Universidade Federal do Amazonas
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Programa de Pós-Graduação em Informática

Métodos de Poda Estática para Índices de Máquina de Busca

Francisca Sancha Azevedo da Silva

Manaus-AM
Julho-2009

Francisca Sancha Azevedo da Silva

Métodos de Poda Estática para Índices de Máquina de Busca

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Departamento de Ciência da Computação da Universidade Federal do Amazonas, como requisito parcial para obtenção do Título de Mestre em Informática. Área de concentração: Bancos de Dados e Recuperação de Informação.

Orientador: Prof. Dr. Edleno Silva de Moura

Francisca Sancha Azevedo da Silva

Métodos de Poda Estática para Índices de Máquina de Busca

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Departamento de Ciência da Computação da Universidade Federal do Amazonas, como requisito parcial para obtenção do Título de Mestre em Informática. Área de concentração: Bancos de Dados e Recuperação de Informação.

Banca Examinadora

Prof. Dr. Edleno Silva de Moura – Orientador
Departamento de Ciência da Computação – UFAM/PPGI

Prof. Dr. Joao Marcos Cavalcanti
Departamento de Ciência da Computação – UFAM/PPGI

Prof. Dr. Marco Antônio Pinheiro de Cristo
FUCAPI

Manaus-AM
Julho-2009

AGRADECIMENTOS

A Deus por permitir a concretização deste momento;

Ao professor Edleno, pelo acompanhamento e paciência providos durante todo o trabalho;

Aos meus familiares pelo apoio;

Aos colegas pelo incentivo;

A todos que contribuíram para realização deste trabalho muito obrigada.

Resumo

Métodos de poda de índices têm sido propostos a fim de melhorar a eficiência de máquinas de busca. Nos métodos de poda estática, as entradas do índice que têm pouca contribuição para ordenação das respostas são descartadas, reduzindo o tempo de acesso ao índice. Neste trabalho é realizado um estudo sobre métodos de poda estática e são propostos dois métodos. O primeiro método proposto leva em consideração a raridade dos termos do vocabulário para realizar a poda. Este método é apropriado para coleções que possuam grande incidência de termos raros nas consultas e um vocabulário com os tamanhos dos termos não concentrado em faixas de tamanho específicas. O segundo método proposto é apropriado para coleções onde os termos das consultas têm baixos valores de idf (inverse document frequency) e baseia-se no tamanho dos termos de *logs* de consultas para realizar a poda. Os resultados apresentados foram satisfatórios e podem ser aplicados a coleções com as características das estudadas neste trabalho.

Palavras-chave: Máquinas de Busca, Indexação, Poda Estática.

Abstract

Index pruning methods have been proposed to increase efficiency of search engines. Static pruning methods discard index entries, whose contributions for the answer ranking are irrelevant, reducing their access time. In this work, static pruning methods are studied and two methods are proposed. The first one considers the *idf* (inverse document frequency) of vocabulary terms to make pruning. This method is appropriated to collections that have many rare query terms and have a vocabulary with term sizes that are not concentrated in specific sizes. The second method proposed, is aimed to collections whose query terms are not so rare and takes into account the query term sizes to discard the list entries. The results obtained are effective and can be applied to collections similar to ones studied here.

Keywords: Search engines, Indexing, Pruning methods

Sumário

Lista de Figuras

Capítulo 1

Introdução

O número de usuários de máquinas de busca tem aumentado nos últimos anos, uma vez que a utilização destas ferramentas proporciona acesso rápido à informação disponível na *Web*. O acesso aos dados armazenados nestes sistemas não é realizado de forma seqüencial, pois a base de texto geralmente é muito grande. É comum que sejam utilizadas estruturas de índices para que os dados sejam acessados de forma eficiente durante o processo de busca.

Os índices geralmente utilizados em máquinas de busca são chamados de índices invertidos ou arquivos invertidos. Estas estruturas possuem vocabulário e listas invertidas. No vocabulário são armazenados todos os termos distintos da coleção. Nas listas invertidas são mantidas as referências para cada termo do vocabulário e a freqüência ou posição de cada documento onde estes termos aparecem.

Estudos têm mostrado que a quantidade de informação na *Web* continua crescendo de maneira acelerada e que a taxa de consultas submetidas às ferramentas de busca é de centenas a milhares por segundo [11,12]. O crescimento contínuo da *Web* implica na necessidade de atualização da base de dados dos sistemas de busca. Esta atualização inclui a inserção de novos documentos na base de texto e no índice mantido aumentando o tamanho deste. O tempo gasto para acessar índices muito grandes pode reduzir a eficiência de máquinas de busca.

Em [12,13] é estimado que cerca de 80% dos usuários vêem apenas entre as 30 e 60 primeiras respostas por consulta submetida. Brin et. al.[11] afirmam que os usuários vêem apenas as 10 primeiras respostas. Isto indica que acessar todo o índice e retornar todos os possíveis resultados por consulta pode ser um esforço desnecessário.

Métodos de poda de índices [6,7,8,9,12,13] têm sido propostos e estudados na

literatura a fim de melhorar a eficiência de máquinas de busca. Ao aplicar tais métodos são descartadas ou não processadas entradas do índice que pouco contribuem para ordenação das respostas. Isto para que seja reduzido o tempo de acesso ao índice e (ou) do tamanho deste.

Quando entradas do índice são removidas do disco antes do processamento de consultas, o método aplicado é chamado de poda estática. Nestes métodos pode ocorrer alguma perda de informação devido ao descarte de entradas. Eles possuem a vantagem de além de reduzir espaço em disco ocupado pelos índices, acelerar o processo de busca. Nas abordagens chamadas de métodos de poda dinâmica, o índice é totalmente mantido e são utilizadas estratégias para selecionar as entradas do índice que devem ser lidas durante o processamento de consultas.

Em [6], foi proposto um método de poda estático no qual é utilizada a função de ordenação da máquina de busca para realizar poda. Cada termo pertencente ao vocabulário da coleção é submetido como uma consulta à função de ordenação. As entradas referentes aos k documentos retornados por consulta são mantidas no índice e as demais são descartadas. Este método aqui chamado de método de Carmel é utilizado como referência a este trabalho.

Com o objetivo de minimizar o impacto da poda sobre a qualidade das respostas, são propostas melhorias ao método de Carmel, neste trabalho. As estratégias propostas apontam duas novas formas de selecionar os k documentos do topo que são mantidos durante a poda.

A primeira consiste em considerar a raridade dos termos durante a escolha do k, ou seja, as listas mais raras são menos afetadas pela poda do que as que possuem raridade menor. A segunda é baseada no tamanho dos termos de consultas previamente submetidas à máquina de busca.

Ao aplicar tais estratégias um tamanho exato do índice é obtido, ou seja, é possível

ter a porcentagem desejada de poda aplicada. Com o nível desejado de poda, é possível realizar experimentos variando as porcentagens de poda do método de Carmel que são utilizadas como entrada para o *LBPM* proposto por Moura et. al.[7].

1.1. Trabalhos Relacionados

1.1.1 Métodos de poda dinâmica

Nos métodos de poda dinâmica, normalmente as entradas das listas invertidas são processadas até que seja encontrada alguma condição de parada. Em Moffat et. al.[14], a condição de parada apresentada consiste em limitar o número de acumuladores (onde são armazenados os cálculos parciais de similaridade). Há duas formas de utilizar esta condição nesta abordagem. Na primeira, o processamento de uma consulta é finalizado quando o limite de acumuladores é alcançado. Na segunda, o processamento continua depois de atingir o número limite de acumuladores, porém novos documentos não são inseridos no conjunto.

A abordagem proposta por Persin et. al.[13] atua sobre índices de freqüência e requer que as entradas de listas invertidas sejam mantidas em ordem decrescente de freqüência. Uma freqüência mínima é indicada através de limiares para determinar se uma entrada deve ser processada. A ordenação das listas por maior freqüência permite que as entradas processadas primeiro sejam as que mais contribuem para a ordenação das respostas. Os termos das consultas são processados por maior idf. Um acumulador é atualizado quando a importância do termo, ponderada por sua freqüência, é grande o suficiente para ter impacto na ordenação final dos documentos.

1.1.2 Métodos de poda estática

Através destes métodos é possível melhorar o desempenho de ferramentas de busca pela redução do tamanho do índice, o que implica na diminuição do tempo gasto para acessar esta estrutura. Tais métodos tentam selecionar quais entradas do índice podem ser removidas sem que a qualidade das respostas do sistema seja comprometida.

Na abordagem centrada em documento, proposta por Clarke et. al.[15], é calculada a relevância dos termos para os documentos. São mantidas nas listas invertidas as entradas com os k termos do topo do documento cujo termo é relevante para ele. Este k pode ser fixo ou proporcional ao número de termos distintos do documento. Este método é voltado para podar o índice de maneira que este caiba completamente em memória e não tem informação posicional.

O método de Carmel é um método de poda estático no qual cada termo pertencente ao vocabulário da coleção é submetido como consulta ao modelo de ordenação do sistema de busca. Sendo k um inteiro positivo, os k documentos do topo do conjunto de respostas serão mantidos na lista do termo correspondente. As demais entradas do índice são removidas do disco. A seleção deste valor k pode ser feita de duas formas, como descrito a seguir:

1. Atribuir a k um valor inteiro positivo. Por exemplo, com k=10 cada termo terá 10 documentos mantidos na lista invertida e os demais serão descartados.

2. Sendo $0 < \delta <= 1$ um valor obtido através de experimentos e *Sim* a pontuação do documento do topo das respostas à consulta *q*, os k documentos com a pontuação mínima δ * *Sim* serão mantidos na lista invertida. Por exemplo, para $\delta = 0.5$, cada documento com uma pontuação de no mínimo 50% do valor de similaridade do documento do topo das respostas será mantido nas listas. Neste caso, o número de documentos mantidos em cada lista não é fixo, pois depende dos valores de similaridade dos documentos e do valor δ

escolhido.

O método de poda baseado em localidade (LBPM), proposto por Moura et al.[7], é um método de poda estático que atua também sobre índices de posição e não somente em índices de freqüência. Esta abordagem utiliza o método de Carmel para selecionar as ocorrências de termo consideradas mais importantes para o texto. Os termos mais importantes de um documento são os que possuem este documento no conjunto de respostas retornado. Quaisquer melhorias ao método de Carmel poderiam facilitar a estimativa de poda do LBPM, uma vez que tornariam melhor a seleção das ocorrências de termo.

