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 Máquinas de Busca

Célia Francisca dos Santos

Manaus – Amazonas
Fevereiro de 2006

Célia Francisca dos Santos

Métodos de Poda Estática para Índices de Máquinas de Busca

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Departamento de Ciência da Computação da Universidade Federal do Amazonas, como requisito para obtenção do Título de Mestre em Informática.

Orientador: Prof. Dr. Edleno Silva de Moura

Célia Francisca dos Santos

Métodos de Poda Estática para Índices de Máquinas de Busca

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Departamento de Ciência da Computação da Universidade Federal do Amazonas, como requisito para obtenção do Título de Mestre em Informática.

Banca Examinadora

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

Prof. Dr. Altigran Soares da Silva
Departamento de Ciência da Computação – UFAM

Prof. Nivio Ziviani, Ph.D.
Departamento de Ciência da Computação – UFMG

Manaus – Amazonas
Fevereiro de 2006

A minha mãe que sempre tem sido minha principal motivadora.

Agradecimentos

Primeiramente a Deus por permitir que este momento se concretizasse.

À minha mãe, Maria Socorro, pelo apoio incondicional.

À professora Rosiane, pelo apoio e incentivo para a entrada no mestrado.

Ao Edleno, por acreditar no meu trabalho e pela paciência provida nestes dois anos.

Ao Daniel Fernandes, pelo material e orientações repassados.

Ao pessoal do GTI, pelo suporte aos experimentos.

Aos amigos Patrícia Peres, Ruan Belém, Márcio Vidal, Daniel Oliveira, Vanessa Mota e Bruno Gadelha pela força e apoio.

À Elienai Nogueira, pelo apoio administrativo.

A todos aqueles que contribuíram de alguma forma na realização deste trabalho, o meu mais profundo agradecimento.

Preciso de serenidade, para aceitar as coisas que não posso mudar. Coragem, para mudar o que posso. E sabedoria, para conhecer a diferença.

Reinhold Niebuhr

Resumo

Neste trabalho são propostos e avaliados experimentalmente novos métodos de poda estática especialmente projetados para máquinas de busca *web*. Os métodos levam em consideração a localidade de ocorrência dos termos nos documentos para realizar a poda em índices de máquinas de busca e, por esta razão, são chamados de "métodos de poda baseados em localidade". Quatro novos métodos de poda que utilizam informação de localidade são propostos aqui: *two-pass lbpm*, *full coverage*, *top fragments* e *random*.

O método *two-pass lbpm* é o mais efetivo dentre os métodos baseados em localidade, mas requer uma construção completa dos índices antes de realizar o processo de poda. Por outro lado, *full coverage*, *top fragments* e *random* são métodos *single-pass* que executam a poda dos índices sem requerer uma construção prévia dos índices originais. Os métodos *single-pass* são úteis para ambientes onde a base de documentos sofre alterações contínuas, como em máquinas de busca de grande escala desenvolvidas para a *web*.

Experimentos utilizando uma máquina de busca real mostram que os métodos propostos neste trabalho podem reduzir o custo de armazenamento dos índices em até 60%, enquanto mantém uma perda mínima de precisão. Mais importante, os resultados dos experimentos indicam que esta mesma redução de 60% no tamanho dos índices pode reduzir o tempo de processamento de consultas para quase 57% do tempo original.

Além disso, os experimentos mostram que, para consultas conjuntivas e frases, os métodos baseados em localidade produzem resultados melhores do que o método de Carmel, melhor método proposto na literatura. Por exemplo, utilizando apenas consultas com frases, com uma redução de 67% no tamanho dos índices, o método baseados em localidade *two-pass lbpm* produziu resultados com uma grau de similaridade de 0.71, em relação aos resultados obtidos com os índices originais, enquanto o método de Carmel produziu resultados com um grau de similaridade de apenas 0.39. Os resultados obtidos mostram que os métodos de poda baseados em localidade são mais efetivos em manter a qualidade dos resultados providos por máquinas de busca.

Palavras-chave: Recuperação de Informação, Máquina de Busca, Web, Poda em Índices.

Abstract

In this work, we propose and experimentally evaluate new static pruning methods targeted for web search engines. These methods take into account the locality of the occurrences of terms in the documents to perform pruning on search engine indices and, for this reason, are called "locality based pruning methods". *Two-pass lbpm*, *full coverage*, *top fragments* and *random* are the four locality based pruning methods proposed here.

Two-pass lbpm method is the most effective among the locality based pruning methods, but requires building a full index before the pruning process. On the other hand, full coverage, top fragments and random are single-pass methods that perform the index pruning without requiring the construction of their non-pruned version. The single-pass methods are useful for pruning in environments where the document database changes continuously, such as large scale web search engines.

Experiments using a real search engine show that our methods can reduce the indices' storage costs by up to 60%, while maintaining the loss in retrieval precision to a minimum. More importantly, our results indicate this same reduction of 60% on the index size can reduce the query processing time to almost 57% of the original time.

In addition, the experiments show that, for conjunctive and phrase queries, the locality based pruning methods produce better results than Carmel's method, the best method proposed in the literature. For instance, using only phrase queries, with a reduction of 67% in the index size, the two-pass lbpm method produced results with a similarity of 0.71, on relation to the results obtained with the original indices, while the Carmel's method produced results with a similarity of 0.39. The results obtained show the locality based pruning methods are more effective than Carmel's method for maintaining the quality of answer results provided by search engines.

Keywords: Information Retrieval, Search Engine, Web, Index Pruning.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Como o número de usuários interessados e capazes de utilizar a *World Wide Web* (*WWW*) cresce continuamente, a demanda por mecanismos de busca rápidos e precisos tornou-se maior do que nunca. Aliada ao aumento do número de usuários, a quantidade de informação disponível na *Web* continua crescendo de forma acelerada e contínua. Este cenário, de fato, não deve mudar, uma vez que usuários comuns, que eram apenas consumidores de informação, agora estão se tornando produtores ativos de conteúdo. Existe portanto uma necessidade constante de se melhorar a eficiência de máquinas de busca, sem que se comprometa a qualidade de seus resultados.

Máquinas de busca utilizam um conjunto de estruturas de dados chamadas índices [24] para permitir o processamento eficiente de consultas sobre documentos armazenados em suas bases de dados. Pelo menos dois tipos de índices são usualmente utilizados: o índice de freqüências e o índice posicional. O primeiro contém, para cada termo t, a freqüência com que t ocorre em cada documento, permitindo assim mensurar a importância relativa dos termos nos documentos indexados. O índice posicional contém, para cada termo t, informação sobre as posições de ocorrência de t em cada documento da base, o que permite o processamento de consultas posicionais como, por exemplo, consultas por frases. Em cada um destes índices a lista de entradas correspondentes a um dado termo, com freqüências ou posições de ocorrências, é chamada de *lista invertida*.

Em máquinas de busca tradicionais, os índices de freqüências e posicional ocupam quase a mesma quantidade de espaço de armazenamento requerida pela base de documentos. Além disso, quase todo o custo computacional associado ao processamento de consultas em máquinas de busca é devido ao tempo necessário para acessar os seus índices. Desta forma, uma redução

no tamanho dos índices reduziria não só custos com armazenamento, mas também melhoraria o desempenho da máquina de busca em termos de eficiência. Por isso, soluções para melhorar a eficiência e reduzir custos de armazenamento em máquinas de busca normalmente atuam sobre seus índices.

Uma das estratégias de maior sucesso para melhorar a eficiência de máquinas de busca é o uso de métodos de compressão de dados. Entretanto, métodos de compressão tradicionalmente utilizados em índices de máquinas de busca impõem limites quanto à taxa máxima de compressão que pode ser obtida, restringindo assim a melhoria de eficiência a ser alcançada [24]. Estes limites existem porque normalmente utiliza-se métodos de compressão sem perda, os quais não descartam informação durante o processo de compressão, permitindo assim a recuperação integral da informação após a descompressão. Técnicas alternativas para melhorar a eficiência de máquinas de busca têm sido propostas na literatura recente. Uma destas alternativas é o uso de métodos de poda [1, 7, 18]. Dentre estes métodos, têm-se os de poda estática, que são o foco deste trabalho. Métodos de poda estática podem ser vistos como técnicas de compressão com perda, uma vez que para reduzir o tamanho dos índices, tais métodos tentam identificar quais entradas dos índices podem ser removidas definitivamente, sem afetar drasticamente a qualidade dos resultados da máquina de busca.

Métodos de poda propostos e experimentados na literatura levam em consideração somente consultas disjuntivas, atuando portanto apenas sobre os índices de freqüências. Esta restrição permite que estes métodos possam predizer o impacto individual de cada termo na ordenação final das respostas de uma consulta. Entretanto, isto não é aplicável a máquinas de busca modernas, onde consultas conjuntivas e frases são não somente permitidas, mas também populares entre usuários [2, 5, 19, 23]. De fato, consultas conjuntivas e frases correspondem juntas a aproximadamente 99.4% das consultas com mais de um termo que eram submetidas à máquina de busca TodoBR[1]. Estes tipos de consultas podem ter um impacto significativo no projeto e desempenho dos métodos de poda. Por exemplo, para alcançar uma redução significativa no tempo para processar uma frase, é necessário remover entradas não somente do índice de freqüências, mas também do índice posicional. Este tipo de índice é o mais caro de se manter e utilizar. Além disso, métodos de poda propostos anteriormente não estão preparados para serem aplicados a índices posicionais, pois realizam a poda com base apenas nas informações individuais de cada

[1] Máquina de busca brasileira, disponibilizada em http://www.todobr.com.br/, existente até 2005.

termo, não levando em consideração relações existentes entre os mesmos. O que significa que tais métodos podem falhar em preservar resultados importantes para consultas conjuntivas e frases.

Diante do cenário apresentado, o objetivo deste trabalho é pesquisar e desenvolver métodos de poda que possam ser utilizados por máquinas de busca *Web* para melhorar sua eficiência em relação ao tempo de processamento de consultas e consumo de recursos, sem que haja perdas significativas na qualidade dos resultados. Para tal, é necessário que se leve em consideração os tipos comuns de consultas e características do ambiente *Web*. Com base neste objetivo, propõe-se aqui uma família de métodos, os quais foram chamados de *Métodos de Poda Baseados em Localidade*. Estes métodos visam predizer que conjuntos de termos de cada documento podem ocorrer juntos em consultas e utilizar esta informação para preservar, para cada documento, as entradas que representem a ocorrência desses conjuntos de termos. Cada um dos métodos desta família utiliza uma heurística diferente para predizer que conjuntos de termos devem ser preservados para cada documento.

A família de métodos de poda baseados em localidade proposta é composta de quatro integrantes: *two-pass lbpm*, *full coverage*, *random* e *top fragments*. O *two-pass lbpm* é um método de duas passagens (*two-pass*), isto é, requer que o processo de poda seja executado após a realização de uma indexação completa da base de documentos. Esta característica tem duas principais desvantagens: primeiro, aumenta o tempo necessário para construir os índices e, segundo, impede que o processo de atualização da base de dados seja realizado diretamente sobre os índices podados. Por outro lado, os métodos *full coverage*, *random* e *top fragments* são métodos de uma única passagem (*single-pass*), ou seja, o processo de poda é executado simultaneamente à construção ou atualização dos índices. Tal fato faz com que o tempo total de construção dos índices com poda utilizando estes métodos seja menor do que o tempo necessário para construir os índices originais. Além disso, os métodos *single-pass* permitem atualização direta dos índices podados, evitando a necessidade de reconstruir os índices toda vez que uma nova informação é adicionada. Estes métodos são especialmente úteis para podar índices em ambientes onde a base de documentos muda continuamente, como em máquinas de busca de larga escala.

Para dar suporte à avaliação dos métodos aqui propostos, foram realizados experimentos extensivos comparando os métodos de poda baseados em localidade com o melhor método de poda proposto previamente na literatura, aqui referenciado como método de Carmel [7]. Curvas de precisão/revocação e similaridade entre os resultados produzidos por índices podados e índices

originais foram utilizadas como forma de mensurar o desempenho dos métodos de poda. Os experimentos mostram que, para consultas conjuntivas e frases, os métodos baseados em localidade produzem resultados melhores do que o método de Carmel. Por exemplo, utilizando apenas consultas contendo frases, com uma redução de 67% no tamanho dos índices, o método baseado em localidade *two-pass lbpm* produziu resultados com uma grau de similaridade de 0.71, em relação aos resultados obtidos com os índices originais, enquanto o método de Carmel produziu resultados com um grau de similaridade de apenas de 0.39. Para este mesmo experimento, o método baseado em localidade *random* produziu resultados com uma grau de similaridade de 0.68, significando uma diferença de somente 3% em relação ao método *two-pass lbpm*.

Além disso, outros experimentos realizados mostraram que uma redução de 67% no tamanho dos índices produzida pelo método *two pass-lbpm*, pode fazer com que se reduza em 48% o tempo necessário para processar consultas sem perda de qualidade nas respostas. Isso mostra que os métodos de poda propostos podem conseguir uma considerável redução no tempo de processamento de consultas, sem comprometer gravemente a qualidade dos resultados produzidos pela máquina de busca.

1.1 Trabalhos relacionados

Uma abordagem bem-sucedida para reduzir o tamanho de índices de máquinas de busca Web é o uso de técnicas de compressão de dados. Estas podem ser técnicas *sem perda*, onde nenhuma informação é descartada, e técnicas *com perda*, onde o tamanho do índice é reduzido pelo descarte de informação que é considerada inútil para o processamento de consultas. Métodos sem perda têm sido extensivamente estudados e experimentados na literatura [1, 3, 9, 16, 24] e podem ser considerados como seguros, uma vez que os arquivos originais podem ser obtidos facilmente a partir da sua versão comprimida. Tais métodos têm a desvantagem, entretanto, de imporem limites quanto à taxa máxima de compressão que pode ser obtida, o que representa um limite prático na melhoria da eficiência e redução de espaço de armazenamento a serem alcançados.

Técnicas de compressão com perda são uma alternativa interessante para superar os limites impostos pelas técnicas sem perda. Exemplos superficiais de tais técnicas são remoção de *stopwords*[2], onde se remove do índice entradas relacionadas a termos muito comuns, e Indexação

[2]Termos que ocorrem freqüentemente no texto de um documento, tais como artigos, preposições e conjunções.

Semântica Latente (LSI) [10]. Nesta última, os documentos são representados por conceitos artificiais independentes que substituem muitos termos individuais que são relacionados (por exemplo, sinônimos). Sendo assim, a maior parte da redundância na caracterização dos documentos, devido o uso de termos relacionados semanticamente, é removido, resultando em uma versão eficientemente comprimida da base de dados.

