UNIVERSIDADE FEDERAL DO PIAUÍ - CAMPUS SHNB

SISTEMAS DE INFORMAÇÃO

ALGORITMOS E PROGRAMAÇÃO II

Gabriel Queiroz Silva

James Sousa

Michael Souza Fernandes

Trabalho Prático

1 Introdução

Com o crescimento exponencial do volume de informações disponíveis em diversos formatos, torna-se cada vez mais difícil para os usuários processarem e assimilarem todo o conteúdo disponível. Diante desse cenário, a utilização de ferramentas capazes de extrair as informações mais relevantes de um texto e apresentá-las de forma resumida e clara é de extrema importância.

Nesse sentido, o desenvolvimento de um sumarizador de texto pode se tornar uma solução viável e eficiente para esse problema. Um sumarizador de texto é uma ferramenta que permite resumir informações em textos longos, tornando a compreensão geral do conteúdo mais acessível e menos demandante de tempo e esforço.

Para esse projeto em específico, o sumarizador de texto foi desenvolvido em um programa em utilizando a linguagem C. A proposta é que o programa analise automaticamente o conteúdo de um texto e, a partir disso, gera um resumo que contenha as informações mais importantes de forma clara e concisa.

Dessa forma, o objetivo principal do projeto é desenvolver uma ferramenta de sumarização de texto eficiente, capaz de otimizar a compreensão e assimilação de conteúdo para os usuários, de maneira rápida e eficaz. O resultado final é um programa em C que oferece aos usuários a oportunidade de processar grandes volumes de informações de maneira simplificada e prática.

2 Solução Proposta

O progama em questão tem o objetivo de servir como um sumarizador de texto, lendo um texto especifico dado antes do inicio do progama e partir da quantidade de palvras chaves alvo que são fornecidas pelo usuario busca gerar um rank das palavras mais ditas.

Ele lê um arquivo de texto e conta a frequência de cada palavra que não é uma stopword (palavras comuns e não significativas como "o", "a", "e", "de", etc.) fornecidas também por um arquivo e, em seguida, grava as palavras mais frequentes em outro arquivo de texto.

O programa começa verificando se foram fornecidos os argumentos corretos na linha de comando. Isso é feito usando uma função chamada "main" que recebe os argumentos da linha de comando e verifica se todos eles estão corretos.

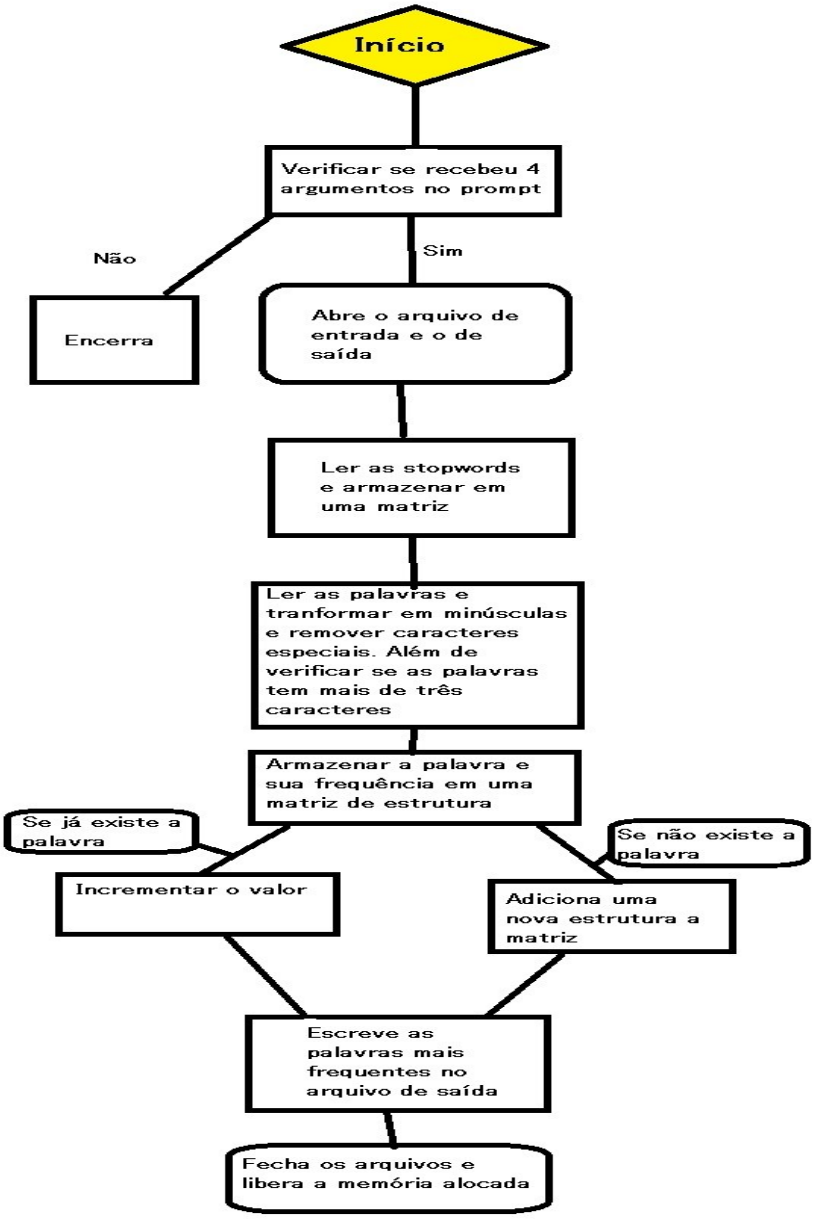
Em seguida, o programa abre o arquivo de entrada e o arquivo de saída e lê as stopwords do arquivo especificado. Isso é feito usando funções como "fopen", "fread" e "fclose", que permitem ao programa abrir e ler arquivos.