Os documentos são divididos em fragmentos e a cada passo é selecionado o fragmento que tem a maior quantidade de termos selecionados como importantes para o documento. Isto é feito até que seja atingido o nível de poda desejado. São mantidas as entradas das listas invertidas que estão no mesmo fragmento e que correspondem a termos importantes para determinado documento. Assim, podem ser exploradas possíveis relações entre os termos.

1.2 Contribuições :

- Modificação do método de Carmel.
- Proposta e avaliação de dois métodos novos baseado no método de Carmel.

1.3 Organização da dissertação

Este trabalho está divido em capítulos dos quais este é o primeiro. O capítulo 2 aborda os principais conceitos necessários à compreensão deste trabalho. O capítulo 3 descreve a estrutura do método proposto e o 4 apresenta os experimentos realizados e resultados obtidos. No capítulo 5, é apresentado um resumo dos resultados obtidos, as conclusões e sugestões para trabalhos futuros.

Capítulo 2

Conceitos Básicos

Neste capítulo, serão apresentados conceitos básicos necessários à compreensão dos métodos propostos neste trabalho.

2.1 Máquinas de busca

Máquinas de busca (ou motores de busca) são ferramentas que realizam recuperação automatizada de documentos, para auxiliar os usuários destes sistemas na busca por informação relevante. Para obter os documentos do seu interesse, o usuário traduz a informação que necessita em uma consulta, escrita na linguagem fornecida pelo sistema. Geralmente isto implica em especificar um conjunto de palavras-chave.

Os principais componentes de um sistema de busca voltado para *web* geralmente são: coletor, indexador e processador de consultas. O coletor realiza a atualização dos documentos armazenados localmente. Ele possui um escalonador que decide quais páginas armazenar e quais deverão ser re-coletadas.

No indexador são construídos os índices do sistema, utilizados para acelerar o processo de recuperação de texto. Os índices mais utilizados nos sistemas de busca são os índices invertidos (ou arquivos invertidos) que possuem vocabulário e listas invertidas. No vocabulário, são armazenados os termos distintos da coleção de documentos do sistema. Nas listas invertidas, são mantidas informações sobre a ocorrência dos termos nos documentos. Esta informação pode ser a freqüência com que os termos aparecem nos documentos ou a posição do texto onde aparecem.

No processador de consultas, o vocabulário é recuperado e as listas invertidas são

manipuladas utilizando algum modelo de recuperação de informação. Estes modelos associam um valor numérico a cada documento indicando sua similaridade com a consulta. Os documentos que possuem os maiores valores de similaridade são mostrados como respostas.

2.2 Modelo vetorial

Neste modelo de recuperação de informação, proposto por Salton [15], documentos e consultas de usuários são representados por vetores em um espaço de dimensão n, onde n correspondente ao total de palavras distintas da coleção.

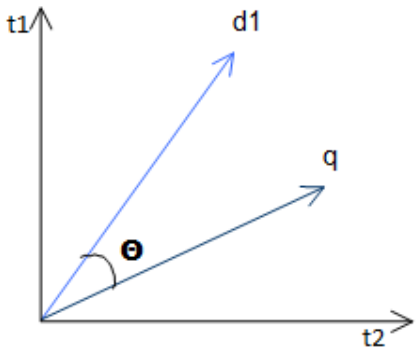

Figura 2.1 - Exemplo de espaço no modelo vetorial

No exemplo do espaço vetorial mostrado na Figura 2.1, temos as palavras distintas da coleção t1 e t2 e o espaço correspondente a estes termos com duas dimensões. O documento d e a consulta q são representados por vetores. Θ corresponde ao ângulo formado entre d e q.

São atribuídos pesos não binários aos termos dos documentos e das consultas. Cada peso representa a importância do termo no documento ou consulta e indica a posição do documento em cada dimensão. Esses pesos são usados para calcular o grau de similaridade entre cada documento da coleção e consulta do usuário. Os pesos são calculados como a seguir:

$$w(t,d) = tf(t,d) \times idf(t)$$

Onde w é o peso do termo t no documento d, tf é a freqüência do termo t no documento d e idf é a freqüência inversa do termo t (ou raridade do termo). Se um termo aparece muitas vezes em um documento este termo é considerado importante para o documento. O idf expressa a importância de um termo na coleção, considerando que se um termo aparece em muitos documentos, então ele não representa bem os documentos. Esta medida obedece à seguinte fórmula:

$$idf(t) = \ln(\frac{N}{n})$$

Onde N é o total de documentos na coleção e n é o número de documentos onde o termo *t* aparece.

Para o cálculo da ordenação final dos documentos, é computada a similaridade através da obtenção do valor do cosseno do ângulo formado entre os vetores documento e consulta (ver Figura 1). Assim, a similaridade a ser computada é calculada pela fórmula:

$$Similarida\ de(q,d) = \cos\Theta = \frac{\sum_{t=1}^{n} w(t,q) \bullet \sum_{t=1}^{n} w(t,d)}{\sqrt{\sum_{t=1}^{n} w(t,q))^{2} \bullet \sum_{t=1}^{n} (w(t,d))^{2}}}$$

Onde *q* representa o vetor de termos da consulta; *d* representa o vetor de termos do documento; *w(t,q)* é o peso do termo *t* na consulta *q*, *w(t,d)* o peso do termo *t* no documento *d*.

2.3 Métricas de Avaliação

Neste trabalho foi utilizada a distância entre listas para avaliar os efeitos dos métodos de poda na qualidade dos resultados do sistema.

2.3.1 Distância entre listas

Na variação do método Kendall's tau proposta em [16] e aqui utilizada, são comparados os primeiros k elementos de dois conjuntos de resposta e é atribuída uma pontuação. Esta pontuação representa o grau de similaridade dos conjuntos. Sendo C1 e C2 conjuntos com os k primeiros documentos resposta, a pontuação S(i,j) é atribuída a cada par {i,j} onde i e j são documentos retornados entre os k do topo de pelo menos um dos conjuntos e são avaliados como a seguir:

- Caso 1: i e j aparecem nos dois topos. Se estes documentos aparecem na mesma ordem, ou seja, i aparece antes de j ou j aparece antes de i nos dois conjuntos, S(i; j) = 0. Caso contrário, S(i; j) = 1.
- Caso 2: i e j aparecem em um conjunto, mas apenas i ou apenas j aparecem no outro. Supondo que i e j apareçam em C1 e apenas i apareça em C2, se i vier antes de j em C1 então S(i; j) = 0, caso contrário S(i; j) = 1. Isto porque se j não aparece em C2 ele deverá estar abaixo do topo na lista, mantendo desta forma os documentos i e j na mesma ordem nas duas listas. Se i e j estivessem na ordem contrária em C1, então pelo mesmo raciocínio a pontuação atribuída a S(i; j) seria 1.
- Caso 3: Apenas i aparece em um dos topos, e apenas j aparece no outro topo. Neste caso, S(i; j) = 1, pois seguindo a mesma idéia descrita no caso 2 o segundo documento estará abaixo do topo nas duas listas, indicando que a ordem de i e j é obrigatoriamente inversa.
- Caso 4: i e j aparecem em um dos topos, mas nenhum dos dois aparece no outro topo. Neste caso não há informação suficiente para atribuir uma pontuação, pois não é

conhecida a ordem entre i e j na outra lista, sendo utilizada neste caso uma pontuação neutra S(i; j) = p. Neste trabalho, é atribuído p = 1/2, como utilizado em [10].

A pontuação aplicada varia de 0 a k(3k-1)/2. A primeira indica topos iguais e a segunda topos completamente diferentes. O resultado é então normalizado através da fórmula mostrada a seguir, proposta em [6]:

$$x' = 1 - \frac{2k}{k(3k-1)}$$

Onde x é o resultado original que, no caso, corresponde à soma de todas as pontuações calculadas. Esta normalização é feita para que a pontuação assuma valores de 0 até 1, onde 0 indica topos extremamente diferentes e 1 aponta para topos iguais.

Capítulo 3

Ambiente experimental

Neste capítulo será apresentado o ambiente utilizado nos experimentos realizados. Este ambiente foi o mesmo utilizado em [10].

3.1 Tipos de consulta

Consultas submetidas por usuários de máquinas de busca podem ser classificadas de duas formas. A primeira classificação está relacionada ao interesse do usuário e a segunda às opções oferecidas pelo sistema de busca. Quanto ao interesse do usuário, consultas podem ser classificadas em[18]:

- Consulta Informacional: o usuário está interessado em obter informação a respeito de um tópico específico. Este tipo de consulta pode ser atendido por uma ou mais páginas de respostas.
- Consulta Navegacional: a intenção do usuário é obter o endereço de uma página e geralmente apenas um resultado é suficiente para atender a necessidade do usuário; por exemplo, se um usuário está interessado encontrar o site do banco do brasil. O resultado www.bb.com.br atenderia bem à requisição solicitada.
- Consulta Transacional: o usuário deseja alcançar um site para realizar uma transação, como fazer compras, procurar serviços web ou arquivos para download, comprar algum produto, pesquisar sobre pacotes de viagem etc.

A segunda classificação pode ser feita em três tipos de consultas considerados importantes para sistemas de busca, descritos a seguir:

- Conjuntivas: todos os termos da consulta devem estar em um documento para que este seja retornado como resposta. Por exemplo, uma consulta com os termos "Biografia Cecília Meireles", deveria ter como resposta apenas os documentos onde os 3 termos aparecem.
- Disjuntivas: Neste tipo de consultas no mínimo um dos termos da consulta deve estar nos documentos resposta. Por exemplo, a consulta “música” e “brasileira” poderia retornar documentos onde aparecesse somente o termo “música” ou “brasileira“ ou ambos.
- Frases: Todos os termos devem estar presentes nos documentos resposta na mesma ordem em que aparecem na consulta submetida. Por exemplo, a consulta "Música popular brasileira" deveria retornar apenas documentos que contém esta frase completa.

