# Análise de sentimento usando ABPs

Lucas L. Nascimento

Universidade Federal do Rio Grande do Sul

INF 01203 – Estruturas de Dados

Viviane P. Moreira

24/11/2021

# Análise de sentimento usando ABPs

Este trabalho tem como objetivo comparar o desempenho de dois tipos de Árvores Binárias de Pesquisa: A Árvore Binária de Pesquisa Convencional (Não balanceada) e a Árvore de Adelson-Velsky and Landis (AVL), que, por sua vez, é balanceada.

## Método

### Análise de Sentimento

A Análise de sentimentos é uma técnica de mineração de dados, usada para extrair sentimento de um texto. A extração do sentimento de um texto pode ser feita usando machine learning, ou, como veremos neste trabalho, usando uma abordagem lexical. Para medirmos o grau de sentimento de cada texto, usaremos um grau de polaridade p, e o classificaremos como positivo para $p>0$, neutro para $p=0$, e negativo para $p<0$.

#### *Léxico*

Para calcularmos o grau de polaridade do texto, usaremos um léxico de opinião. O léxico de opinião é formado por uma lista de palavras e sua respectiva polaridade. Por exemplo, a palavra “legal” tem um grau de polaridade positivo de 0.9, já a palavra “chato” tem polaridade negativa de -1.1. Para determinarmos o grau de polaridade de um texto, será efetuada a soma dos valores numéricos de polaridade das palavras que compõem o texto.

**Textos Opinativos.** Usaremos uma série de textos opinativos e calcularemos a polaridade de cada um deles.

# Implementação

## Linguagem

Para implementar a análise de sentimento usaremos um programa em C e usaremos uma série de algoritmos.

## Estruturação do programa

O programa será dividido em três arquivos principais:

- analiseDeSentimento.c

  Que servirá como main() no arquivo.

- Abp.c e Abp.h

  Arquivo TAD para a estrutura da ABP.

- Avl.c e Avl.h

  Arquivo TAD para a estrutura da AVL.

## Chamada do programa

A chamada do programa será feita em console da seguinte maneira:

<nome_do_programa.exe> <arquivo_lexico.txt> <arquivo_de_entrada.txt> <arquivo_de_saida.txt> <tipo de árvore (abp ou avl)>

**Exemplo de chamada:**

Ex 1. analiseDeSentimento.exe _léxico_shuf.txt _reviews.txt _saida.txt abp

## Funcionamento do programa

### *Mapeamento das palavras do léxico*

Para calcular a polaridade dos textos informados na chamada, primeiro o programa irá mapear todas as palavras fornecidas no arquivo passado como primeiro argumento do programa

(_léxico_shuf.txt) no Ex. 1 acima. As palavras retiradas do arquivo serão lidas e inseridas em uma AVL, ou em uma ABP, dependendo do valor do quarto parâmetro informado ao programa.

***Cálculo da polaridade dos textos***

A seguir, o programa segue para ler o arquivo contendo os textos opinativos, informado como segundo parâmetro, (_reviews.txt) no Ex. 1 acima. O programa então calcula a polaridade de cada texto informado no arquivo parâmetro, calculando a polaridade de cada palavra do texto e fazendo uma soma.

***Retorno dos valores calculados***

Depois de calculados os valores de polaridade dos textos, os mesmos serão impressos em um arquivo, passado como terceiro argumento na chamada do programa, arquivo (_saida.txt) no Ex. 1 acima.

***Métricas de comparação de eficiência***

Após a execução do programa, será informado o número de comparações e inserções na árvore necessárias para a execução do programa, e, também será informado o número de rotações, caso a árvore escolhida for a AVL.

## Comparação de eficiência

Foram realizados inúmeros testes comparando a eficiência das duas ABPs usadas no programa. Foram testados os seguintes parâmetros:

**Quantidade de inserções:**

Como os dados a serem inseridos nas duas árvores são idênticos, o número de inserções será exatamente o mesmo entre as duas. Porém, a AVL não permite inserções de nodos com valor chave repetido. Isso é computado para termos certeza da diferença de eficiência entre as duas árvores.

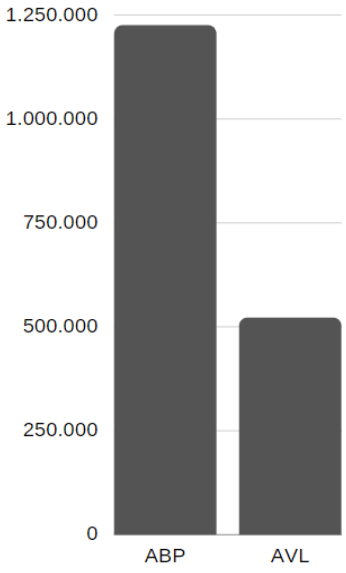
**Nos testes efetuados o número de inserções foi:**

ABP: 5033 AVL: 5027 nodos.

Como podemos verificar, o número de nodos inseridos é praticamente igual nas duas árvores, com exceção dos nodos repetidos que não foram inseridos na AVL.