O arquivo de entrada é lido palavra por palavra, verificando se cada palavra é uma stopword e, se não for, adicionando-a a uma lista de frequências de palavras. Isso é feito usando um loop "while" que lê cada palavra do arquivo e verifica se ela é uma stopword. Se não for, a palavra é adicionada a uma lista usando uma estrutura de dados chamada "array" que armazena cada palavra e sua frequência.

Se a palavra já está na lista, sua frequência é incrementada. Caso contrário, a palavra é adicionada à lista com uma frequência de 1. Isso é feito usando uma estrutura de controle "if-else" que verifica se a palavra já está na lista ou não.

Quando todo o arquivo de entrada é lido, a lista de frequências é ordenada em ordem decrescente de frequência e as palavras mais frequentes (o número especificado pelo usuário) são gravadas no arquivo de saída. Isso é feito usando uma função chamada "qsort" que ordena a lista de palavras em ordem decrescente de frequência e uma estrutura de controle "for" que escreve as palavras mais frequentes no arquivo de saída.

2.1 fluxograma do promaga feito:



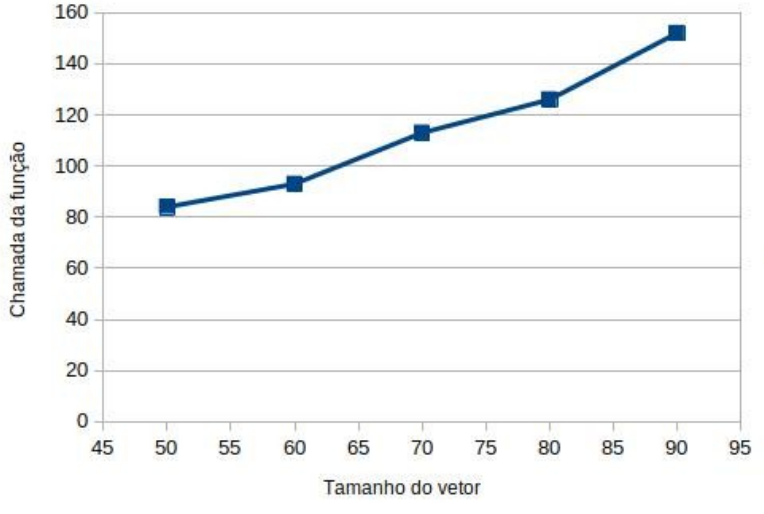
2.2 Recursos usados

Para que o programa funcionasse, foram utilizados recursos como structs para salvar as palavras, sua frequência e uma contagem, juntamente com três funções int externas que serviram para ler as stopwords, verificar se as palavras são stopwords e comparar suas frequências. Além disso, a biblioteca string.h foi utilizada para usar suas funções na comparação do programa e na leitura. No entanto, o mais importante foi o conteúdo relacionado a arquivos ministrado, que foi utilizado em diversas partes do programa para que ocorresse o seu devido funcionamento.