Aproximadamente 81,1% das consultas do *log* do WBR com mais de um termo são conjuntivas, em contraste a consultas por frase e disjuntivas, que constituem respectivamente 18.3% e 0.6% das consultas submetidas com mais de um termo [10].

Como consultas com apenas um termo são independentes do tipo classificado, foram submetidas nos experimentos consultas com mais de um termo.

3.2 Coleções de referência

Grande parte das pesquisas realizadas em sistemas de recuperação de informação utiliza coleções homogêneas, como coleções compostas por artigos específicos [11]. Para representar este tipo de coleção, foi utilizada neste trabalho a coleção Latimes que faz parte da *trec*[17].

A Latimes contém artigos publicados no jornal Los Angeles Times, com aproximadamente 132.000 documentos. As consultas utilizadas para esta coleção foram extraídas dos títulos dos tópicos 401 a 450 pertencentes às sessões ad-hoc da *trec* 8. Estas consultas foram submetidas para a máquina de busca como consultas disjuntivas, conjuntivas e frases.

O funcionamento de quaisquer métodos aplicados sobre coleções homogêneas, como a *trec*, pode não produzir resultados semelhantes aos obtidos em coleções Web, devido à diferença existente entre elas [11]. Uma das diferenças existentes é que as coleções web são compostas por documentos heterogêneos e não há controle sobre o conteúdo publicado na web.

Para verificar possíveis diferenças entre a aplicação dos métodos de poda em coleções Web e coleções como a Latimes, e para aproximar o ambiente experimental de um ambiente real de um sistema de busca, foram utilizadas duas versões da coleção WBR. As duas versões da WBR foram escolhidas devido à intenção de se verificar possíveis mudanças no comportamento dos resultados dos métodos, propostos neste trabalho, quando coleções Web sofrem atualizações. A primeira versão, será aqui chamada de WBR10 constituída por 10.077.722 páginas coletadas da Web brasileira em 1999. A segunda, é uma versão atualizada da WBR10 coletada em 2003, com aproximadamente 12 milhões de documentos e aqui chamada de WBR.

Um *log* de 3 milhões de consultas submetidas por usuários reais foi utilizado nos experimentos. Este *log* é uma das razões da escolha da coleção WBR para os experimentos, pois fornece informações úteis sobre as preferências dos usuários da máquina de busca. Foram selecionadas de forma aleatória 1000 consultas do *log* de consultas para cada tipo de consulta, disjuntiva, conjuntiva e por frase. Também foram selecionadas consultas informacionais e navegacionais, cada conjunto contendo 200 consultas.

Foram analisadas algumas propriedades da coleção como o tamanho das listas invertidas dos termos na coleção e o tamanho das listas de termos presentes em *logs* de consultas. As tabelas 3.4, 3.5 e 3.6 mostram respectivamente o tamanho (em kilobytes) das listas invertidas dos termos das coleções Latimes, WBR10 e WBR e a porcentagem dos termos com listas nos tamanhos correspondentes.

A coleção Latimes apresenta 248.758 termos dos quais 92,23% apresentam listas menores que 1kb (ver tabela 3.1) e o maior tamanho de lista invertida nesta coleção é de 1 megabyte. Os vocabulários das coleções WBR10 e WBR contêm respectivamente 2.965.474 e 17.009.825 termos distintos. Os tamanhos das listas chegam a 51,58Mb na WBR10 e 64,20Mb na WBR.

Tamanho das Listas Invertidas (Kb)	*(%) Termos do Vocabulário*
<1	*92,23*
1 a 10	*6,49*
10 a 50	*0,12*
>=50	*1,16*

Tabela 3.1 - Vocabulário Latimes: porcentagem das faixas de tamanho quando consideradas as palavras do vocabulário

Tamanho das Listas Invertidas (Kb)	*(%)Termos do Vocabulário*
<1	91,85
1 a 10	5,967
10 a 50	1,388
>=50	0,7945

Tabela 3.2 - Vocabulário WBR10: porcentagem das faixas de tamanho quando consideradas as palavras do vocabulário

Tamanho das Listas Invertidas (Kb)	*(%) Termos do Vocabulário*
<1	97,659
1 a 10	1,76
10 a 50	0,3556
>=50	0,2244

Tabela 3.3 - Vocabulário WBR: porcentagem das faixas de tamanho quando consideradas as palavras do vocabulário.

As tabelas 3.4, 3.5 e 3.6 mostram respectivamente o tamanho das listas invertidas dos termos em kilobytes e a porcentagem dos termos que possuem os tamanhos correspondentes das coleções Latimes, WBR10 e WBR. Apesar das distribuições de tamanho serem parecidas nas três coleções quando considerado o vocabulário completo, a distribuição muda quando consideradas as consultas. Nas consultas, há uma concentração maior de termos com listas menores (termos raros) na Latimes. Já os termos das consultas selecionadas da WBR10 e da WBR não são raros como os da Latimes, com 81,07% dos termos das consultas da WBR10 e 87,56% dos termos das consultas WBR com listas de tamanhos iguais ou maiores que 50kb. A idéia dos métodos de poda estática propostos (ver capítulo 4) é baseada nestas informações sobre a raridade dos termos.

Tamanho das Listas Invertidas (Kb)	**(%) *Termos das Consultas***
<1	12,23
1 a 10	6,49
10 a 50	0,12
>=50	11, 63

Tabela 3.4 - Consultas Latimes: porcentagem das faixas de tamanho quando consideradas apenas palavras das consultas.

Tamanho das Listas Invertidas (Kb)	**(%) *Termos das Consultas***
<1	2,17
1 a 10	4,76
10 a 50	12,00
>=50	81,07

Tabela 3.5 - Consultas WBR10: porcentagem das faixas de tamanho quando consideradas apenas palavras das consultas.

Tamanho das Listas Invertidas (Kb)	(%) *Termos das Consultas*
<1	2,56
1 a 10	2,77
10 a 50	7,11
>=50	87,56

Tabela 3.6 - Consultas WBR: porcentagem das faixas de tamanho quando consideradas apenas palavras das consultas.

Capítulo 4

Métodos de poda estática

Os métodos de poda estática tentam identificar quais entradas do índice podem ser removidas sem que haja perda na qualidade das respostas [9], de forma que o tamanho do índice seja reduzido, economizando assim espaço de armazenamento e reduzindo o custo para processar consultas.

O método de poda estática proposto por Carmel et al[6], reduz significantemente o tamanho do índice utilizado em sistemas de recuperação de informação. Este método pode ser aplicado de maneira uniforme quando mantidas k entradas do topo, ou através de um limiar fixo de corte quando são mantidas as entradas mais similares à do topo. Entretanto, estas formas de realizar poda dificultam o controle do nível de poda aplicado, uma vez que todas as listas têm o mesmo limiar de corte ou mesmo número de entradas mantidas. Como o índice obtido depende do valor de entrada, o nível de poda pode ser diferente do desejado. Além disso, ambas as estratégias assumem de certo modo que todos os termos devem ser podados com o mesmo critério, o que pode levar à perda da qualidade quando houver diferença na importância de cada termo para a coleção.

As modificações ao método de Carmel propostas neste trabalho visam obter mais controle sobre o nível de poda e aumentar a qualidade das respostas, através do descarte não uniforme nas entradas das listas. Neste caso, as listas não terão um limiar fixo de corte, ou seja, cada lista terá seu próprio número de entradas mantidas.

4.1 Método de poda baseado em ordenação por idf

No método de Carmel, cada termo é submetido como consulta à função de ordenação da máquina de busca. Cada lista de documentos de resposta está ordenada por maior grau de similaridade com o termo correspondente, ou seja, os documentos mais relevantes para determinado termo estão no topo da lista de respostas.

Vocabulário	Listas de Resposta
Ta	<D9, 0.90>, <D7, 0.85>, <D1, 0.66>, <D2, 0.55>, <D3, 0.3>, <D5, 0.1>
Tb	<D8, 0.93>, <D3, 0.78>,<D7, 0.72>, <D1, 0.55>
Tc	<D11, 0.88>
Td	<D3, 0.78>, < D7, 0.45>, <D2, 0.32>
Te	<D2, 0.99>, <D1, 0.70>
Tf	<D1, 0.91>, <D3, 0.80>, <D11, 0.65> , <D6, 0.58>
Tg	<D2, 0.87>, <D1, 0.66>, <D7, 0.45>

Figura 4.1.a - Exemplo de listas de respostas para termos de um vocabulário

A Figura 4.1 serve para ilustrar o funcionamento do método de Carmel. A Figura 4.1.a mostra um vocabulário de uma coleção e as listas retornadas como resposta para os termos submetidos à função de ordenação de um sistema de busca. A Figura 4.1.b mostra quais referências serão mantidas ao aplicar o método de Carmel uniforme utilizando k=1. A Figura 4.1.c mostra as referências que serão mantidas ao aplicar o delta-top com delta=0.8.

Carmel com K=1:

Ta	<D9, 0.90>
Tb	<D8, 0.93>
Tc	<D11, 0.88>
Td	<D3, 0.78>
Te	<D2, 0.99>
Tf	<D1, 0.91>
Tg	<D2, 0.87>

Figura 4.1.b - Poda com Método k-top de Carmel

Carmel com Delta= 0.8:

Ta	<D9, 0.90>, <D7, 0.85>
Tb	<D8, 0.93>, <D3, 0.78>
Tc	<D11, 0.88>
Td	<D3, 0.78>
Te	<D2, 0.99>
Tf	<D1, 0.91>, <D3, 0.80>
Tg	<D2, 0.87>

Figura 4.1. c - Poda com o delta-top do carmel

No exemplo dado, ao aplicar k=1, apenas uma entrada de cada lista permanece e no máximo duas entradas da segunda aplicação de poda. Como pode ser visto nas Figuras acima, ao realizar a poda através de um limiar fixo de corte, não há controle direto sobre os níveis de poda aplicados. Se o objetivo fosse manter 65% deste índice, seria necessário encontrar outro k ou delta para obter um nível de poda aproximado.

Como no método de Carmel a ordenação é feita por maior similaridade entre os documentos e o termo correspondente, é possível inferir que se o número de entradas que permanece no índice variar de acordo com alguma propriedade que influencie a poda, a qualidade dos resultados obtida pode ser aumentada. Com base nessa observação, pode-se tentar apresentar o processo de poda com uma estratégia denominada aqui de método de poda baseado em ordenação por idf. A modificação proposta consiste na idéia de variar o k, o número de entradas mantidas por lista, de forma que se tenha controle sobre o nível de poda desejado e o aumento na qualidade das respostas.

