UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL INSTITUTO DE INFORMÁTICA INF 01203 – Estruturas de Dados

Trabalho Final: Similaridade Textual

Nome: Henrique Lorentz Trein — 00341853

1. Introdução:

1.1 Situação Problema

O objetivo do trabalho se resume na comparação de dois arquivos textos, ignorando certas palavras que se encontram em outro arquivo do tipo texto constituído, principalmente, por artigos, preposições e algumas outras palavras selecionadas. Além disso, o código-solução deve retornar ao usuário, por meio da linha de comando, o coeficiente de Jaccard. Esse cálculo computa a taxa de palavras compartilhadas entre dois textos. O escore resultante estará entre 0 e 1, sendo 1 é a similaridade máxima e 0 indica que os textos não têm nenhuma palavra em comum. Além disso, o programa deve retornar o tempo de execução e lidar com algumas restrições que foram pré-determinadas no enunciado do trabalho.

1.2 Lógica Geral

Antes da construção do código em C no CodeBlocks, foi necessário elaborar uma lógica que envolvesse a solução do problema em questão (1.1) e também, os pré-requisitos da proposta do trabalho, em especial a utilização de pelo menos uma estrutura de dados do tipo árvore (AVL, Splay ou Rubro-Negra). Com isso em mente, foi planejado, em um primeiro momento, que seriam designadas árvores para a fragmentação dos textos e para a lista de “stopwords”, que também era um arquivo do tipo texto. Em ordem, para melhor clareza, do que foi arquitetado: 1) Procedimentos padrões, verificações de sucesso da abertura dos arquivos e da devida coerência na passagem de parâmetros do usuário. 2) Início do contador de tempo de execução do programa. 3) Criação da árvore para as “stopwords”. 4) Criação da árvore para o primeiro arquivo texto, mas já ignorando as palavras contidas na árvore de “stopwords”. 5) Criação da terceira árvore para o segundo arquivo texto, também já ignorando os nodos da árvore de “stopwords”. 6) Cálculo da união dos textos. 7) Cálculo da intersecção dos textos. 8) Cálculo do coeficiente de Jaccard. 9) Fim do contador de tempo e do programa.

2. Código:

2.1 Primeira Implementação

Para a construção inicial do código, foram designadas árvores do tipo AVL para as três estruturas de dados previstas pelos passos 3, 4 e 5 da lógica geral (1.2).

Motivações: estruturas do tipo AVL são bem balanceadas, otimizando tempo de consulta e inserção, e por essa razão, conseguem armazenar inúmeros

elementos na árvore. Além disso, por manterem a altura da árvore sempre equilibrada, possuem uma caminhamento descendente mais curto e consequentemente uma busca mais eficiente. Árvores AVL permitem a organização dos dados ordenadamente, o que é útil para a comparação de palavras em ordem alfabética, facilitando uma comparação mais precisa e eficiente das palavras compartilhadas entre os textos.

Porque não árvores Splay ou Rubro-Negras: as árvores Splay possuem muitas rotações para cada operação feita, e no caso de busca de palavras, colocaria as mais visitadas próximas à raiz da árvore, o que poderia ser um benefício, mas acaba por não fazer diferença em nossa implementação e até pioraria o desempenho. Isso se deve ao fato de que as palavras na árvore seriam melhor organizadas lexicograficamente e não há fatores de frequência envolvidos e as árvores Splay iriam alterar a ordem a cada operação. Quanto às Rubro-Negras, em um primeiro momento não foram consideradas pelo motivo de que são mais utilizadas quando há mais inserções e remoções, visto que possui um balanceamento menos rígido que as AVL. Então, já que existem mais consultas do que inserções (comparações entre o par de árvores e antes de adicionar a palavra na árvore em si, consultando a estrutura que representa as “stopwords”), pareceu mais eficiente utilizar as AVLs.

2.2 Segunda Implementação

A fim de melhorar o desempenho do algoritmo em tempo de execução, criou-se uma nova implementação, baseada no código anterior, mas com alterações relacionadas principalmente às estruturas de dados anteriormente escolhidas. Dessa vez, foram escolhidas árvores Rubro-Negras para os arquivos textos e manteve-se a AVL para as stopwords, por representarem um arquivo pequeno. Nada mais esclarecedor do que fazer ambas as implementações e comparar os resultados.

3. Conclusão

Após serem conduzidas diversas análises e verificações de ambos códigos e também testagens usando dados de exemplos disponibilizados na proposta do trabalho (Veja imagem 1), concluiu-se que a primeira implementação tinha um desempenho significativamente melhor do que a segunda. Em termos de precisão do coeficiente de Jaccard, ambos códigos eram equivalentes, mas quanto ao tempo de execução, o programa que utilizava árvores Rubro-Negras para os arquivos textos tinha um desempenho mais lento. A razão dessa diferença se deve a algumas possíveis causas, como a complexidade da implementação, como o código foi otimizado ou até mesmo a escolha dos parâmetros e configurações utilizadas. No caso específico da comparação entre árvores AVL e Rubro-Negras, pode ser interessante explorar outras implementações (estruturas que não sejam árvores) e técnicas de otimização, para que se obtenham execuções ainda mais rápidas.

Imagem 1 (AVL para stopwords e Rubro-Negra para os textos à esquerda e AVL para stopwords e textos à direita)