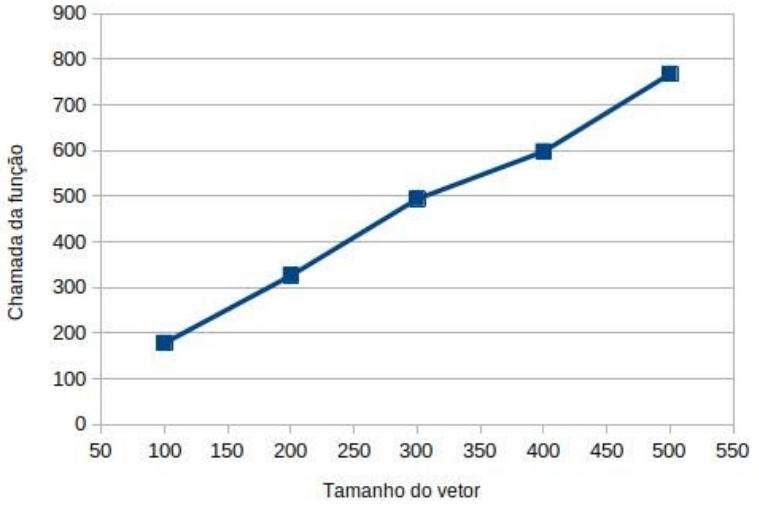
3 Análise de Desempenho

A análise de desempenho de um algoritmo é uma etapa fundamental no processo de desenvolvimento de software. Ela permite avaliar o desempenho de um algoritmo em termos de tempo de execução e uso de recursos, como memória e processamento. Através da análise de desempenho, é possível identificar gargalos no algoritmo e buscar maneiras de otimizá-lo, tornando-o mais eficiente e reduzindo o tempo de execução e o uso de recursos. Nessa seção, vamos avaliar o desempenho do algoritmo de sumarização de texto em relação ao texto de entrada do usuário. Para isso, vamos analisar o número de vezes que a função de comparação de frequência é chamada ao longo do processo de execução do algoritmo em diferentes cenários de entrada, para verificar como o desempenho é afetado pela complexidade do problema fornecido pelo usuário. Para isso escolhemos três cenário classificados como melhor, médio e pior caso lembrando que a quantidade de palavras escolhida pelo usuário para serem rankeadas como as mais frequêntes não interfere na chamada da função os resultados são mostrados a seguir.

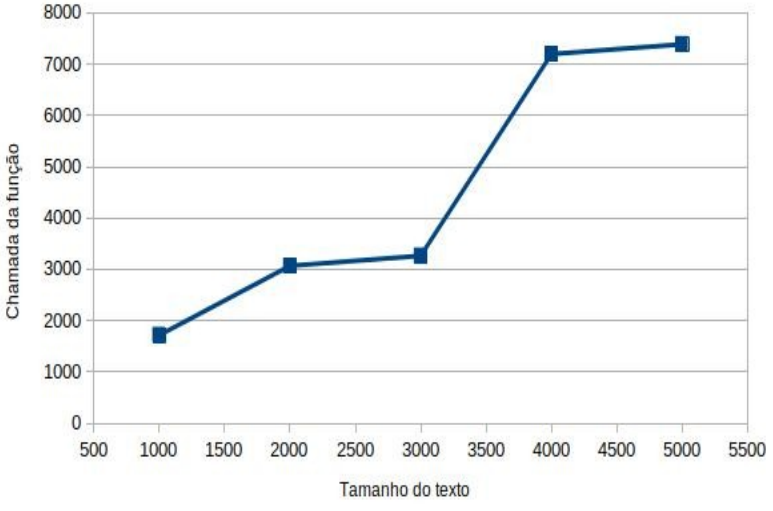
No melhor caso consideramos um texto de entrada de 50 a 90 palavras



No médio caso consideramos um texto de 100 a 500 palavras



No pior caso consideramos um texto de entrada de 1000 a 5000 palavras



A função de comparação de frequencia é usada para ordenar um array de estruturas PalavraFreq em ordem decrescente com base em sua frequência de ocorrência. A função recebe dois ponteiros void como argumentos e os converte em ponteiros PalavraFreq. Em seguida, ela compara as contagens de frequência de cada estrutura e retorna um valor negativo, positivo ou zero, dependendo da relação de ordem entre os elementos como mostrado pelos graficos vemos que quanto maior for o texto de entrada maior será a

quantidade de vezes que essa função é chamada pelo programa e conseguentemente maior será o consumo de recursos como memória e processamento.

4 Conclusão

A sumarização de textos é uma técnica que consiste em extrair informações relevantes e resumidas de um texto original, de forma a apresentar ao leitor as ideias mais importantes de maneira concisa e objetiva.
Para resolver esse problema desenvolvemos um programa em linguagem c que sumariza um texto dado pelo usuário ignorando as stopwords e mostrando as palavras mais frequentes do texto de forma decrecente e informando a quantidade de vezes que ela apareceu. Os aspectos positivos do programa implementado são respectivamente:
Eficiência: O algoritmo é capaz de resumir um certo volume de texto em um tempo relativamente curto, o que pode economizar tempo para os usuários que precisam de informações rápidas.
Precisão: O algoritmo é capaz de capturar as principais ideias e informações importantes do texto, o que pode ajudar os usuários a entenderem o conteúdo de forma mais clara e concisa.
Personalização: Dependendo das preferências do usuário, o algoritmo pode ser personalizado para fornecer um resumo com mais ou menos palavras.
Os aspectos negativos e que podem ser mudado em no futuro é a leitura das stopwords a função que faz a leitura assume que o arquivo de stopwords está formatado de uma maneira específica, com cada palavra em uma linha separada, sem caracteres extras. Se o arquivo tiver algum formato diferente, o programa pode não funcionar corretamente.. Caso elas fossem separadas por espaço e vírgula no arquivo armazenado, algumas não seriam lidas corretamente. No geral foi um desafio, exploramos muitas coisas novas e aprendemos, mesmo que levemente a utilizar funções novas e maneiras novas de se progamar.