Para indicar como deve ser o corte de listas, ou seja, de que forma atribuir valores a k, são utilizadas propriedades das coleções. Para seleção dos valores de k, neste primeiro método proposto, é considerada a influência do idf dos termos do vocabulário sobre a poda. O idf é aqui utilizado por indicar a importância de um termo na coleção. Esta informação apresenta-se como um forte indicador para o valor de corte de listas de uma coleção. Com o uso do idf, listas mais raras ou de tamanhos menores recebem prioridade durante a poda, sendo menos afetadas, ou até não afetadas, enquanto listas menos raras são mais afetadas. As listas menores (de termos mais raros), são mais preservadas para evitar perda de informação nestas listas. O método baseado em ordenação por idf é propício para coleções com consultas usando termos raros, como é o caso da Latimes (ver capítulo 3).

O algoritmo *Poda por Maior Idf* dado a seguir, recebe como entrada o nível do índice a ser mantido (*taxa)*.

Algoritmo *Poda por Maior Idf*:

Entrada: *taxa* - taxa do índice que será mantida

Variáveis: *tamanhoMax* - tamanho do índice final

tamÍndice - tamanho inicial do índice

V - conjunto de termos do vocabulário

V' - conjunto de termos do vocabulário ordenado por maiores valores de idf

B – conjunto dos números mínimos de entradas das listas com s entradas

S - conjunto com as quantidades de entradas que os tamanhos das listas possuem

N- número de listas com s entradas (s∈ S)

n*umPal* - número de palavras da coleção

tentrada -tamanho que corresponde uma entrada de lista

soma – acumulador

```
início
|
|   tamanhoMax ← taxa * tamÍndice;      //tamanho do índice após a poda
|   V' ← Ordena_Vocabulário(V);
|   blocoMin← 0;
|   numPal ← |V|;
|   soma←0; n← s ∈ S;
|   repita
|   |   soma←soma+(tentrada*numPal);
|   |   blocoMin← blocoMin +1;
|   |   n← s ∈ S;
|   |   Se (blocomin=n) então
|   |   |   nListas← n ∈ N;
|   |   |   numPal← numPal –nListas;
|   |   |   B←blocoMin;
|   |   fim
|   até(soma=tamanhoMax ∨ numPal=0)
|   fim
|   Calcula_k(blocoMin, tamanhoMax, V');
Fim
```

Calcula_k(*blocoMin, tamanhoMax, V', B*)

Entrada: *B* – conjunto com os números de entradas mínimo para as listas
tamanhoMax - tamanho máximo do índice final
V' - vocabulário ordenado por maiores valores de idf

Variáveis: *tamanhoLista* - tamanho da lista do termo *t*
k, ndocs- número de documentos mantidos no índice final
C- conjunto de documentos da coleção
R - Conjunto de documentos retornados pela função de similaridade
F - Conjunto de documentos mantidos na lista invertida
nentradas - número de entradas da lista atual *t*, cada entrada possui tamanho *tam*
acumulador - tamanho do índice

```
Início
|   para cada t ∈ V' faça
|   |   k← Recupera_BlocoMin(s);     //recupera valor de blocoMin
|   |   se (k < nentradas) então
|   |   |// incrementa o valor de k até que este seja o mesmo número de entradas da
|   |   |//    lista, ou seja alcançado o nível do índice desejado.
|   |   |   repita
|   |   |   |   k ← k+1;
|   |   |   |   acumulador ← k*tam;
|   |   |   até((acumulador=tamanhoMax) ∨ (k=nentradas));
|   |   |   fim
|   |   fim
|   |   ndocs ←0;
|   |   R ← Função_Ordenação(t, C);
|   |   enquanto (ndocs<k) faça      //mantém os k primeiros documentos de R
|   |   |   F ← d ∈ R;
|   |   |   ndocs←nTuplas+1;
|   |   fim
|   fim
fim
```

O nível do índice (*taxa*) dado na entrada do algoritmo *Poda por Maior Idf*, é utilizado para determinar o tamanho final do índice após o processo de poda. Por exemplo, se *taxa*=0.4

indica que 40% do índice será mantido e 60% das entradas serão removidas. O tamanho final do índice (*tamanhoMax*) é então calculado através do produto de *taxa* por *tamanhoÍndice* .

Na função Ordena_Vocabulário, o vocabulário é ordenado por maiores valores de idf(ou menores tamanhos das lista invertidas). No laço *repita* são calculados os valores de bloco mínimo para listas com s entradas, que consiste em um número mínimo de entradas para as listas de *V'*. A soma do valor de *blocoMin* de toda lista existente deve ser menor ou igual ao tamanho máximo do índice final. Se esta soma for menor, mais entradas serão armazenadas.

O valor de k de cada lista, o número de entradas mantidas, é calculado na função Calcula_k e consiste de um bloco mínimo somado ao número de entradas que não ultrapassam o tamanho final do índice (*tamanhoMax*).

Na Função_Ordenação é computada a similaridade de cada termo com os documentos de *C* (conjunto de documentos da coleção) e é obtido o conjunto de respostas (*R*). Os valores de similaridade neste trabalho foram calculados através do modelo vetorial descrito na seção 2. Após o valor de k ser calculado para um termo e obtido o conjunto de documentos reposta (*R*), o conjunto *F* (documentos que serão mantidos em uma lista) terá os *k* primeiros documentos de *R*. Isto é feito para cada termo pertencente à *V'*, conjunto de termos ordenados por maiores valores de idf.

No exemplo dado pela Figura 4.1.a, a poda com o método baseado em ordenação por idf funcionaria como nas Figuras a seguir:

Lista de Respostas Ordenadas Por Idf dos Termos:

Tc	<D11,0.88>
Te	<D2, 0.99>, <D1, 0.70>
Td	<D3, 0.78>, <D7, 0.45>, <D2, 0.32>
Tg	<D2, 0.87>, <D1, 0.66>, <D7, 0.45>
Tf	<D1, 0.91>, <D3, 0.80>, <D11, 0.65>, <D6, 0.58>
Tb	<D8, 0.93>, <D3, 0.78>, <D7, 0.72>, <D1, 0.55>
Ta	<D9, 0.90>, <D7, 0.85>, <D1, 0.66>, <D2, 0.55>, <D3, 0.3>, <D5, 0.1>

Figura 4.1.d - Ordenação de listas por idf

A Figura 4.1.d, mostra as listas dos termos da Figura 4.1.a ordenadas por maior raridade. A Figura 4.1.f a seguir, aponta quais entradas seriam mantidas se fosse aplicada a poda com o método baseado em ordenação por idf com um nível de 65% do índice. O índice final deve ter 15 entradas (correspondente ao nível desejado). O valor inicial de um bloco mínimo é de 1 entrada de lista invertida, totalizando 7 entradas de lista para o índice. Como este número é inferior ao limite de 15 entradas, pode ser adicionado mais um bloco às listas maiores, que corresponde a mais 1 entrada de lista.

O valor de k para Tc corresponde ao número de entradas que a lista possui, neste caso 1 entrada de lista. Para Te, k tem o mesmo valor de um bloco mínimo, já que o número de entradas da lista de Te é o mesmo de um bloco mínimo.

O valor de k para Td e Tg é de um bloco mínimo somado a 1 entrada, pois ainda é possível adicionar entradas para alcançar o nível do índice desejado. Para Tf, Tb e Ta o valor de k é de apenas um bloco mínimo, pois não é possível adicionar mais entradas. As listas mais raras, Tc ,Te, Td e Tg não são afetadas pela poda ao contrário das demais listas. Neste exemplo, é possível visualizar o controle direto sobre o nível desejado e o uso da raridade como indicador de corte das listas.

Tc	<D11,0.88>		
Te	<D2, 0.99>	<D1, 0.70>	
Td	<D3, 0.78>	<D7, 0.45>	<D2, 0.32>
Tg	<D2, 0.87>	<D1, 0.66>	<D7, 0.45>
Tf	<D1, 0.91>	<D3, 0.80>	<D11, 0.65>, <D6, 0.58>
Tb	<D8, 0.93>	<D3, 0.78>	<D7, 0.72>, <D1, 0.55>
Ta	<D9, 0.90>	<D7, 0.85>	<D1, 0.66>, <D2, 0.55>, <D3, 0.3>, <D5, 0.1>

Figura 4.1.e - Poda com o método baseado em ordenação por idf

4.2 Método de poda baseado em faixas de tamanhos

O idf dos termos, o tamanho da coleção e a freqüência com que os termos, tanto do vocabulário quanto das consultas, aparecem nos documentos são fatores que podem influenciar na qualidade dos resultados de índices podados. Em coleções *Web* é esperado que termos das consultas tendam a não ser tão raros quanto em coleções como a Latimes (ver capítulo 3). Se um termo raro começa a aparecer em consultas, então é provável que novos documentos apareçam rapidamente para falar sobre o assunto. Com o surgimento de novos documentos contendo os termos consultados, estes tornam-se mais comuns na coleção.

Para tratar tais coleções, é proposto o método de poda baseado em faixas de tamanhos que utiliza a informação do tamanho das listas invertidas do vocabulário da coleção e de um conjunto de consultas previamente submetidas à máquina de busca. Considerando este conjunto de consultas, arbitrariamente selecionadas, grande o suficiente para representar as faixas de tamanhos em que estão as consultas submetidas, é possível indicar quantas entradas devem ser mantidas nas listas de cada faixa.

As faixas de tamanhos em que se concentram os termos que aparecem nas consultas apontam para as listas que devem receber prioridade durante o processo de poda, ou seja, indicam quais listas são consideradas importantes durante a poda justamente por estas tenderem a aparecer nos termos das consultas. Por exemplo, temos classificados os termos do

vocabulário em dois conjuntos *A* ou *B*. O conjunto *A* possui termos cujo tamanho da lista invertida está entre *x* e *y* e em *B* estão os termos com listas de tamanho de *z* a *w*. Se em um conjunto de consultas for aplicada esta mesma classificação e a maioria dos termos fizer parte do conjunto *B*, significa que os termos que estão em *B* são considerados mais importantes e devem ser menos podados que os de *A*. Isto porque em *B* está a maior parte dos termos do conjunto de consultas. A seguir é apresentado o algoritmo de poda proposto para utilizar essa idéia.