**Quantidade de Comparações:**

A quantidade de comparações é a melhor maneira de vermos a diferença de eficiência entre os dois tipos de árvores. Como podemos ver neste gráfico:




**Número exato de comparações:**

ABP: 1.224.472 comparações

AVL: 520.952 comparações

Como podemos perceber, o número de comparações na AVL é 80.6% menor do que o número de comparações na ABP.

A diferença se dá pelo fato da AVL ser balanceada, e por isso ser muito mais completa do que a ABP comum. Com 5000 nodos já podemos ver a grande diferença, que tenderá a aumentar conforme mais nodos são adicionados à árvore.

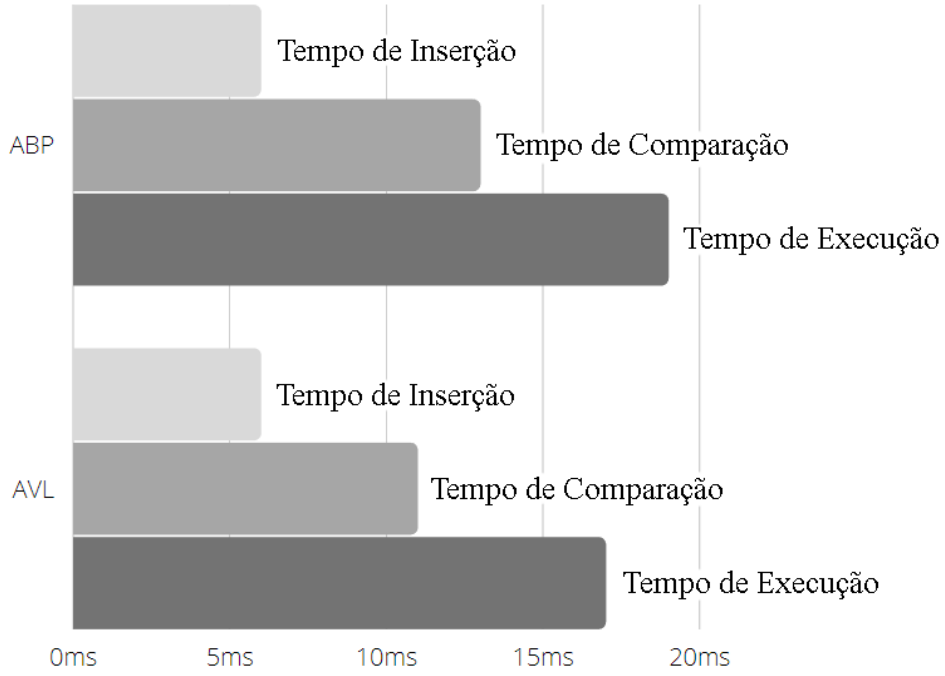
**Quantidade de Rotações:**

Para balancear a AVL e ter como resultado uma árvore mais completa e eficiente, precisamos realizar rotações para manter o fator de balanceamento da árvore. Nos testes, com 5027 nodos, a AVL teve que ser balanceada diversas vezes, e para isso, foram necessárias 2542 rotações. Que sim, demandam processamento, mas o resultado, como podemos ver no gráfico de comparação, resulta em uma maior performance no geral.

**Tempo de Execução:**

Também foi comparado o tempo de execução dos dois tipos de árvores. Também é uma boa maneira de medirmos a eficiência de cada uma.

Como podemos ver neste gráfico:



Os tempos exatos de inserção, comparação e execução da **ABP** foram:

Tempo de Inserção: 6ms

Tempo de Comparação: 13ms

Tempo de Execução: 19ms

Os tempos da **AVL** foram:

Tempo de Inserção: 6ms

Tempo de Comparação: 11ms

Tempo de Execução: 17ms

O tempo de inserção entre as duas árvores ficou exatamente igual, já o tempo de comparação teve uma diferença de 16.6%, que em milissegundos é uma diferença bem significativa. E o tempo de Execução teve uma diferença de 11.1%, que também é bem significativa.

## Conclusão

Ao fim do trabalho, podemos concluir que a pesquisa realizada ampliou e comprovou o que já era esperado, a AVL ser significativamente mais eficiente do que a ABP. Foi possível provar com números reais e exemplos práticos a superioridade na eficiência da AVL e consequentemente das árvores balanceadas.

### Implementação da aplicação em C

A implementação dos algoritmos em C foi bastante complexa, mas estão satisfatoriamente estáveis. E o programa foi capaz de realizar todos os testes e apresentar os números para comparar a eficiência das árvores.

### Comparação entre a eficiência das árvores

Comparando a eficiência das árvores, vemos uma significativa diferença entre a AVL e a ABP, onde a AVL se mostra significativamente mais eficiente, por ser balanceada. Vimos também que os passos extras de rotação na inserção de novos nodos, não desequilibra a vantagem no tempo de comparação da AVL, comparado com a ABP.

## References

Bush, D. (2021). Assistência com conceitos de ponteiros.

*https://stackoverflow.com/questions/70023520/initializing-a-null-pointer-before-fscanf*