Albit - PSE

Cristhian Grundman
Danilo Lemos Cardoso
Igor Cortes Junqueira

Junho de 2020

Conteúdo

I. Resumo

ALBIT-PSE é um mecanismo de busca por palavras chave em um conjunto de textos pré-processados, do banco de dados Raw. Utilizando uma estrutura de Trie, a busca é executada em C++ no servidor e retornada ao usuário por meio da interface em HTML+CSS, interconectados pela linguagem PHP. Além disso, o usuário conta com informações como tempo de execução da busca e quantidade de resultados, todos dispostos convenientemente na interface. Simples, eficiente e elegante, ALBIT-PSE é, certamente, a raiz de suas buscas.

Na etapa de pré-processamento, os textos recebem chaves numéricas identificadoras e, mais importante, notas de acordo com um sistema de classificação para cada palavra, atribuindo pesos à posição da mesma no texto (título, corpo) e também à sua densidade. Assim, os resultados exibidos para aquela busca podem ser notoriamente ordenados de acordo com a prioridade de correspondência do texto com a busca, indicando ao usuário o que é mais relevante como as primeiras posições dentre as correspondências.

Todo esse sistema de pré-processamento se resume à montagem de uma árvore cujos nós terminadores portam os ID's correspondentes às buscas. Para isso, a árvore é construída e comprimida a cada data-base, e depois armazenada e preparada para ser carregada no momento da inicialização do sistema para as buscas. Além disso, parte das informações são carregadas para a RAM do servidor para melhor desempenho.

Na construção, são utilizados dois tipos de nó. O BASICNODE, com indicação de letra e apenas um filho, é utilizado na geração de caminhos inexistentes. Se surge no processo a necessidade de mais de uma ramificação, esse nó é trocado por um MULTINODE, com 36 filhos e identificação, e seu espaço é aproveitado para a criação do novo filho, que é um BASICNODE. Desta maneira, a árvore se apresenta sempre compacta, tornando desnecessária uma etapa extra de limpeza da mesma.

Por fim, mas não menos importante, uma vez inicializado o ambiente e tudo preparado, a palavra-chave é enviada pelo usuário. A busca então corresponde a localizá-la na Trie e retornar os títulos dos textos correspondentes aos ID's associados a ela. A ordem já estabelecida no processamento prévio garante os melhores resultados e a otimização da árvore o melhor tempo. Desta forma, o usuário pode então analisar aquilo que lhe for conveniente clicando nos títulos dos textos para abri-los, ou mesmo realizando uma nova busca.

II. Descrição

1. A estrutura de dados

A estrutura de dados utilizada para a busca de palavras é uma *trie*, uma árvore onde cada nó representa a palavra formada pelo caminho da raiz ao nó. Cada nó possui um atributo que indica a lista de todos os documentos que possuem a palavra associada ao nó. Uma análise revelou que 89% dos nós possuem no máximo um filho. Foi necessária a seguinte compressão para a trie: há dois tipos de nó, **basic** e **multi**. Um nó básico só pode ter um filho, enquanto um nó multi pode ter até 36 (um para cada elemento do alfabeto reduzido). Um nó básico ocupa 16 vezes menos espaço de memória que um nó multi (desconsiderando *struct padding*).

2. O pré-processamento

Para o nosso projeto, utilizamos a versão *raw* do corpus.

A primeira etapa do processamento foi uma formatação simples. Os arquivos foram renomeados para o formato `db*`, onde * é qualquer inteiro entre 0 e 163. As marcações que indicam o início de um documento foram simplificadas para indicarem somente o título, como em `#doc Henry Hallam`. As marcações de fim de arquivo(inconsistentes) foram removidas.

A segunda etapa do processamento foi a construção da trie em si. Para cada palavra, sua versão no alfabeto reduzido é adicionada na trie, de tal modo que a trie continue sempre comprimida. Para adicionar uma palavra, suas letras no alfabeto reduzido ditam um caminho na árvore. Se o caminho não continua para uma letra, é necessário incluir mais um nó na árvore. Um nó básico é adicionado. Há dois casos a serem considerados: quando se abre caminho por um nó multi e por um básico. O caso multi é trivial. No caso de nó básico, este é substituído por um nó multi que recebe dois filhos: o que o nó básico tinha antes da adição e o novo filho. Novamente, se escolhe um nó básico para o novo filho (aproveitando o nó básico que foi destruído). Desse modo, é impossível haver um nó multi com um filho só.

O nó (que representa a palavra a ser adicionada) indica uma lista de documentos que a contém. Então à essa lista é adicionado o índice do documento. Os resultados recebem pesos que indicam sua relevância para a busca, levando um consideração a proporção dessa palavra no documento e se ela aparece no título. A lista de resultados é ordenada por ordem decrescente de peso, facilitando a busca.

Imediatamente após a construção da trie, esta é serializada e armazenada em disco em 3 arquivos: um para a lista de nós básicos, uma para os nós multi e um para todas as listas de resultados. Essa divisão permitiu que a serialização fosse trivial.

3. A interface

A interface foi elaborada em um conjunto de páginas HTML5, aprimoradas com CSS. A conexão do cliente com o servidor e a dinâmica das páginas foi programada em PHP. As páginas foram elaboradas e forma simples e elegante para tornar o uso fácil e intuitivo. Ademais, o sistema usa memória compartilhada, fato usado pela interface web.

A página inicial do buscador conta com uma barra de pesquisa, um botão de envio e a logo do buscador. A pesquisa pode ser feita pela mesma ou pela própria URL, visto que o método utilizado é o GET. Por isso, a qualquer momento pode-se utilizar o endereço */search.php?word=**palavra*** para pesquisar a palavra em questão.