O algoritmo *Poda por Faixas de Tamanho* também recebe como entrada o nível do índice a ser mantido *taxa* que também é utilizada para determinar o tamanho final do índice após o processo de poda. O tamanho final do índice é calculado através do produto de *taxa* por *tamanhoÍndice*.

Algoritmo *Poda por Faixas de Tamanho*:
Entrada: *taxa* - nível do índice que será mantido
numFaixas - número de faixas
Variáveis: *tamanhoMax* - tamanho do índice podado
tamÍndice - tamanho do índice inicial
totalTermos - total de termos do log de consultas
S – tamanhos das faixas *f*

```
Início
|   tamanhoMax ← taxa * tamÍndice;
|   F ←Classifica_Tamanhos_em_Faixas(tamÍndice,numFaixas);
|   Classifica_Termos_Consultas(F);
|   S ←Calcula_Peso_Faixas(tamanhoMax, totalTermos)
|   CalculaKs(S);
|   Poda_Listas();
Fim
```

Na função Classifica_Tamanhos_em_Faixas a seguir, é criado um conjunto de faixas *F*, para que os termos das consultas sejam classificado nestas faixas. O valor de entrada

numFaixas determina quantas faixas serão criadas. É calculado o tamanho de cada faixa do índice (*tam)* dividindo o tamanho do índice pelo número de faixas desejado. Os tamanhos das listas são somados até que alcancem o tamanho de uma faixa. *Se* ultrapassarem, é criada uma nova faixa *f*.

```
Classifica_Tamanhos_em_Faixas(tamÍndice, numFaixas)
        Entrada: tamÍndice- tamanho do índice antes da poda
                 numFaixas- número de faixas
        Variáveis:  F - conjunto de faixas
                    L - conjunto de tamanhos de listas invertidas da coleção
                    tam - tamanho das faixas
                    acm - acumulador de tamanhos
início
|       F← ϕ;
|       tam   ← tamÍndice/numFaixas;
|       acm ← 0;
|       F← f;       //primeira faixa
|       para cada tamLista ∈ L faça
|       |       se ((tamLista+ acm) <= tam) então
|       |       |       acm← acm+ tamLista;
|       |       fim
|       |       senão
|       |       |       F← f;    //Criar Faixas somando tamanhos das listas dos termos do voc..
|       |       |       acm←0;
|       |       fim
|       fim
fim
```

Em Classifica_Termos_Consultas, cada termo *t* do conjunto de consultas *C* é classificado em uma faixa $f \in F$ (conjunto de faixas) . Para cada faixa é acumulado o número de listas classificadas (*numListas*). O total de termos classificados em uma faixa de tamanho, é

utilizado na Calcula_Peso_Faixas.

```
Classifica_Termos_Consultas(F)
        Entrada:
        Variáveis:      F – conjunto de faixas
                        C - conjunto de consultas
                        x - tamanho mínimo de lista em uma faixa
                        y - tamanho máximo de lista em uma faixa
                        numListas - número de listas que pertencem à uma faixa
                        tc - tamanho da lista da consulta
Início
|       para cada f ∈ F faça
|       |       enquanto( (C≠ϕ) ∧ (tc∈ [x,y]) )    //classifica consulta na faixa f e acumula o numero
|       |       |                                   //de termos em cada faixa.
|       |       |       numListas ← numListas+ 1;
|       |       até;
|       |       Fim
|       Fim
Fim

Calcula_Peso_Faixas(tamanhoMax, totalTermos)
        Entrada: tamanhoMax- tamanho do índice final
                 totalTermos – número de termos do conjunto de consultas
        Variáveis: p – peso aplicado a uma faixa f
                   numListas – número de listas da faixa
                   S – tamanho das faixas
início
|       para cada faixa f ∈ F faça
|       |       peso ← (numListas x 100) / totalTermos ;    //Peso é atribuído a cada faixa
|       |       P ← peso;
|       |       tamanho ← (peso * tamanhoMax);    //tamanho total da faixa
|       |       S ← tamanho;
|       |       fim
|       fim
fim
```

Na função Calcula_Peso_Faixas são calculados os pesos dados a cada faixa *f*. Isto é feito para identificar quais faixas são comuns aos termos das consultas e para priorizar os termos

classificados em faixas com maiores pesos, que são as faixas que se espera os termos das consultas estejam concentrados.O cálculo do peso de uma faixa é a porcentagem que os termos desta faixa representam para o total. É criado o conjunto S com os tamanhos que cada faixa terá no índice podado. O tamanho máximo que uma faixa pode assumir é calculado multiplicando o peso que a faixa possui pelo tamanho máximo do índice final (*tamanhoMax*), controlando o nível de poda.

```
Calcula_k(S)
        Entrada: S – tamanho das faixas
        Variáveis:   V -vocabulário
                     acumulador - acumulador de tamanhos
                     k ,ktemp -  número de documentos
                     numEntradas -número de entradas de uma lista com tamanho tamanhoLista
início
|       para cada t ∈ V
|       |       k = 0;          //inicializa os valores de k
|       |       s←Recupera_Tam(S);   //lê tamanho limite da faixa que o termo pertence
|       Fim
|       acumulador ←0;
|       para cada t ∈ V faça
|       |       ktemp←log(tamanhoLista);
|       |       enquanto ((acumulador < s) ^ ((k+ktemp)<numEntradas)) faça    //s ∈ S
|       |       |       acumulador ← acumulador+ (k);
|       |       |       k ←k+ktemp;
|       |       fim
|       fim
fim
```

Em Calcula_k o valor de *k* é selecionado considerando o valor máximo permitido por faixa ($s \in S$). A soma do número de entradas não pode ultrapassar ao tamanho limite *s* de uma faixa *f*. O número de entradas das listas é adicionado utilizando o logaritmo do tamanho de

cada lista como uma entrada mínima, para não priorizar as listas mais raras da faixa.

```
Poda_Listas()
        Variáveis: V - vocabulário
                        C- conjunto de documentos da coleção
                        R - Conjunto de documentos retornados pela função de similaridade
                        F - Conjunto de documentos mantidos na lista invertida
                        ndocs - número de documentos
                        k - número de documentos mantidos na lista invertida
início
|       para cada t ∈ V faça
|       |       ndocs ←0;
|       |       R ← Função_Ordenação(t, C);
|       |       enquanto (ndocs<k) faça
|       |       |       F ← d ∈ R;
|       |       |       ndocs←ndocs+1;
|       |       fim
|       fim
fim
```

Na função Poda_Listas é utilizada a Funcao_Ordenação para computar a similaridade dos termos com os documentos. Nesta abordagem também foi utilizado o modelo vetorial para o cálculo de similaridade. São mantidos no conjunto final de documentos *F*, os *k* primeiros documentos do conjunto de respostas *R* correspondentes a cada lista. Ao final se obtém o índice podado com o nível controlado de ((1-*taxa*)*100) % de poda.

As Figuras 4.2.a e 4.2.b, dadas a seguir, servem para ilustrar o funcionamento do método de poda baseado em faixas de tamanho. A Figura 4.2.a mostra um exemplo de listas de respostas dos termos de um vocabulário. A figura 4.2.b aponta quais entradas seriam mantidas se fosse aplicada a poda com o método de poda baseado em faixas de tamanho com um nível de 70% do índice.

Ta	<D11,0.88>, <D2,0.27>
Tb	<D3, 0.78>, <D7, 0.45>, <D2, 0.32>
Tc	<D2, 0.87>, <D1, 0.66>, <D7, 0.45>
Td	<D1, 0.91>, <D3, 0.80>, <D11, 0.65>, <D6, 0.58>
Te	<D9, 0.89>, <D5, 0.85>, <D1, 0.66>, <D2, 0.55>
Tf	<D9, 0.90>, <D7, 0.83>, <D1, 0.72>, <D2, 0.68>, <D3, 0.5>
Tg	<D9, 0.95>, <D11, 0.80>, <D1, 0.76>, <D2, 0.55>, <D3, 0.43>,<D8,0.4>

Figura 4.2.a - Exemplo de Listas de respostas para termos de um vocabulário

O índice final de acordo com o nível desejado deve ter 19 entradas. Se os tamanhos das listas deste índice forem classificados em três faixas de tamanho, então teremos na primeira faixa (f1) as listas dos termos Ta,Tb e Tc, na segunda faixa (f2) as listas de Td e Te e na terceira faixa (f3) as listas de Tf e Tg.

Supondo que em um conjunto de consultas temos 30% dos tamanhos das listas dos termos das consultas classificados em f1, 45% em f2 e 25% em f3, ao aplicar estes pesos ao tamanho final do índice, teremos o máximo de 6 entradas mantidas nas listas de f1, 8 entradas mantidas nas listas de f2 e 5 entradas em f3.

Ta	<D11,0.88>, <D2,0.27>
Tb	<D3, 0.78>, <D7, 0.45>, <D2, 0.32>
Tc	<D2, 0.87>, <D1, 0.66>, <D7, 0.45>
Td	<D1, 0.91>, <D3, 0.80>, <D11, 0.65>, <D6, 0.58>
Te	<D9, 0.89>, <D5, 0.85>, <D1, 0.66>, <D2, 0.55>
Tf	<D9, 0.90>, <D7, 0.83> <D1, 0.72>, <D2, 0.68>, <D3, 0.5>
Tg	<D9, 0.95>, <D11, 0.80>, <D1, 0.76>, <D2, 0.55>, <D3, 0.43>,<D8,0.4>

Figura 4.2.b - Entradas das Listas mantidas com a poda

Ao final, temos nas listas de Ta, Tb e Tc 2 entradas mantidas, totalizando 6 que é o máximo da faixa f1. Td e Tc seriam totalmente mantidas e em Tf e Tg seriam mantidas respectivamente 2 e 3 entradas.

CAPÍTULO 5

Resultados dos experimentos

Os experimentos descritos neste capítulo foram realizados utilizando o ambiente descrito no capítulo 3 desta dissertação. O objetivo geral da realização destes experimentos foi comparar os métodos aqui propostos com o método de Carmel.

