

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

[1 Introdução 1](#_Toc38837552)

[2 Desenvolvimento dos processamento de dados 3](#_Toc38837553)

[2.1 Fonte de Dados 3](#_Toc38837554)

[2.2 Calculando frequencias e demais medidas 5](#_Toc38837555)

[3 Conclusão 10](#_Toc38837556)

Lista de Figuras

Figura 1: Estrutura de uma lista de categorias gramaticais 3

Figura 2: Estrutura de uma lista de palavras 3

Figura 3: Estrutura de dados aninhada 4

Figura 4: Estrutura para listas de frequências 5

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

Figura 6: Tabela de frequências do tamanho das palavras 6

Figura 7: Lista das médias e desvios padrões com base na certeza de etiquetação 6

Figura 8: Output tabela de média e desvio padrão 7

Figura 9: medidas de dispersão e localização 7

Figura 10: variância e desvio padrão 8

Figura 11: output medidas de dispersão e localização 8

Figura 12: exemplo de cálculo de quartis 9

Figura 13: parte da tabela de palavras e suas frequencias 9

Figura 14: valores de quartil 9

Figura 15: busca de quartil segundo palavra dada pelo usuário 9

Figura 16: busca de quartil segundo palavra dada pelo usuário 9

# 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 tambén 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>.

# Desenvolvimento dos processamento de dados

## 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:

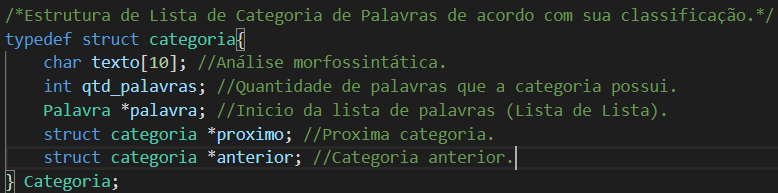


Figura 1: Estrutura de uma lista de categorias gramaticais

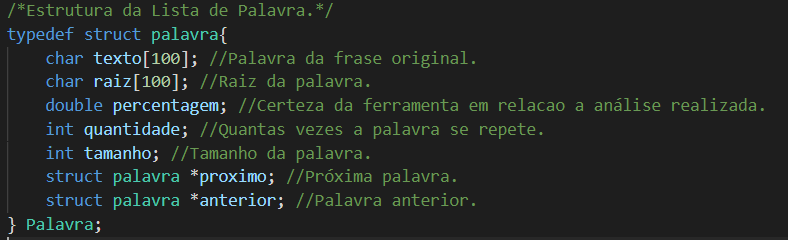


Figura 2: Estrutura de uma lista de palavras

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.

1. insereCategoria()

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

1. *buscaBinariaPalavra()*

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

1. *inserePalavra()*

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

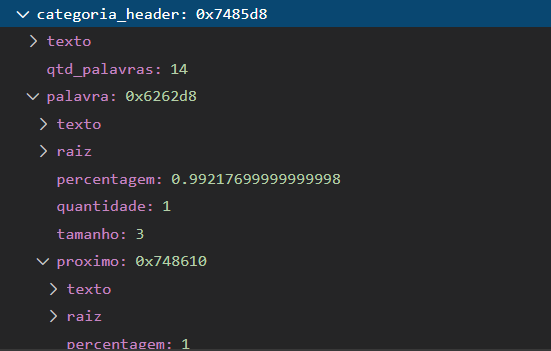


Figura 3: Estrutura de dados aninhada

## 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:

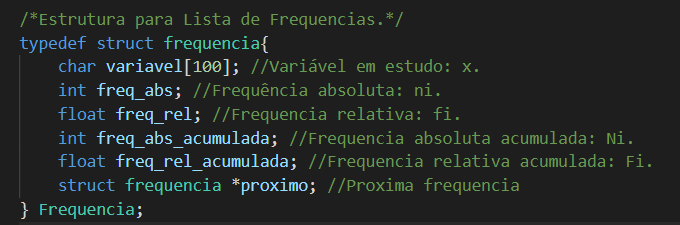


Figura 4: Estrutura para listas de frequências

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, comprementendo a performance do mesmo.