Embora estes métodos tenham sido propostos originalmente para remover ruído dos índices, estes podem ser vistos como técnicas de compressão com perda, uma vez que eles também reduzem o tamanho dos índices. Entretanto, estes dois métodos possuem desvantagens. LSI envolve um alto custo computacional, fazendo sua aplicação inviável para máquinas de busca que lidam com coleções de tamanho não trivial. Remoção de *stop-words* tipicamente ajuda na melhoria do desempenho da busca, mas é uma técnica de compressão limitada, uma vez que remove somente listas invertidas de um conjunto restrito de termos e é freqüentemente dependente de domínio.

Uma alternativa de compressão sem perda mais efetiva para índices de máquinas de busca é o uso de métodos de poda [1, 7, 18]. A idéia de tais métodos é remover dos índices entradas cuja potencial contribuição para a pontuação de relevância dos documentos nos quais ocorrem é tão pequena, que a remoção destas entradas terá pouco impacto na ordenação final dos documentos. O impacto esperado desta remoção é uma notável redução nos requisitos de armazenamento, operações de entrada/saída e custo computacional.

As abordagens de poda encontradas na literatura podem ser classificadas em duas categorias: dinâmica [1, 18] e estática [7]. Na abordagem dinâmica, a poda é executada durante o tempo de processamento de consultas. Este fato permite que o número de entradas dos índices a serem processadas varie de acordo com a consulta do usuário, o que representa uma vantagem, já que a poda pode ser melhor adaptada a cada consulta específica. Por outro lado, na abordagem estática a poda é executada durante o tempo de construção dos índices. Neste caso, a poda é executada diretamente nos índices, criando uma nova versão destes. Por esta razão, métodos de poda estática são vistos como métodos de compressão com perda, reduzindo não só o tempo de processamento de consultas, mas também custos com espaço de armazenamento.

O método de poda proposto por Persin et al. [18] é um exemplo de método que realiza poda dinâmica e atua apenas sobre o índice de freqüências. Nesta abordagem, os termos de uma consulta devem ser processados em ordem decrescente de *idf* (*inverse document frequency*) [21]

e as listas invertidas devem ter suas entradas ordenadas por freqüência. Para cada termo da consulta a ser processado, são computados limiares de freqüência que indicam que entradas da lista invertida de cada termo deverão ser processadas.

O método proposto por Carmel et al. [7] é um método de poda estática que atua apenas sobre o índice de freqüências. Tal método será referenciado como *método de Carmel* e utilizado como base de comparação entre os métodos aqui propostos. O método de Carmel faz uso do critério de ordenação de respostas da máquina de busca para computar a importância de cada entrada da lista invertida de um termo e assim poder determinar quais entradas podem ser removidas. Esta informação é obtida submetendo para a máquina de busca cada um dos termos do vocabulário da coleção como uma consulta. Para um termo t, a lista de documentos resultantes, denominada por R_t, contém os documentos relacionados a t dispostos em ordem decrescente de importância. A importância de cada documento é definida de acordo com o critério de ordenação de respostas da máquina de busca. A porção superior de R_t, denominada $R_t(\delta)$, é determinada usando o limiar $\delta - top$. Este limiar é computado com base no valor do parâmetro δ e no valor de importância do documento topo de R_t. O valor de δ é definido no intervalo $[0;1]$ e escolhido através de experimentos. O documento topo de R_t corresponde ao documento de maior importância em R_t. O limiar $\delta - top$ preserva em $R_t(\delta)$ somente os documentos de R_t que possuem uma importância cujo valor é pelo menos δ vezes o valor de importância do documento topo de R_t. Por exemplo, se $\delta = 0.7$, todo documento em $R_t(\delta)$ deve ter um valor de importância que corresponde no mínimo a 70% do valor de importância do documento topo de R_t. O processo de poda é guiado pela porção superior de R_t, ou seja, $R_t(\delta)$. Durante o processo de poda, uma entrada do índice de freqüências, que representa a ocorrência de um termo t em um documento D, é removida se D não está presente em $R_t(\delta)$.

Uma outra alternativa para reduzir o tamanho de índices de máquinas de busca é utilizar representações parciais de documentos. Isto pode ser realizado por métodos de geração de resumos, que consistem em condensar um texto preservando seu conteúdo de informação e mantendo a sua legibilidade. Alguns métodos de geração de resumo na literatura não focam na legibilidade, mas sim em não perder informação contida nos documentos [12, 15, 17, 20].

Estes métodos podem ser facilmente adaptados para atuar como métodos de poda estática, uma vez que a remoção de algumas entradas das listas invertidas é uma conseqüência natural da indexação de resumos em vez do documento original. Entretanto, existe uma pequena, mas

importante, diferença entre os métodos de poda e os de geração de resumos. Os métodos de poda removem entradas de listas invertidas, mantendo informações estatísticas originais sobre a base de documentos. Por outro lado, quando se trabalha com métodos de geração de resumos, é comum não se usar qualquer informação estatística obtida da base original de documentos. Como conseqüência, informações importantes tais como freqüência, posição de ocorrência e o valor de *idf* [21] originais dos termos são perdidas durante o processo de geração de resumos. Este tipo de informação é importante para a obtenção de resultados mais próximos a aqueles obtidos com a base de documentos original.

Dentre os métodos de geração de resumos que objetivam não perder informação contida nos documentos, tem-se uma abordagem não supervisionada, proposta por Nomoto e Matsumoto [17], que é baseada na diversidade de conceitos de um documento. Esta abordagem consiste de duas etapas: encontrar diversidade e reduzir redundância. Na primeira etapa são encontradas áreas de tópicos diversos no documento utilizando o algoritmo de classificação *X-Means*, que é uma variação do *K-Means* com cálculo prévio do número de classes. Na segunda etapa, seleciona-se a sentença mais importante de cada área de tópico, isto é, a sentença mais representativa de cada classe. O resumo de um documento é criado a partir de suas sentenças selecionadas.

O método de geração de resumos *Full Coverage*, proposto por Mallett et al. [15], é outro método que tenta não perder informação contida nos documentos ao gerar resumos. Neste caso, o objetivo é gerar resumos de documentos para máquinas de busca, tentando preservar a utilidade destes para os usuários da máquina. Esta abordagem calcula a similaridade entre cada sentença de um documento e seu conteúdo inteiro. A cada iteração, a sentença com maior valor de similaridade é selecionada e os termos presentes na mesma são excluídos dos vetores que representam o documento e as sentenças ainda não selecionadas. No cálculo da similaridade, os valores de *idf* dos termos dizem respeito apenas a importância dos mesmos para o documento e não para a coleção. Na verdade, ao aplicar este método cada sentença passa a ser enxergada como um documento e o seu conjunto forma uma coleção. O resumo de um documento é gerado a partir das suas sentenças selecionadas. Nesta dissertação, este método foi adaptado para trabalhar como método de poda, que corresponde a um dos métodos baseados em localidade.

Fernandes [12] propôs o uso da geração de resumos como estratégia para melhorar o desempenho de máquinas de busca. A abordagem de Fernandes gera o resumo de cada documento da coleção utilizando os termos mais importantes deste. Tais termos são obtidos utilizando o

método de Carmel. Esta idéia foi o ponto de partida para o método baseado em localidade *two-pass lbpm*.

1.2 Contribuições

As principais contribuições deste trabalho são:

- A proposta e implementação de métodos de poda que possam ser utilizados por máquinas de busca *Web* para melhorar sua eficiência, sem perdas significativas na qualidade dos resultados [8].
- Um estudo comparativo entre os métodos propostos e os existentes na literatura.
- Implementação de um método de poda encontrado na literatura [7].

1.3 Organização da dissertação

Esta dissertação está organizada em cinco capítulos, dos quais este é o primeiro. No Capítulo 2 são apresentados os principais conceitos necessários à compreensão deste trabalho. O Capítulo 3 descreve a estrutura geral dos métodos de poda propostos e detalha o funcionamento de cada um destes métodos: *two-pass lbpm*, *full coverage*, *top fragments* e *random*. O Capítulo 4 apresenta os experimentos realizados e discute os resultados obtidos. Informações sobre coleções de teste, estratégias de ordenação de respostas, métricas de avaliação de desempenho e tipos de consultas utilizadas nos experimentos também são apresentados. Finalmente, o Capítulo 5 apresenta o resumo dos resultados e as conclusões obtidas, discute as contribuições do trabalho desenvolvido e apresenta sugestões de trabalhos futuros.

Capítulo 2

Conceitos Básicos

Neste capítulo serão apresentados e discutidos de forma sucinta alguns conceitos básicos necessários à compreensão dos experimentos e métodos propostos neste trabalho.

2.1 Máquinas de Busca

O crescimento contínuo do volume de informação e do número de usuários da World Wide Web (WWW) criou a necessidade de se disponibilizar mecanismos de busca de informação eficientes. Máquinas de busca são mecanismos de busca que coletam continuamente documentos disponíveis na Web, armazenando-os em uma base de dados local e disponibilizando-os para que o usuário final possa realizar consultas sobre tais documentos. Uma máquina de busca *web* é composta basicamente de três elementos principais: o coletor de páginas, o indexador e o processador de consultas [4].

O coletor de páginas, também conhecido como *crawler*, *spider* ou *robot*, é o elemento responsável por atualizar a base de documentos da máquina de busca. Ele trabalha visitando e coletando páginas *web* que serão armazenadas localmente na base de dados da máquina de busca. A atualização que o coletor faz sobre a base de documentos não se restringe somente à coleta de novas páginas, mas também a atualização das páginas já coletadas, se estas tiverem sofrido qualquer alteração de conteúdo. Para atualizar as páginas já existentes na base, o coletor não pode simplesmente decidir re-coletar todas as páginas de uma vez, devido o volume de páginas ser muito grande e cada página ter o seu próprio ciclo de alterações. Enquanto algumas poucas páginas sofrem alterações quase que diariamente, outras páginas ficam meses sem sofrer

qualquer mudança. Sendo assim, o coletor de páginas é um dos elementos mais complexos da máquina de busca.

Uma vez que as páginas *web* foram coletadas, é necessário que estas sejam adicionadas a base de documentos, fazendo parte das estruturas de dados utilizadas pelas máquinas de busca durante o processamento de consultas. Nesta etapa, o indexador de documentos entra em operação. O indexador é responsável por processar cada documento *web*, extraindo os termos (palavras) encontrados nos mesmos para adicioná-los ao vocabulário da base de dados. Outras informações extraídas são a freqüência com que cada termo ocorre nos documentos e a posição em que estes ocorrem. Estas informações são armazenadas em estruturas de dados chamadas índices [24]. Dois tipos de índices são usualmente utilizados: o índice de freqüências e o índice posicional. O índice de freqüências mantém a freqüência com que cada termo t, do vocabulário, ocorre em cada documento da base e, o índice posicional mantém informação sobre as posições de ocorrência dos termos nos documentos. Em cada um destes índices, a lista de entradas de freqüências ou posições de ocorrências para cada termo t é chamada de lista invertida.

Após a indexação dos documentos *web*, entra em operação o processador de consultas. Este é responsável por verificar quais documentos da base de dados atendem uma determinada consulta, e apresentá-los ao usuário final de acordo com uma determina ordem de importância. Com base no tipo de consulta submetida ao sistema e nos termos presentes na mesma, o processador de consultas define que tipos de índices serão processados (freqüência ou posicional) e que termos terão suas listas invertidas processadas. De posse das listas invertidas, o processador tem como descobrir que documentos podem responder a consulta submetida ao sistema. Além disso, o processador de consultas utiliza métricas que definam a ordem em que os documentos resultantes devem ser apresentados ao usuário final. Uma das métricas mais conhecidas e utilizada é o tradicional modelo vetorial [21].

Embora as máquinas de busca sejam, em geral, compostas destes três elementos, cada máquina de busca tem suas particularidades com relação à forma de mensurar a importância de um documento para uma consulta. Além disso, as máquinas se diferenciam com relação ao tamanho de suas bases de dados e a freqüência com que estas são atualizadas.

2.2 Modelo Vetorial

O modelo vetorial é um clássico da área de recuperação de informação, que visa recuperar informação de forma simples e eficiente em coleções de documentos [21]. Neste modelo, documentos e consultas são representados como vetores em um espaço de dimensão igual ao número de termos do vocabulário da coleção [24].

No modelo vetorial, um documento ou consulta é representado por um vetor onde cada coordenada representa a importância de um termo do vocabulário da coleção para o documento ou consulta em questão. Desta forma, o documento d_j e a consulta q são representados, respectivamente, pelos vetores $\vec{d_j} =< w_{1j};\ w_{2j};\ \ldots;\ w_{kj} >$ e $\vec{q} =< w_{1q};\ w_{2q};\ \ldots;\ w_{kq} >$, onde k é o número de termos no vocabulário da coleção e w_{ix} representa a importância do termo t_i para o documento ou consulta x. O peso w_{ix} pode ser computado da seguinte forma:

$$w_{ix} = tf_{ix} \times idf_i \tag{2.1}$$

onde tf_{ix} (*term frequency*) corresponde à freqüência com que o termo t_i ocorre no documento ou consulta x e idf_i (*inverse document frequency*) corresponde à importância do termo t_i para a coleção de documentos. O valor do idf_i é dado por:

$$idf_i = log\frac{N}{n_i} \tag{2.2}$$

onde N é o número de documentos da coleção e n_i é o número de documentos onde ocorre o termo t_i.

Neste modelo, documentos e consultas podem ser comparados utilizando qualquer medida de relação entre vetores. A medida mais utilizada é a similaridade entre cossenos (*cosine similarity*), isto é, o valor de cosseno do ângulo formado entre os vetores do documento e da consulta. Desta forma, a similaridade entre o documento d_j e a consulta q é definido da seguinte maneira:

$$sim(d_j, q) = \frac{\sum_{i=1}^{k} w_{ij} \times w_{iq}}{\sqrt{\sum_{i=1}^{k} w_{ij}^2} \times \sqrt{\sum_{i=1}^{k} w_{iq}^2}} \tag{2.3}$$

Na Figura 2.1, é exibido um exemplo de espaço vetorial de duas dimensões (termos k_a e k_b), composto por um documento d_j e a consulta q. A similaridade entre d_j e q é o cosseno de θ.

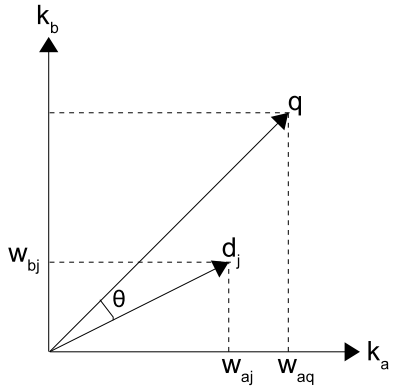


Figura 2.1: Similaridade entre um documento d_j e uma consulta q, representada no espaço vetorial.

2.3 Poda em Índices

Em máquinas de busca de larga escala, o custo para processar consultas costuma ser bastante elevado. Isso se deve principalmente ao tempo necessário para acessar os índices da máquina de busca, isto é, tempo para acessar a lista invertida de cada termo presente em uma consulta. O custo de acesso à lista invertida de um termo cresce à medida que este se encontra presente em um número maior de documentos.

Uma redução no número de entradas a serem lidas de cada lista invertida poderia reduzir consideravelmente o tempo de processamento de consultas. Tal redução pode ser obtida utilizando-se os chamados métodos de poda em índices. Tais métodos consistem em remover, das listas invertidas dos termos, entradas cuja remoção supostamente afetaria muito pouco a qualidade dos resultados providos por máquinas de busca.

Existem basicamente duas abordagens de poda: a dinâmica [1, 18] e a estática [7]. A principal diferença entre as duas abordagens está relacionado ao momento em que a poda é realizada. Na abordagem dinâmica, os índices originais permanecem armazenados em disco e a poda é executada durante o tempo de processamento das consultas, o que permite que a poda possa ser melhor adaptada a cada consulta específica. Na abordagem estática, a poda é executada durante o tempo de construção do índice, onde se remove as entradas que supostamente poderiam não ser úteis durante o tempo de processamento de consultas. Os métodos de poda estática têm a vantagem de reduzir também o espaço de armazenamento em disco visto que a poda é executada diretamente nos índices, criando uma nova versão destes.

2.4 Métricas de Avaliação

Para estudar o efeito dos métodos de poda na qualidade dos resultados gerados pela máquina de busca, foram utilizadas duas métricas distintas: similaridade entre lista de respostas, utilizando uma variação do método *Kendall's tau*, e curvas de precisão e revocação. Estas duas medidas são descritas a seguir.

2.4.1 Similaridade de *Kendall's tau*

A similaridade (distância) entre duas listas de respostas para uma dada consulta pode ser computada utilizando-se uma variação do método *Kendall´s tau* proposta por Fagin [11]. Tal variação compara os k primeiros elementos de duas listas de respostas e produz uma pontuação que pode dizer o quão similar são estas duas listas.

A variação do método *Kendall´s tau* proposta por Fagin [11] atribui uma penalidade $S(i, j)$, para cada par $\{i, j\}$, onde i e j são documentos que aparecem entre as k primeiras entradas de pelo menos uma das duas listas que estão sendo avaliadas, como definida a seguir:

- Caso 1: i e j aparecem entre as k primeiras entradas de ambas as listas. Neste caso, se i e j estão na mesma ordem (por exemplo, i aparece antes de j em ambas as listas), então $S(i, j) = 0$, caso contrário $S(i, j) = 1$.

- Caso 2: i e j aparecem entre as k primeiras entradas da lista 1, mas somente um destes, i ou j, aparece entre as k primeiras entradas da lista 2. Assuma que i aparece entre as k primeiras entradas da lista 2. Se i aparece primeiro do que j na lista 1, então $S(i, j) = 0$, caso contrário $S(i, j) = 1$.

- Caso 3: se somente i aparece entre as k primeiras entradas da lista 1 e somente j aparece entre as k primeiras entradas da lista 2, então $S(i, j) = 1$.

- Caso 4: se i e j aparecem entre as k primeiras entradas da lista 1, e nenhum destes, nem i ou j, aparece entre as k primeiras entradas da lista 2, então $S(i, j) = p$.

Neste trabalho, decidiu-se atribuir para p uma pontuação de penalidade neutra de $1/2$ assim como fora feito por Carmel et al. [7]. Sendo assim, a soma total das penalidades recebidas por todos os pares $\{i, j\}$ rende uma pontuação entre 0, quando as k primeiras entradas das duas

listas são idênticas, e $k(3k-1)/2$, quando as k primeiras entradas das listas são completamente diferentes. Os resultados são normalizados utilizando-se a formula $x' = 1 - \frac{2x}{k(3k-1)}$, como explicado por Carmel et al. [7], onde x é o valor não normalizado de *Kendall's tau*. Portanto, a comparação final entre duas listas, varia de 0, representando listas completamente diferentes, a 1, representando listas iguais. Quanto maior o valor de x', mais similares são as listas comparadas.

2.4.2 Precisão e Revocação

Precisão e revocação são duas medidas comumente utilizadas na avaliação de sistemas de Recuperação de Informação (RI). Neste trabalho, estas medidas são utilizadas para gerar as curvas de precisão e revocação obtidas por cada método de poda.

Dada uma consulta Q, um conjunto R de documentos relevantes para Q e um conjunto A de documentos retornados como resposta para Q por um dado sistema de RI, compara-se o conjunto A com o conjunto R, calculando-se a interseção entre estes. Quanto maior for a interseção, melhor será o resultado. Com base nisto, tem-se que:

Precisão é a porcentagem de documentos em A, recuperados pelo sistema de RI, que são relevantes, ou seja:

$$\text{Precisão} = \frac{|R \cap A|}{|R|}$$

Revocação é a porcentagem de documentos relevantes, que estão em R, e foram recuperados pelo sistema de RI, ou seja:

$$\text{Revocação} = \frac{|R \cap A|}{|A|}$$

Para obter curvas de precisão e revocação para um consulta, é necessário que se calcule o valor de precisão de cada um dos onze pontos padrão de revocação: 0%, 10%, 20%, ..., e 100%. Como em geral utiliza-se mais de uma consulta para avaliar o desempenho de um sistema, é necessário que ao final seja calculada a média aritmética do valor de precisão para cada um dos onze pontos de revocação, de forma a gerar apenas um gráfico de precisão e revocação para cada sistema.

Capítulo 3

Métodos de Poda Baseados em Localidade

Métodos de poda visam remover entradas dos índices de máquinas de busca sem comprometer a qualidade dos resultados providos pelas mesmas. Em particular, alguns tipos de consultas tipicamente submetidas para máquinas de busca, tais como consultas conjuntivas e frases, requerem que métodos de poda preservem entradas nos índices para documentos que ocorrem ao mesmo tempo em listas invertidas de diferentes termos, ou seja, documentos onde dois ou mais termos da consulta ocorrem próximos no texto. Visto que os métodos existentes na literatura analisam as entradas de cada termo individualmente para realizar a poda, os índices resultantes podem não possuir essa propriedade. Nesta situação, documentos importantes, antes retornados como resultado de uma dada consulta, podem não ser mais retornados quando esta mesma consulta for realizada nos índices podados.

Tal problema é ilustrado na Figura 3.1, onde a lista invertida do termo A começa com os documentos 3, 15, 1, 8 e 14, e termina com os documentos 2, 31, 4 e 13. A lista invertida do termo B começa com os documentos 2, 1, 7, 9 e 43, e termina com os documentos 25, 16, 3 e 21. Supondo que as áreas sombreadas correspondam às porções podadas de ambas as listas, uma consulta após a poda requerendo documentos que contenham ambos os termos A e B não incluiria os documentos 2 e 3. Entretanto, estes documentos poderiam estar originalmente na resposta antes da poda, podendo inclusive serem documentos importantes para a consulta. Este exemplo mostra que soluções de poda que não levem em consideração relações entre os termos podem falhar em preservar resultados importantes para consultas conjuntivas e frases.

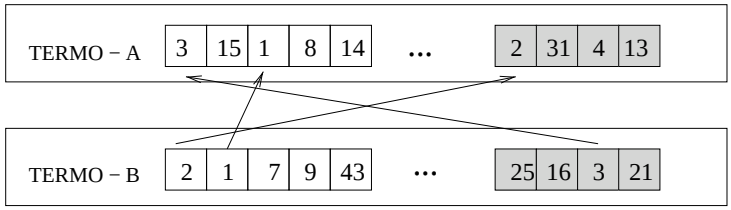


Figura 3.1: Possível perda de informação causada pela poda de entradas de cada termo individualmente.

Baseado nesta observação, propõe-se neste trabalho uma família de métodos que predizem quais conjuntos de termos podem ocorrer juntos em consultas e utilizam esta informação para preservar documentos comuns em listas invertidas de diferentes termos. Para predizer tais conjuntos de termos, os métodos propostos analisam individualmente cada documento da base. O algoritmo base dos métodos propostos é exibido na Figura 3.2. Neste algoritmo, cada documento é visualizado como um conjunto de fragmentos. Tais fragmentos podem ser simples passagens de texto com um número exato de termos ou ainda fragmentos que representem unidades lógicas como, por exemplo, sentenças ou parágrafos. Neste trabalho, optou-se pelo tipo de fragmento que representasse uma aproximação de sentenças em linguagem natural. As sentenças são extraídas através da pontuação encontrada em cada documento. Sinais de pontuação como “.", “?", “!"e “;" são interpretados como delimitadores de sentenças. Estudos comparativos quanto ao melhor tipo de fragmento a ser utilizado, ou seja, o impacto que o tipo de fragmento pode ter sobre o desempenho de um método de poda, foram deixados para trabalhos futuros.

Sendo assim, o algoritmo base da Figura 3.2 recebe basicamente dois parâmetros de entrada: $S(D)$ (conjunto de fragmentos extraídos do documento D) e p (nível de poda desejado). O nível de poda p é utilizado para determinar o número de entradas a serem removidas dos índices, o que define conseqüentemente o tamanho final dos índices após o processo de poda. Por exemplo, um nível de poda $p = 0.3$ indica que se deseja uma remoção de 30% das entradas existentes em cada índice da máquina de busca, permitindo que ao final cada índice possua apenas 70% do seu tamanho original. De fato, o parâmetro p é utilizado para controlar o número de fragmentos selecionados de cada documento, uma vez que o algoritmo realiza, na verdade, uma poda no conteúdo dos documentos, para então realizar a poda nos índices.

O algoritmo base da Figura 3.2 é dividido em três fases distintas: seleção de fragmentos, poda do índice de freqüências e poda do índice posicional. A primeira fase do algoritmo seleciona fragmentos do documento D até que não existam mais fragmentos a serem selecionados (isto é, $S(D)$ é

vazio), ou até que o tamanho, em número de termos, de todos os fragmentos selecionados alcance o percentual $1 - p$ do tamanho de D. Cada iteração do laço de repetição adiciona ao conjunto de fragmentos significativos, $S'(D)$, o fragmento $S_{importante}$ de $S(D)$ que é o fragmento mais importante de acordo com a heurística utilizada pela função $FragmentoMaisImportante$. A variável *tamanho* é então incrementada com o tamanho do fragmento $S_{importante}$, que é removido de $S(D)$.

```
Algoritmo Base de Poda
início
    entrada: D - documento
             S(D) - conjunto de fragmentos extraídos do documento D
             p - nível (ou taxa) de poda desejado
    variáveis: S'(D) - conjunto de fragmentos selecionados para D
               S_importante - fragmento mais importante de S(D)
               tamanho - tamanho total dos fragmentos em S'(D)
               termo - termo do vocabulário
               freq - freqüência de um termo no documento
               pos - posição de ocorrência de um termo no documento

    /* Seleção de fragmentos */
    S'(D) ← ∅;
    tamanho ← 0;
    Inicializa();
    repita
        S_importante ← FragmentoMaisImportante(S(D));
        S'(D) ← S'(D) ∪ S_importante;
        tamanho ← tamanho + |S_importante|;
        S(D) ← S(D) − S_importante;
    até (tamanho ≥ ((1 − p) × |D|)) ou (S(D) = ∅) ;

    /* Poda do índice de freqüências */
    para cada ⟨termo, freq⟩ ∈ D faça
        se termo ∉ s, ∀s ∈ S'(D) então
            remove ⟨termo, freq⟩ do índice de freqüências;
        fim
    fim

    /* Poda do índice posicional */
    para cada s ∈ S(D) faça
        se ⟨termo, pos⟩ ∈ s então
            remove ⟨termo, pos⟩ do índice posicional;
        fim
    fim
fim
```

Figura 3.2: Algoritmo base de poda baseada em localidade.

A função $FragmentoMaisImportante$ é o núcleo dos métodos propostos neste trabalho e determina a diferença entre eles. Cada um dos métodos de poda propostos segue o al-

goritmo base exibido na Figura 3.2 utilizando uma heurística diferente para mensurar a importância dos fragmentos. Como cada heurística pode trabalhar com uma informação diferente sobre os fragmentos, tem-se também um procedimento auxiliar chamado *Inicializa* que é usado para declarar e inicializar variáveis necessárias para o funcionamento correto da função $FragmentoMaisImportante$.

Uma vez que os fragmentos significativos foram selecionados, o algoritmo inicia a fase de poda, primeiro para o índice de freqüências e depois para o índice posicional. Para o índice de freqüências, a entrada que representa a freqüência de um termo t em um documento D é preservada somente se t ocorrer em pelo menos um dos fragmentos em $S'(D)$, que corresponde aos fragmentos selecionados para D. Todas as entradas restantes são removidas. No índice posicional, somente as ocorrências de termos em fragmentos selecionados são preservadas, as restantes, remanescentes em $S(D)$, são removidas.

A família de métodos proposta neste trabalho foi dividida em duas categorias: *single-pass* e *two-pass*. Os métodos *single-pass* são aqueles que não necessitam criar o índice duas vezes, ou seja, o índice podado é criado ao mesmo tempo em que se indexa a coleção, removendo entradas antes de armazená-las em disco. Em contraste, os métodos *two-pass* usualmente utilizam informações obtidas a partir de uma indexação completa prévia para guiar o processo de poda. Portanto, nos métodos *two-pass*, a base de documentos é indexada duas vezes: uma primeira vez para obter um índice completo e outra para executar o processo de poda. Claramente, os métodos *single-pass* tendem a ser mais eficientes e mais facilmente ajustados para trabalhar como métodos de poda incrementais do que os de métodos *two-pass*. A possibilidade de se utilizar métodos incrementais reduz o custo de atualização da base de dados e índices das máquinas de busca.

Nas subseções seguintes, serão apresentados quatro métodos de poda baseados em localidade, sendo um método *two-pass*, chamado *two-pass lbpm*, e três métodos *single-pass*: *full coverage*, *top fragments* e *random*. Todos estes métodos seguem o algoritmo base descrito na Figura 3.2. A principal diferença entre eles reside na fase de seleção de fragmentos que é executada pela função $FragmentoMaisImportante$. Além disso, a função *Inicializa* também pode vir a ter alguma diferença em decorrência de informações utilizadas por cada função $FragmentoMaisImportante$ para mensurar a importância dos fragmentos. Os métodos de poda baseados em localidade são detalhados a seguir.

3.1 Two-pass LBPM

O método *two-pass lbpm* inicia seu processamento utilizando o método de Carmel, que foi apresentado na Seção 1.1 do Capítulo 1, para determinar que ocorrências de termos são individualmente importantes para cada documento. Esta informação é então usada para ordenar os fragmentos de texto presentes no documento e determinar os fragmentos mais significativos. Mais especificamente, *two-pass lbpm* determina os termos importantes para cada documento D computando, para cada termo t do vocabulário $\mathcal{V}$, a lista ordenada $R_t(\delta)$, exatamente como faz o método de Carmel. Sendo assim, os termos importantes de um documento D ficam definidos da seguinte maneira:

$$T(D) = \{t | D \in R_t(\delta), \forall t \in \mathcal{V}\}$$

Os termos importantes de um documento D, denotados por $T(D)$, são todos os termos t cuja lista $R_t(\delta)$ contém D.

O método *two-pass lbpm* trabalha de acordo com o algoritmo base exibido na Figura 3.2, utilizando as funções $Iniciliza$ e $FragmentoMaisImportante$ da forma como estão detalhadas na Figura 3.3. O procedimento $Inicializa$ é responsável por declarar $T(D)$, o conjunto de termos importantes do documento D, e copiar $T(D)$ para a variável global $T'(D)$, usada para armazenar os termos importantes que ainda não foram cobertos pelos fragmentos selecionados. Estas variáveis são manipuladas pela função $FragmentoMaisImportante$.

Considere a existência da função $MaiorNumeroTermosComuns(S(D), T'(D))$ que retorna o fragmento em $S(D)$ que possui o maior número de termos em comum com $T'(D)$. O método *two-pass lbpm* considera este como o fragmento mais importante em $S(D)$. Portanto, cada chamada à função $FragmentoMaisImportante$ retorna o fragmento em $S(D)$ que possui mais termos em comum com $T'(D)$, armazenado em S_{comum}. Os termos em S_{comum} são removidos do conjunto $T'(D)$. A remoção de termos de $T'(D)$ permite executar cada seleção de fragmento baseada em diferentes conjuntos de termos importantes. Cada execução da função $FragmentoMaisImportante$ atualiza a variável global $T'(D)$ para que uma próxima chamada a função selecione um fragmento baseado somente nos termos ainda não cobertos, ou seja, aqueles remanescentes em $T'(D)$. Este processo tem o objetivo de preservar o maior número de termos importantes possível durante a seleção de fragmentos. Se os fragmentos já selecionados

cobrem todos os termos importantes, $T'(D)$ torna-se vazio, o que faz com que o algoritmo execute $T'(D) \leftarrow T(D)$, iniciando uma nova rodada de escolhas.

```
Procedimento Inicializa
início
    entrada: T(D) - conjunto de termos importantes de D (variável global)
             T'(D) - termos de T(D) ainda não cobertos (variável global)

    T'(D) ← T(D);
fim

Função FragmentoMaisImportante
início
    entrada: S(D) - conjunto de fragmentos de D (parâmetro)
             T(D) - conjuntos de termos importantes de D (variável global)
             T'(D) - termos de T(D) ainda não cobertos (variável global)
    saída: S_comum - fragmento mais importante de S(D)
    variáveis: t - termo do vocabulário

    S_comum ← MaiorNumeroTermosComuns(S(D), T'(D));
    T'(D) ← T'(D) − {t|t ∈ S_comum ∧ t ∈ T'(D), ∀t ∈ S_comum};
    se T'(D) = ∅ então
        T'(D) ← T(D);
    fim
    retorna S_comum;
fim
```

Figura 3.3: Procedimento *Inicializa* e função *FragmentoMaisImportante* do método *two-pass lbpm*.

O algoritmo pode ser implementado em $O(s^2)$, onde s é o número de fragmentos em cada documento. Como o tamanho de cada documento é independente do número de documentos indexados, s pode ser considerado uma constante e o custo para executar a função *FragmentoMaisImportante* para todos os documentos é linear em relação ao tamanho da coleção. Definindo n como o número de documentos da coleção, o custo será $O(n)$. Entretanto, uma das entradas para o algoritmo é o conjunto de termos importantes de cada documento, o que é obtido executando o método de Carmel. Embora o custo do método de Carmel também seja $O(n)$, para executá-lo é necessário realizar previamente uma indexação completa da base de documentos, o que torna o método *two-pass lbpm* bastante lento.

3.2 Método Full Coverage

O método de poda *full coverage* computa a importância de cada fragmento utilizando uma adaptação de uma abordagem bem-sucedida de geração de resumos proposta por Mallet et al.

[15] para gerar resumos de textos processados por máquinas de busca. A idéia por trás de tal gerador de resumos é selecionar fragmentos que “cubram completamente"o espaço de conceitos de um documento. Isto é feito mensurando iterativamente a similaridade de cada fragmento com um texto que, a princípio, é o conteúdo original do documento, mas que a cada iteração é alterado pela remoção das palavras já cobertas pelos fragmentos até então selecionados. Esta idéia foi adotada para compor o método de poda *full coverage*.

É importante notar que, diferente do método de geração de resumos de Mallet, o método de poda *full coverage* remove entradas do índice enquanto preserva informações estatísticas originais, tais como *idf* e freqüências dos termos e norma dos documentos. Tais informações são essenciais em uma máquina de busca, visto que estas contribuem fortemente para manter os resultados obtidos com os índices podados, para uma dada consulta, próximos aos obtidos com a versão original dos índices.

O procedimento $Inicializa$ e a função $FragmentoMaisImportante$ para o *full coverage* são apresentados na Figura 3.4. O procedimento $Inicializa$ declara e inicializa a variável Q com o texto original do documento D. A variável Q é global e é acessada pela função $FragmentoMaisImportante$. Tal função trata cada fragmento em $S(D)$ como um documento e Q como uma consulta de forma a obter, a cada chamada, o fragmento em $S(D)$ de maior similaridade com Q. De fato, quem computa tal similaridade é a função $CalculaSimilaridade(S(D), Q)$, utilizando para isso o modelo vetorial [21]. Esta função computa a similaridade entre cada fragmento e o valor corrente de Q, e retorna o fragmento que obteve o maior valor de similaridade, dentre os fragmentos em $S(D)$. Após a seleção do fragmento mais importante, S_{sim}, os termos presentes no mesmo são removidos de Q. Esta remoção de termos tem o objetivo de minimizar a redundância de conceitos no conjunto de fragmentos selecionados pelo algoritmo.

Como a versão podada dos índices é construída a partir dos documentos e não é necessário nenhum processo prévio de indexação da base, o qual é um requisito dos métodos *two-pass*, o método *full coverage* foi classificado como um método *single-pass*. Note que algumas informações globais, como *idf* dos termos, necessitam ser computadas de qualquer maneira em um passo independente do processo de poda. Entretanto, este novo passo pode ser executado a um custo menor do que a realização de um processo prévio de construção dos índices.

O método $FragmentoMaisImportante$ do *full-coverage* pode ser implementado em $O(s^2)$, onde s é o número de fragmentos de cada documento, considerando que o tamanho de cada

documento é independente do número de documentos indexados. O custo para executar tal algoritmo para todos os documentos da coleção é linear em relação ao tamanho da coleção.

```
Procedimento Inicializa
início
    entrada: Q - texto da consulta (variável global)
             D - documento (variável global)

    Q ← D;
fim

Função FragmentoMaisImportante
início
    entrada: S(D) - conjunto de fragmentos de D (parâmetro)
             Q - texto da consulta (variável global)
    saída: S_sim - fragmento mais importante de S(D)
    variáveis: t - termo do vocabulário

    S_sim ← CalculaSimilaridade(S(D), Q);
    Q ← Q − {t|t ∈ S_sim ∧ t ∈ Q, ∀t ∈ S_sim};
    retorna S_sim;
fim
```

Figura 3.4: Procedimento *Inicializa* e função *FragmentoMaisImportante* do método *full coverage*.

3.3 Método Top Fragments

O método *top fragments* define a importância de cada fragmento com base na sua posição no texto. Esta estratégia assume que os primeiros fragmentos de cada documento sãos os que melhor podem descrevê-lo, isto é, o método assume que a idéia principal de um documento está concentrada nos seus fragmentos do topo. Esta idéia foi usada para criar a função *FragmentoMaisImportante* do método *top fragments*, que a cada chamada retorna o próximo fragmento do topo ainda não selecionado.

Se k for definido como o número médio de fragmentos selecionados para cada documento, este método pode ser implementado em $O(k)$ por documento, sendo também linear no tamanho da coleção, uma vez que k pode ser considerado uma constante.

3.4 Método Random

Ao contrário do método *top fragments*, o método *random* assume que a idéia principal de um documento está igualmente distribuída entre os seus fragmentos. Desta forma, a cada chamada à função $FragmentoMaisImportante$ do método *random*, esta seleciona aleatoriamente um fragmento, dentre os que ainda não foram selecionados, e o retorna como o mais importante. Este método também pode ser implementado em tempo linear, com um custo similar ao do método *top fragments*.

Capítulo 4

Experimentos

Neste capítulo são apresentados experimentos realizados para avaliar o desempenho dos métodos aqui propostos. Os experimentos foram realizados sobre uma máquina de busca comum e mostram eventuais perdas na qualidade dos resultados, diferenças entre os resultados gerados com índices podados e os gerados com índices originais, a redução que pode ser obtida no tamanho dos índices, o impacto dos métodos de poda no tempo de construção dos índices e os ganhos de eficiência obtidos com diferentes níveis de poda.

4.1 Ambiente de Experimentação

Nesta seção, é apresentado o ambiente de experimentação adotado para avaliar os métodos propostos neste trabalho. São apresentadas informações sobre coleções de teste, estratégias de ordenação de respostas, tipos de consultas, métricas de avaliação de desempenho e configuração das máquinas utilizadas nos experimentos de eficiência.

4.1.1 Coleções de Teste

Os experimentos com métodos de poda foram executados sobre duas coleções: TodoBR e Los Angeles Times (LAT) da TREC [13]. A coleção LAT contém aproximadamente 132.000 documentos (467 MB) extraídos de artigos publicados no jornal *Los Angeles Times* e foi utilizada nos experimentos com o propósito de descobrir possíveis diferenças entre o comportamento de métodos de poda em coleções *web* e coleções de documentos convencionais. As consultas utilizadas nos experimentos com esta coleção foram selecionadas da sessão *ad-hoc* da TREC 8, tópicos de

401 a 450. Os títulos dos tópicos foram utilizados como consultas curtas, pois para a maior parte das consultas longas, a coleção LAT retornava resultados vazios.

A coleção TodoBR é composta de um conjunto de 10.077.722 páginas coletadas da *web* brasileira por uma máquina de busca real de mesmo nome. A escolha desta coleção para os experimentos se deu por esta representar um ambiente real de máquina de busca e possibilitar uma melhor validação das idéias apresentadas neste trabalho. As consultas utilizadas nos experimentos com esta coleção foram extraídas de um *log* de 3 milhões de consultas submetidas à máquina de busca TodoBR. Este *log* é uma das principais razões para utilização da coleção TodoBR nos experimentos, já que este provê informação útil sobre as preferências de usuários de máquinas de busca.

Todos os índices adotados, incluindo os originais (sem poda), foram comprimidos utilizando o método de compressão sem perda *Elias-δ* para codificar número de documentos, freqüências e posições de ocorrência dos termos, como descrito por Witten [24]. O tamanho dos índices da coleção TodoBR após compressão com *Elias-δ* é 8.4 GB para o índice posicional e 1.4 GB para o índice de freqüências, totalizando 9.8 GB de índices comprimidos. As taxas de poda (compressão) computadas para os métodos de poda consideram estes tamanhos de índices como referencial. Por exemplo, a taxa de poda de 80%, nos experimentos com a coleção TodoBR, significa que o tamanho total dos índices após a poda é 1.96 GB. Decidiu-se combinar métodos de compressão “com perda"e “sem perda"para exibir um cenário com índices completamente comprimidos, tirando vantagem dos dois tipos de compressão. Os índices do TodoBR, sem compressão alguma, requerem 31.7 GB de espaço de armazenamento, quando representa-se entradas de freqüência com 8 *bits*, número de documentos com 28 *bits* e posições de ocorrência com 28 *bits*. No caso da coleção LAT, os índices sem compressão requerem 738 MB de espaço de armazenamento e a versão dos índices comprimida com *Elias-δ* requer 184.5 MB.

Documentos Atômicos

Os métodos de poda propostos aqui são baseados na extração de fragmentos dos documentos da base. Neste trabalho, os fragmentos de texto representam uma aproximação de sentenças em linguagem natural extraídas através da pontuação encontrada nos documentos. Embora os métodos tentem podar todos os documentos igualmente, existe um conjunto de documentos que não é afetado pelos métodos de poda. Este conjunto é composto por documentos vazios e

documentos que possuam somente uma sentença. Os documentos vazios são retornados como resultados de consultas devido a outras evidências de informação, tais como valor de autoridade, texto de âncora, etc. Os documentos com apenas uma sentença não são afetados porque os métodos de poda propostos têm a característica de manter pelo menos uma sentença em cada documento. Este conjunto de documentos não afetados pelos métodos baseados em localidade foi chamado de *documentos atômicos*. Apesar dos documentos atômicos possuírem pouca ou nenhuma informação textual, estes são freqüentes nos resultados e usualmente relevantes para o usuário, já que podem conter, por exemplo, imagens, *scripts* que produzem informação textual útil, animações e, em muitos casos, pontos de entrada para *web sites*.

A Figura 4.1(a) exibe como os documentos da coleção TodoBR estão distribuídos em relação ao número de fragmentos (sentenças) que possuem. Mais de 10% dos documentos da coleção TodoBR são vazios e quase 15% têm o seu conteúdo composto por somente uma sentença. Desta forma, aproximadamente 25% dos documentos da coleção TodoBR são atômicos, isto é, não são afetados pelos métodos de poda. Isto não é problema, entretanto, pois estes documentos representam juntos menos de 2% do tamanho da coleção, como ilustrado na Figura 4.1(b).