5.1 Distância entre listas

Serão mostrados a seguir os valores de similaridade entre as 20 respostas do topo do resultado obtido com o índice íntegro e as 20 respostas do topo do resultado obtido com os métodos de poda. A similaridade foi medida utilizando-se a variação do método Kendall's tau descrita na seção 2.1, na qual valores mais próximos de 1 indicam maior similaridade, enquanto os mais próximos de 0 indicam menor similaridade.

Os valores do percentual do índice mantido no método de Carmel não foram exatos, mas por ficarem próximos dos demais foram considerados do mesmo nível que o dos outros métodos. Nos gráficos referentes à coleção Latimes, o método baseado em faixas de tamanho não foi apresentado pois não havia *log* de consultas para esta coleção.

O gráfico da Figura 5.1 mostra os resultados obtidos por métodos de poda aplicados à coleção Latimes em consultas conjuntivas. Os resultados obtidos pelo método baseado no idf dos termos mostram-se satisfatórios quando comparados ao método de Carmel, sendo menores que este somente quando mantidos apenas 10% do índice. De 20% em diante, os valores de kendall do método proposto são maiores que o método de Carmel, alcançando uma ótima diferença quando a taxa mantida é de 40%, onde a similaridade do método proposto é de 0,6190 e do método base é de apenas 0,3903.

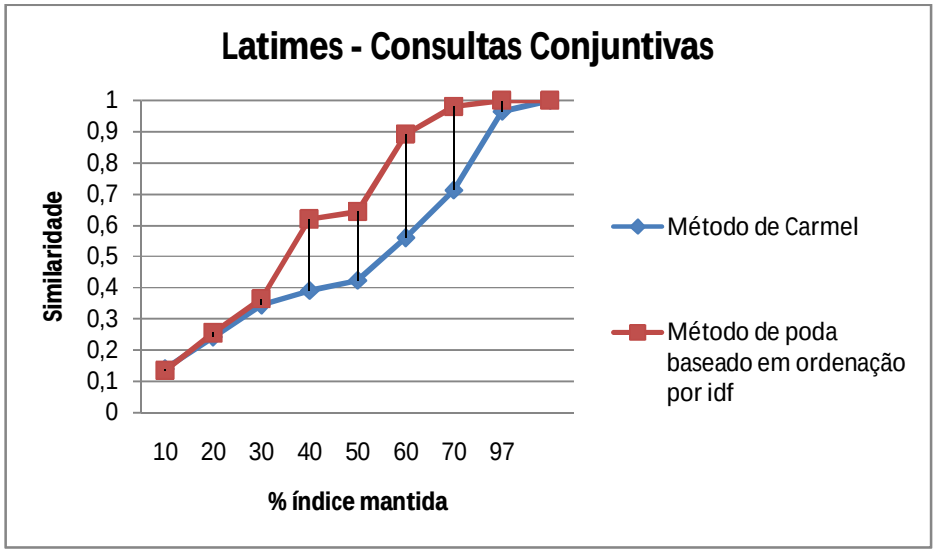

Figura 5.1- Latimes – Consultas Conjuntivas: Similaridade média entre as listas do índice podado e íntegro, em função do tamanho final do índice.

Os resultados obtidos em consultas disjuntivas apresentam-se, independente do nível de poda aplicado, para a poda baseada em ordenação por idf são próximos aos obtidos com o Carmel. Os resultados para consultas por frase (ver Figura 5.3) foram ligeiramente melhores do que os obtidos com Carmel a partir de 20%, semelhante ao que houve com as consultas conjuntivas.

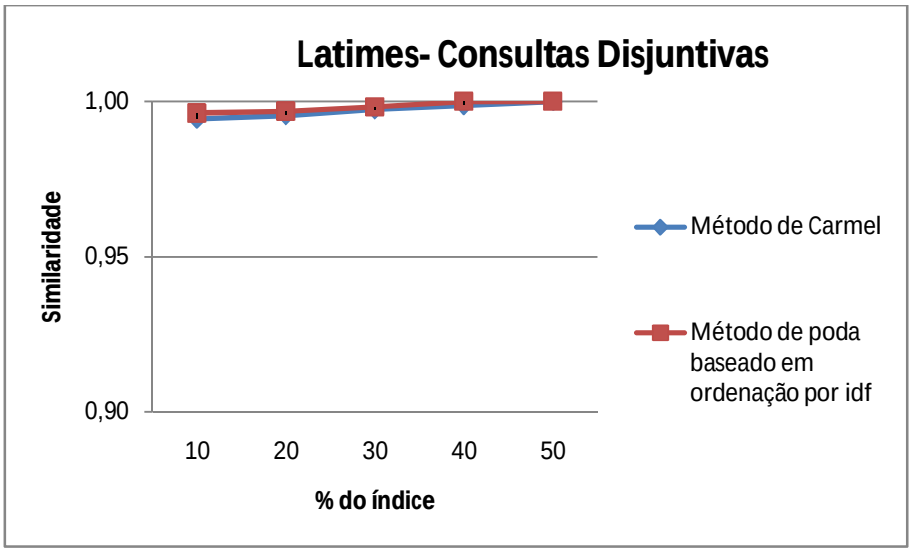

Figura 5.2 – Latimes- Consultas disjuntivas: Similaridade média entre as listas do índice podado e íntegro, em função do

tamanho final do índice.

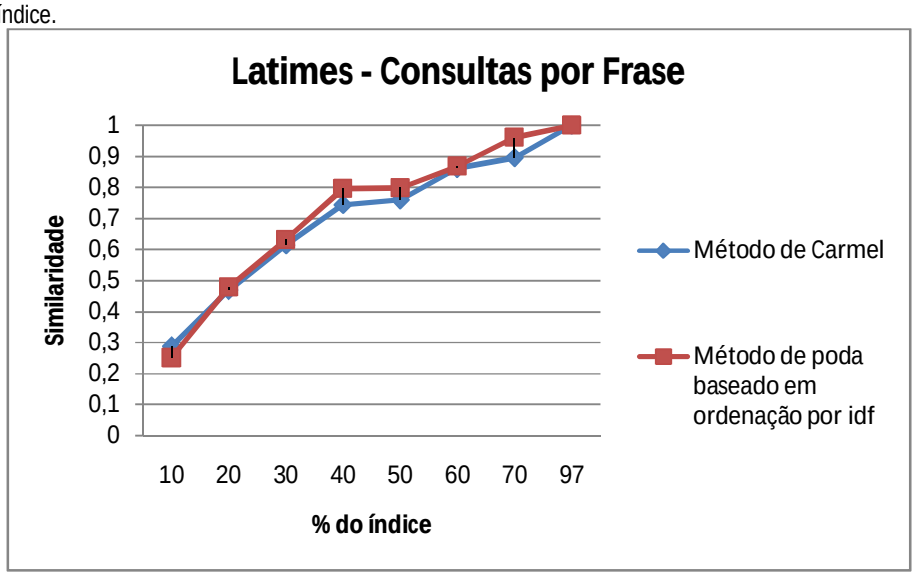

Figura 5.3 - Latimes- Consultas por frase: Similaridade média entre as listas do índice podado e íntegro, em função tamanho final do índice.

É possível visualizar nos gráficos das Figuras que ao aumentar a taxa do índice mantida os valores de similaridade média aumentam. Em consultas disjuntivas (ver Figura 5.2) a similaridade média mesmo com apenas 10% do índice apresenta valores acima de 0.99 nos dois métodos comparados. Apesar de mantida a eficácia do método baseado no idf nos três tipos de consultas, os melhores resultados obtidos foram com consultas conjuntivas.

Na coleção Latimes a raridade é um bom indicio para determinar os valores de k, pois os termos que aparecem nas consultas são raros (ver tabelas 3.1 e 3.2). Nela, percebe-se que o tamanho médio das listas das palavras das consultas é bem menor que o das palavras que ocorrem em média no vocabulário.

Nas Figura 5.4, 5.5 e 5.6 temos respectivamente a similaridade média obtida com consultas conjuntivas, disjuntivas e por frase dos dois métodos de poda propostos e do método de Carmel aplicados à coleção WBR10. Os resultados obtidos com o método de poda baseado no idf não tiveram ganho algum, com nenhum dos tipos de consulta submetidos. Isto aconteceu porque ao contrário do que ocorreu na Latimes, os termos raros não são tão comuns

nas consultas apresentadas na WBR.

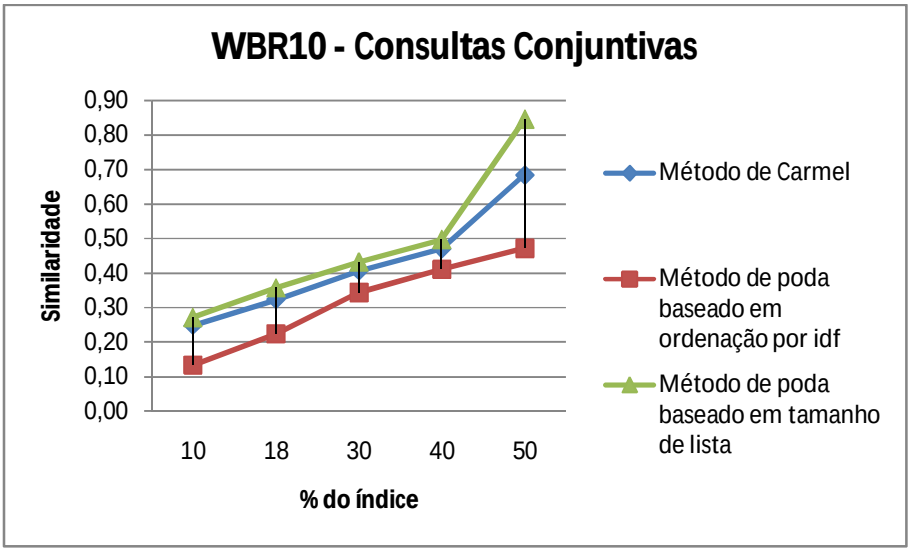

Figura 5.4 – WBR10 Consultas conjuntivas: Similaridade média entre as listas do índice podado e íntegro, em função do tamanho final do índice.

