGARITMICA

Um contador já um pouco mais complexo, que tem um funcionamento idêntico ao de probabilidade fixa sendo que a principal diferença é o facto de que a probabilidade é dada em função de uma base e do valor atual do contador (probabilidade = $\frac{1}{base^{contador}}$). No final para obter o número de ocorrências aplica-se a seguinte fórmula:

$$\frac{base^{contador} - base + 1}{base - 1}$$

Tal como no contador anterior pode-se calcular a média para obter um valor mais próximo do real.

III. Estratégia

Para a resolução do problema proposto, foram então implementados os três contadores indicados. Estes estão definidos em três classes:

- Classe Counter: implementa um contador exato;
- Classe LogarithmicCounter: implementa um contador aproximado com probabilidade decrescente logarítmica;
- Classe ProbabilisticCounter: implementa um contador aproximado com probabilidade fixa.

Estas classes implementam todas os mesmos métodos:

- get: retorna o número de ocorrências estimado;
- increment: incrementa o número de ocorrências;
- reset: reinicializa o contador a 0.

```
class Counter:
    def __init__(self):
        self.reset()

    def get(self):
        return self.count

    def increment(self):
        self.count += 1

    def reset(self):
        self.count = 0

class ProbabilisticCounter(Counter):
    def __init__(self, a = 2):
        super().__init__()
        self._probability = 1 / a
        self._a = a

    def get(self):
        return super().get() * self._a

    def increment(self):
        if random.random() < self._probability:
            self.count += 1

class LogarithmicCounter(ProbabilisticCounter):
    def get(self):
        b = self._a - 1
        return (self._a ** self.count - b) / b

    def increment(self):
        if random.random() < self._probability ** self.count:
            self.count += 1
```

Posto isto, o livro nas suas várias versões é carregado para memória. Neste processo toda a pontuação é removida, as palavras passam a ser todas em minúsculas e são então colocadas todas numa lista pelo ordem em que aparecem no livro. Finalmente algumas palavras como artigos definidos e indefinidos, entre outros, são removidas. Este processo consiste em carregar para memória um ficheiro com as palavras a serem removidas e então removê-las verificando a sua existência nesse ficheiro (este tipo de lista é comumente denominado de *stopwords list*).

```
def load_book(path, stopwords, encoding = 'ascii'):
    with open(path, 'r', encoding = encoding) as fin:
        return [ word for word in re.split(r' +', fin.read().translate
(translator).strip().lower()) if len(word) > 2 and word not in st
opwords ]
    return [ ]

def load_stopwords(path):
    with open(path, 'r', encoding = 'utf-8') as fin:
        return [ line.strip() for line in fin.readlines() if len(line.st
rip()) != 0 ]
    return [ ]

def main():
    languages = [ 'french', 'german', 'english', 'italian' ]
    encodings = [ 'utf-8', 'iso-8859-1', 'iso-8859-1', 'iso-8859-
1' ]
    stopwords = { language : load_stopwords('stopwords\\{}.t
xt'.format(language)) for language in languages }
    books = { language : load_book('books\\stripped_{}.txt'.fo
rmat(language), stopwords[language], encoding = encoding)
for language, encoding in zip(languages, encodings) }
```

Finalmente realiza-se então os testes de *performance*. Nesta fase, cada contador faz a contagem das palavras para cada uma das versões do livro 10, 100, 1000 e 10000 vezes. Para cada uma destas instâncias e para cada uma das versões é calculado o tempo gasto a realizar a contagem, o número de *bits* máximo necessário para o contador (valor real e utilizado, onde o real é um "arredondamento para cima"), além disto para os contadores aproximados são calculadas várias métricas para cada uma das palavras, nomeadamente:

- Errro relativo máximo, erro relativo mínimo, error relativo médio, erro absoluto médio,

valor esperado, valor máximo contado, valor mínimo contado, valor médio contado, desvio médio absoluto, desvio máximo, desvio padrão, variância, número esperado de ocorrências e número de ocorrências registado.

De seguida são apresentados os resultados para às 10 primeiras palavras com maior ocurrência, uma média de todas as palavras e os resultados de tempo gasto e de memória máxima necessária por contador.

IV. Implementação

A implementação do algoritmo e da resolução do problema é feita na linguagem *Python3*. A seguir estão apresentados os módulos implementados e as bibliotecas utilizadas na sua implementação.