Ao clicar sobre a logo da página inicial, o usuário é direcionado para uma página de apresentação do projeto, com um texto resumo do mesmo e o vídeo de explicação. Vale ressaltar que, assim como em todas as páginas do buscador, é possível pesquisar diretamente da mesma em uma barra de busca convenientemente localizada.

Por fim mas não menos importante, pesquisar uma palavra leva o usuário a uma página que lista os títulos dos resultados de acordo com a ordem atribuída pelo sistema de pesos, garantindo uma boa ordenação por relevância. Nesse ponto, o usuário pode clicar sobre qualquer título para inspecionar o respectivo artigo.

Para utilização da mesma, um servidor pode ser emulado, no nosso caso com Apache rodando em Linux. A partir disso, o buscador fica acessível por qualquer dispositivo na mesma rede pelo endereço IP do servidor, inclusive em dispositivos móveis. Assim, a acessibilidade do sistema é imensamente aprimorada pela interface web.

III. Resultados

1. Espaço e armazenamento

Quanto ao quesito espaço, a adaptação do sistema de Trie permitiu a criação de uma árvore incrivelmente compacta, que ocupa aproximadamente 200MB. Juntamente com um vetor de resultados de tamanho 1GB, a árvore é carregada na RAM para melhor desempenho na busca.

Utilizamos na implementação um espaço de memória compartilhado para o qual os arquivos são carregados ao se inicializar o servidor. Assim, a interface web interage direto com essa memória, o que corrobora para a otimização do sistema.

2. Tempo

As buscas são feitas com o tempo de se percorrer a árvore. O nosso tempo está na ordem de 20 microsegundos para buscas simples. Uma busca que envolve mais de uma palavra demora mais. Primeiro, as palavras são buscadas individualmente, e suas listas de resultados são ordenadas em ordem crescente de tamanho. Assim, ao se fazer interseções dois a dois, se faz primeiro com entre as listas mais restritivas, fazendo com que palavras específicas diminuem o tempo de busca consideravelmente. A interseção é feita em tempo quadrático, pois as listas não estão ordenadas por índice de artigo, mas por peso, que não está presente nas listas.

IV. Limitações e trabalhos futuros

Como todo projeto, estamos sempre em evolução. Há mais detalhes que gostaríamos de acrescentar e ficam pendentes para possíveis futuras implementações, bem como limitações do sistema elaborado. Nas principais, podemos citar a sugestão de palavras, a acentuação, a paginação na interface e a possibilidade de operadores com múltiplas palavras.

Quanto ao primeiro, era de intuito do grupo implementar um algoritmo baseado na distênica de Levenshtein para sugerir palavras caso não fossem encontrados resultados para a pesquisa efetuada. Contudo, algumas ideias paralelas que não foram bem sucedidas na implementação e, especialmente, a limitação de tempo, impossibilitou que esta parte estivesse operante na finalização do trabalho.

Quanto à acentuação, a árvore foi desenvolvida em um alfabeto de 36 letras, sendo o alfabeto e os algarismos. Optamos por esse modelo por eficiência, visto que a perca não é muito grande. Ainda assim é uma limitação considerável, uma vez que o sistema não distingue coisas como "pais" e "país". No banco em inglês o prejuízo é mínimo, mas ainda é existente no projeto como um todo.

Quanto à paginação na interface, não foi encontrada à tempo solução viável para a paginação de forma eficiente. A paginação é feita na pesquisa no terminal de 20 em 20 resultados, satisfazendo portanto os requisitos do trabalho. Sendo assim, teoricamente foi cumprido com os pontos exigidos. Mesmo assim, é um detalhe inconveniente que desejaríamos ter contornado.

Por fim, quando são pesquisadas duas ou mais palavras, o buscador retorna a interseção das mesmas, ou seja, os artigos que possuem todas elas simultaneamente. Esse detalhe foi inclusive acrescentado pós apresentação, bem a tempo da entrega, o que é uma grande evolução. Um adicional seria poder trabalhar com negação ou união no conjunto pesquisado, dando liberdade ao usuário de especificar e combinar os operadores. Embora o padrão seja apenas a intersecção, essa conveniente amplitude de operadores fica também para uma possibilidade de melhoria futura.

V. Conclusões

A elaboração do trabalho como um todo foi desafiadora. Seguiram-se ao longo do projeto diversos caminhos e experimentações que tiveram que ser ocasionalmente abandonadas ou selecionadas. A amplitude de possibilidades e a situação de isolamento também foram dificultadores presentes na elaboração.

Mesmo assim, o grupo reconhece uma grande evolução na gama de ferramentas nas diversas áreas exigidas por um projeto de tal porte. Toda a estruturação em quesito de integração dos membros, delegação de tarefas e conhecimento prático foi certamente muito construtiva para a trajetória acadêmica de cada um dos integrantes de forma singular.

Assim, adoramos elaborar e executar um trabalho tão desafiador e engrandecedor, e agradecemos a oportunidade de fazê-lo. Ficamos extremamente contentes com nossos resultados, excepcionalmente com o quesito eficiência, e alcançamos, ao nosso ver, um ótimo patamar no projeto.

VI. Distribuição

Para a execução do trabalho, os nichos de implementações foram divididos pelo grupo. A forma adotada seguiu-se como especificado a seguir:

A estrutura, o pré processamento e o mecanismo de busca foi desenvolvido em conjunto por Cristhian Grundman e Danilo Cardoso. A interface web e comunicação cliente servidor, bem como vídeo de apresentação, foram desenvolvidos por Igor Junqueira.

VII. Links

1. Repositório

Todo o projeto se encontra no seguinte repositório:
`https://github.com/cristhiangrundmann/Albit`

2. Vídeo do projeto

O vídeo do projeto pode ser acessado no link:
`https://www.youtube.com/watch?v=azfLyqZMez0`