Já os resultados obtidos com o método de poda baseado em faixas de tamanho, foram satisfatórios quando aplicados a todos os tipos de consulta submetidos. Com consultas conjuntivas, os resultados foram melhores em todos os níveis, alcançando similaridade de 0,85 com 50% do índice enquanto no método base a similaridade foi de 0,68.

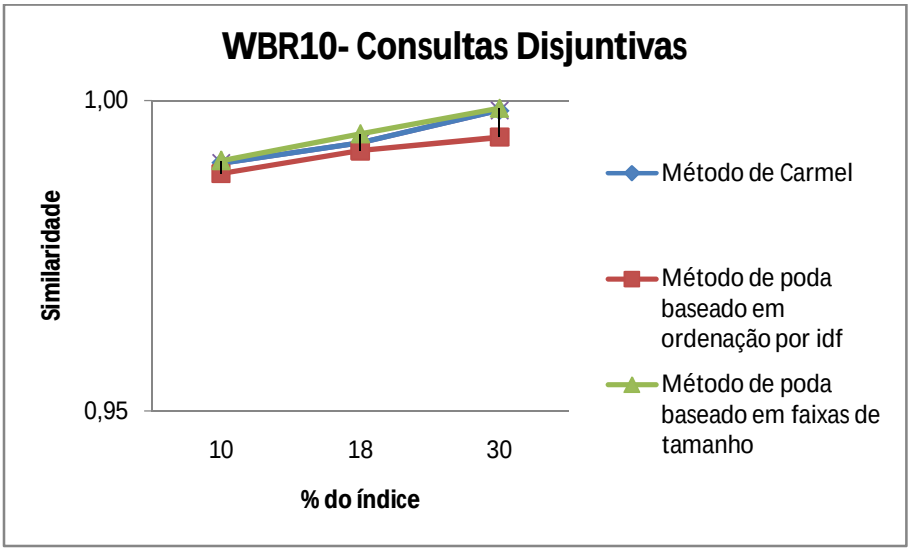

Figura 5.5 : WBR10- Consultas disjuntivas: Similaridade média entre as listas do índice podado e íntegro, em função do tamanho final do índice.

Com consultas disjuntivas a similaridade em todos os métodos experimentados ficou acima de 0.98, independente dos níveis de poda aplicados, mostrando que para este tipo de consulta todos os métodos de poda estudados podem ser aplicados ao índice e este pode ser mais comprimido. Já para consultas conjuntivas e por frase valores de similaridade indicam que para obter resultados melhores é necessário manter mais entradas no índice.

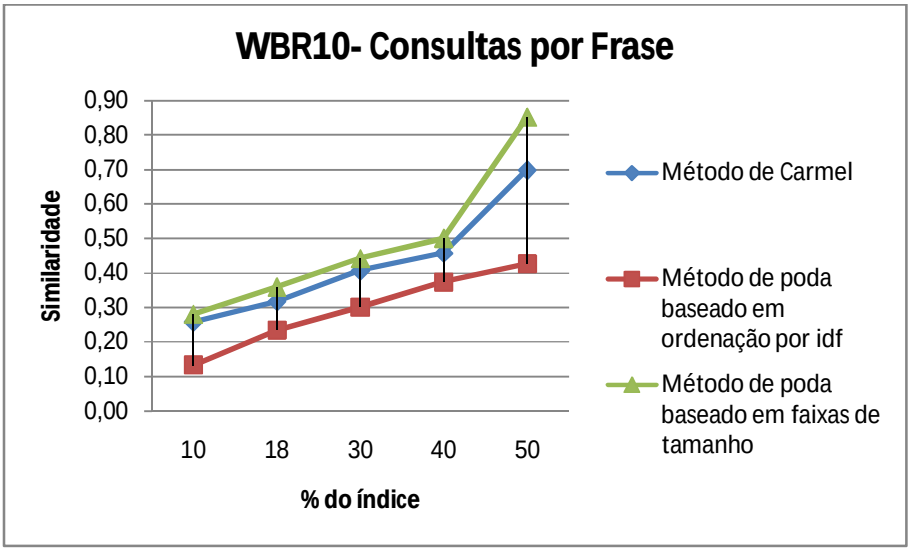

Figura 5.6 – WBR10- Consultas por frase: Similaridade média entre as listas do índice podado e íntegro, em função do tamanho final do índice.

Nas Figura 5.7, 5.8 e 5.9 temos os gráficos dos valores obtidos ao aplicar os métodos de poda sobre a coleção WBR.

Os resultados obtidos com a WBR se comportaram de forma semelhante aos da WBR10 sendo satisfatórios para o método de poda baseado em faixas de tamanho e os menores valores de similaridade obtidos com o método baseado no idf.

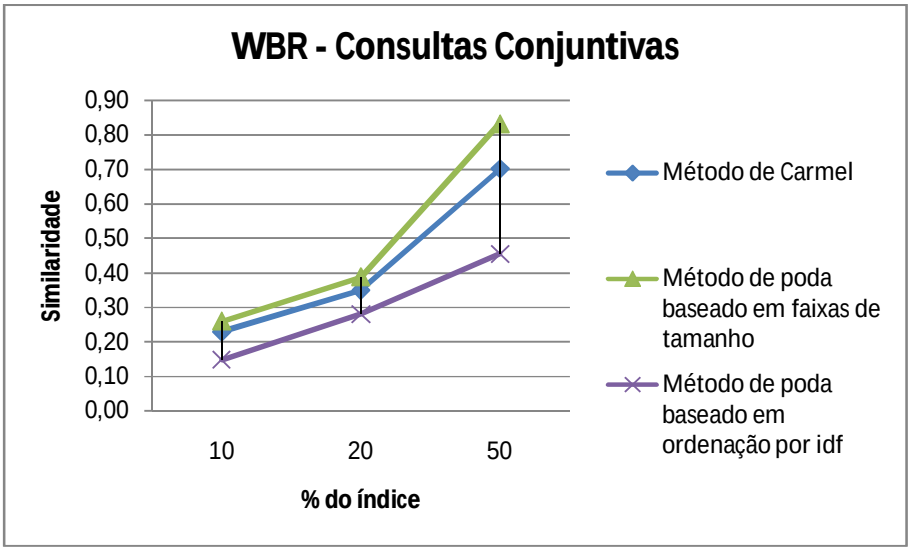

Figura 5.7 - Consultas conjuntivas na WBR: Similaridade média entre as listas do índice podado e íntegro, em função do tamanho final do índice.

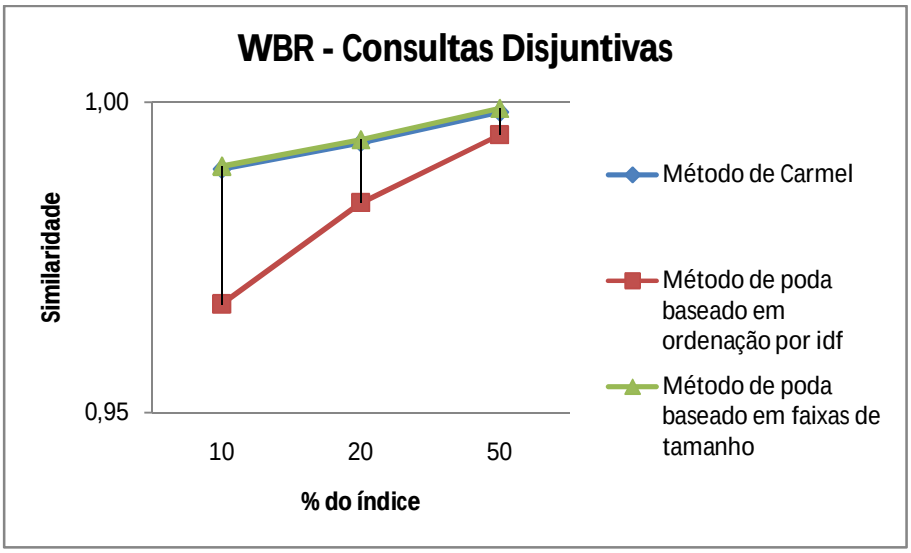

Figura 5.8- Consultas disjuntivas na WBR: Similaridade média entre as listas do índice podado e íntegro, em função do tamanho final do índice.

Nas coleções WBR e WBR10, a ocorrência maior de termos com baixo idf nas consultas influencia negativamente o método baseado em idf. Isto porque em coleções como a WBR as consultas não apresentam muitos termos raros e a maioria das listas dos vocabulário

são raras. Mais que 80% dos tamanhos das listas dos termos das consultas desta coleção têm valores maiores ou iguais à 50kb (Ver tabela 3.4 e 3.6).

O método baseado em faixas de tamanho foi novamente eficaz em todos os tipos de consulta.

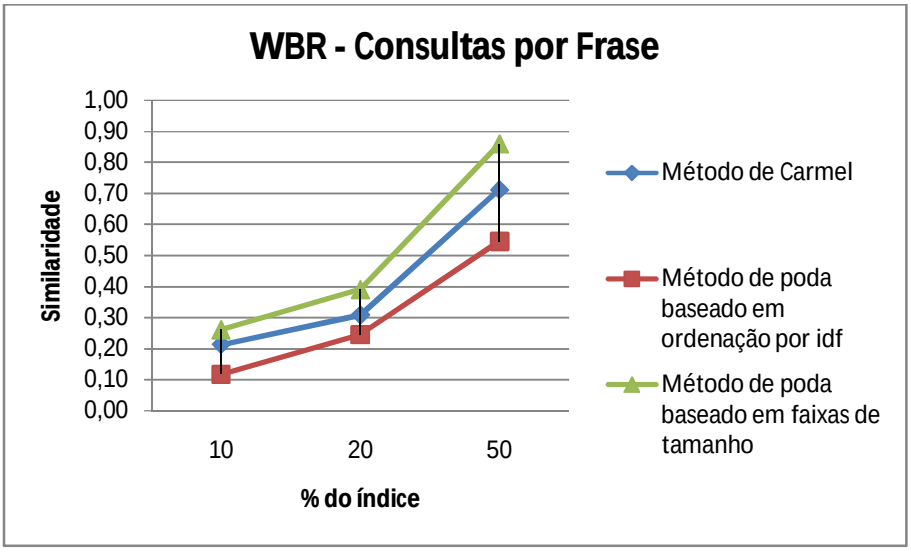

Figura 5.9- Consultas por frase na WBR: Similaridade média entre as listas do índice podado e íntegro, em função do tamanho final do índice.