A. Módulos

a. *counters.py* – módulo que contém as três classes de contadores;
b. *main.py* – este módulo contém as funções para carregar os livros e as listas de palavras a filtrar, bem como funções para realizar os testes de *performance*. Além disto, contém uma função main quando o módulo é utilizado como *main script*.

B. Bibliotecas Utilizadas

a. *Itertools* – módulo nativo ao *Python3* que fornece ferramentas para a geração. bem como manipulação de iteradores. Neste trabalho em particular foi utilizada para selecionar as 10 primeiras palavras em conjunto com a função "sorted";

b. *Sys* – módulo nativo ao *Python3* que fornece ferramentas de interação com o sistema nativo da máquina. Neste projeto foi utilizado para imprimir para o *standard output*;

c. *Math* – Também um módulo nativo ao *Python3*, que fornece ferramentas matemáticas. Utilizado para arredondamentos e cálculos de logaritmos e raízes quadradadas;

d. *Random* – Outro módulo nativo ao *Python3*. Contém ferramentas de geração de valores aleatórios. Neste trabalho utilizado para a geração de números aleatórios utilizados nos contadores aproximados;

e. *Time* – Último módulo nativo ao *Python3* utilizado. Disponibiliza ferramentas que permitem trabalhar com o tempo, tais como obter o *timestamp* atual, entre outros;

f. *Re* – módulo nativo do *Python3*. Este módulo permite a utilização de expressões regulares e foi utilizado para dividir a *string* com o conteúdo do livro numa lista de *strings*;

g. *String* – módulo nativo do *Python3*. Este módulo contém algumas funcionalidades relacionadas com *strings*. Foi utilizado neste trabalho para remover sinais de pontuação;

h. *Argparse* – módulo nativo do *Python3*. Este módulo permite de forma simples realizar o *parsing* dos argumentos do programa e foi utilizado para detetar se o utilizar quer escrever os resultados em disco e para que ficheiro o quer fazer;

i. *Venv* – módulo nativo do *Python3* (a partir da versão 3.3). Este módulo permite tornar o código *python* portátil e sem necessidade de instalar bibliotecas no ambiente *default* do *python*, o que por outro lado permite remover todas as bibliotecas e módulos não nativos do *python* apagando uma pasta. Neste trabalho é utilizado para que não haja a necessidade de instalar os módulos *prettytable* e *progressbar2*;

j. *Prettytable* – módulo não nativo do *Python3*. Este módulo permite de forma simples imprimir tabelas em ficheiros ou no *standard output* e neste trabalho foi utilizado para mostrar os resultados dos testes;

k. *Progressbar2* – módulo não nativo do *Python3*. Este módulo permite de forma simples imprimir uma *progresso bar* na consola para mostrar ao utilizador o estado do processo em execução. Neste trabalho é utilizado para mostrar o estado dos testes para as diferentes versões do livro.

V. Testes de *Performance*

Para análise de *performance* foi então executado o conjunto de testes criados. No caso do tempo gasto e do número de bits utilizado por contador, foi utilizada apenas a média dos testes com 10000 contagens já que nos permite obter uma melhor média dos valores dada o maior número de execuções.

Os resultados encontram-se enão no ficheiro *results.txt* e, portanto, neste relatório apenas serão apresentados as partes mais relevantes desses resultados.

VI. RESULTADOS

Language	Exact Counter	Logarithmic Counter (base = 2)	Probabilistic Counter (probability = 1/64)
french	0.0062	0.0120	0.0099
german	0.0065	0.0101	0.0085
english	0.0049	0.0086	0.0072
italian	0.0076	0.0127	0.0107

Figura 1 – Média do tempo gasto por cantador para cada palavra em cada uma das versões do livro

Language	Exact Counter	Logarithmic Counter (base = 2)	Probabilistic Counter (probability = 1/64)
french	Used: 8.6257 bits, Real: 9 bits	Used: 3.0630 bits, Real: 4 bits	Used: 2.6321 bits, Real: 3 bits
german	Used: 8.5999 bits, Real: 9 bits	Used: 3.0590 bits, Real: 4 bits	Used: 2.6054 bits, Real: 3 bits
english	Used: 8.6330 bits, Real: 9 bits	Used: 3.0632 bits, Real: 4 bits	Used: 2.6395 bits, Real: 3 bits
italian	Used: 8.6582 bits, Real: 9 bits	Used: 3.0670 bits, Real: 4 bits	Used: 2.6518 bits, Real: 3 bits

Figura 2 – Média de *bits* máximos utilizados por cada contador em cada uma das versões do livro