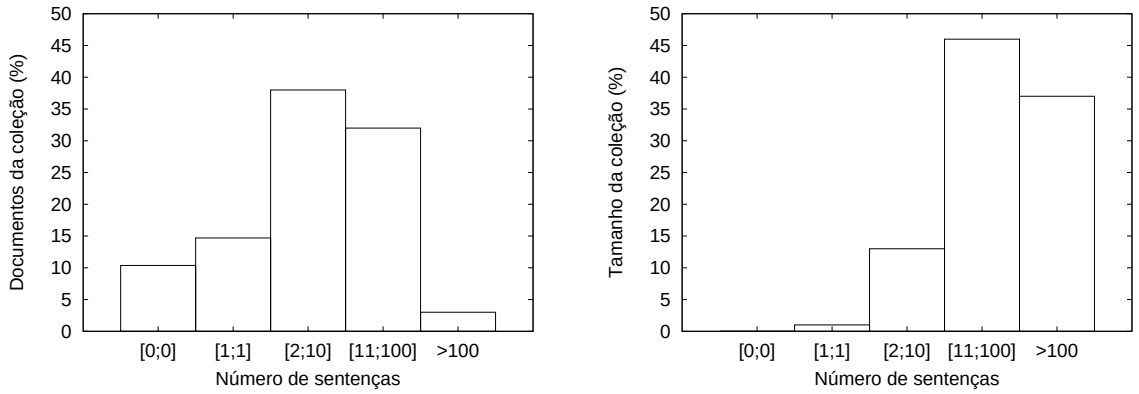
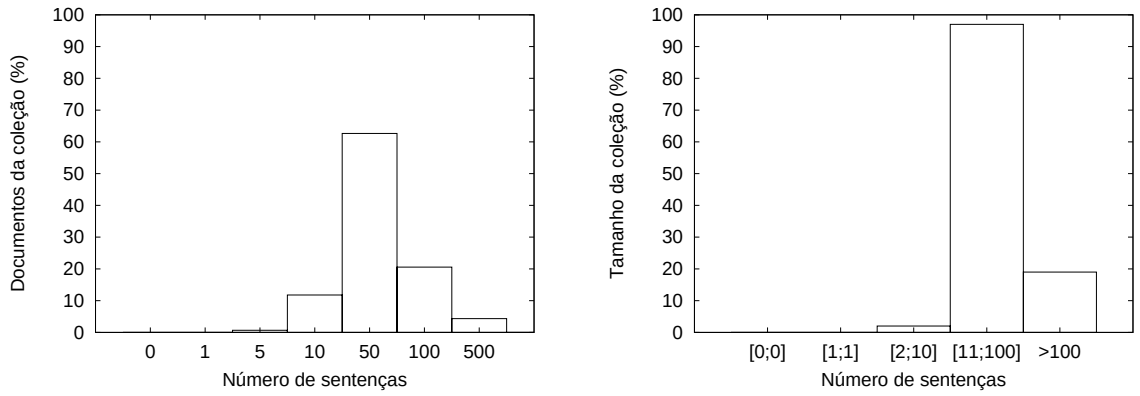


Figura 4.1: Número percentual de documentos da coleção TodoBR (a) e tamanho percentual dos documentos da coleção TodoBR (b), de acordo com o número de sentenças nos documentos.

A Figura 4.2(a) ilustra como os documentos da coleção LAT são distribuídos em relação ao número de sentenças que possuem. Diferentemente da coleção TodoBR, a LAT não possui documentos atômicos. Os documentos da coleção LAT são artigos publicados no jornal *Los Angeles Times* e, como tal, são usualmente bem-escritos e possuem mais de uma sentença. O tamanho percentual dos documentos da coleção LAT de acordo com o número de sentenças é exibido na Figura 4.2(b).



(a) (b)

Figura 4.2: Número percentual de documentos da coleção LAT (a) e tamanho percentual dos documentos da coleção LAT (b), de acordo com o número de sentenças presentes nos documentos.

Documentos atômicos podem influenciar fortemente os resultados obtidos por métodos de poda, uma vez que não são afetados por estes métodos. Consultas que retornam originalmente muitos documentos atômicos como resultado podem ter quase nenhuma alteração na ordenação de suas respostas após a poda e, conseqüentemente, nenhuma perda na qualidade dos seus resultados, independente do método de poda aplicado.

A existência de um grande número de documentos atômicos em coleções *web* torna os métodos de poda ainda mais atrativos no ambiente *web*, pois independente da estratégia adotada, seja esta simples ou mais agressiva, os documentos atômicos não são afetados. Este fato faz com que a perda na qualidade dos resultados causada pela poda seja amortizada pela presença dos documentos atômicos.

4.1.2 Estratégias de Ordenação de Respostas

Um detalhe importante sobre experimentos com máquinas de busca é a estratégia de ordenação de respostas utilizada nos mesmos. Como em toda máquina de busca de grande escala, o algoritmo de ordenação de respostas aplicado à coleção TodoBR foi computado utilizando não somente o texto dos documentos, mas também outras evidências auxiliares. Para tal algoritmo, foram combinadas três diferentes evidências de informação: conteúdo dos documentos, concatenação dos textos de âncora e valores de autoridade de cada documento. A função de ordenação que combina estas três evidências foi implementada como sugerido por Calado et al. [6].

A evidência de conteúdo do documento consiste na similaridade entre o texto do documento e a consulta do usuário, computada utilizando o modelo vetorial [21]. A concatenação do texto de âncora consiste na similaridade entre o texto de todas as âncora que apontam para um documento e a consulta do usuário, também computada utilizando o modelo vetorial. O valor de autoridade de cada documento é dado pelo algoritmo de *HITS* global. O algoritmo de *HITS* global consiste em aplicar o algoritmo de *HITS* [14] para o gráfico de *link* completo da coleção *web*, como fora definido por Calado et al. [6], em vez de aplicar somente nos documentos relacionados a consulta do usuário. O valor de similaridade final de cada documento é dada da seguinte forma:

$$s(d,q) = [1 - (1 - s_c(d,q)) \times (1 - s_a(d,q)) \times (1 - s_h(d,q))] \tag{4.1}$$

onde s_c é a similaridade vetorial da consulta q com o conteúdo do documento d, s_a é a similaridade de q com a concatenação do texto de âncora associado com d, e s_h é o valor de autoridade de d.

Para a coleção LAT, como esta não é uma coleção *web*, todas as consultas foram processadas utilizando somente o modelo vetorial tradicional [1, 21] para ordenar os documentos dados como resposta.

É importante lembrar que o objetivo deste trabalho não é avaliar a qualidade dos resultados produzidos pelos algoritmos de ordenação de respostas, mas sim avaliar o impacto dos métodos de poda sobre os resultados das consultas. A escolha de um algoritmo de ordenação de respostas proposto especificamente para máquinas de busca se deu para que fosse possível produzir resultados mais realísticos nos experimentos. Não seria justo executar experimentos com somente uma fonte de evidência, quando na verdade, máquinas de busca reais usualmente levam em consideração outras evidências de informação para ordenar as respostas de uma consulta. Um outro detalhe importante sobre o algoritmo de ordenação é que o custo principal de espaço de armazenamento e esforço computacional advém dos índices que estão sendo tratados neste trabalho. O texto de âncora, na coleção TodoBR, tem os seus próprios índices que representam somente 10% do tamanho dos índices principais de freqüências e posicional. Além disso, os valores de *HITS* são pré-computados e carregados para memória principal. Mais informações sobre o algoritmo de ordenação de respostas utilizado nos experimentos apresentados aqui podem ser obtidas em [6].

4.1.3 Tipos de Consultas

Máquinas de busca usualmente provêem ricas opções de consultas para seus usuários. Por exemplo, a máquina de busca Google[1] permite que seus usuários incluam opções como inserção de frases nas consultas ou tornem um termo da consulta obrigatório. Em geral, pelo menos três tipos de consultas são importantes para ambientes de máquinas de busca: conjuntivas, disjuntivas e frases.

Em máquinas de busca como Google, Altavista[2] e também no TodoBR, consultas são por padrão submetidas como conjuntivas e, devido a isso, consultas conjuntivas são muito comuns. De fato, no *log* de consultas do TodoBR, aproximadamente 81.1% das consultas com mais de um termo são conjuntivas. Frases e consultas disjuntivas são muito menos comuns. Por exemplo, no *log* do TodoBR elas representam respectivamente 18.3% e 0.6% das consultas com mais de um termo submetidas. Todavia, estes tipos de consultas também serão investigados devido uma possível importância para outros sistemas de busca.

Consultas com um termo também podem ser vistas como um caso particular. Experimentos realizados neste trabalho mostram que o impacto dos métodos de poda sobre consultas com um termo é diferente do impacto sobre os outros três tipos de consultas. Na coleção TodoBR, consultas com um termo correspondem a 34% das consultas existentes no *log*. A Figura 4.3 exibe a distribuição de consultas do *log* do TodoBR de acordo com o número de termos presentes nas mesmas. Como pode ser visualizado, mais de 81% das consultas possuem no máximo três termos.

Na coleção TodoBR, para entender o impacto dos métodos de poda propostos sobre todos os tipos de consulta, definiu-se quatro conjuntos de consultas, um para cada tipo. Cada conjunto contem 1000 consultas selecionadas aleatoriamente do *log* do TodoBR. Para avaliar o impacto dos métodos de poda propostos sobre consultas de diferentes tamanhos (ou seja, diferentes números de termos), foram realizados experimentos com o método proposto que produziu os melhores resultados, submetendo para cada tamanho específico de consulta, um conjunto de 1000 consultas também extraídas do *log* do TodoBR.

Para a coleção LAT, foram utilizados os títulos dos tópicos 401 a 450, extraídos da TREC 8, como consultas conjuntivas e disjuntivas. Consultas frases não foram incluídas nos experimen-

[1] http://www.google.com
[2] http://www.av.com

tos pois a maior parte destas produziam resultados vazio ou com poucos documentos. Poucas consultas com um termo foram encontradas, e por esta razão, este tipo de consulta também não foi incluído nos experimentos para a coleção LAT.

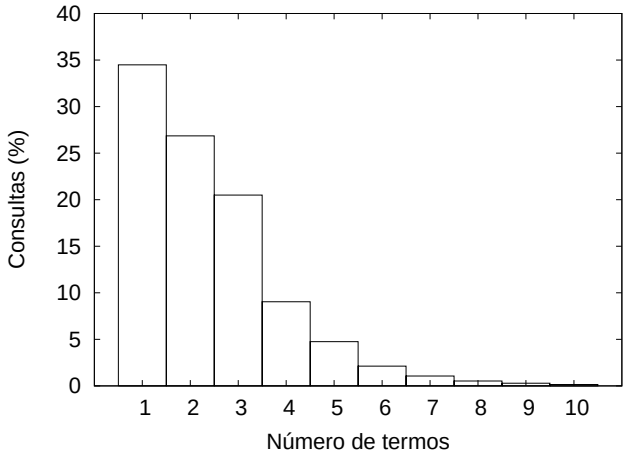


Figura 4.3: Número percentual de consultas do *log* do TodoBR em relação ao número de termos presentes nas mesmas.

4.1.4 Métricas de Avaliação de Desempenho

Para estudar o efeito dos métodos de poda na qualidade dos resultados, foram utilizadas duas métricas distintas: similaridade (distância) entre lista de respostas (documentos) e curvas de precisão e revocação. A primeira medida é baseada na similaridade entre as duas listas de respostas obtidas para cada consulta utilizada nos experimentos, sendo a primeira lista referente às respostas obtidas com os índices originais e a segunda referente às respostas obtidas com os índices gerados por um dado método de poda. Esta medida é computada utilizando a variação do método *Kendall´s tau* proposta por Fagin [11]. A segunda medida, curvas de precisão e revocação, é baseada nos valores de precisão e revocação obtidos para cada consulta. Ambas as métricas foram detalhadas no Capítulo 2, Seção 2.4 - Métricas de Avaliação.

Para cada consulta testada, foram avaliadas as 20 primeiras respostas. Escolheu-se este número pois usuários de máquinas de busca freqüentemente visualizam somente as 20 primeiras respostas de uma consulta. De fato, a maioria dos usuários visualiza somente as 10 primeiras respostas. Por exemplo, no TodoBR, para 70% das consultas, somente as 10 primeiras respostas eram visualizadas [22] e, para aproximadamente 90% das consultas, somente as 20 primeiras respostas eram visualizadas.

Consultas conjuntivas e frases, por serem tipos de consultas mais restritivas, muitas vezes podem produzir listas com poucas respostas. De fato, pode ocorrer do número de respostas ser menor do que 20 para algumas consultas. Nesse caso, após a execução do processo de poda, tais consultas podem produzir um número menor ainda de respostas, fazendo com que a lista de respostas produzida pelos índices podados tenha um tamanho diferente da lista produzida pelos índices originais. Originalmente, o método *Kendall's tau* é voltado apenas para comparar listas de tamanhos iguais. Por esta razão, precisou-se fazer uma adaptação nas listas de respostas para que estas pudessem ser comparadas utilizando este método. A solução adotada aqui foi adicionar respostas falsas a lista de menor tamanho, produzida pelos índices podados, até que esta alcance o tamanho da lista de maior tamanho, produzida pelos índices originais. As entradas falsas consistem de número de documentos não existentes na base de documentos. Esta modificação leva em consideração o fato de que a ausência de algumas respostas no resultado pós poda de uma consulta indica que houve uma perda de informação que estava presente no resultado original. Com esta modificação é possível, por exemplo, comparar uma lista vazia, obtida após o processo de poda, com uma lista de 20 elementos produzida pelos índices originais. Simplesmente são adicionadas 20 respostas falsas a lista vazia e, então, tem-se assim duas listas completamente diferentes, gerando um valor de *Kendall's tau* igual a 0.

Nos experimentos de precisão e revocação, utilizou-se o conceito de revocação relativa, que é baseada somente nos documentos relevantes recuperados utilizando os índices originais e as versões produzidas pelos métodos de poda. Os documentos relevantes oriundos de todos os experimentos são agrupados na tentativa de aproximar-se do número total de documentos relevantes da coleção. Desta forma, as taxas de revocação de um dado sistema consiste dos documentos relevantes recuperados pelo mesmo sobre todos os documentos relevantes recuperados por todos os sistemas.

4.1.5 Teste de Eficiência

O ambiente para avaliação do tempo de execução dos sistemas compreende duas máquinas rodando o sistema operacional GNU/Linux, *kernel* versão 2.4.21. O servidor da máquina de busca roda em uma máquina Pentium 4 2.4 GHz com 1 Gb de memória principal e um disco IDE de 200 Gb. O cliente da máquina de busca roda em uma outra máquina com a mesma configuração.

As duas máquinas são conectadas diretamente (utilizando um cabo *crossover*) por uma conexão *Ethernet* rápida de 100-*megabit*.

4.2 Resultados

Esta seção apresenta os resultados dos experimentos realizados com os métodos de poda propostos neste trabalho. Os resultados do método de Carmel foram utilizados como base de comparação entre os métodos propostos. Os experimentos foram divididos em três partes: similaridade entre resultados de consultas, curvas de precisão e revocação e tempo de eficiência. No primeiro, foi estudada a similaridade (distância) entre os resultados obtidos com os índices originais e os obtidos com os índices gerados por cada um dos método de poda avaliados. Em seguida, estudou-se o impacto da poda sobre as medidas de precisão e revocação. Finalmente, foi estudado o tempo de eficiência dos métodos de poda, incluindo tempos para a construção dos índices e de processamento de consultas.

4.2.1 Similaridade entre resultados de consultas

Os gráficos seguintes mostram a similaridade entre os resultados obtidos com índices gerados pelos métodos de poda avaliados e os obtidos com os índices originais, a medida que aumenta-se o nível de poda. Cinco níveis de poda de aproximadamente 50%, 60%, 70%, 80% e 90% foram testados. É importante citar que os níveis de poda não puderam ser completamente controlados por nenhum dos métodos avaliados. Sendo assim, os níveis produzidos para cada método não são exatamente os mesmos, mas são próximos o suficiente para permitir uma comparação justa.

A Figura 4.4 apresenta os resultados obtidos para a coleção TodoBR, utilizando um conjunto de consultas disjuntivas. Neste caso, todos os métodos produziram resultados muito próximos dos obtidos com os índices originais. A similaridade obtida por cada método, em todos os níveis de poda, foi superior a 0.96. Isto já era esperado pois a probabilidade de se conservar pelo menos um termo da consulta no documento, após o processo de poda, é muito maior do que a probabilidade de se conservar todos os termos.

A Figura 4.5 exibe os resultados obtidos com consultas conjuntivas sobre a coleção TodoBR. Vale lembrar que consultas conjuntivas representam o tipo de consulta mais popular em máquinas de busca. Neste caso, todos os métodos baseados em localidade (*two-pass lbpm*, *full coverage*,

top fragments e *random*) produziram resultados melhores do que o método de Carmel. Por exemplo, para o nível de poda de 87%, o método *top fragments*, que é a abordagem mais simples, obteve uma similaridade de 0.47, enquanto o método de Carmel obteve uma similaridade de 0.21. Embora o método *two-pass lbpm* tenha alcançado melhores resultados na maioria dos níveis de poda, os resultados obtidos pelos métodos single-pass (*full coverage*, *top fragments* e *random*) também foram bem próximos dos obtidos com os índices originais. Para a taxa de poda 60%, o método *two-pass lbpm* produziu uma similaridade de 0.70 enquanto o método *top fragments* rendeu uma similaridade de 0.66, significando uma diferença de somente 4%. O *full coverage* foi o que produziu os melhores resultados dentre os métodos *single-pass*, para consultas conjuntivas.

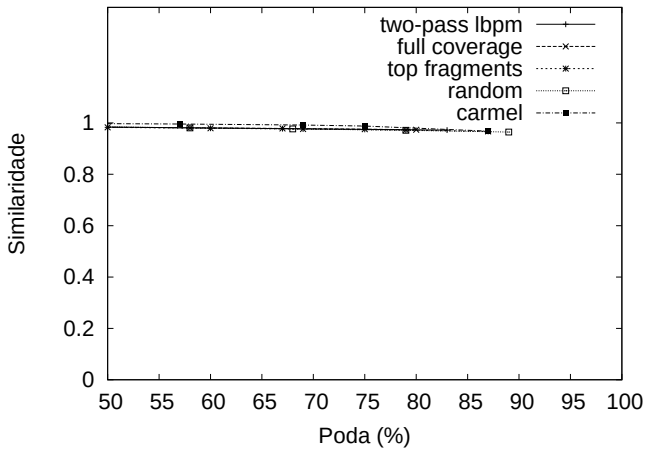


Figura 4.4: Similaridade de *Kendall's tau* obtida para os métodos *two-pass lbpm*, *full coverage*, *top fragments*, *random* e Carmel sobre a coleção TodoBR, utilizando somente consultas disjuntivas.

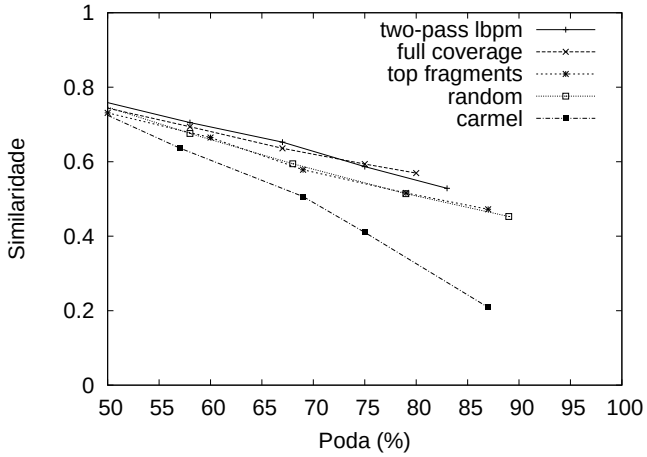


Figura 4.5: Similaridade de *Kendall's tau* obtida para os métodos *two-pass lbpm*, *full coverage*, *top fragments*, *random* e Carmel sobre a coleção TodoBR, utilizando consultas conjuntivas.

Na Figura 4.6 são apresentados os resultados obtidos com consultas frases sobre a coleção TodoBR. Os métodos de poda baseados em localidade produziram consistentemente resultados melhores neste caso. Por exemplo, para o nível de poda de 88%, o método *random*, a abordagem mais ingênua, produziu uma similaridade de 0.44, sendo este o menor valor de similaridade obtido dentre todos os métodos baseados em localidade. O maior nível de poda que permitiu essa mesma similaridade para o método de Carmel foi 57%. O método de Carmel apresenta um valor de similaridade muito baixo inclusive para o nível de poda de 50%, onde foi obtida uma similaridade de 0.5. Por outro lado, para este mesmo nível de poda, o método *random* produziu uma similaridade de quase 0.8. Novamente, o método *two-pass lbpm* produziu os resultados mais similares ao sistema original, embora muito próximos dos resultados obtidos pelos métodos *single-pass*. Para o nível de poda de 67%, o método *two-pass lbpm* gerou uma similaridade de 0.71, enquanto o método *random*, produziu uma similaridade de 0.68, significando uma diferença de 3%. Em geral, os níveis de similaridade obtidos com consultas frases são melhores do que os obtidos com consultas conjuntivas. De fato, uma frase, que freqüentemente ocorre em uma única sentença, é mais provável de ser preservada do que uma consulta conjuntiva que pode está distribuída entre muitas sentenças.

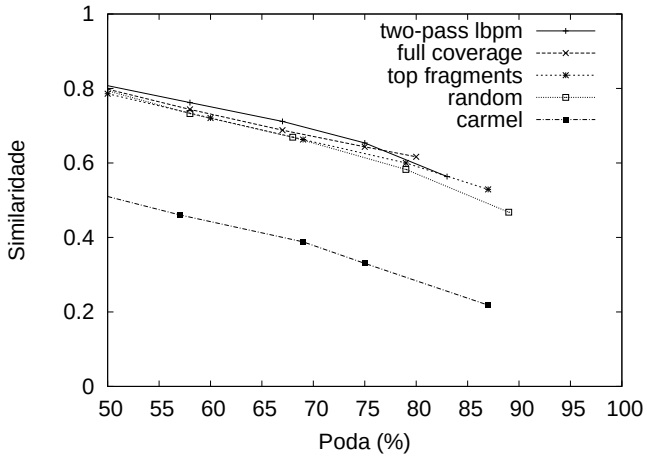


Figura 4.6: Similaridade de *Kendall's tau* obtida para os métodos *two-pass lbpm*, *full coverage*, *top fragments*, *random* e Carmel sobre a coleção TodoBR, utilizando somente consultas frases.

A Figura 4.7 exibe os resultados para a coleção TodoBR, utilizando somente consultas de um termo. Neste caso, o método de Carmel produziu os resultados mais similares aos obtidos com os índices originais em todos os níveis de poda. Uma pequena perda na similaridade era esperada para os métodos baseados em localidade, uma vez que estes métodos removem algumas entradas

do topo da lista invertida de cada termo individualmente, para incluir entradas que ajudem a melhorar os resultados de consultas conjuntivas e frases. Apesar disso, tais métodos apresentam resultados muito próximos dos obtidos pelo método de Carmel. Além disso, em todos os níveis de poda, a similaridade obtida pelos métodos baseados em localidade foi superior a 0.89, o que indica que os resultados produzidos são bastante similares aos do sistema original. É importante lembrar que consultas com um termo correspondem a 34% das consultas submetidas ao TodoBR.

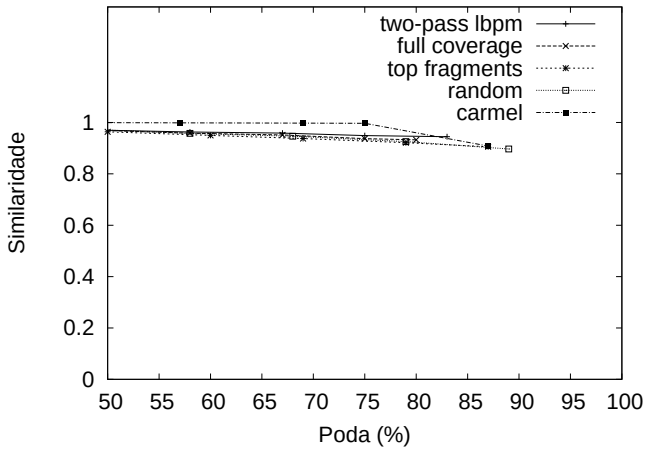


Figura 4.7: Similaridade de *Kendall's tau* obtida para os métodos *two-pass lbpm*, *full coverage*, *top fragments*, *random* e Carmel sobre a coleção TodoBR, utilizando somente consultas de um termo.

Os experimentos realizados com a coleção TodoBR mostram que os métodos baseados em localidade claramente produzem resultados melhores do que os produzidos pelo método de Carmel para consultas conjuntivas e frases. De certa forma, estes resultados já eram esperados, pois os métodos baseados em localidade tendem a preservar as ocorrências de termos que aparecem próximas no texto dos documentos. Vale lembrar que consultas conjuntivas e frases representam juntas mais de 65% das consultas encontradas no *log* do TodoBR. Os experimentos também mostram que os métodos *single-pass*, principalmente os métodos *top fragments* e *full coverage*, produzem, em muitos casos, resultados próximos dos obtidos com o método *two-pass lbpm*. Isto significa que, apesar da simplicidade destes métodos, estes podem ser utilizados por máquinas de busca *web* sem uma perda significativa na qualidade dos resultados das consultas.

Em geral, no caso da coleção TodoBR, os métodos de poda baseados em localidade produziram resultados próximos entre si. Este fato pode ser explicado pelo número de documentos atômicos retornados como respostas das consultas utilizadas nos experimentos. Por exemplo,

mais de 40% das respostas de cada consulta conjuntiva testada eram documentos atômicos. É importante lembrar que os documentos atômicos não são afetados pelos métodos baseados em localidade e que usualmente são documentos importantes para os usuários. Estes resultados mostram que documentos atômicos podem afetar bastante o desempenho de métodos de poda na *Web*.

Uma outra informação importante sobre os métodos de poda é o impacto que estes têm sobre o resultado de consultas à medida que se aumenta o número de termos nas mesmas. Na Figura 4.8, tem-se os valores de similaridade produzidos pelo método *two-pass lbpm*, que obteve o melhor desempenho dentre os métodos baseados em localidade sobre a coleção TodoBR. Neste experimento, foram utilizadas consultas com no máximo 5 termos, o que corresponde a 95% das consultas encontradas no *log* do TodoBR. Para cada tamanho de consulta, ou seja, número de termos, utilizou-se um conjunto de 1000 consultas. Como já era esperado, os níveis de similaridade são degradados consideravelmente a medida que se aumenta o número de termos nas consultas. Entretanto, os níveis de similaridade obtidos para as taxas de poda de 50% e 60% são aceitáveis, especialmente para consultas com até 3 termos, que correspondem aproximadamente a 82% das consultas do *log* do TodoBR.

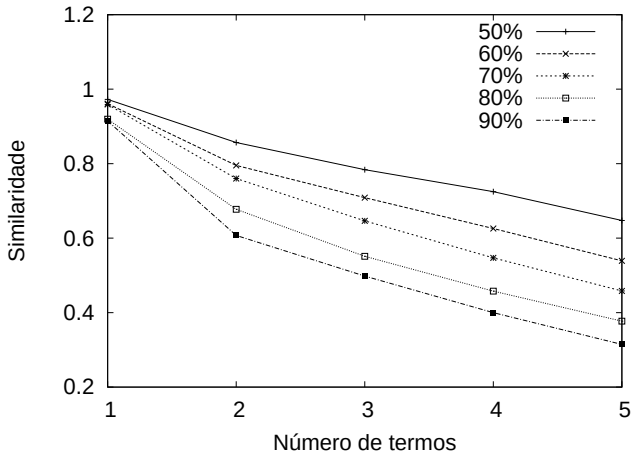
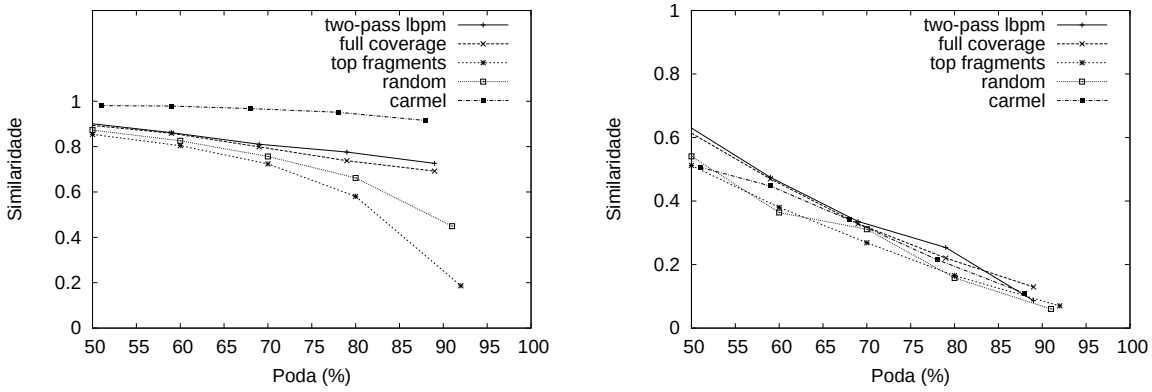


Figura 4.8: Similaridade de *Kendall's tau* obtida pelo método *two-pass lbpm* sobre a coleção TodoBR, de acordo com o número de termos nas consultas.

Finalmente, a Figura 4.9 apresenta resultados para a coleção LAT utilizando consultas curtas extraídas do título de tópicos e submetidas como consultas disjuntivas e conjuntivas. Para consultas disjuntivas, Figura 4.9(a), pode-se observar que os resultados são levemente diferentes daqueles obtidos com a coleção TodoBR. Neste caso, o método de Carmel obteve melhores

resultados para consultas disjuntivas com uma diferença significativa. Além disso, os resultados produzidos pelos métodos *single-pass* são degradados consideravelmente a medida que se aumenta o nível de poda aplicado. Por exemplo, para o nível de poda de 80%, o método *top fragments* produziu uma similaridade de 0.97 para a coleção TodoBR. Por outro lado, na coleção LAT, o método *top fragments* obteve uma similaridade de 0.58 para o nível de poda de 80%.



(a) disjuntiva (b) conjuntiva

Figura 4.9: Similaridade de *Kendall's tau* obtida para os métodos *two-pass lbpm*, *full coverage*, *top fragments*, *random* e Carmel sobre a coleção LAT. As figuras (a) e (b) exibem, respectivamente, os resultados para consultas disjuntivas e conjuntivas.

Em geral, os valores de similaridade obtidos para a coleção LAT são menores que aqueles obtidos para a coleção TodoBR. Isto pode ser explicado principalmente pelo fato de não existirem documentos atômicos na coleção LAT. Além disso, muitos termos do vocabulário da coleção LAT são termos raros, que aparecem em poucos documentos e com uma freqüência pequena, o que faz com que tenham uma probabilidade menor de serem mantidos nos índices após um processo de poda.

Na coleção LAT, um termo raro é usualmente correto e importante para discriminar um documento de outros. Isto provavelmente não é válido na *Web*, onde muitos termos raros não fazem sentido, sendo em geral lixo produzido por erros de digitação. Além disso, a natureza coletiva da *Web* faz com que os termos que aparecem em uma consulta tenham uma alta probabilidade de aparecerem nos documentos. Isto acontece porque um termo que é importante para um usuário, provavelmente é também importante para outros usuários. Em outras palavras, termos comuns em consultas de máquinas de busca tendem também a ser comuns nos documentos *web*.

O método de Carmel preserva uma porção significativa das entradas para termos raros, enquanto os métodos baseados em localidade tendem a substituir algumas entradas dos termos

raros por entradas de termos que aparecem próximos nos documentos. Isso não é um problema para coleções *web*, mas pode ser uma desvantagem para a coleção LAT. De fato, os resultados obtidos com a coleção LAT não são conclusivos, uma vez que não se utilizou nos experimentos da LAT consultas reais, criadas por usuários de máquinas de busca, mas sim consultas sintéticas, criadas a partir do título de tópicos da TREC. Ainda assim, os resultados dos experimentos enfatizam a necessidade de se projetar métodos de poda voltados para máquinas de busca *web*, e também indicam que métodos de poda baseados em localidade podem ser uma boa alternativa nestes casos.

4.2.2 Precisão e Revocação

Para avaliar o impacto dos métodos de poda sobre a precisão das máquinas de busca, foram executados experimentos com a coleção TodoBR, submetendo um conjunto de 40 consultas extraídas aleatoriamente do *log* de consultas do TodoBR. O conjunto é dividido, de acordo com a distribuição real de consultas no *log*, em 14 consultas de um termo, 21 consultas conjuntivas e 5 frases. Os resultados das consultas foram avaliados por um grupo de 15 pessoas. O número médio de respostas para cada consulta foi 33 e o número médio de respostas relevantes foi 17.3.

A Figura 4.10 exibe a evolução da precisão média obtida para cada um dos métodos avaliados, a medida que aumenta-se o nível de poda. Para níveis de poda de até 70%, os valores de precisão média obtidos para o método *two-pass lbpm* são próximos dos obtidos com os índices originais (0% de poda), fazendo com que o *two-pass lbpm* seja o melhor método em termos de precisão média. Os métodos *single-pass* apresentaram resultados próximos aos gerados pelo método de Carmel. O que significa que os métodos de Carmel e *full coverage* não são superiores aos métodos *top fragments* e *random*, que são abordagens mais simples.

As Figuras 4.11 a 4.13 apresentam curvas de precisão e revocação obtidas com os índices originais e os índices gerados pelos métodos avaliados, para níveis de poda variando de 60% a 80%. Na Figura 4.11, tem-se os resultados para o nível de poda 60%, onde o método *two-pass lbpm* produziu resultados muito similares aos obtidos com os índices originais (não podado). Os resultados gerados pelos métodos *single-pass* e Carmel foram bastante próximos aos obtidos com o método *two-pass lbpm*.

Para o nível de poda de 70%, exibido na Figura 4.12, a perda na precisão é maior. Entretanto, o método *two-pass lbpm* continua gerando resultados consideravelmente próximos aos obtidos com

os índices originais, para níveis de revocação de até 50%. Como os níveis de revocação menores são os mais importantes para as máquinas de busca, apesar da perda de 4 pontos em precisão média, as perdas na qualidade dos resultados para o nível de poda de 70% são aceitáveis.

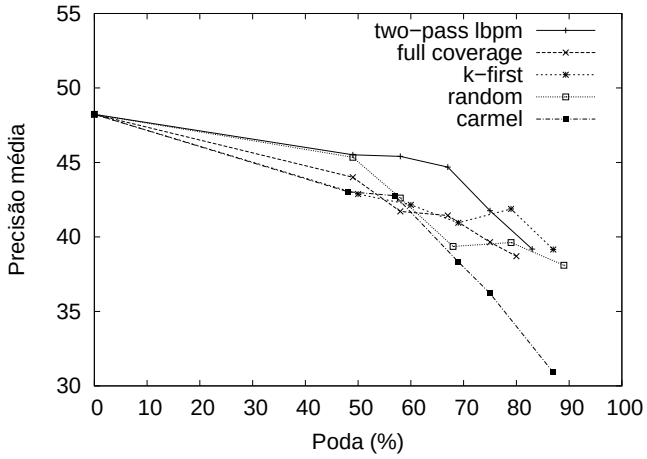


Figura 4.10: Precisão média obtida para consultas processadas utilizando os índices gerados pelos métodos *two-pass lbpm*, *full coverage*, *top fragments*, *random* e Carmel.

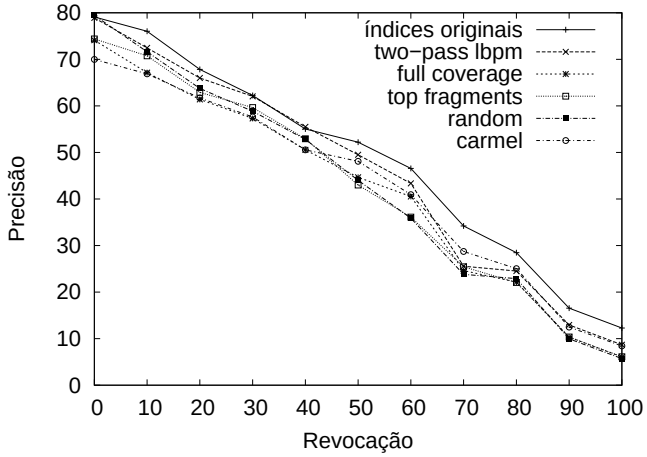


Figura 4.11: Curvas de precisão e revocação para a coleção TodoBR, utilizando os índices originais e índices comprimidos pelos métodos *two-pass lbpm*, *full coverage*, *top fragments*, *random* and Carmel, com uma taxa de poda de aproximadamente 60%.

Finalmente, a Figura 4.13 mostra que todos os métodos não são eficazes em manter os níveis de precisão quando a taxa de poda é maior que 80%.

Utilizando o mesmo conjunto de 40 consultas testado nos experimentos anteriores de precisão e revocação, foram realizados outros experimentos com o método *two-pass lbpm* para avaliar o

impacto deste método sobre a precisão das consultas à medida que se aumenta o número de termos nas mesmas(ver Figura 4.14). Os experimentos foram realizados sobre a coleção TodoBR. Os resultados obtidos com consultas de um termo são apresentadas na Figura 4.14(a). Como já fora observado nos experimentos de similaridade apresentados neste trabalho, as consultas de um termo sofrem quase nenhum impacto da poda realizada nos índices da máquina de busca, independente da taxa de poda aplicada.

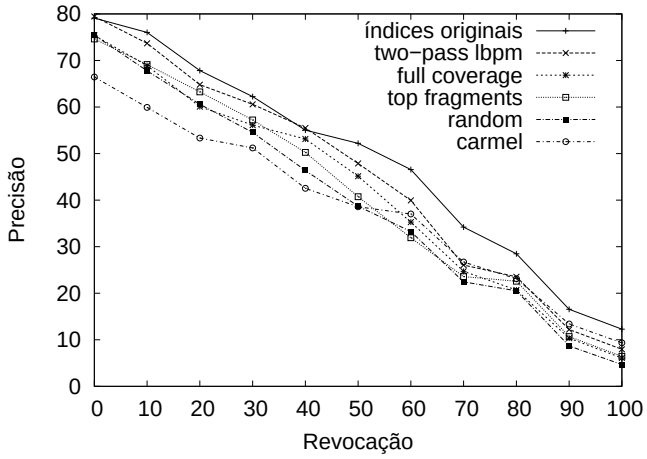


Figura 4.12: Curvas de precisão e revocação para a coleção TodoBR, utilizando os índices originais e índices comprimidos pelos métodos *two-pass lbpm*, *full coverage*, *top fragments*, *random* and Carmel, com uma taxa de poda de aproximadamente 70%.

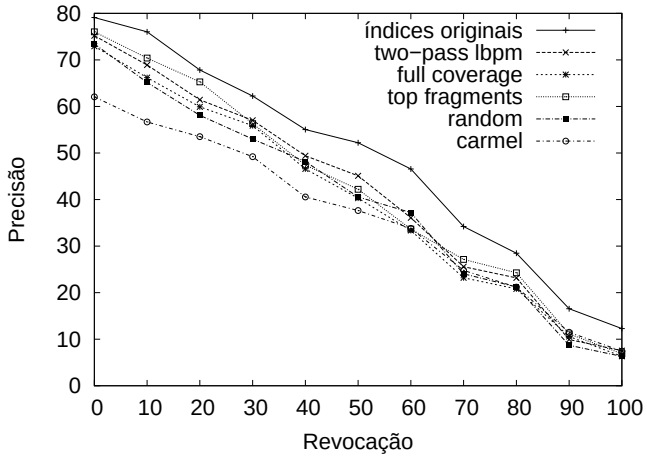
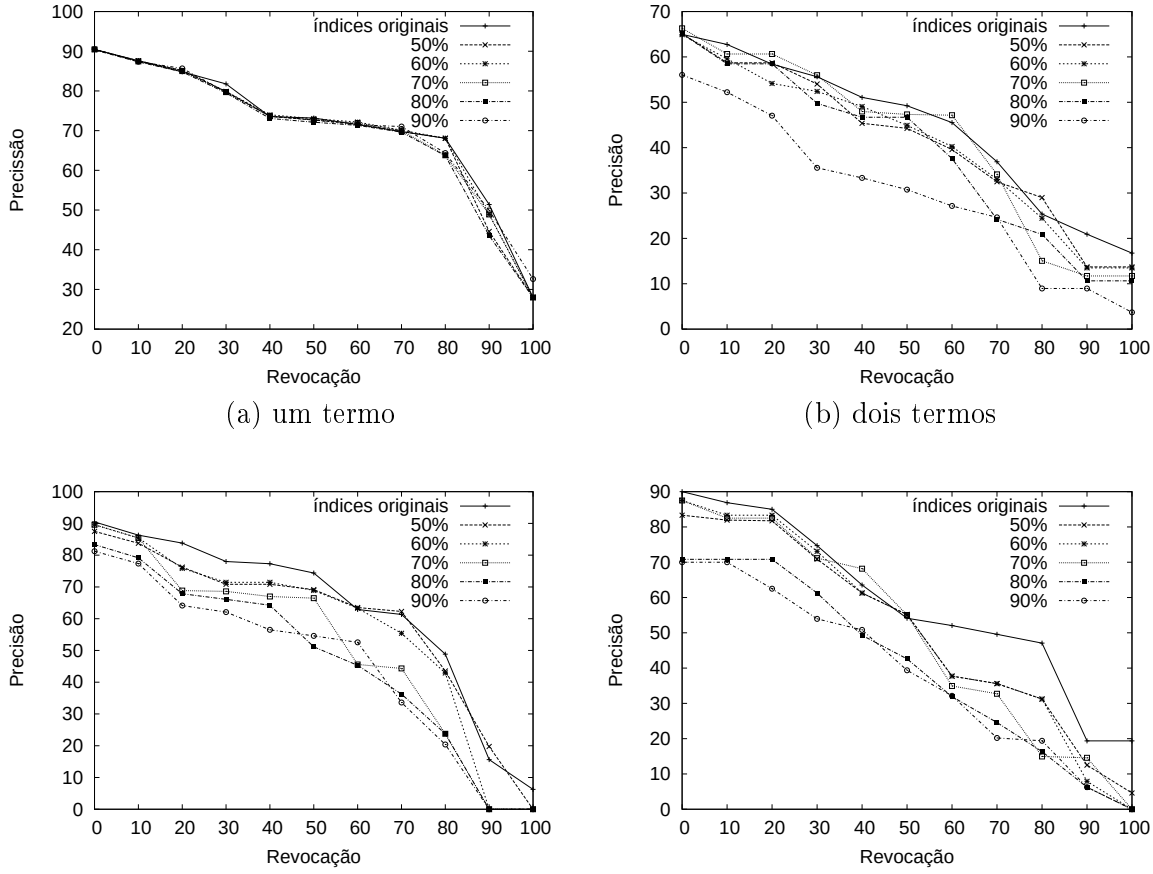


Figura 4.13: Curvas de precisão e revocação para a coleção TodoBR, utilizando os índices originais e índices comprimidos pelos métodos *two-pass lbpm*, *full coverage*, *top fragments*, *random* and Carmel, com uma taxa de poda de aproximadamente 80%.

A Figura 4.14(b) apresenta curvas de precisão e revocação para consultas com dois termos. Pode-se notar que, embora os níveis de precisão degradem à medida que se aumenta a taxa de poda, os níveis de precisão são próximos dos obtidos com os índices originais para taxas de poda até 80%. Para consultas de três termos, os resultados exibidos na Figura 4.14(c) mostram que o método *two-pass lbpm* não é eficaz em manter os níveis de precisão, embora estes sejam aceitáveis até o nível de poda de 60%. A Figura 4.14(d) exibe os resultados para consultas de quatro termos. Como se pode observar, para as taxas de poda de até 70%, a precisão para os níveis de revocação topo (até 50%) são bastante próximos dos obtidos com os índices originais.



(a) um termo (b) dois termos

(c) três termos (d) quatro termos

Figura 4.14: Curvas de precisão e revocação para a coleção TodoBR, utilizando índices comprimidos pelo método *two-pass lbpm*. As figuras (a), (b), (c) e (d) exibem, respectivamente, resultados dos experimentos com consultas de um, dois, três e quatro termos.

Estes resultados permitem concluir que, embora o processo de poda cause alterações nos resultados das consultas, como foi exibido pelos experimentos de similaridade utilizando a medida *Kendall's tau*, as alterações na precisão são pequenas. Na verdade, isto implica que pode-se

reduzir significativamente o custo com o armazenamento dos índices das máquinas de busca, melhorando o seu desempenho, sem comprometer a satisfação dos usuários com os resultados.

4.2.3 Eficiência

Finalmente, têm-se os experimentos conduzidos para avaliar o ganho de eficiência obtido com a redução no tamanho dos índices. Primeiro, é apresentado o tempo necessário para podar os índices, utilizando cada um dos métodos propostos aqui. Neste caso, os métodos *single-pass* têm a vantagem de serem mais eficientes. A Figura 4.15 exibe o tempo necessário para podar os índices da coleção TodoBR utilizando os métodos propostos, em cada um dos cinco níveis de poda experimentados. Neste gráfico, o tempo é expresso como um valor percentual, em função do tempo necessário para criar os índices originais, que é representado no gráfico pela reta no ponto $y = 100\%$.

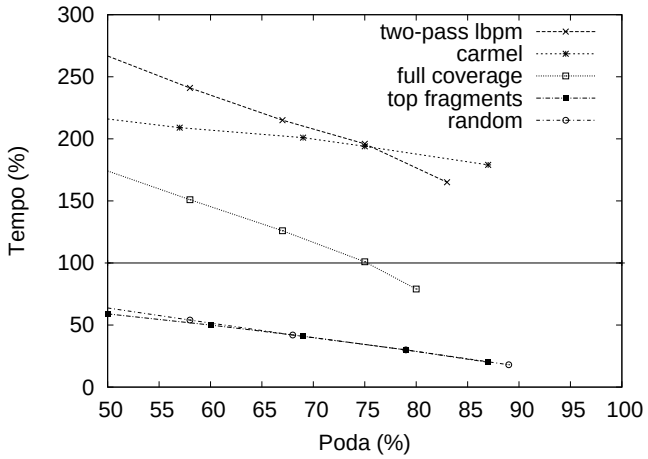


Figura 4.15: Tempo percentual, em função do tempo de criação dos índices originais, necessário para criar os índices pelos métodos de poda propostos.

Note que para os métodos *two-pass lbpm* e Carmel, o tempo necessário para podar os índices é sempre superior ao tempo de geração dos índices originais (100%), visto que estes métodos necessitam primeiro realizar um processo de indexação completo da base de documentos, criando os índices originais, para então realizar o processo de poda. Nos métodos *single-pass*, o processo de poda é executado juntamente com o processo de indexação, o que resulta em um ganho significativo de eficiência ao gerar os índices. Por exemplo, o tempo de construção dos índices para os métodos *top fragments* and *random* são proporcionais a redução alcançada no tamanho final dos índices. O *full coverage* é o único método *single-pass* que não obteve ganho de eficiência

ao gerar os índices. Isto é conseqüência do seu custo computacional, que é quadrático em relação ao número de fragmentos selecionados para cada documento da base.

Uma outra vantagem importante da poda é a redução no tempo de processamento das consultas. A Figura 4.16 exibe os resultados obtidos com a coleção TodoBR ao se processar consultas sobre índices com diferentes níveis de poda gerados pelo método *two pass lbpm*. O tempo é expresso como um valor percentual em função do tempo de processamento das consultas utilizando os índices originais. Note que os métodos de poda reduzem somente o tamanho dos índices de freqüências e posicional construídos para o texto dos documentos. Outros componentes da máquina de busca, tais como índices para texto de âncora, não são afetados pelos métodos de poda propostos aqui.

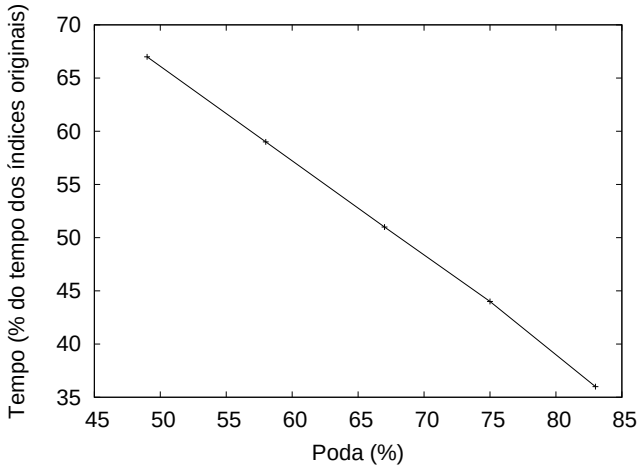


Figura 4.16: Tempo necessário para processar um total de 10.000 consultas extraídas do *log* de consultas do TodoBR, utilizando diferentes níveis de poda.

Obviamente, como os índices gerados pelos métodos de poda sofrem redução de tamanho, o tempo para processar consultas é também reduzido. Nos experimentos, esta redução segue um comportamento quase linear. É importante notar que o impacto da poda no processamento de consultas depende do sistema de busca utilizado, da configuração de *hardware* disponível e da quantidade de informação processada pelo sistema de busca. Portanto, os resultados apresentados na Figura 4.16 são úteis apenas para ilustrar como os métodos de poda podem ter um impacto significativo sobre o tempo de processamento de consultas.

Capítulo 5

Conclusão e Trabalhos Futuros

Neste trabalho, foram propostos e experimentados novos métodos de poda estática especialmente projetados para máquinas de busca *web*. Baseado em características observadas em coleções *web*, tais como a necessidade de conservar as relações entre termos e a presença dos chamados documentos atômicos, foi proposta uma família de métodos de poda baseados em localidade. Quatro métodos de poda fazem parte desta família: *two-pass lbpm*, *full coverage*, *random* e *top fragments*. Estes métodos tentam preservar nos índices conjuntos de termos que podem ocorrer juntos em consultas, visando manter a qualidade dos resultados fornecidos pelas máquinas de busca a seus usuários.

O *two-pass lbpm* é o método que consegue melhor manter a qualidade dos resultados providos por máquinas de busca. Entretanto, por ser um método *two-pass*, apresenta duas desvantagens: aumenta o tempo de geração dos índices e não permite a atualização direta dos índices podados. Os métodos *single-pass* (*full coverage*, *random* e *top fragments*) são abordagens mais simples, que produzem resultados ligeiramente inferiores aos obtidos com o método *two-pass lbpm*, mas que não possuem suas principais desvantagens. Nos métodos *single-pass*, o processo de poda é executado simultaneamente à construção ou atualização dos índices. Neste caso, gerar os índices podados é mais rápido inclusive do que gerar os índices originais e qualquer atualização pode ser realizada diretamente na versão podada dos índices, evitando a necessidade de reconstruir os índices toda vez que uma nova informação é adicionada.

Quando comparados ao método *two-pass lbpm*, os métodos *single-pass* têm uma pequena perda na qualidade dos resultados, mas esta perda é compensada pela redução no tempo de construção e atualização do índices. Por exemplo, utilizando apenas consultas frases, para o

nível de poda de 67%, o método *two-pass lbpm* gerou uma similaridade de 0.71, enquanto o método *random* produziu uma similaridade de 0.68, significando uma diferença de somente 3%. Para esta mesma taxa de poda, o método de Carmel, melhor método proposto na literatura, produziu uma similaridade de apenas 0.39.

Os experimentos realizados comprovam que localidade é uma informação essencial para projetar métodos de poda para máquinas de busca *web*. Mesmo métodos de poda baseados em localidade simples são superiores aos métodos que não fazem uso da informação de localidade. Por exemplo, para a taxa de poda de 87%, utilizando apenas consultas conjuntivas, o método *top fragments*, que é a abordagem mais simples dentre os métodos baseados em localidade, obteve uma similaridade de 0.48, enquanto o método de Carmel obteve uma similaridade de 0.21. Os métodos baseados em localidade produzem resultados melhores do que o método de Carmel, principalmente para consultas conjuntivas e frases, que correspondem a 99.4% das consultas com mais de um termo encontradas no *log* de consultas do TodoBR.

Além disso, vale lembrar que o objetivo principal de se utilizar métodos de poda em índices de máquinas de busca é melhorar sua eficiência em termos de tempo de processamento de consultas. Experimentos realizados mostram, por exemplo, que uma redução de 67% no tamanho dos índices, obtida com o método *two pass-lbpm*, consegue alcançar uma redução de 48% no tempo necessário para processar um conjunto de 10.000 consultas. Isso mostra que os métodos de poda podem alcançar uma redução significativa no tempo de processamento de consultas em máquinas de busca, sem afetar drasticamente a qualidade dos resultados providos pelas mesmas. Os experimentos mostram que a perda na qualidade das respostas de consultas é mínima para uma redução de até 50% no tamanho dos índices. Para uma redução de 60% no tamanho dos índices, a perda na qualidade continua aceitável para consultas de até 3 termos, que correspondem a 82% das consultas encontradas no *log* do TodoBR.

O uso de métodos de poda estática abre a possibilidade de melhorias na ordenação de respostas através da adição de estratégias de ordenação mais sofisticadas. Isto é possível devido a quantidade de dados gerenciada durante o processamento de consultas ser menor após a poda, o que economiza tempo para a utilização de uma estratégia de ordenação de respostas mais robusta. A decisão de adotar um método *two-pass* ou *single-pass* depende do projetista da máquina de busca e é uma questão de custo versus benefício entre qualidade dos resultados e eficiência do processo de criação e atualização dos índices.

Trabalhos futuros

Os métodos de poda estática aqui propostos visualizam cada documento da base como um conjunto de fragmentos. Estes fragmentos podem ser passagens de texto com um número definido de termos, passagens que representem unidades lógicas como sentenças ou parágrafos, entre outros. Neste trabalho, os experimentos foram realizados utilizando fragmentos do tipo sentença. Entretanto, seria interessante estudar como os métodos de poda baseados em localidade se comportam quando utilizados outros tipos de fragmentos, ou seja, verificar o impacto do tipo de fragmento sobre o desempenho destes métodos. Além disso, estes métodos identificam que conjuntos de termos são possíveis de ocorrerem juntos em consultas, analisando apenas os documentos da base. Uma outra possível abordagem seria analisar os *logs* de consultas da máquina de busca para obter este tipo de informação.

Embora os métodos de poda estática propostos tenham apresentado bons resultados quanto à manutenção da qualidade das respostas produzidas pela máquina de busca, estes métodos podem e devem ser melhorados quanto a isso. Experimentos realizados neste trabalho, para avaliar o impacto do processo de poda sobre consultas com diferentes números de termos, mostram que a qualidade dos resultados é degradada consideravelmente quando se aumenta o número de termos nas consultas. Além disso, à medida que se eleva o nível de poda aplicado, estes resultados degradam ainda mais, principalmente para consultas com um número maior de termos.

Portanto, uma abordagem interessante de se estudar seria desenvolver métodos de poda dinâmica, que levassem em consideração o número de termos presentes em uma consulta ao definir o nível de poda a ser aplicado na mesma. Métodos de poda dinâmica realizam o processo de poda durante o tempo de processamento das consultas e por isso podem ajustar o nível de poda aplicado a cada consulta específica. Nesta abordagem, as entradas da lista invertida de cada termo devem estar ordenadas de modo a permitir o processamento apenas das entradas consideradas relevantes. Sendo assim, consultas com um número menor de termos, que costumam ser mais abrangentes, poderiam sofrer uma poda maior, enquanto consultas com um número maior de termos, que costumam ser mais específicas e menos abrangentes, poderiam sofrer uma poda menor. De fato, identificar o nível de poda a ser aplicado para uma dada consulta pode não depender exclusivamente do número de termos encontrados na mesma, mas também do tipo de consulta, se esta é popular ou não, entre outros fatores.

Referências Bibliográficas

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley, 1999.

[2] D. Bahle, H. E. Williams, and J. Zobel. Efficient phrase querying with and auxiliary index. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 215–221, 2002.

[3] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text compression.* Prentice Hall, 1990.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International World Wide Web Conference*, pages 107–117, 1998.

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

[6] P. P. Calado, E. S. de Moura, B. Ribeiro-Neto, I. Silva, and N. Ziviani. Local versus global link information in the web. *ACM Transactions on Information Systems (TOIS)*, 21(1):42–63, 2003.

[7] D. Carmel, D. Cohen, R. Fagin, E. Farchi, M. Herscovici, Y. S. Maarek, and A. Soffer. 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, 2001.

[8] E. S. de Moura, C. F. dos Santos, D. R. Fernandes, A. S. Silva, P. Calado, and M. A. Nascimento. Improving web search efficiency via a locality based static pruning method. In *Proceedings of the 14th International World Wide Web Conference*, pages 235–244, 2005.

[9] E. S. de Moura, G. Navarro, N. Ziviani, and R. Baeza-Yates. Fast and flexible word searching on compressed text. *ACM Transactions on Information Systems (ACM TOIS)*, 18(2):113–139, 2000.

[10] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of American Society for Information Science*, 41(6):391–407, 1990.

[11] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 28–36, 2003.

[12] D. R. Fernandes. *O uso de geração de resumos como estratégia para melhorar o desempenho de máquinas de busca.* Dissertação de Mestrado, Universidade Federal do Amazonas, 2004.

[13] D. Hawking, E. Voorhees, P. Bailey, and N. Craswell. Overview of trec-8 web track. In *Proceedings of TREC-8*, pages 131–150, 1999.

[14] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, 1998.

[15] D. Mallett, J. Elding, and M. A. Nascimento. Information-content based sentence extraction for text summarization. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2*, pages 214–218, 2004.

[16] G. Navarro, E. S. de Moura, M. Neubert, N. Ziviani, and R. Baeza-Yates. Fast and flexible word searching on compressed text. *Information Retrieval Journal*, 3(1):49–77, 2000.

[17] T. Nomoto and Y. Matsumoto. A new approach to unsupervised text summarization. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 26–34, 2001.

[18] M. Persin, J. Zobel, and R. Sacks-Davis. Filtered document retrieval with frequency-sorted indexes. *Journal of the American Society of Information Science*, 47(10):749–764, 1996.

[19] Daniel E. Rose and Danny Levinson. Understanding user goals in web search. In *Proceedings of the 13th International World Wide Web Conference*, pages 13–19, 2004.

[20] T. Sakai and K. Sparck-Jones. Generic summaries for indexing in ir. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 190–198, 2001.

[21] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[22] P. C. Saraiva, E. S. Moura, N. Ziviani, B. Ribeiro-Neto, W. Meira, and R. L. C. Fonseca. Rank-preserving two-level caching for scalable search engines. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 51–58, 2001.

[23] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[24] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images.* Morgan Kaufmann Publishers, 1999.