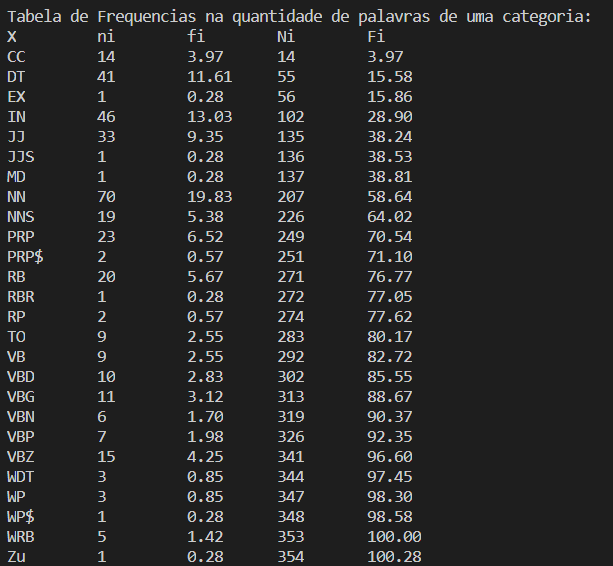


Figura 5: Tabela de frequências na quantidade de palavras de uma categoria

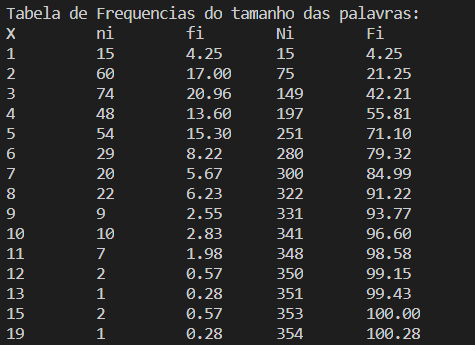


Figura 6: Tabela de frequências do tamanho das palavras

Significado em estatística das colunas:

1. X: variável em estudo
2. ni: frequência absoluta
3. fi: frequência relativa
4. Ni: frequência absoluta acumulada
5. 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:

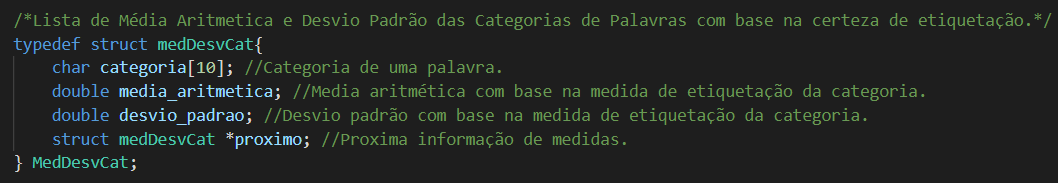


Figura 7: Lista das médias e desvios padrões com base na certeza de etiquetação

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

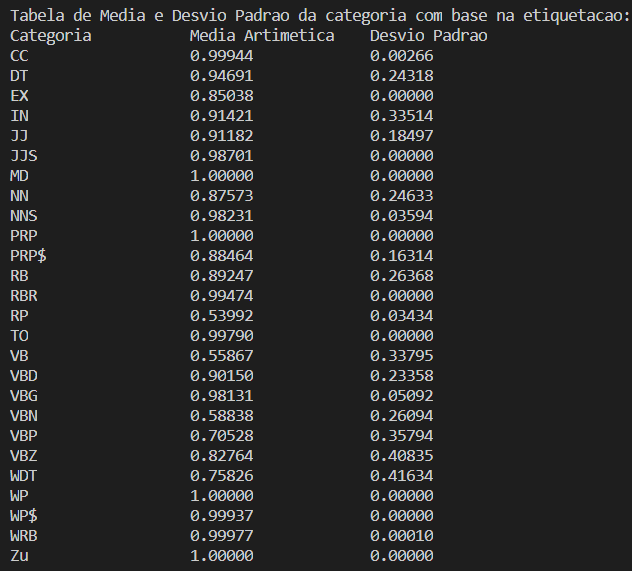


Figura 8: Output tabela de média e desvio padrão

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:

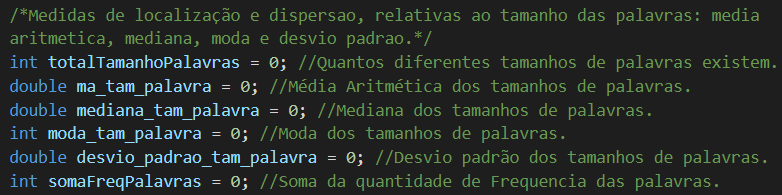


Figura 9: medidas de dispersão e localização

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.