Trials	MaxRE	MinRE	MeanRE	MeanAE	ExpValue	MaxCV	MinCV	MeanCV	MeanAD	MD	SD	Variance	ExpEvents	Events
10	0.3819	0.0002	0.1346	0.2066	1.3701	1.8237	1.1665	1.4833	0.2046	0.4006	1.5167	2.9805	2.9602	2.9427
100	0.3524	0.0000	0.1362	0.2083	1.3701	1.9742	1.0802	1.4863	0.2172	0.5082	1.5227	2.9961	2.9602	2.9710
1000	0.3824	0.0000	0.1359	0.2078	1.3701	2.0572	1.0468	1.4858	0.2188	0.5755	1.5225	2.9936	2.9602	2.9625
10000	0.4049	0.0000	0.1359	0.2079	1.3701	2.1207	1.0302	1.4859	0.2189	0.6357	1.5226	2.9939	2.9602	2.9593

Figura 3 – Média total de cada uma das métricas para o contador aproximado com probabilidade decrescente logarítmica em cada uma das versões do livro

Trials	MaxRE	MinRE	MeanRE	MeanAE	ExpValue	MaxCV	MinCV	MeanCV	MeanAD	MD	SD	Variance	ExpEvents	Events
10	9.0583	0.9956	1.9177	0.0755	0.0463	0.3005	0.0007	0.0463	0.0679	0.2545	0.1106	0.0689	2.9602	2.9662
100	38.0568	0.9955	1.9221	0.0755	0.0463	0.5048	0.0002	0.0464	0.0749	0.9177	0.1771	0.0684	2.9602	2.9670
1000	48.6891	0.9955	1.9166	0.0754	0.0463	1.3037	0.0001	0.0463	0.0754	1.2574	0.1892	0.0680	2.9602	2.9637
10000	54.1604	0.9955	1.9184	0.0754	0.0463	1.5580	0.0000	0.0463	0.0754	1.5117	0.1900	0.0679	2.9602	2.9605

Figura 4 – Média total de cada uma das métricas para o contador aproximado com probabilidade fixa em cada uma das versões do livro

Word	MaxRE	MinRE	MeanRE	MeanAE	ExpValue	MaxCV	MinCV	MeanCV	MeanAD	MD	SD	Variance	ExpEvents	Events
alice	5e+01%	0.0%	8.3%	0.66	8	12	6	8.36	0.72	3.64	8.40	70.62	397	397.07
queen	6.7e+01%	0.0%	1e+01%	0.61	6	10	3	6.04	0.63	3.96	6.11	37.28	78	78.77
time	6.7e+01%	0.0%	1e+01%	0.61	6	10	3	5.91	0.65	4.09	5.97	35.62	71	71.23
king	5e+01%	0.0%	1.1e+01%	0.64	6	9	3	5.72	0.69	3.28	5.79	33.47	63	62.14
turtle	8e+01%	0.0%	1.6e+01%	0.80	5	9	3	5.68	0.70	3.32	5.74	32.97	60	60.28
mock	8e+01%	0.0%	1.5e+01%	0.76	5	9	3	5.59	0.72	3.41	5.65	31.94	57	56.43
hatter	8e+01%	0.0%	1.5e+01%	0.74	5	9	3	5.56	0.72	3.44	5.62	31.63	56	55.41
gryphon	8e+01%	0.0%	1.5e+01%	0.73	5	9	3	5.54	0.72	3.46	5.61	31.42	55	54.67
head	8e+01%	0.0%	1.4e+01%	0.69	5	9	3	5.46	0.72	3.54	5.53	30.54	52	51.74
rabbit	6e+01%	0.0%	1.3e+01%	0.67	5	8	3	5.43	0.72	2.57	5.49	30.16	51	50.39

Figura 5 – *Top-10* com as métricas para o contador aproximado com probabilidade decrescente logarítmica na versão inglesa do livro

Word	MaxRE	MinRE	MeanRE	MeanAE	ExpValue	MaxCV	MinCV	MeanCV	MeanAD	MD	SD	Variance	ExpEvents	Events
alice	1.9e+02%	3.3%	3.2e+01%	1.97	6.20	18	0	6.23	1.98	11.77	6.70	44.89	397	398.79
queen	5.6e+02%	1.8e+01%	7.1e+01%	0.87	1.22	8	0	1.21	0.87	6.79	1.63	2.66	78	77.66
time	5.3e+02%	9.9%	7.2e+01%	0.88	1.11	7	0	1.11	0.80	5.89	1.51	2.29	71	70.78
king	5.1e+02%	1.6%	7.4e+01%	0.73	0.98	6	0	0.99	0.73	5.01	1.39	1.94	63	63.48
turtle	6.5e+02%	6.7%	7.7e+01%	0.72	0.94	7	0	0.93	0.73	6.07	1.33	1.78	60	59.53
mock	6.9e+02%	1.2e+01%	8.2e+01%	0.73	0.89	7	0	0.90	0.73	6.10	1.30	1.68	57	57.45
hatter	7e+02%	1.4e+01%	8.4e+01%	0.73	0.88	7	0	0.88	0.73	6.12	1.28	1.64	56	56.04
gryphon	6e+02%	1.6e+01%	8.4e+01%	0.72	0.86	6	0	0.86	0.72	5.14	1.26	1.60	55	55.32
head	7.6e+02%	2.3e+01%	8.8e+01%	0.71	0.81	7	0	0.82	0.71	6.18	1.22	1.48	52	52.74
rabbit	6.5e+02%	2.5e+01%	9e+01%	0.72	0.80	6	0	0.81	0.72	5.19	1.20	1.44	51	51.57

Figura 6 – *Top-10* com as métricas para o contador aproximado com probabilidade fixa na versão inglesa do livro

A. ANÁLISE DOS RESULTADOS

Olhando pros resultados pudemos verificar alguns factos, entre eles, o facto de que em velocidade de execução o contador exato é o mais rápido. No entanto, ocupa uma quantidade de *bits* por contagem consideralmente mais elevada em relação aos outros dois contadores, e esta tendência apenas tende a acentuar à medida que se vai aumentando o volume de dados.

Levando isto em conta, puderá ser uma boa solução utilizar um dos outros contadores caso se tenha um volume muito elevado de dados. Olhando para os dados, o contador probabilístico parece à primeira vista o mais indicado já que tem uma menor média de tempo gasto e uma menor quantidade de *bits* gastos. No entanto o segundo dado induz em erro, pois se olharmos para o *top-10* das palavras mais encontradas numa das versões do livro (figuras 5 e 6), verificamos que o número de *bits* do contador probabilístico começa muito baixo e vai aumentado à medidade que se vai para uma palavra com um maior número de ocorrências. Por outro lado, o contador logarítmico tem um número de *bits* já elevado nas palavras com menor ocorrência desse *top-10*, mas vai aumentando muito devagar, estando quase que estagnado, o que indica que eventualmente, o logarítmico para volumes de dados enormes deverá ocupar menos espaço do que o probabilístico (estes dados são verificados também nas outras versões). Sendo assim, é preciso levar-se então em conta outros fatores para decidir qual dos dois utilizar, principalmente o nível de exatidão dos resultados. Se a exatidão não for um fator determinante, e o volume de dados for enorme, então sem dúvida o logarítmico é o contador ideal. No entanto, se a precisão for importante, a partir de certo ponto, puderá ser melhor utilizar o probabilístico, ou, eventualmente, o exato. (De lembrar que para uma melhor exatidão, tanto para o logarítmico como para o probabilístico é necessário realizar várias contagens e calcular uma média, o que significa que quanto maior a exatidão necessária, maior será o tempo de execução, sendo que dos dois aproximados o logarítmico deverá ser o que demora mais, daí as afirmações anteriores).

Outro facto que pudemos retirar dos resultados é o de que apenas 5 palavras se mantêm no *top-10* em todas as versões:

1. Alice – que mantém a sua posição em todas
2. Raínha – que apenas muda de posição na versão italiana
3. Rei – muda de posição
4. Tartaruga – muda de posição
5. Chapeleiro – muda de posição

as outras palavras não se mantém sendo que com a exceção do ingês as outras versões têm verbos no *top-10*.

VII. CONCLUSÃO

Com este trabalho pudemos concluir que cada contador tem a sua utilidade sendo que o mais indicado para quando temos elevados volumes de dados e a exatidão é pouco importante é o contador aproximado com probabilidade decrescente logarítmica.

Por outro lado, pudemos verificar que as linguagens têm algumas diferenças entre si, mas também têm semelhanças, isto porque quando se verifica o *top-10* para o mesmo livro nas diferentes linguagens, têm algumas que se mantêm em todas as linguagens, parte delas até ocupam a mesma posição, e outras que são diferentes.