INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

RELATÓRIO DE TRABALHO PRÁTICO I

Processamento de Linguagem Natural com Estrutura de Dados

LUCAS BRAGA MENDONÇA

NELSON CUNHA

ALUNOS Nº 17870 E 19241

Estrutura de Dados II

Licenciatura em Engenharia de Sistemas Informáticos

Barcelos, Abril de 2020

Índice

Lista de Figuras

1 Introdução

O Objetivo deste trabalho é utilizar as Estruturas de Dados na Linguagem de Programação C, de forma que as perguntas do Trabalho Prático I sejam respondidas. Desta forma, houve o desenvolvimento de funções que apresentam estruturas de dados aninhadas, eficientes com código modular e estruturado com a finalidade de não somente obter o resultado das questões abordas, mas também trazer clareza, eficácia e alta performance no programa desenvolvido.

Por sua vez, o programa é desenvolvido com base em listas dinâmicas duplamente e simplesmente encadeadas, assim como também utiliza-se de estruturas aninhadas para ligar diferentes listas. A busca binária é utilizada no programa, facilitando a inserção de dados ordenados nas listas quando necessário. O presente relatório tem como foco fudamental apresentar as informações pertinentes com relação a toda arquitetura da lógica de processamento dos dados.

Ainda, tem o propósito de descrever e tratar informações relativas ao Processamento de Linguagem Natural, uma sub-área da Inteligência Artificial, usando funções estatísticas, onde o dado é coletado, organizado, descrito, cálculado e interpretado de forma a apoiar os resultados e probabilidades em estudo.

É importante ressaltar que as figuras e imagens que descrevem os dados presente neste relatório são baseadas somente em parte do ficheiro disponibilizado para análise e não no ficheiro em sua totalidade, ou seja, com um número reduzido de linhas. Entretanto, a lógica se aplica tanto a um ficheiro que disponha de dezenas de dados, quanto um ficheiro que disponha de milhares. Testes poderão ser feitos com ficheiros em grande quantidade de dados, embora, obviamente, a performance não seja a mesma.

Todo código desenvolvido, assim como o presente relatório encontra-se disponível no GitHub, através do link https://github.com/lucasbmendonca/ED2TP1.

2 Desenvolvimento dos processamento de dados

2.1 Fonte de Dados

A fonte de dados do programa é gerada através de um ficheiro .TXT etiquetado com informações morfossintáticas. Essas informações morfossintáticas são passadas para a função denominada *insere()*, que é responsável por inserir as informações do ficheiro em uma lista duplamente encadeada disposta da seguinte forma:

```c
/*Estrutura de Lista de Categoria de Palavras de acordo com sua classificação.*/
typedef struct categoria{
    char texto[10]; //Análise morfossintática.
    int qtd_palavras; //Quantidade de palavras que a categoria possui.
    Palavra *palavra; //Inicio da lista de palavras (Lista de Lista).
    struct categoria *proximo; //Proxima categoria.
    struct categoria *anterior; //Categoria anterior.
} Categoria;
```

Figura 1: Estrutura de uma lista de categorias gramaticais

```c
/*Estrutura da Lista de Palavra.*/
typedef struct palavra{
    char texto[100]; //Palavra da frase original.
    char raiz[100]; //Raiz da palavra.
    double percentagem; //Certeza da ferramenta em relacao a análise realizada.
    int quantidade; //Quantas vezes a palavra se repete.
    int tamanho; //Tamanho da palavra.
    struct palavra *proximo; //Próxima palavra.
    struct palavra *anterior; //Palavra anterior.
} Palavra;
```

Figura 2: Estrutura de uma lista de palavras

Como pode ser visto na figura 1, a lista principal é a de categorias, estruturada como uma lista duplamente encadeada e que possui um campo que aponta para uma lista de palavras associadas. A figura 2 representa a lista duplamente encadeada de palavras, nas quais estas palavras obrigatoriamente estão associadas a uma categoria. Logo, a estrutura que guarda a informação dos dados do .TXT é uma lista de lista: as categorias possuem listas de palavras.

Dentro da função *insere()* estão dispostas funções que auxiliam na inserção destes dados de forma ordenada. Inicialmente, a ordenação é feita com base na **ordem alfabética das categorias** e em seguida na **ordem alfabética das palavras associadas**. As funções associadas podem ser vistas a seguir:

1. *buscaBinariaCategoria()*

   Função que dispõe de uma lógica com a finalidade de realizar uma busca binária na lista de categorias tendo como parâmetro a nova inserção de categoria a ser feita.

   Dentro desta função, há uma outra função auxiliar *acharMeioCategoria()* que retorna o elemento que está no meio da lista. Este elemento é fundamental na lógica de busca binária.

2. insereCategoria()

   Função que insere um elemento do tipo Categoria na lista de categorias de forma ordenada.

3. *buscaBinariaPalavra()*

   Função que realiza a busca binária na lista de palavras, tal como na busca binária de categorias. Ainda, apresenta a função auxiliar *acharMeioPalavra()*.

4. *inserePalavra()*

   Função que insere um elemento do tipo Palavra na lista de palavras de uma determinada categoria.

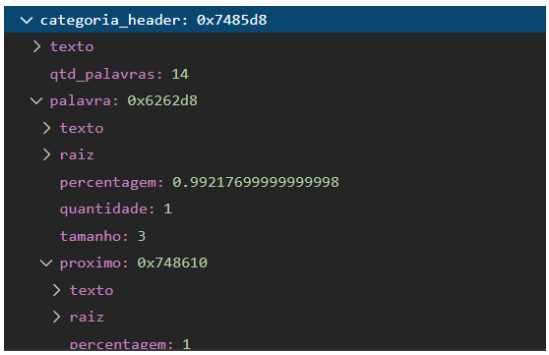

Figura 3: Estrutura de dados aninhada

2.2 Calculando frequencias e demais medidas

As estrutura feita para processar este e qualquer outro dado que necessite de análise com relação a sua frequencia pode ser vista a seguir:

```c
/*Estrutura para Lista de Frequencias.*/
typedef struct frequencia{
    char variavel[100]; //Variável em estudo: x.
    int freq_abs; //Frequência absoluta: ni.
    float freq_rel; //Frequencia relativa: fi.
    int freq_abs_acumulada; //Frequencia absoluta acumulada: Ni.
    float freq_rel_acumulada; //Frequencia relativa acumulada: Fi.
    struct frequencia *proximo; //Proxima frequencia
} Frequencia;
```

Figura 4: Estrutura para listas de frequências

A figura 4 ilustra a forma como a lista de frequencias, dada uma variável X de estudo, deve ser estruturada. Dentre as questões abordadas no trabalho prático, a construção de uma tabela de frequencias absolutas, relativas e acumuladas dado a categoria gramatical usada e outra tabela de frequencias dado o tamanho das palavras existentes, utilizam-se destra estrutura para guardar as informações.

Toda a leitura das categorias e suas palavras associadas é feita dentro da função *calcFreqMed()* de forma que toda a função seja usada para estruturas a maioria dos dados das questões abordas sem a necessidade de realizar mais iterações desnecessárias em outras funções, compremententendo a performance do mesmo.

Tabela de Frequencias na quantidade de palavras de uma categoria:

X	ni	fi	Ni	Fi
CC	14	3.97	14	3.97
DT	41	11.61	55	15.58
EX	1	0.28	56	15.86
IN	46	13.03	102	28.90
JJ	33	9.35	135	38.24
JJS	1	0.28	136	38.53
MD	1	0.28	137	38.81
NN	70	19.83	207	58.64
NNS	19	5.38	226	64.02
PRP	23	6.52	249	70.54
PRP$	2	0.57	251	71.10
RB	20	5.67	271	76.77
RBR	1	0.28	272	77.05
RP	2	0.57	274	77.62
TO	9	2.55	283	80.17
VB	9	2.55	292	82.72
VBD	10	2.83	302	85.55
VBG	11	3.12	313	88.67
VBN	6	1.70	319	90.37
VBP	7	1.98	326	92.35
VBZ	15	4.25	341	96.60
WDT	3	0.85	344	97.45
WP	3	0.85	347	98.30
WP$	1	0.28	348	98.58
WRB	5	1.42	353	100.00
Zu	1	0.28	354	100.28

Figura 5: Tabela de frequências na quantidade de palavras de uma categoria

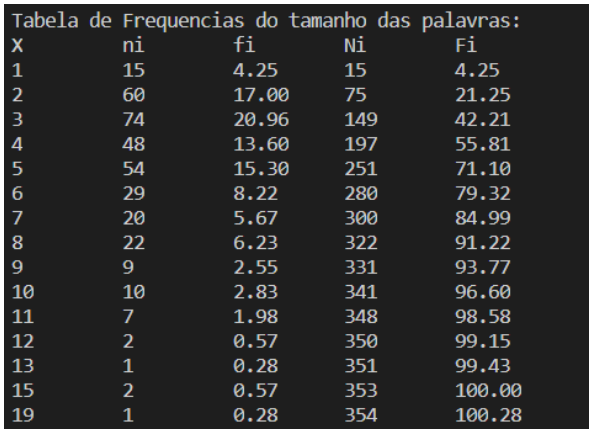

Tabela de Frequencias do tamanho das palavras:

X	ni	fi	Ni	Fi
1	15	4.25	15	4.25
2	60	17.00	75	21.25
3	74	20.96	149	42.21
4	48	13.60	197	55.81
5	54	15.30	251	71.10
6	29	8.22	280	79.32
7	20	5.67	300	84.99
8	22	6.23	322	91.22
9	9	2.55	331	93.77
10	10	2.83	341	96.60
11	7	1.98	348	98.58
12	2	0.57	350	99.15
13	1	0.28	351	99.43
15	2	0.57	353	100.00
19	1	0.28	354	100.28

Figura 6: Tabela de frequências do tamanho das palavras

Significado em estatística das colunas:

a) X: variável em estudo

b) ni: frequência absoluta

c) fi: frequência relativa

d) Ni: frequência absoluta acumulada

e) Fi: frequência relativa acumulada

Para a tabela das médias e devios padrões com base na medidas de certeza de etiquetação, foi utilizada uma lista simplesmente encadeada:

```c
/*Lista de Média Aritmetica e Desvio Padrão das Categorias de Palavras com base na certeza de etiquetação.*/
typedef struct medDesvCat{
    char categoria[10]; //Categoria de uma palavra.
    double media_aritmetica; //Media aritmética com base na medida de etiquetação da categoria.
    double desvio_padrao; //Desvio padrão com base na medida de etiquetação da categoria.
    struct medDesvCat *proximo; //Proxima informação de medidas.
} MedDesvCat;
```

Figura 7: Lista das médias e desvios padrões com base na certeza de etiquetação

Nesta lista estão dispostas as informações necessárias para a construção da tabela associadas a uma categoria. O resultado pode ser visto na figura 8.

```
Tabela de Media e Desvio Padrao da categoria com base na etiquetacao:
Categoria           Media Artimetica    Desvio Padrao
CC                  0.99944             0.00266
DT                  0.94691             0.24318
EX                  0.85038             0.00000
IN                  0.91421             0.33514
JJ                  0.91182             0.18497
JJS                 0.98701             0.00000
MD                  1.00000             0.00000
NN                  0.87573             0.24633
NNS                 0.98231             0.03594
PRP                 1.00000             0.00000
PRP$                0.88464             0.16314
RB                  0.89247             0.26368
RBR                 0.99474             0.00000
RP                  0.53992             0.03434
TO                  0.99790             0.00000
VB                  0.55867             0.33795
VBD                 0.90150             0.23358
VBG                 0.98131             0.05092
VBN                 0.58838             0.26094
VBP                 0.70528             0.35794
VBZ                 0.82764             0.40835
WDT                 0.75826             0.41634
WP                  1.00000             0.00000
WP$                 0.99937             0.00000
WRB                 0.99977             0.00010
Zu                  1.00000             0.00000
```

Figura 8: Output tabela de média e desvio padrão

Para o cálculo das medidas de dispersão e localização com base no tamanho das palavras, a lista principal de categorias e palavras foi utilizada de forma que preenchesse os seguintes campos do programa:

```c
/*Medidas de localização e dispersao, relativas ao tamanho das palavras: media
aritmetica, mediana, moda e desvio padrao.*/
int totalTamanhoPalavras = 0; //Quantos diferentes tamanhos de palavras existem.
double ma_tam_palavra = 0; //Média Aritmética dos tamanhos de palavras.
double mediana_tam_palavra = 0; //Mediana dos tamanhos de palavras.
int moda_tam_palavra = 0; //Moda dos tamanhos de palavras.
double desvio_padrao_tam_palavra = 0; //Desvio padrão dos tamanhos de palavras.
int somaFreqPalavras = 0; //Soma da quantidade de Frequencia das palavras.
```

Figura 9: medidas de dispersão e localização

O cálculo, tal como o processamento dos dados já apresentados nesse relatório, é feito na função *calcFreqMed()* com a finalidade de reaproveitamento de iterações. Para todos esses cálculos, as fórmulas matemáticas da estátistica descritiva são transportadas e transformadas em lógica de processamento na linguagem C. Um exemplo de cálculo de uma dessas variáveis pode ser visto nas figuras seguintes.

```
/*INI - COD-003*/
media = (somatorioCertezaFrequencia / categoria->qtd_palavras);
med_desv_cat->media_aritmetica = media;
double variancia = 0;
palavra = categoria->palavra;
while (palavra != NULL)
{
    variancia = variancia + pow(((palavra->percentagem - media) * palavra->quantidade), 2);
    palavra = palavra->proximo;
}
if (categoria->qtd_palavras - 1 > 0)
{
    variancia = variancia / (categoria->qtd_palavras - 1);
    med_desv_cat->desvio_padrao = sqrt(variancia);
}
else
{
    med_desv_cat->desvio_padrao = 0;
}
insereMedDesv(&med_desv_cat);
/*FIM - COD-003*/
```

Figura 10: variância e desvio padrão

```
Medidas de localizacao e dispersao relativas ao tamanho das palavras:
Media Aritmetica    Moda                Mediana             Desvio Padrao
4.68272             3                   9.00000             13.26883
```

Figura 11: output medidas de dispersão e localização

a) Fórmula da variância:

$$s^2 = \frac{\sum_{i=1}^{n} (X_i - \overline{X})^2}{n-1}$$

b) Fórmula do desvio padão:

$$s = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n-1}}$$

Para calcular os quartis com base nas frequências no número de ocorrência das palavras, foram criados os seguintes campos:

Float quartil_1;

Float quartil_2;

Float quartil_3;

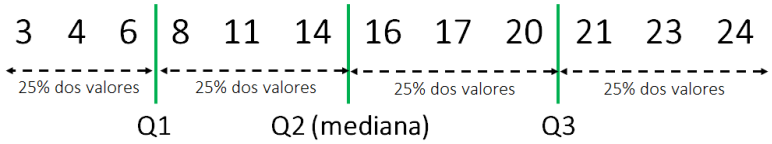

Figura 12: exemplo de cálculo de quartis

Para o preenchimento dessas variáveis, a função *calculaQuartil()* foi utilizada. O resultado pode ser visto a seguir:

```
me                  4
first               4
that                4
on                  4
about               4
was                 5
is                  6
in                  6
to                  9
and                 11
a                   12
of                  14
the                 18
```

Figura 13: parte da tabela de palavras e suas frequencias

```
Quartil 1: 57.00 | Quartil 2: 114.00 | Quartil 3: 172.00
```

Figura 14: valores de quartil

Foi desenvolvida ainda uma função que retorna o quartil dado uma palavra qualquer como parâmetro de importação e que esteja presente no rol de dados: *getQuartil(palavra)*.

```
Digite uma palavra para busca de quartil:the

Palavra the esta no Quartil 4. Posicao 228 no Rol.
```

Figura 15: busca de quartil segundo palavra dada pelo usuário

Para a construção do histograma foi usada a função *calcHistograma()* e o resultado pode ser visto a seguir:

```
Histograma:

Classes                 Frequencia abs.     Ponto Medio
4.58        9.50        4                   7.04
10.14       19.00       6                   14.57
20.29       27.75       5                   24.02
30.43       39.66       5                   35.05
40.57       50.00       5                   45.29
50.71       59.32       9                   55.02
60.86       68.17       9                   64.51
71.00       80.58       8                   75.79
81.14       90.17       17                  85.65
91.28       100.00      160                 95.64
```

Figura 16: busca de quartil segundo palavra dada pelo usuário

3 Conclusão

Conclui-se que com a abordagem utilizada na construção deste programa, o presente aluno seja capaz de estruturar e encontrar soluções para novos problemas encontrados no mundo real. Uma das características mais importantes de uma lista encadeada é seu caráter dinâmico, que permite armazenar um número de elementos limitado apenas pela memória disponível. E face as questões apresentadas, os dados foram devidamente estruturados seguinte os padrões de construção destas listas de forma a facilitar o cálculo de todas as medidas estatísticas em questão.

Mas informações e processos podem ser encontradas no código desenvolvido, como por exemplo em funções auxiliares de processamento, retorno e destruição das listas utilizadas, e, não obstante, descrições mais detalhadas sobre cada processo também estão dispostas ao longo do código.