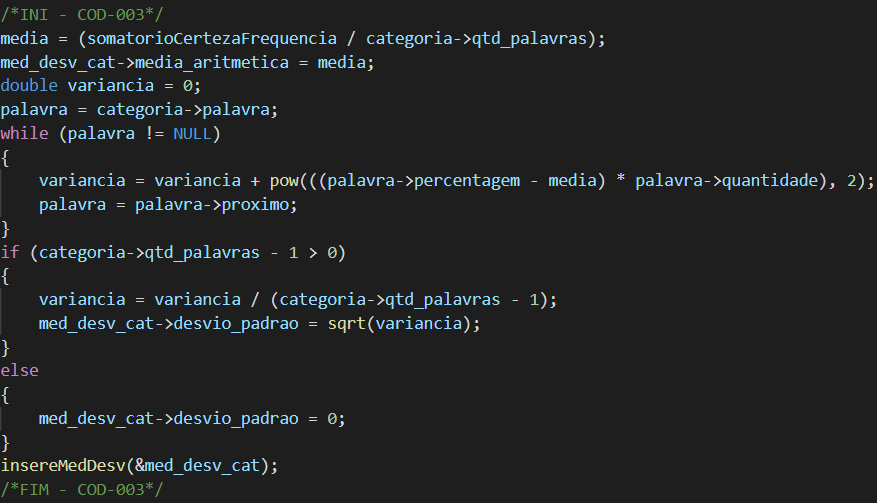


Figura 10: variância e desvio padrão

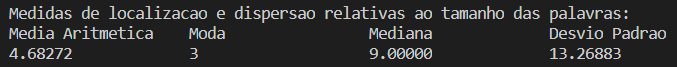
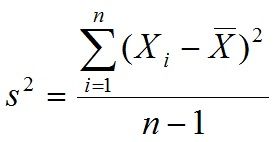
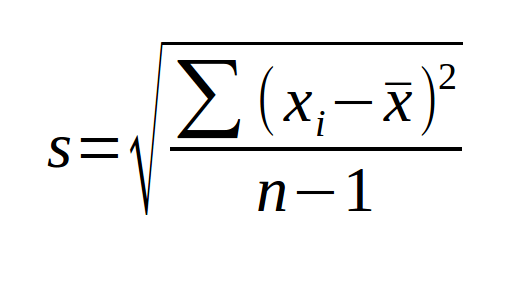


Figura 11: output medidas de dispersão e localização

1. Fórmula da variância:



1. Fórmula do desvio padão:



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;

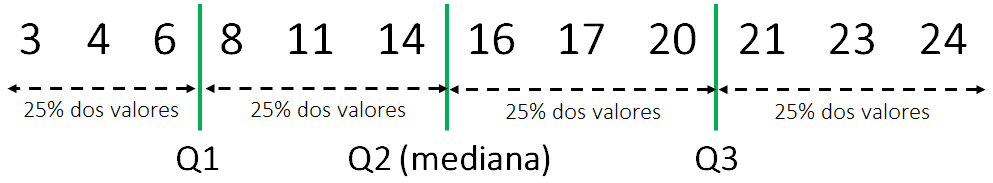


Figura 12: exemplo de cálculo de quartis

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

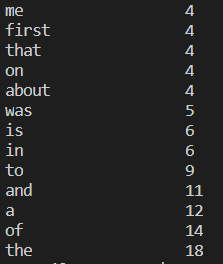


Figura 13: parte da tabela de palavras e suas frequencias



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)*.



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:

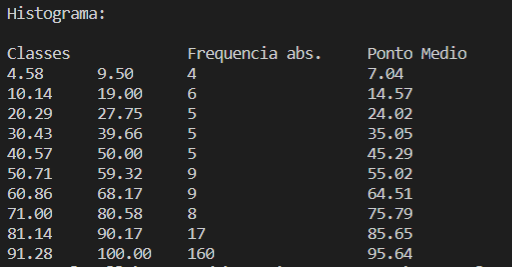


Figura 16: busca de quartil segundo palavra dada pelo usuário

# 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.