Nas Figuras 5.10 e 5.11 temos os gráficos dos valores obtidos ao aplicar os métodos de poda utilizando consultas informacionais e navegacionais sobre a coleção WBR. Para consultas navegacionais foi utilizada também a evidência texto de âncora. Os resultados com consultas navegacionais são melhores que os obtidos com consultas informacionais, devido à influência do texto de âncora utilizado.

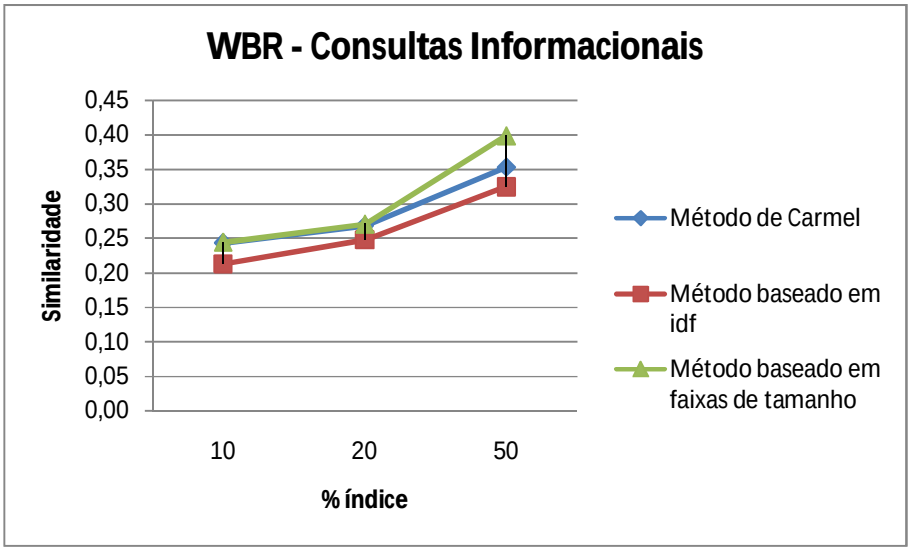

Figura 5.10- Consultas informacionais na WBR: Similaridade média entre as listas do índice podado e íntegro, em função do tamanho final do índice.

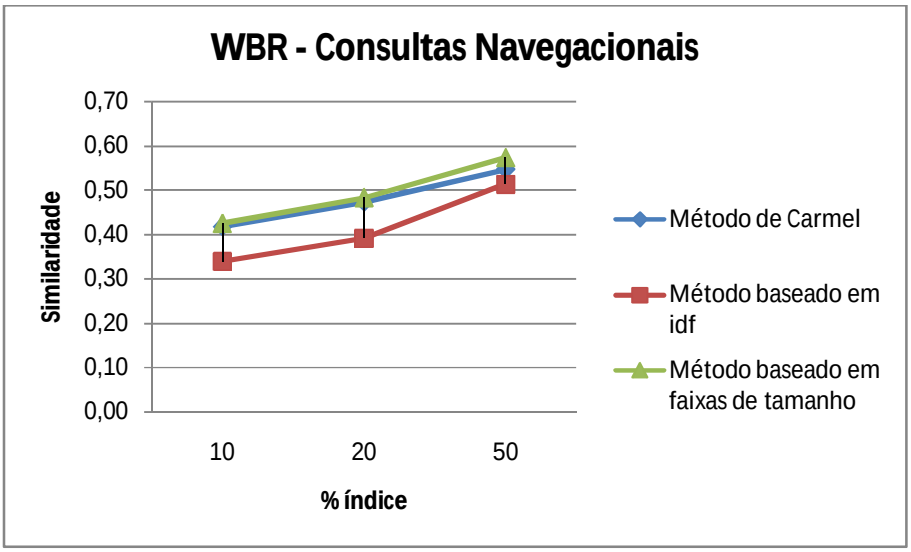

Figura 5.11- Consultas navegacionais na WBR: Similaridade média entre as listas do índice podado e íntegro, em função do tamanho final do índice.

Capítulo 6

Conclusão

Neste trabalho foram propostos e experimentados dois métodos de poda estática a fim alcançar melhorias nos resultados obtidos pelo método de Carmel e também para obter maior controle sobre os níveis de poda aplicados. Os novos métodos foram baseados na influência que características das coleções têm sobre os métodos de poda. As características consideradas foram o tamanho das coleções, a raridade dos termos do vocabulário e das consultas.

A primeira abordagem proposta, baseada em idf, quando aplicada a coleção Latimes apresentou resultados satisfatórios. Para consultas conjuntivas os valores de similaridade obtidos com 20% do índice em diante, são maiores que o método de Carmel, alcançando um ótimo ponto quando a taxa mantida é de 40%, onde a similaridade do método proposto é de 0,6190 e do método base é de apenas 0,3903. Para consultas disjuntivas, os resultados foram praticamente iguais em todos os níveis de poda aplicados, acima de 0.95 em ambos os métodos. Para consultas por frase os valores comportaram-se semelhantes ao que houve com as consultas conjuntivas, sendo os melhores resultados obtidos com consultas conjuntivas. Em coleções como a WBR, os resultados do método de poda baseado em ordenação por idf não foram melhores que o método de Carmel, devido à grande presença de consultas com termos comuns nessa coleção.

Esta abordagem é apropriada para coleções pequenas e convencionais que possuam grande incidência de termos raros nas consultas e um vocabulário com os tamanhos dos termos não concentrado em faixas de tamanho específicas.

Para tratar coleções com as características da WBR, foi proposto um método baseado

nos tamanhos de lista invertida que mais aparecem em um conjunto de consultas. Este método apresentou ganhos consistentes, com o comportamento semelhante nas duas versões da WBR. Com consultas disjuntivas os resultados foram praticamente iguais aos obtidos com o método base. Com consultas conjuntivas e disjuntivas os resultados obtidos foram melhores que com o método de Carmel em todos os níveis de poda aplicados. As consultas conjuntivas dominaram os resultados, alcançando similaridade de 0,85 com 50% do índice enquanto no método base a similaridade foi de 0,68.

Os dois métodos podem ser aplicados tanto a listas de freqüência quanto de posição, como visto nos resultados apresentados, e podem ser aplicados a coleções com as mesmas características das experimentadas.

Como trabalho futuro, seria interessante experimentar os métodos propostos utilizando outras evidências para ordenação das respostas tais como Pagerank.

Outro trabalho que pode ser realizado futuramente é experimentar os métodos propostos como entrada para o LBPM[7], um método de poda que realiza um refinamento no resultado do método de Carmel.

Referências Bibliográficas

[1] Baeza, Y Ricardo; Ribeiro, N Berthier. *Modern Infomation Retrieval*. Editora Addison Wesley. ACM Press, 1999.

[2] Chakrabarti, Samen. *Mining the Web: Discovering Knowledge for Hipertext Data*.Morgan Kaufmann Publishers, 2003.

[3] Stefan Büttcher; Charles L. A. Clarke. *Indexing Time vs. Query Time Tradeoffs in Dynamic Information Retrieval Systems.* Proceedings of the 14th ACM international conference on Information and knowledge management, 2005.

[4] Witten, Ian; Moffat, Alistair and Bell, Timothy. *Managing Gigabytes -Compressing and Indexing Documents and Images,* the second edition.Morgan Kaufmann Publishers, 1999.

[5] Zobel**,** Justin; Moffat, Alistair**.** *Inverted Files for Text Search Engines***.** The Australian Research Council and the Victorian Partnership for Advanced Computing have provided valuable financial support as have our two home institutions. ACM, 2006.

[6] D. Carmel, D. Cohen, R. Fagin, E. Farchi, M. Herscovici, Y. S. Maarek, and A. So_er. *Static Index Pruning for Information Retrieval Systems*. In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 43{50, New Orleans, Louisiana, USA, September 2001.

[7] Edleno S. de Moura, Célia F. dos Santos, Daniel R. Fernandes, Altigran S. Silva, Pavel Calado, and Mário A. Nascimento. *Improving web search eficiency via a locality based static pruning method*. In Proceedings of the 14th International World Wide Web Conference, pages 235-244, Chiba, Japan, Maio 2005.

[8] Edleno Silva de Moura, Celia Francisca dos Santos, Bruno dos Santos de Araujo, Altigran Soares da Silva, Pavel Calado, and Mario A. Nascimento. *Locality-based pruning methods for web search. ACM Trans. Inf. Syst., 26(2):128, 2008.*

[9] Célia Francisca dos Santos. *Métodos de Poda Estática para Índices de Máquinas de Busca*. Dissertação de mestrado, Universidade Federal do Amazonas, 2006.

[10] Bruno Araújo dos Santos, *Avaliação de localidade em métodos de poda de Índices para a Web*. Dissertação de mestrado, Universidade Federal do Amazonas, 2008.

[11] Sergey Brin, Lawrence Page. *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. In Proceedings of the 7th International World Wide Web Conference, pages 107-117, April 1998.

[12] Cho Jungo and Ntoulas Alexandros.*Pruning Policies for Two-Tiered Inverted Index with Correctness Guarantee* **.** In SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval.

[13] Michael Filtered, *Document Retrieval with Frequency-Sorted Indexes.* Department of Computer Science, RMIT, Persin

[14] A. Moffat and J. Zobel. *Self-indexing inverted files for fast text retrieval.*ACM Transactions on Information Systems, in press.

[15] G. Salton, A. Wong, and C. S. Yang. *A vector space model for automatic indexing*. Commun.ACM, 18(11):613-620, 1975.

[16]R. Fagin, R. Kumar, and D. Sivakumar. *Comparing top k lists*. In Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms, pages 28-36, 2003.

[17] D. Hawking, E. Voorhees, P. Bailey, and N. Craswell. *Overview of trec-8 web track*. In Proc. of TREC-8, pages 131-150, Gaithersburg MD, November 1999.

[18] Andrei Broder. A taxonomy of web search. SIGIR Forum, 36(2):3-10, 2002.

[19] Michael Persin. *Document filtering for fast ranking*. In Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval.Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum), pages 339-348, 1994.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo