Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Computação Aplicada

Processamento Distribuído da Consulta Espaço Textual Top-k

Tiago Fernandes de Athayde Novaes

Feira de Santana

2017

Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Computação Aplicada

Tiago Fernandes de Athayde Novaes

Processamento Distribuído da Consulta Espaço Textual Top-k

Dissertação apresentada à Universidade Estadual de Feira de Santana como parte dos requisitos para a obtenção do título de Mestre em Computação Aplicada.

Orientador: João B. Rocha-Junior

Coorientador: Angelo A. Duarte

Feira de Santana

2017

Ficha Catalográfica - Biblioteca Central Julieta Carteado

N819p Novaes, Tiago Fernandes de Athayde
Processamento distribuído da consulta espaço textual top-*k*/ Tiago Fernandes de Athayde Novaes. Feira de Santana, 2017.
64f.: il.

Orientador: João B. Rocha-Júnior.
Coorientador: Angelo A. Duarte
Dissertação (mestrado) - Universidade Estadual de Feira de Santana, Programa de Pós-Graduação em Computação Aplicada, 2017.

1. Particionamento de dados (Computação). 2. Sistemas distribuídos. 3. Sistemas de gerenciamento de bases de dados. 4. Consulta Espaço textual Top-k. (Computação). 5. Sistemas de informação. I. Rocha- Júnior, João B., orient. II. Duarte, Angelo A., coorient. III. Universidade Estadual de Feira de Santana. IV. Título.

CDU: 004.65

Tiago Fernandes de Athayde Novaes

Processamento Distribuído da Consulta Espaço Textual Top-k

Dissertação apresentada à Universidade Estadual de Feira de Santana como parte dos requisitos para a obtenção do título de Mestre em Computação Aplicada.

Feira de Santana, 17 de julho de 2017

BANCA EXAMINADORA

Dr. João Batista da Rocha Junior (Orientador)
Universidade Estadual de Feira de Santana

Dr. Luciano de Andrade Barbosa
Universidade Federal de Pernambuco

Dr. Daniel Gouveia Costa
Universidade Estadual de Feira de Santana

Abstract

With the popularization of databases containing objects with spatial and textual information (spatio-textual object), the interest in new queries and techniques for retrieving these objects have increased. In this scenario, the main query is the the top-k spatio-textual query. This query retrieves the k best spatio-textual objects considering the distance of the object to the query location and the textual similarity between the query keywords and the textual information of the objects. However, most the studies related to top-k spatio-textual query are performed in centralized environments, not addressing real world problems such as scalability. In this paper, we study different strategies for partitioning the data and processing the top-k spatio-textual query in a distributed environment. We evaluate each strategy in a real distributed environment, employing real datasets.

Keywords: data partitioning, distributed query processing, spatio-textual query, information systems, information retrieval

Resumo

Com a popularização de bases de dados contendo objetos que possuem informação espacial e textual (objeto espaço-textual), aumentou o interesse por novas consultas e técnicas capazes de recuperar esses objetos de forma eficiente. Uma das principais consultas para objetos espaço-textuais é a consulta espaço-textual top-k. Essa consulta visa recuperar os k melhores objetos considerando a distância do objeto até um local informado na consulta e a similaridade textual entre palavras-chave de busca e a informação textual dos objetos. No entanto, a maioria dos estudos para consultas espaço-textual top-k assumem ambientes centralizados, não abordando problemas frequentes em aplicações do mundo real como escalabilidade. Nesta dissertação são estudadas diferentes formas de particionar os dados e o impacto destes particionamentos no processamento da consulta espaço-textual top-k em um ambiente distribuído. Todas as estratégias propostas são avaliadas em um ambiente distribuído real, utilizando dados reais.

Palavras-chave: particionamento de dados, processamento de consultas distribuídas, consultas espaço-textuais, sistemas de informação, recuperação de informação

Prefácio

Esta dissertação de mestrado foi submetida a Universidade Estadual de Feira de Santana (UEFS) como requisito parcial para obtenção do grau de Mestre em Computação Aplicada.

A dissertação foi desenvolvida dentro do Programa de Pós-Graduação em Computação Aplicada (PGCA) tendo como orientador o Dr. **João B. Rocha-Junior** e coorientador o Dr. **Angelo A. Duarte**.

Agradecimentos

Primeiramente gostaria de agradecer ao meu orientador Prof. João B. Rocha-Junior pela paciência e orientação durante todo o programa do PGCA. O seu auxílio foi imprescindível para conclusão deste trabalho.

Ao meu coorientador, Prof. Angelo A. Duarte, agradeço pelas aulas e todo conhecimento que me foi passado durante o PGCA.

Agradeço a minha esposa Tallita Menezes e aos meus filhos Felipe e Rafael que por muitas vezes foram privados da minha atenção e disponibilidade durante a elaboração deste trabalho. Obrigado pela compreensão e por tornarem os meus dias mais felizes, amo vocês.

Aos meus pais, Djalma Araújo Novaes e Esther Fernandes de Athayde Novaes, agradeço bastante por dedicaram parte de suas vidas para me fornecer uma educação de qualidade. Sem o empenho de vocês este trabalho não seria possível.

Agradeço a minha irmã Rafaela Athayde e seu namorado Leandro Pedreira pelas conversas via facebook, durante a madrugada, em momentos de descontração entre a rotina de estudos.

A minha irmã Maria Amelia e seu marido Igor Fraga, um agradecimento especial pelo incentivo e por disponibilizarem seus carros para que eu voltasse de Cruz das Almas a tempo de assistir as aulas.

E por fim, agradeço a minha família e aos amigos pelo apoio e incentivo durante esta jornada.

Sumário

Lista de Publicações

- Proposta de Particionamento de Dados para processamento de Consultas Espaço-Textual Top-k distribuídas. Tiago Fernandes de Athayde Novaes, João B. Rocha-Junior. (2016). *In WPOS/ERBASE.*

- S2I+: Armazenamento Eficiente em Índice Espaço-Textual. Tiago F. Athayde-Novaes, Fellipe L. Fonseca and João B. Rocha-Junior. (2016). *In SBBD.*

Lista de Tabelas

Lista de Figuras

Lista de Abreviações

Abreviação	**Descrição**
S2I	*Spatial Inverted Index*
S2I+	Variação do *Spatial Inverted Index*
MBR	*Minimum Bounding Rectangle*
aR-Tree	*Aggregated* R-Tree
MA	Modelo de Particionamento Aleatório
SMAleatório	Sistema que utiliza o Modelo de Particionamento Aleatório
MLE	Modelo de Particionamento por Localização Espacial
SMEspacial	Sistema que utiliza o MLE com processamento espacial puro
SMHíbrido	Sistema que utiliza o MLE com processamento híbrido
MST	Modelo de Particionamento por Similaridade Textual
SMTextual	Sistema que utiliza o Modelo de Particionamento por Similaridade Textual
MT	Modelo de Particionamento por Termos
SMTermo	Sistema que utiliza o Modelo de Particionamento por Termos
$q.l$	Localização da consulta
$o.l$	Localização do objeto
$q.d$	Conjunto de palavras-chave de uma consulta
$o.d$	Descrição textual de um objeto

Lista de Símbolos

Símbolo	Descrição
θ	Função de cálculo da relevância textual entre dois conjuntos de caracteres
δ	Função de cálculo da distância entre dois objetos
λ	Função de cálculo da relevância textual de um termo em um objeto
α	Parâmetro que define uma relevância maior entre a medida textual e a espacial

Capítulo 1

Introdução

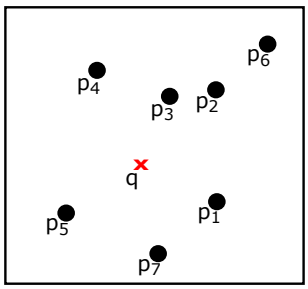
"Acredite que você pode, assim você já está no meio do caminho."
– Theodore Roosevelt

Hoje em dia é fácil encontrar bases de dados com objetos que possuem informação espacial (latitude e longitude) e textual. O *OpenStreetMap*[1] é um exemplo de site que disponibiliza informações sobre objetos como hotéis, restaurantes, bares, farmácias, dentre outros, contendo a informação espacial referente à localização do objeto e também a informação textual que descreve este objeto. Por exemplo, um restaurante pode ter a indicação da sua localização espacial e também informações textuais, tais como: nome do estabelecimento, serviços prestados, cardápio, dentre outros. Estes objetos que contem informação espacial e textual são chamados de objetos espaço-textuais.

As bases de dados contendo objetos espaço-textuais têm crescido consideravelmente nos últimos anos. Twitter[2] e Facebook[3] são exemplos de aplicações que armazenam diariamente uma quantidade enorme de objetos espaço-textuais, tornando um desafio extrair informações relevantes destas bases de dados. Isto justifica o interesse por novas consultas e técnicas capazes de selecionar objetos espaço-textuais em bases de dados volumosas [Rocha-Junior et al. 2011, Cong et al. 2009, Li et al. 2011, Cao et al. 2012].

Uma das principais consultas para objetos espaço-textuais é a consulta espaço-textual top-k [Rocha-Junior et al. 2011, Cong et al. 2009, Li et al. 2011, Cao et al. 2012, Chen et al. 2013, Kwon et al. 2015]. Esta consulta recebe como parâmetro um local de interesse, um conjunto de palavras-chave e a quantidade

[1] www.openstreetmap.org
[2] www.twitter.com
[3] www.facebook.com.br



Objetos	Termos
p_1	**restaurante** chinês
p_2	chinês chinês
p_3	**restaurante japonês**
p_4	**restaurante**
p_5	italiano
p_6	**restaurante** mexicano
p_7	chinês

Figura 1.1: Região espacial e informações textuais do objetos.

de objetos desejados, retornando os k melhores objetos espaço-textuais. Para cada objeto espaço-textual é computado um escore, medidos por uma combinação da distância do objeto até o local informado na consulta e a relevância da descrição textual dos objetos para as palavras pesquisadas.

Por exemplo, a Figura 1.1 apresenta uma área espacial com uma coleção de objetos p, onde q representa a localização de um usuário, os objetos p_1 até p_7 representam restaurantes e a tabela apresenta o texto associado a cada objeto espaço-textual p. O resultado da consulta espaço-textual top-3 nessa coleção para q e palavras-chave "restaurante japonês" são os objetos p_3, p_4, p_1, assumindo que as descrições textuais com maior número de palavras-chave presentes na consulta são mais relevantes.

Uma forma de executar a consulta espaço-textual top-k de forma eficiente é utilizando estruturas de indexação. A maioria das estruturas de indexação para objetos espaço-textuais são híbridas, combinando índices textuais e espaciais. Os índices mais utilizados para implementação dessas estruturas híbridas são o Arquivo Invertido [Zobel e Moffat 2006] para busca textual e a R-tree [Guttman 1984] para busca espacial. [Rocha-Junior et al. 2011, Li et al. 2011, Cong et al. 2009].

Diante de bases de dados bastante volumosas, a utilização de estruturas de indexação pode não ser suficiente para atender às requisições dos usuários em um tempo satisfatório. Isso acontece porque o processamento é realizado de forma centralizada e os recursos computacionais das máquinas são limitados, influenciando no desempenho do processamento da consulta quando esse limite é alcançado, mesmo utilizando estruturas de indexação.

Uma forma de processar a consulta espaço-textual top-k em volumosas bases de dados é utilizando sistemas distribuídos. Esta pesquisa estuda o processamento distribuído e paralelo da consulta espaço-textual top-k com diferentes formas de particionar o conjunto de objetos espaço-textuais e avalia o desempenho de cada uma das formas em um cenário distribuído.

Este capítulo é organizado da seguinte forma: a Seção 1.1 contém a motivação desta pesquisa, a Seção 1.2 apresenta as contribuições. Na Seção 1.3 as questões de pesquisa são apresentadas, enquanto a Seção 1.4 expõe o método de pesquisa

utilizado. Por fim, a Seção 1.5 lista os artigos publicados durante o mestrado e a Seção 1.6 faz um esboço da estrutura desta dissertação.

1.1 Motivação

Para o melhor do nosso conhecimento, todos os estudos realizados com consultas espaço-textuais top-k são em ambientes centralizados. Nesses estudos, dados e estruturas ficam armazenados de forma centralizada em um único servidor, limitando a quantidade de objetos utilizados e refém da capacidade de armazenamento e processamento de somente uma máquina. Isso não é comum em aplicações do mundo real que devem garantir escalabilidade ao sistema, sendo capazes de funcionarem com uma quantidade enorme de dados e requisições de usuários.

Sistemas de Informação como Google, Twitter, Facebook, dentre outros, têm adotado sistemas distribuídos para resolver problemas relacionados a escalabilidade. Esses sistemas atendem melhor a demanda dos usuários da Internet, por garantirem um melhor poder de processamento e armazenamento com um baixo custo. Em meados de 2004, o engenho de busca do Google utilizou um sistema distribuído com 20.000 computadores para processar mais de 200 milhões de consultas por dia em mais de 20TB de dados indexados [Zobel e Moffat 2006].

Uma forma simples de processar uma consulta em um sistema distribuído é particionar a coleção de objetos e alocar cada partição em um servidor do sistema. Dessa forma, o processamento da consulta pode ser dividido entre os servidores, onde cada servidor processa a consulta em uma partição. Ao final do processamento local, cada servidor gera um resultado parcial que são combinados e formam um resultado final.

A maioria dos estudos com processamento distribuído de consultas para dados espaço-textuais realizam essa forma simples de distribuir uma consulta [He et al. 2015, Porwal e Shiwani 2015, Doulkeridis et al. 2017]. No entanto, nenhum realiza o processamento distribuído da consulta espaço-textual top-k.

A seguir são apresentadas aplicações que podem se beneficiar do processamento distribuído da consulta espaço-textual top-k (Seção 1.1.1).

1.1.1 Aplicações

Muitas aplicações podem se beneficiar por novas técnicas para processar a consulta espaço-textual top-k de forma distribuída. Dentre essas aplicações, é possível destacar aplicativos de trânsito e navegação (Waze[4] e Google Maps[5]), Redes Sociais (Twitter e Facebook) e Sistemas de Segurança Pública.

[4] www.waze.com
[5] maps.google.com

Os sistemas de trânsito e navegação já possuem bases de dados volumosas formadas por objetos espaço-textuais. Esses aplicativos podem se beneficiar da consulta espaço-textual top-k para permitir que seus usuários encontrem os melhores estabelecimentos (restaurantes, bares, farmácias, hotéis, dentre outros) próximos a sua localização e com as palavras-chave pesquisadas. O Waze é um exemplo de aplicação que permite consultas por palavras-chave para que os usuários encontrem um local de destino desejado e visualize rotas entre o local de partida e o local de destino escolhido. No entanto, não realiza um ranqueamento dos objetos consultados avaliando a proximidade do local de partida e a similaridade textual dos objetos.

As redes sociais como Twitter e Facebook também podem se beneficiar. A maioria das mensagens enviadas para o Twitter através de Smartphones com GPS possuem a informação textual e espacial. Essas aplicações podem realizar a consulta espaço-textuais top-k distribuída para permitir que seus usuários consultem as melhores mensagens enviadas próximo à sua localização e que possuam as palavras-chave de busca. Por exemplo, um usuário do Twitter deseja encontrar mensagens que relatem opiniões sobre bares próximos a sua casa. O Twitter já permite pesquisas por palavras-chave dentro de uma região espacial, mas não realiza um ranqueamento dos objetos considerando a proximidade do local de busca e a similaridade textual dos objetos.

Os Órgãos de Segurança Pública podem se beneficiar da consulta espaço-textual top-k distribuída para auxiliar o combate à criminalidade. Sistemas de segurança pública podem utilizar grandes bases de dados formadas por objetos espaço-textuais disponíveis pelas redes sociais para permitir que seus usuários (policiais, seguranças, guardas municipais) consultem termos como “assalto”, “furto”, “revolver”. Dessa forma, é possível encontrar possíveis ocorrências de criminalidade próximas as viaturas, as delegacias, dentre outros pontos da cidade e permitir que o poder público execute as medidas de segurança necessárias.

1.2 Contribuições

1. Apresentar formas de particionamento de dados para permitir o processamento distribuído da consulta espaço-textual top-k;
2. Apresentar adaptações no processamento distribuído da consulta espaço-textual top-k;
3. Identificar através de uma extensa avaliação experimental o melhor modelo em um cenário distribuído, considerando parâmetros como: tempo para construção dos modelos, tempo de resposta e número de páginas lidas (I/O);
4. Disponibilizar as bases de dados utilizadas nos experimentos para que outros pesquisadores possam utilizá-las em suas pesquisas.

1.3 Questões de Pesquisa

A questão de pesquisa geral desta dissertação é – Como processar a consulta espaço-textual top-k em um sistema distribuído de forma eficiente? Esta questão conduz a questões mais específicas que são apresentadas a seguir:

Q 1. Como processar a consulta espaço-textual top-k distribuída em paralelo e sequencial?

Q 2. Qual a melhor forma de particionar os dados para o processamento em paralelo e sequencial?

Q 3. Qual o desempenho do sistema para cada uma das abordagens em termos de tempo de resposta, I/O e tráfego de dados na rede?

1.4 Método de Pesquisa

Esta seção contém a metodologia adotada nesta pesquisa. As principais etapas são: 1) levantamento bibliográfico, 2) formalização dos modelos para particionamento dos dados, 3) levantamento de bases de dados para validação, 4) validação dos modelos, 5) realização de experimentos, 6) publicação dos resultados obtidos e 7) a escrita da dissertação.

O levantamento bibliográfico consiste em uma revisão detalhada da literatura em tópicos como consultas textuais, consultas espaciais, consultas espaço-textuais e consultas distribuídas. Esses tópicos são pesquisados em bases de dados como o periódicos Capes e Google Scholar para encontrar artigos e livros relacionados ao tema, dando prioridade aos mais recentes e com maior número de citações. O objetivo desse levantamento é conhecer mais sobre o tema e as estruturas mais utilizadas para processamento das consultas espaço-textual top-k, durante este levantamento é elaborado, para cada artigo, um resumo com as ideias principais dos documentos selecionados.

Após o levantamento bibliográfico, novos modelos são propostos para permitir que as consultas espaço-textuais top-k sejam processadas em um sistema distribuído, de forma paralela ou sequencial e com uma maior escala de dados. A definição desses modelos deve se basear em soluções similares apresentadas em áreas como Recuperação de Informação, Mineração de Dados e Engenhos de Busca.

A etapa de levantamento de bases de dados para validação consiste em coletar objetos espaço-textuais reais em sistemas como OpenStreetMap (OSM), Twitter e Wikipédia. Tanto o OSM como Wikipédia disponibilizam suas bases de dados para download de forma gratuita. O Twitter também disponibiliza uma parte dos seus dados de forma gratuita, no entanto é necessário implementar uma aplicação utilizando a *API (Twitter Search API)* disponibiliza pelo próprio Twitter para realizar a coleta.

A etapa de validação dos modelos consiste em desenvolver sistemas que funcione em um sistema distribuído permitindo a troca de informações entre as máquinas e a divisão do processamento das consultas espaço-textual top-k.

Durante os experimentos, as técnicas de particionamento são avaliadas em um sistema distribuído real com as bases de dados coletadas. Em cada experimento pretende-se avaliar o impacto sobre I/O, tráfego de dados e tempo de resposta ao variar o número de palavras-chave da consulta, o número de resultados, número de alfa e com base de dados diferentes.

Os resultados obtidos são organizados, analisados e discutidos para serem publicados em eventos e periódicos. O resultados parciais são publicados em eventos e os resultados finais em periódico para compartilhar os resultados obtidos com a comunidade científica.

A etapa final é a escrita da dissertação. Esta etapa consiste em descrever os modelos de particionamentos de forma detalhada, assim como os conceitos necessários para a compreensão desta consulta. Nesta etapa, também são descritos os experimentos realizados, incluindo todos os resultados obtidos.

1.5 Publicações

Esta seção apresenta as publicações parciais originadas desta pesquisa de mestrado, acompanhadas por uma breve descrição.

- Proposta de Particionamento de Dados para processamento de Consultas Espaço-Textual Top-k distribuídas. Tiago Fernandes de Athayde Novaes, João B. Rocha-Junior. (2016). *In WPOS/ERBASE*, Alagoas, Maceió, Abril 2016.

  Artigo apresentado no *Workshop* de pós-graduação da Escola Regional de Computação Bahia-Alagoas-Sergipe (ERBASE). Neste artigo foi apresentado apenas uma definição dos modelos de particionamento de dados e algumas informações relacionadas a proposta da pesquisa de mestrado. Durante o WPOS, o artigo concorreu ao prêmio *Best Wpos Paper*, onde foi analisado por uma banca de doutores e acabou sendo considerado o segundo melhor artigo do *Workshop*.

- S2I+: Armazenamento Eficiente em Índice Espaço-Textual. Tiago F. Athayde-Novaes, Fellipe L. Fonseca and João B. Rocha-Junior *XXXI Simpósio Brasileiro de Banco de Dados (SBBD 2016)*, Bahia, Salvador, Outubro 2016.

  O artigo curto publicado no SBBD apresenta um estudo na distribuição dos dados armazenados no S2I e dois métodos de armazenamento para reduzir

o espaço utilizado em disco. Os dois métodos propostos apresentaram uma redução significativa em relação ao tamanho do índice além de apresentarem melhores tempo de resposta no processamento da consulta espaço-textual top-k.

1.6 Sinopse da Dissertação

Nesta seção é descrita a organização desta dissertação.

Capítulo 1: Introdução. Este capítulo oferece a introdução desta dissertação e é subdividido em motivação, contribuições, questões de pesquisa, método de pesquisa e publicações realizadas.

Capítulo 2: Fundamentação Teórica. Neste capítulo são abordados os temas indispensáveis para o entendimento da pesquisa, bem como os trabalhos relacionados.

Capítulo 3: Definição do Problema. Neste capítulo é feita a definição do problema de pesquisa.

Capítulo 4: Modelos de Particionamento. Este capítulo descreve em detalhes os modelos propostos para realizar o processamento distribuído da consulta espaço-textual top-k.

Capítulo 5: Avaliação Experimental. Neste capítulo é realizada a avaliação experimental com abordagens que utilizam os modelos propostos para realizar o processamento distribuído da consulta espaço-textual top-k.

Capítulo 6: Consideração Finais. Este capítulo conclui a dissertação, apresentando as principais contribuições e os trabalhos futuros.

Capítulo 2

Fundamentação Teórica

"A vida é muito curta para ser gasta nutrindo animosidade ou registrando erros."

– Charlotte Bronte

Este capítulo apresenta estudos que contextualizam o tema proposto neste trabalho. O capítulo inicia com uma apresentação das consultas distribuídas (Seção 2.1), em seguida, apresenta as consultas textuais (Seção 2.2) e espaciais (Seção 2.3) e por fim as consultas espaço-textuais (Seção 2.4).

2.1 Consultas Distribuídas

O processamento de consultas distribuídas é uma importante ferramenta para lidar com as limitações de recursos computacionais das máquinas e a grande escala de dados disponíveis [Zobel e Moffat 2006]. Antes de abordar sobre como é realizado o processamento distribuído das consultas é importante apresentar o conceito de sistemas distribuídos.

2.1.1 Sistemas Distribuídos

Os sistemas distribuídos são formados por componentes (hardware ou software) localizados em computadores interligados em rede que se comunicam e coordenam suas tarefas através da troca de mensagens [Coulouris et al. 2011]. O principal objetivo para construir esses sistemas é facilitar o compartilhamento de recursos entre usuários e aplicações dentro de uma rede [Tanenbaum e Steen 2006]. Esses recursos podem ser entendidos como impressoras, unidades de armazenamento, páginas web,

banco de dados, ou seja, qualquer coisa considerada útil que possa ser compartilhados em uma rede de computadores [Coulouris et al. 2011].

Os sistemas distribuídos devem ter as seguintes características: 1) possuir abertura para permitir que componentes sejam adicionados ou substituídos facilmente com o sistema em execução, 2) ser transparente fazendo com que os usuários pensem que são atendidos por um único sistema, ocultando as transações executadas entre os serviços, 3) possuir escalabilidade, sendo capaz de funcionar bem quando o número de acessos ou a carga de trabalho aumentarem, 4) possuir um bom tratamento de falhas para evitar que a falha de um recurso prejudique todo o sistema, permitindo que o sistema sempre fique disponível mesmo que somente com uma parte dos recursos, e 5) permitir que várias tarefas sejam executadas ao mesmo tempo.

A maioria dos sistemas distribuídos são construídos na arquitetura cliente-servidor [Tanenbaum e Steen 2006]. Nesta arquitetura os recursos são fisicamente encapsulados dentro de computadores e só podem ser acessados por meio de comunicação, tanto o servidor como o cliente são considerados processos que interagem por meio de mensagens. Um processo cliente solicita um serviço de um processo servidor através de requisições e aguarda a resposta desse servidor. Os servidores podem ser clientes de outros servidores como por exemplo um servidor web pode ser cliente de um servidor de banco de dados que pode ser cliente de um servidor de e-mail.

Um exemplo de sistema distribuído que utiliza a arquitetura cliente-servidor é um engenho de busca na web. O engenho de busca na web é responsável por realizar buscas através de palavras-chave em todo o conteúdo disponível na internet [Coulouris et al. 2011]. O processo cliente fica responsável por interagir com os usuários, requisitando as consultas ao processo servidor e apresentando os resultados retornados. Já o processo servidor fica responsável por encontrar o conteúdo pesquisado dentro da enorme quantidade de informações disponíveis na internet. Diante da grande quantidade de requisições e do grande volume de dados processados, o processo servidor pode ser um outro sistema distribuído para realizar o processamento das consultas.

O sistema distribuído em *cluster* é muito utilizado por engenhos de busca na web para implementar o processo servidor. Esse sistema é formado por um conjunto de computadores conectados por uma rede de alta velocidade com objetivo de fornecer um único recurso computacional integrado de alto desempenho [Coulouris et al. 2011]. Geralmente os computadores que formam um *cluster* são semelhantes, possuindo mesmo hardware e software, com a finalidade de usar a programação paralela para permitir que um único programa seja executado em vários computadores ao mesmo tempo [Tanenbaum e Steen 2006].

Um exemplo muito conhecido de *cluster* é o *cluster* de *Beowulf*. Esse *cluster* é formado por computadores, chamadas de nós, onde são gerenciadas por um único nó (mestre) que é responsável por manter a lista de tarefas, dividir as tarefas em sub-tarefas, distribuir as sub-tarefas entre os outros nós do *cluster* (escravos)

e ao mesmo tempo interagir com os usuários retornando o resultado das tarefas [Tanenbaum e Steen 2006].

Na Seção 2.1.2 são apresentadas formas de processar uma consulta em um sistema distribuído.

2.1.2 Processamento Distribuído

Uma forma de processar uma consulta em um sistema distribuído é através do particionamento de dados [Zobel e Moffat 2006]. O particionamento de dados consiste em dividir um conjunto de dados em subconjuntos para que servidores diferentes do sistema distribuído possam acessar e processar consultas em um subconjunto. Assim o processamento de uma consulta pode ser distribuído entre os servidores do sistema, onde cada servidor processa a consulta em um subconjunto. Ao final do processamento local, cada servidor gera um resultado parcial que são combinados e formam um resultado final.

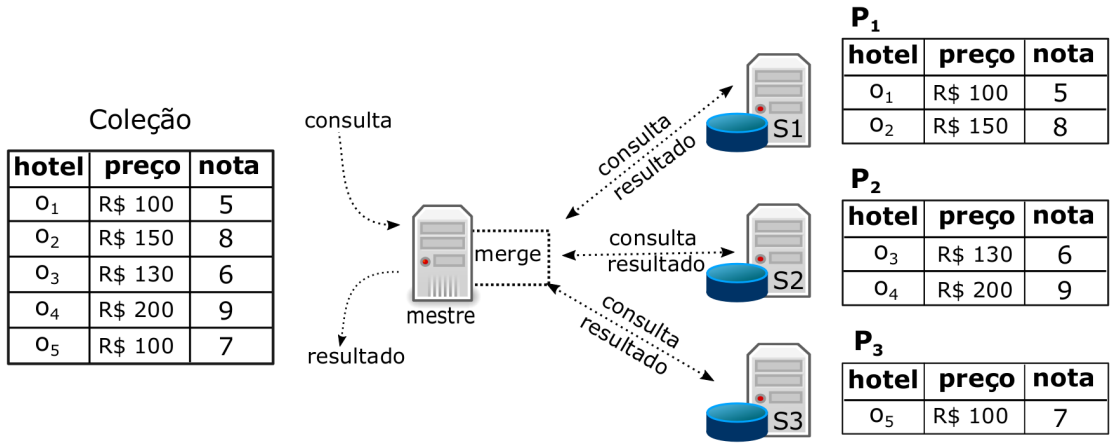


Figura 2.1: Modelo básico para distribuição do processamento de uma consulta.

O particionamento por objetos é uma estrategia de particionamento muito utilizada em sistemas distribuídos [Zobel e Moffat 2006]. Neste particionamento, uma coleção de objetos é dividida e distribuída de forma disjuntas, ou seja, todas as informações de um objeto são alocadas exclusivamente em um servidor. Por exemplo, a Figura 2.1 apresenta um sistema distribuído com 4 servidores, sendo um mestre e 3 escravos. Neste sistema, uma coleção com 5 objetos ($o_1..o_5$) que representam hotéis é particionada em 3 subcoleções (P_1, P_2 e P_3), onde cada subcoleção é formada por objetos distintos e é alocada em um servidor escravo. O servidor S_1 armazena a partição P_1 contendo os objetos o_1 e o_2, o S_2 armazena P_2 contendo o_3 e o_4 e o S_3 armazena P_3 contendo o_5.

O processamento da consulta pode ser feito de forma paralela e sequencial. No processamento paralelo a consulta é enviada simultaneamente para os servidores escravos do sistema distribuído, onde cada servidor processa a consulta de forma concorrente em sua subcoleção [Zobel e Moffat 2006]. Por exemplo, ao consultar os hotéis que possuem preço menor que R$ 150 no sistema distribuído apresentado na Figura 2.1, a consulta é recebida pelo servidor mestre e enviada para os 3 servidores escravos que processam a consulta concorrentemente. Cada escravo retorna um resultado parcial, onde S_1 retorna o objeto o_1, S_2 o objeto o_3 e S_3 o objeto o_5. O servidor mestre retorna o resultado final com os objetos o_1, o_3 e o_5.

No processamento sequencial, a consulta é processada sequencialmente, acessando um servidor por vez [Cambazoglu et al. 2013]. Cada servidor do sistema distribuído processa a consulta na sua subcoleção, mescla com os resultados recebidos da execução anterior e repassa um novo resultado para o próximo servidor, esse processo se repete até que o último servidor do sistema seja alcançada, retornando o resultado final para o servidor de origem. Um plano de execução pode ser adotado para definir a ordem de acesso aos servidores.

Por exemplo, na Figura 2.1 o servidor mestre pode criar um plano de execução com a ordem S_1 -> S_2 -> S_3 para consultar os hotéis que possuem preço menor que R$ 150. No escravo S_1, o objeto o_1 é recuperado e repassado para o escravo S_2 que recupera o objeto o_3 e repassa os objetos o_1 e o_3 para o escravo S_3 que, por fim, recupera o_5 e devolve para o servidor mestre o resultado final com os objetos o_1, o_3 e o_5.

O processamento sequencial geralmente apresenta um tempo de resposta superior (latência) ao processamento paralelo, no entanto, pode reduzir a transferência de dados na rede quando objetos recuperados nos primeiros servidores auxiliam a poda do processamento nos servidores subsequentes [Cambazoglu et al. 2013].

2.2 Consultas Textuais

Existem diversos tipos de documentos disponíveis na internet. Uma parte desses documentos são páginas web, artigos de jornais, publicações acadêmicas, relatórios de empresas, registros históricos, e-mails, dentre outros [Zobel e Moffat 2006]. Além da informação textual, esses documentos podem incluir outros tipos de mídias como: imagens, áudio, vídeos, entretanto, somente o conteúdo textual é considerado para formar coleções (coleção textual) que são utilizadas pelas consultas textuais.

A Tabela 2.1 apresenta uma coleção textual com 7 documentos. Nesta tabela, cada linha representa um documento que são formados por um identificar e um texto, onde o documento 1 possui o texto “restaurante chinês”, o documento 2 o texto “restaurante chinês chinês”, o documento 3 o texto “restaurante japonês”, o documento 4 o texto “restaurante”, o documento 5 o texto “restaurante italiano”, o documento 6 o texto “restaurante mexicano”, o documento 7 o texto “restaurante chinês”.

ID	Texto
1	restaurante chinês
2	restaurante chinês chinês
3	restaurante japonês
4	restaurante
5	restaurante italiano
6	restaurante mexicano
7	restaurante chinês

Tabela 2.1: Coleção textual.

As consultas textuais têm por objetivo recuperar documentos relevantes, em uma coleção textual, para as palavras-chave informadas pelo usuário [Zobel e Moffat 2006]. Essa consulta é utilizada em aplicações como: sistema de ajuda e de busca de arquivos integrados aos sistemas operacionais desktop, bem como em sistemas de busca na web. Para pequenas coleções de dados textuais, a consulta textual pode ser processada acessando todos os documentos existentes, no entanto, para grandes coleções é necessário utilizar estruturas de indexação [Cambazoglu et al. 2006]. A estrutura de indexação mais utilizada para o processamento de consultas textuais é o Arquivo Invertido.

O Arquivo Invertido é composto por dois componentes principais: vocabulário e lista invertida. O vocabulário é formado por todas as palavras distintas encontradas nos documentos de uma coleção, sendo que para cada palavra presente no vocabulário é armazenada a quantidade de documentos que possuem esta palavra (f_t) e uma referência para uma lista invertida. Por exemplo, a Figura 2.2 apresenta um vocabulário composto pelos termos "restaurante", "chinês", "japonês", "italiano" e "mexicano". O termo "restaurante" aparece em 7 documentos, o "chinês" em 3 e o restante em somente um documento da coleção.

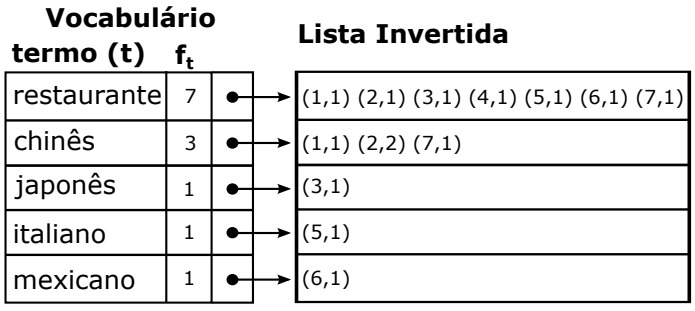


Figura 2.2: Arquivo Invertido.

A Lista invertida (*posting list*) é uma lista com referências para todos os documentos que contém aquela palavra. É formada por pares $(id, f_{d,t})$ onde o id indica o identificador do documento e $f_{d,t}$ a quantidade de vezes que a palavra ocorre dentro

do documento. A Figura 2.2 apresenta a estrutura de um Arquivo Invertido, onde a maior lista invertida é do termo "restaurante" que possui a identificação de 7 documentos e o termo só aparece uma vez em cada documento. O termo "chinês" possui uma lista com a identificação de 3 documentos, onde o termo aparece duas vezes no documento 2 e uma vez nos documentos 1 e 7. Os restante dos termos possuem listas invertidas com a identificação de um documento e cada termo só aparece uma vez neste documento.

Para realizar uma consulta textual no Arquivo Invertido são necessário os seguintes passos: 1) buscar no vocabulário os termos e recuperar as listas invertidas correspondentes, 2) calcular a similaridade entre os termos da consulta e os documentos das listas retornadas, 3) classificar os documentos a partir da relevância textual, e 4) retornar os documentos ordenados pela relevância.

Uma forma de verificar a similaridade textual entre palavras-chave de busca e os documentos de uma coleção é através do modelo vetorial [Zobel e Moffat 2006]. Neste modelo, cada termo pesquisado representa uma dimensão em um espaço n-dimensional e tanto os documentos como a consulta são representados por meio de vetores. Para cada vetor é calculado o ângulo cosseno entre o vetor de consulta e o vetor do documento, quanto menor a distância do vetor de consulta mais similar o documento é considerado. Para realizar este cálculo as seguintes medidas são consideradas: 1) a frequência $f_{d,t}$ do termo t no documento d, 2) a frequência $f_{q,t}$ do termo t na consulta q, 3) a frequência f_t de documentos que contém um termo t, e 4) o número total N de documentos da coleção.

Existem muitas variações da formulação do cosseno, segue um exemplo apresentado por Zobel e Moffat (2006):

$$W_{q,t} = ln(1 + \tfrac{N}{f_t}) \qquad W_{d,t} = 1 + ln f_{d,t}$$

$$W_d = \sqrt{\textstyle\sum_t w_{d,t}^2} \qquad W_q = \sqrt{\textstyle\sum_t w_{q,t}^2}$$

$$S_{q,d} = \frac{\sum_t w_{d,t} \cdot w_{q,t}}{W_d \cdot W_q}$$

Nesta variação o vetor de consulta é representado por $W_{q,t}$ e o vetor de documento por $W_{d,t}$, sendo o *ranking* da pontuação dos documentos calculados pela equação $S_{q,d}$. A equação W_d e W_q representam o comprimento do vetor do documento d e da consulta q. O W_q pode ser desprezado por ser um valor constante na consulta, entretanto é utilizado para manter os resultados normalizados dentro do intervalo 0 a 1.

2.2.1 Consultas Textuais Distribuídas

Para que as consultas textuais possam ser processadas de forma distribuída, inicialmente é necessario realizar o particionamento da coleção textual entre os servi-

dores de um sistema distribuído. O particionamento para coleções textuais geralmente é executado de duas maneiras: uma através do particionamento por objetos apresentado na Seção 2.1.2, que é conhecido como particionamento por documentos e uma outra chamada de particionamento por termos que é o particionamento da estrutura do Arquivo Invertido [Zobel e Moffat 2006, Cambazoglu et al. 2013, Jonassen e Bratsberg 2010, Moffat et al. 2007].

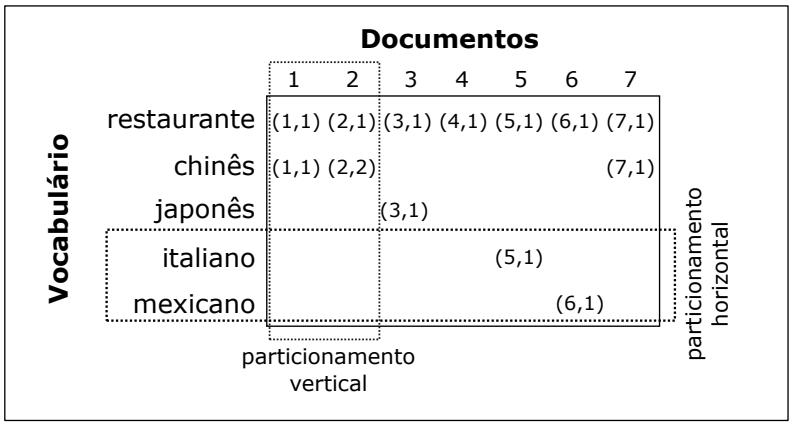


Figura 2.3: Formas de particionamento entre a relação de termos com documentos em uma coleção textual.

A Figura 2.3 exibe uma relação entre termos e documentos de uma coleção textual e a forma como os particionamentos podem ser realizados.

No particionamento por documentos (vertical), a coleção de documentos é particionada em subcoleções e essas são distribuídas entre os servidores de um sistema distribuído. Para cada servidor do sistema é construído um Arquivo Invertido local com a sua subcoleção. Um servidor fica responsável por distribuir as consultas entre todos os servidores e realizar a junção dos subconjuntos retornados. Este modelo tem como vantagem a divisão na carga de trabalho e a facilidade para crescimento da coleção porque somente um servidor necessita atualizar o seu índice para inserir um novo documento [Zobel e Moffat 2006].

No particionamento por termos (horizontal) é realizado uma divisão da estrutura do Arquivo Invertido. O índice é particionado de forma horizontal, onde subconjuntos de termos e suas respectivas lista invertidas são distribuídas entre os servidores de um sistema distribuído. Na Figura 2.3 é possível observar como é realizado o particionamento do índice através dos termos.

Para realizar o particionamento horizontal é necessário que um servidor fique responsável por: construir um índice global com toda a coleção, distribuir partes do índice entre os outros servidores do sistema distribuído e armazenar um vocabulário global que aponta, para cada termo, o servidor que armazena a lista invertida do

termo. Cada servidor escravo do sistema distribuído fica com um Arquivo Invertido para somente alguns termos da coleção.

A Figura 2.4 apresenta um particionamento vertical de um Arquivo Invertido.

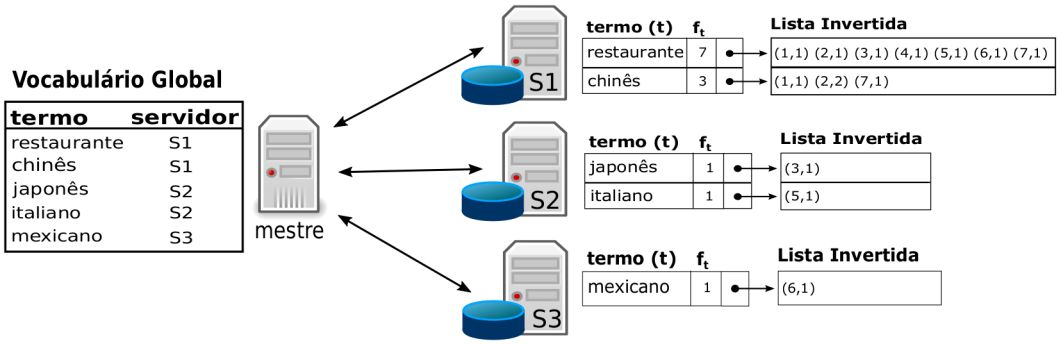


Figura 2.4: Particionamento por termos - Inverted File.

O servidor mestre é responsável por receber as consultas, distribuir entre os servidores que possuem os termos pesquisados, realizar o *merge* dos resultados parciais e retornar para os usuários. Por exemplo, ao realizar uma consulta no modelo que utiliza o particionamento horizontal como apresentado na Figura 2.4 com os termos "restaurante japonês" o servidor mestre distribui a consulta para os servidores S_1 e S_2 e recebe a lista invertida de cada termo para realizar o *merge* e apresentar o resultado final. Neste modelo, uma consulta só pode ser iniciada pelo servidor mestre porque é o único que possui a informação global do índice e somente alguns servidores (os que possuem as listas invertidas dos termos) são utilizados no processamento da consulta, o que permite que várias consultas sejam executadas ao mesmo tempo.

O particionamento horizontal diminui o acesso a disco e a taxa de transferência na rede em comparação com o particionamento por documentos [Zobel e Moffat 2006, Cambazoglu et al. 2013]. No entanto, acaba gerando um desbalanceamento de carga computacional no sistema distribuído porque alguns servidores são sobrecarregados enquanto outros são subutilizados.

Trabalhos foram propostos com o objetivo de minimizar a subutilização dos servidores no modelo que utiliza particionamento horizontal [Cambazoglu et al. 2013]. Algumas propostas replicam as listas invertidas mais consultadas entre os servidores escravos do sistema, enquanto outras criam partições através de hipergrafos e reconstroem o índice periodicamente através de uma avaliação do log das consultas para juntar nos servidores escravos os termos que são geralmente consultados juntos.

Os dois modelos apresentados sofrem com sobrecarga de trabalho no servidor mestre, isso acontece porque muitos resultados parciais são retornados pela rede e esse servidor não é capaz de processar em um tempo satisfatório para o usuário [Moffat et al. 2007, Jonassen e Bratsberg 2010]. Para resolver esse problema de

sobrecarga uma arquitetura em pipeline (processamento sequencial Seção 2.1) foi proposta por [Moffat et al. 2007] e posteriormente por [Jonassen e Bratsberg 2010] para que qualquer servidor envolvido na consulta pudesse realizar uma junção e o ranqueamento dos documentos, restando somente ao servidor mestre apresentar o resultado ao usuário. Entretanto, esse modelo gera uma latência maior no tempo de resposta do sistema distribuído porque não existe paralelismo no processamento da consulta.

2.3 Consultas Espaciais

Atualmente, uma enorme quantidade de objetos espaciais estão sendo criados e compartilhados na internet. Esses objetos representam pontos e formas geométricas como linhas e polígonos para modelar entidades dentro de uma região espacial [Güting 1994]. Por exemplo, considerando a cidade de Feira de Santana como uma região espacial, os estabelecimentos comerciais podem ser representados por pontos, as ruas representadas por linhas e os bairros por polígonos. Neste contexto, o espaço Euclidiano pode ser assumido, onde os objetos são representados através de latitude e longitude e a distância entre os objetos é calculada através da distância Euclidiana.

Muitas aplicações realizam consultas por objetos espaciais (consultas espaciais) para oferecer serviços baseados em localização. O *Foursquare*[1] é um exemplo de sistema que permite aos usuários localizarem outros usuários próximos a sua localização. As consultas espaciais mais utilizadas em Sistemas de Informação é a seleção espacial *range* e a consulta por vizinhos mais próximos (kNN).

A consulta espacial por *range* recupera um conjunto de objetos dentro de uma região espacial. Uma forma de realizar esta consulta é informando como parâmetro uma localização espacial $q.l$ e uma distância r de $q.l$, obtendo com resposta todos os objetos espaciais que possuem uma distância menor ou igual a r da localização $q.l$. Por exemplo: em uma base de dados contendo objetos espaciais que representam todos os bares de Salvador, um turista no centro da cidade pode representar uma consulta espacial *range* para encontrar todos os bares que estão próximos a sua localização dentro de um raio de 150 metros.

A consulta por vizinhos mais próximos visa encontrar os k objetos que estão mais próximos a um local $q.l$ informado no momento da consulta kNN [Güting 1994]. Esta consulta possui como parâmetro uma localização espacial $q.l$ e um valor de k que representa a quantidade de objetos solicitados, a consulta retorna os k objetos mais próximos. Por exemplo: encontre os 4 bares que estão mais próximos da localização especificado pelo usuário.

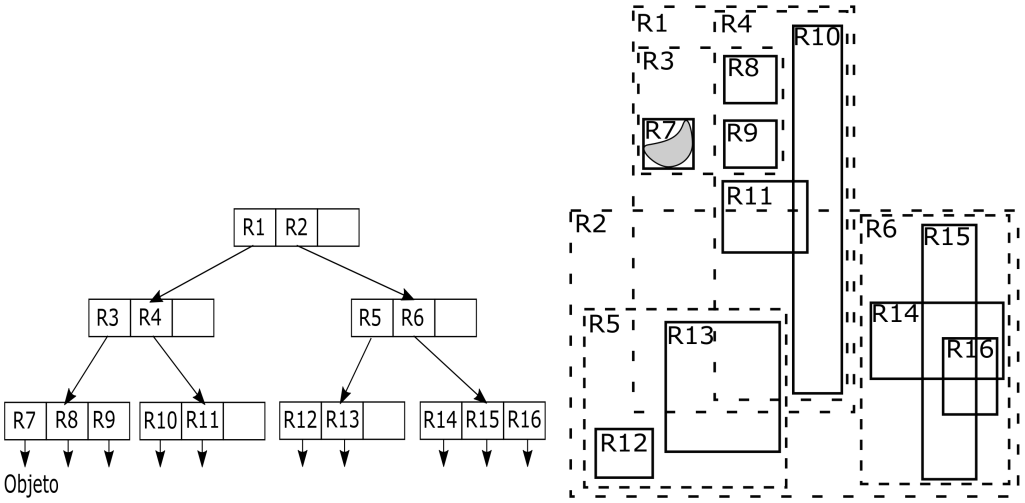
Devido a enorme quantidade de objetos e a necessidade de responder em um tempo satisfatório aos usuários, as consultas espaciais utilizam estruturas de indexação.

[1] https://pt.foursquare.com/

Uma estrutura de indexação espacial organiza o espaço e os objetos para que apenas partes do espaço e dos objetos sejam considerados para responder uma consulta [Güting 1994]. Uma das estruturas de indexação mais utilizadas para dados espaciais é a R-tree que é apresentada na Seção 2.3.1.

2.3.1 R-tree

A R-tree [Guttman 1984] é uma estrutura de dados similar ao índice B-tree, que organiza seus objetos hierarquicamente e as suas rotinas de inserção e exclusão são suficientes para fundir ou dividir nós, mantendo a árvore balanceada. Os nós da R-tree são representados da seguinte forma (*MBR, id*), onde o MBR (*Minimum Bounding Rectangle*) é um retângulo que envolve os objetos indexados e o *id* representa um número identificador quando o nó é folha ou um ponteiro para outro nó quando o nó é intermediário. O MBR de um nó intermediário envolve os MBRs de todos os nós filhos, enquanto o MBR de um nó folha é o menor retângulo que circunda um objeto espacial [Guttman 1984].



(a) Árvore. (b) MBRs.

Figura 2.5: Árvore e MBRs de uma R-Tree. Fonte: Adapdado de Guttman 1984 [Guttman 1984]

A Figura 2.5 apresenta a estrutura de uma R-tree, sendo a Figura 2.5(a) em forma de árvore e a Figura 2.5(b) através de MBRs. Na árvore, os nós R1 e R2 que estão na raiz envolvem todos os outros objetos contidos na árvore, representando a estrutura hierárquica. Essa estrutura permite evitar acessar sub-árvores que não sejam relevantes para a consulta.

A busca na R-tree inicia no nó raiz e percorre a árvore da raiz para as folhas, sempre acessando os nós que são relevantes, à medida que a árvore é percorrida, uma parte

significativa do espaço (nós irrelevantes) é descartada. Por exemplo, para localizar o objeto na região R7 da Figura 2.5 somente os nós R1, R3 e R7 são acessados, sendo todo o restante dos nós descartados.

Quando ocorre sobreposição de retângulos, onde existe uma interseção entre os mesmos, a busca pode percorrer mais de uma subárvore das áreas envolvidas. Por exemplo, para localizar o objeto na região R13 na Figura 2.5 a busca percorre uma subárvore formada pelos nós R1, R3, R4, R10, R11 para depois percorrer a subárvore R2, R5, R13 onde realmente está o objeto. No entanto, nem sempre quando ocorre sobreposição de retângulos existe a necessidade de percorrer todos os nós, por exemplo, para buscar um objeto na região R12 que possui interseção entre R1 e R2 somente os nós R1, R4 e R12 são acessados.

A R-tree é uma das estruturas mais utilizadas para processar consultas espaciais, sendo muito eficiente para processar consultas do tipo *range* e *kNN*.

2.3.2 Consultas Espaciais Distribuídas

A maioria das pesquisas na área de processamento de consultas espaciais distribuídas utilizam o particionamento por objetos apresentado na Seção 2.1.2, onde as partições são criadas levando em consideração à proximidade espacial dos objetos. No entanto, estratégias são criadas para permitir que a consulta seja realizada sem a necessidade de acessar todos os servidores do sistema distribuído.

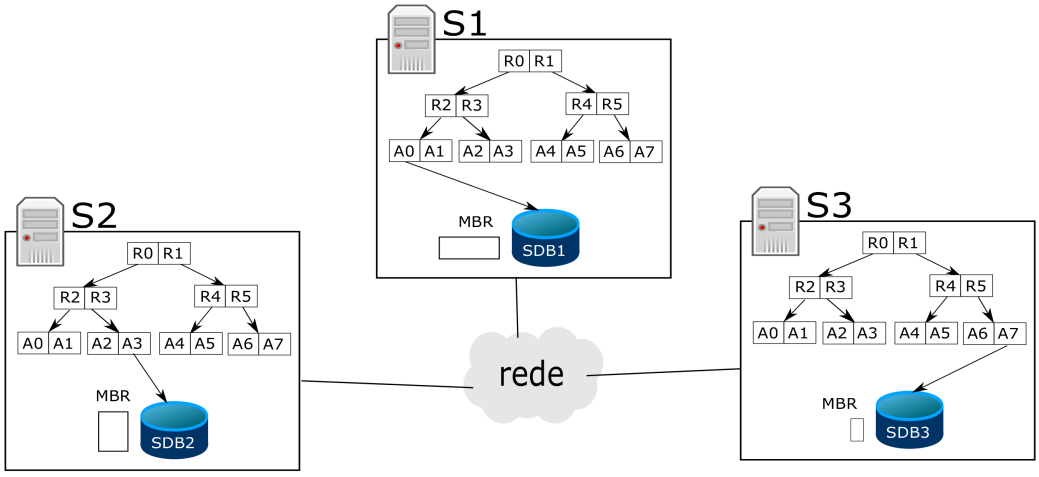


Figura 2.6: R-tree distribuída.

No estudo realizado por Zimmermann et al. [Zimmermann et al. 2004] foi proposto uma abordagem de distribuição do processamento para aumentar a eficiência da consulta espacial e diminuir a troca de mensagens entre os servidores do sistema

distribuído. Neste estudo, cada servidor do sistema distribuído armazena uma subcoleção de dados espaciais em um MBR e constrói um índice espacial (R-tree ou QuadTree) local com as informações dos MBRs dos outros servidores do sistema distribuído. Ou seja, cada servidor tem um índice espacial local com toda a dimensão do conjunto de dados, mas só fica responsável por processar as consultas que solicitam objetos dentro do seu MBR, quando os objetos estão em outro MBR a consulta é repassada automaticamente para o servidor responsável.

A Figura 2.6 apresenta um sistema distribuído com 3 servidores que possuem um índice R-tree, onde cada índice possui um MBR armazenado no servidor local e os outros são referências para outros servidores. Quando uma consulta é realizada, a mesma é repassada automaticamente para o servidor responsável ou para os servidores que possuem sobreposição de regiões no caso da R-tree. Isso diminui o tráfego de informação na rede e a necessidade de consultar todos os servidores [Zimmermann et al. 2004].

2.4 Consulta Espaço-Textual Top-k

A consulta espaço-textual top-k recupera objetos em ordem decrescente de escore, onde o escore de cada objeto é composto por uma combinação entre a distância do objeto até o local da consulta e a relevância da descrição textual do objeto para as palavras-chave pesquisadas [Rocha-Junior et al. 2011, Cao et al. 2012]. Para realizar uma consulta espaço-textual top-k é necessário informar três parâmetros: 1) um local de interesse (latitude e longitude), 2) um conjunto de palavras-chave , e 3) a quantidade de objetos desejados (k).

A Pontuação do *ranking* dos objetos para uma consulta espaço-textual top-k é definida pela seguinte equação:

$$rank(p, q) = \alpha \cdot \delta(p.l, q.l) + (1 - \alpha) \cdot \theta(p.d, q.d)$$

O escore espacial $\delta(p.l, q.l)$ é um valor normalizado definido pela equação $\delta(p.l, q.l) = 1 - (d(p.l, q.l)/dmax)$, onde $d(p.l, q.l)$ é a distância entre os objetos $p.l$ e $q.l$, e $dmax$ é a maior distância entre dois objetos da região espacial analisada. Para que os objetos mais próximos da localização da consulta $q.l$ possuam maiores escores, é utilizado o maior valor do escore (1) menos a distância espacial normalizada.

A relevância textual $\theta(o.d, p.d)$ computa a similaridade entre o texto do objeto $o.d$ e o texto das palavras-chave da consulta $q.d$. Essa similaridade pode ser calculada de diversas formas diferentes, sendo o modelo vetorial (Seção 2.2) uma das formas mais utilizadas.

As duas medidas, retornam valores normalizados dentro do intervalo [0, 1] e podem ter seus pesos diferenciados pelo parâmetro $\alpha \in (0, 1)$ que define a importância de uma medida sobre a outra [Rocha-Junior et al. 2011]. Por exemplo, $\alpha = 0{,}3$

significa que a proximidade espacial tem uma importância menor que a relevância textual.

Na Seção 2.4.1 contém a descrição de estruturas de indexação utilizadas para processar consultas espaço-textuais.

2.4.1 Estruturas de indexação para consultas Espaço-Textuais

A maioria das estruturas de indexação para objetos espaço-textuais combinam índices textuais e espaciais. Os índices textuais são utilizados para realizar a busca através das palavras-chave e índices espaciais para realizarem a pesquisa por objetos que atendem ao local de interesse especificado pelo usuário.

Zhou et al. [Zhou et al. 2005] combinou índices espaciais e textuais para criar três estruturas eficientes no processamento das consultas espaço-textuais. A primeira estrutura foi criada com dois índices independentes, sendo um Arquivo Invertido e uma R*-tree. A segunda utilizou o Arquivo Invertido como base para o índice e cada termo do vocabulário apontou para uma R*-tree. A última estrutura utilizou a R*-tree como base, sendo adicionado um Arquivo Invertido em cada nó intermediário da R*-tree com a informação textual das suas sub-árvores.

As estruturas híbridas apresentaram um melhor desempenho para o processamento da consulta perante a estrutura com índices independentes. Dentre as estruturas híbridas, a que utilizou o Arquivo Invertido como base para o índice foi a que obteve o melhor desempenho [Zhou et al. 2005]. Entretanto, estas estruturas não realizam consultas espaço-textuais top-k, realizam somente consultas Booleanas, onde todos os termos pesquisados devem conter na descrição textual dos objetos.

Outras estruturas de índices híbridos que utilizam a R-tree como base foram propostas para otimizar as consultas espaço-textuais [Cong et al. 2009, Zhou et al. 2005, Li et al. 2011, De Felipe et al. 2008]. A IR^2-tree proposta por [De Felipe et al. 2008] agrega arquivos de assinaturas aos nós da R-tree para indicar a presença de termos nos objetos espaço-textuais armazenados nas suas sub-árvores. Esta estrutura realiza uma consulta que se aproxima da consulta espaço-textual top-k por levar em consideração uma posição inicial e retornar os k objetos solicitados na consulta, porém, se diferencia por realizar a consulta Booleana.

Os primeiros índices híbridos desenvolvidos para consultas espaço-textuais top-k foram propostos por [Cong et al. 2009] e [Li et al. 2011]. As duas estruturas receberam o nome IR-Tree e ambas agregam Arquivos Invertidos aos nós da árvore R-tree com as informações textuais dos objetos contidos nas suas sub-árvores. A junção desses dois índices permite que a IR-Tree execute uma poda simultânea entre distância espacial e relevância textual, deixando a consulta eficiente.

A Figura 2.7 apresenta a estrutura de uma IR-Tree criada com uma coleção de 7 objetos ($o_1..o_7$), onde R_i representa os MBRs que envolvem os objetos e os InvFile$_i$ os

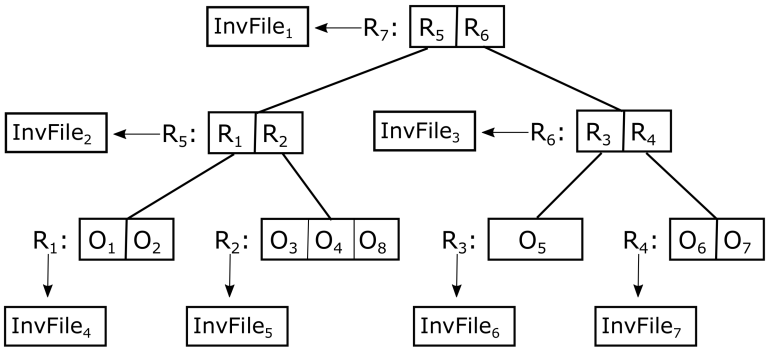


Figura 2.7: Estrutura de uma IR-tree

Arquivos Invertidos de cada região contendo os termos dos objetos das sub-árvores de cada nó.

O Estudo realizado por Cong et al. ainda propôs variações da IR-Tree com os nomes de DIR-tree, CIR-tree, CDIR-tree. A DIR-Tree utiliza no momento da construção da árvore R-Tree a localização espacial e a descrição textual dos objetos, onde os objetos presentes em um mesmo MBR possuem proximidade espacial e semelhança na descrição textual. A CIR-tree é uma derivação da IR-Tree que busca aperfeiçoar o processamento das consultas através de agrupamentos dos objetos com base em suas descrições textuais, porém, é gerada mais uma informação agregada aos nós da R-tree. O CDIR-Tree é uma derivação da DIR-Tree com os agrupamentos da CIR-Tree [Cong et al. 2009].

Rocha-Junior et al. [Rocha-Junior et al. 2011] propôs um índice chamado índice invertido espacial (S2I) que utilizou a estrutura do Arquivo Invertido para mapear os termos distintos das informações textuais dos objetos, entretanto, a lista com as referências desses objetos é armazenada de forma diferente, podendo ser alocada em blocos ou em árvores multidimensionais. A escolha da forma de armazenamento dos objetos depende da frequência do termo na coleção. Termos frequentes têm seus objetos armazenados em árvores multidimensionais e termos com pouca frequência têm seus objetos armazenados em blocos.

Chen et al. [Chen et al. 2013] realizou um estudo que avaliou 12 (doze) estruturas de indexação propostas para otimizar as consultas espaço-textuais. Segundo este estudo o S2I está entre os índices com melhor desempenho para consultas espaço-textuais top-k.

Enquanto novas estruturas híbridas continuavam surgindo como o I^3 [Zhang et al. 2013] que utilizou a mesma metodologia do S2I para termos frequentes e não frequentes, apenas alterando o índice espacial R-tree por uma QuadTree [Samet 1984] para os termos frequentes. Outros índices surgiram sem utilizar estruturas híbridas. Neste caso, índices espaciais e textuais são criados separadamente, tendo como maior desafio o desenvolvimento de algoritmos que possibilitem realizar a junção dos resultados de forma eficiente. O RASIM

[Kwon et al. 2013] criou estruturas de indexação separadas sendo uma um Arquivo Invertido e a outra uma variação da R-tree. No Arquivo Invertido os objetos são ordenados pela frequência dos termos no objeto, enquanto na R-tree são ordenados por *ID*. Um algoritmo de acesso aleatório é utilizado para realizar a junção das listas de objetos de cada índice específicos e selecionar os melhores resultados.

Recentemente, um estudo foi realizado com as estruturas de indexação mais utilizadas para consultas espaço-textuais top-k. Um framework chamado G-Index [Kwon et al. 2015] foi implementado com os métodos de pesquisas utilizados para buscas nas estruturas de indexação citadas anteriormente S2I, IR-TREE e RASIM. O framework utiliza o método mais adequado para consultas espaço-textuais top-k de acordo com a consulta realizada pelo usuário. Por exemplo, para consultas com poucas palavras-chave um método de consulta igual ao índice S2I é utilizado, para muitas é utilizado o método similar ao RASIM.

Todas as estruturas citadas anteriormente foram desenvolvidas para serem executadas de forma centralizada. As estruturas que são baseadas no Arquivo Invertido facilitam o particionamento dos dados, podendo ser realizado tanto pelos termos do vocabulário como pelos objetos. Diante dessa facilidade, o índice S2I é utilizado nesta pesquisa para avaliar e identificar a melhor estratégia de particionamento de dados. Por isso o S2I é abordado em maiores detalhes na Seção 2.4.2.

2.4.2 S2I

O S2I (*Spatial Inverted Index*) [Rocha-Junior et al. 2011] é semelhante a um Arquivo Invertido (Figura 2.2). Essa estrutura mantém um vocabulário com todos os termos distintos encontrados nas informações textuais dos objetos, e para cada termo armazena o conjunto de objetos espaço-textuais relevantes. Ao invés de armazenar os objetos em listas, o S2I armazena os objetos em uma estrutura de bloco ou de árvore, onde a escolha da forma de armazenamento dos objetos depende da frequência do termo na coleção. Termos frequentes têm seus objetos armazenados em árvores multidimensionais e termos com pouca frequência têm seus objetos armazenados em blocos.

O S2I utiliza um método simples para definir se o termo é frequente ou infrequente. Inicialmente, todos os termos são considerados infrequentes e seus objetos são armazenados em blocos de tamanho fixo, quando a quantidade de objetos supera a capacidade máxima de armazenamento do bloco, esse termo passa a ser considerado frequente e seus objetos são armazenados em uma árvore multidimensional.

Além de termos distintos, o vocabulário armazena para cada termo a quantidade de objetos que contém esse termo, um flag indicando se o armazenamento dos objetos é através de bloco ou árvore e um indicador para a estrutura que os objetos estão armazenados [Rocha-Junior et al. 2011]. A estrutura de bloco tem tamanho fixo, definido no momento da criação do S2I e é capaz de armazenar uma quantidade

fixa de objetos. Em cada bloco os objetos são armazenados sem nenhum critério de ordenação e contendo as seguintes informações: identificação, localização espacial (latitude e longitude) e o impacto do termo nesse objeto.

O impacto λ do termo é um valor normalizado que define o peso de um termo em um objeto. Esse valor pode ser utilizado para comparar a relevância textual de um tempo em dois objetos diferentes que possuam esse termo nas suas informações textuais [Rocha-Junior et al. 2011]. O S2I utiliza a abordagem empregada por Zobel e Moffat [Zobel e Moffat 2006] (Seção 2.2) para calcular o escore textual dos objetos, entretanto a equação é reescrita em termos de impacto, onde o impacto de um termo t em um documento d é definido por: $\lambda_{d,t} = \frac{W_{d,t}}{W_d}$ e o impacto de um termo t em uma consulta q é definido por: $\lambda_{q,t} = \frac{W_{q,t}}{W_q}$, reescrevendo a equação do escore textual para $\theta(o.d, q.d) = \sum_{t \in q.d} \lambda_{q,t} \cdot \lambda_{d,t}$.

A estrutura de árvore utilizada pelo S2I é a aR-tree. A aR-tree [Papadias et al. 2001] possui a mesma estrutura de uma R-tree (Seção 2.3.1) tradicional, porém uma informação não espacial é agregada em cada nó para representar os objetos que estão na sua sub-árvore. O S2I utiliza esse atributo para armazenar o maior impacto entre os objetos presentes nas subárvores de um nó, permitindo evitar o acesso a subárvores que não possuam objetos relevantes [Rocha-Junior et al. 2011].

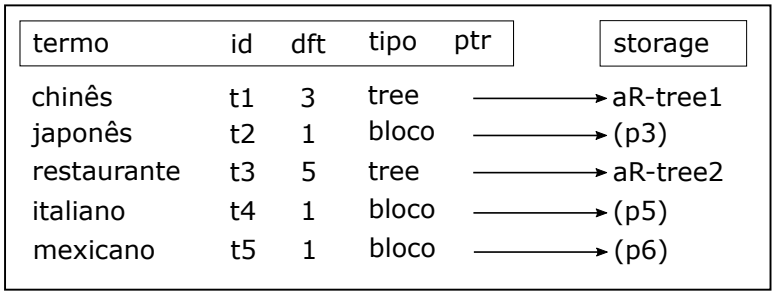


Figura 2.8: Estrutura do S2I.

A Figura 2.8 apresenta a estrutura do S2I. Os termos "restaurante" e "chinês", que aparecem com uma frequência maior na coleção de objetos espaço-textuais, apontam para estruturas aR-tree e o restante dos termos apontam para uma estrutura de bloco.

O S2I utiliza dois algoritmos (SKA e MKA) para processar as consultas espaço-textual top-k. O algoritmo SKA apresenta um bom desempenho para processar consultas com um termo porque acessa somente uma árvore aR-tree ou um bloco para o seu processamento. Já o MKA executa as consultas com mais de um termo e também apresenta bom desempenho porque acessa somente alguns nós de um conjunto de árvores de pequeno porte ou blocos. Ambos os algoritmos (SKA e MKA) recuperam os objetos de forma incremental e ordenados pelo escore, armazenando

os k melhores em uma pilha enquanto novos objetos são recuperados até que o valor do menor escore entre os objetos que estão na pilha não consiga ser alcançado.

Apesar de apresentar bom desempenho no processamento da consulta espaço-textual top-k, o S2I utiliza muito espaço em disco para armazenar sua estrutura [Chen et al. 2013]. A estratégia utilizada no S2I em armazenar os objetos em um bloco de tamanho fixo para cada termo infrequente, permite que blocos sejam reservados e não sejam preenchidos em sua totalidade [Athayde-Novaes et al. 2016]. No decorrer desta pesquisa, uma variação do S2I foi proposta com um método diferente para definir termos frequentes e infrequentes e com novas formas de armazenar os dados para reduzir o espaço de armazenamento em disco (S2I+ [Athayde-Novaes et al. 2016]).

No S2I+ [Athayde-Novaes et al. 2016] um termo só é considerado frequente quando a quantidade de objetos relevantes para esse termo não podem ser armazenadas em 6 (seis) blocos. Em um experimento realizado comparando a estrutura de bloco e árvore, onde permitiu-se que os termos infrequentes armazenassem seus objetos em mais de um bloco, apresentou que a aR-tree só consegue ter um desempenho melhor (acesso a páginas do disco e tempo de resposta) que a estrutura de blocos quando o número de objetos é superior a quantidade suportada por 6 blocos.

O S2I+ [Athayde-Novaes et al. 2016] utiliza duas novas formas de armazenar os objetos (MTB e SEQ). O MTB (Múltiplos termos por bloco) permite que os objetos de mais de um termo infrequente ocupem o mesmo bloco, reduzindo o espaço vazio e evitando que o índice reserve espaço sem necessidade. Já o SEQ (Armazenamento Sequencial) armazena os objetos de forma sequencial em um arquivo, onde cada bloco tem tamanho ideal para armazenar os objetos de um termo. No SEQ cada termo infrequente aponta para uma posição do arquivo onde inicia o bloco dos seus objetos.

As duas formas (MTB e SEQ) proporcionam um redução significatica do espaço utilizado pelo índice além de apresentarem melhores tempos de resposta para as consultas realizadas [Athayde-Novaes et al. 2016].

2.4.3 Consultas Espaço-Textuais Distribuídas

Alguns estudos foram realizados utilizando sistemas distribuídos para permitir o processamento de consultas espaço-textuais distribuídas [He et al. 2015, Porwal e Shiwani 2015, Doulkeridis et al. 2017]. Nestes estudos, as abordagens propostas utilizam o particionamento por objetos apresentado na Seção 2.1.2 e consideram a informação espacial para criar partições de objetos próximos um dos outros. No entanto, são diferentes na forma como os objetos espaciais são agrupados e no tipo de consulta espaço-textual utilizada para recuperar os objetos no ambiente distribuído.

He et. al [He et al. 2015] utilizou um índice espacial em grade para dividir a região espacial em celulas quadradas de tamanhos iguais. Para realizar a consulta textual, foi criado um vocabulário com todas as palavras-chave distintas do conjunto de dados e para cada termo do vocabulário é armazenada uma lista com as células que possuem objetos com esse termo [He et al. 2015]. O processamento da consulta inicia na célula onde está o local informado para consulta e é verificado se nessa célula existem todas as palavras-chave pesquisadas, caso não existam, a consulta é ampliada para as células vizinhas até encontrar os objetos que cubram todas as palavras-chave [He et al. 2015]. Neste artigo a abordagem proposta avalia somente uma forma de particionamento e não realiza a consulta espaço-textual top-k.

Em 2015, outro artigo foi publicado propondo o processamento de uma consulta por objetos espaço-textuais de forma distribuída [Porwal e Shiwani 2015]. Um índice espacial chamado GeoHash, que é similar ao índice em grade, foi utilizado para particionar a coleção. Neste estudo, o método proposto realiza a consulta em duas etapas: 1) os objetos são recuperados através da proximidade espacial com o local informado na consulta, e 2) cada objeto é processado para verificar se satisfaz para as palavras-chave informada na consulta. Neste artigo, além de só estudar uma forma de particionamento, o método proposto utiliza uma consulta textual diferente da consulta estudada nesta pesquisa.

Recentemente, Doulkeridis et. al [Doulkeridis et al. 2017] publicou um trabalho propondo o processamento paralelo e distribuído da consulta espacial preferencial usando palavras-chave. A coleção de objetos é particionada em células utilizando um algoritmo de grade. Para processar uma consulta, além dos parâmetros exigidos pela consulta espaço-textual top-k (localização, palavras-chave e k), é exigido um valor de raio para definir um limite na região espacial pesquisada. O processamento da consulta inicia calculando a distância mínima do ponto informado para cada célula da grade, as células com distância maior que o raio são eliminadas e a consulta é processada em paralelo nas células selecionadas. A abordagem proposta neste artigo também só utiliza uma forma de particionamento e não realiza a consulta estudada nesta pesquisa por necessitar de um parâmetro r para limitar a região espacial pesquisada.

Capítulo 3

Definição do Problema

"Sorte é aquilo que acontece quando a preparação encontra a oportunidade."

– Elmer Letterman

Dado um conjunto C de objetos espaço-textuais, onde cada objeto $o \in C$ é formado por uma identificação $o.id$, uma localização espacial $o.l$ (latitude e longitude) e uma informação textual $o.d$ que descreve este objeto, como por exemplo o nome de um estabelecimento. Ao realizar uma consulta espaço-textual top-k q composta por um conjunto de palavras-chave $q.d$, uma localização espacial $q.l$ e a quantidade de objetos desejados $q.k$. A consulta q retorna um conjunto com $q.k$ objetos $\{o_1..o_k\} \in C$ com os maiores escores, sendo que o escore de cada objeto é calculado utilizando a seguinte equação:

$$rank(o, q) = \alpha \cdot \delta(o.l, q.l) + (1 - \alpha) \cdot \theta(o.d, q.d)$$

O escore espacial $\delta(o.l,\ q.l)$ é calculado através da distância euclidiana entre a localização do objeto selecionado $o.l$ e a localização da consulta $q.l$ (Seção 2.4), enquanto a relevância textual $\theta(o.d, p.d)$ utiliza a equação do modelo vetorial (Seção 2.2) reescrita em termos de impacto (Seção 2.4.2) para computar a similaridade entre o texto do objeto $o.d$ e as palavras-chave da consulta $q.d$. As duas medidas, retornam valores normalizados dentro do intervalo [0, 1] e podem ter seus pesos diferenciados pelo parâmetro $\alpha \in (0,\ 1)$ que define a importância de uma medida sobre a outra [Rocha-Junior et al. 2011]. Por exemplo, $\alpha = 0{,}3$ significa que a proximidade espacial tem uma importância menor que a relevância textual. Existem outras formas, como *learn to rank* [Li 2011], que podem ser utilizadas para ranquear os objetos, no entanto não fazem parte do escopo desta pesquisa.

O objetivo dessa pesquisa é realizar o processamento distribuído da consulta espaço-textual top-k, estudando diferentes formas de particionar o conjunto C de objetos

espaço-textuais. Nesta pesquisa, assume-se um ambiente distribuído composto por 1 servidor mestre M e n servidores escravos S_i, onde cada servidor escravo S_i armazena uma parte da coleção $C_i \subset C$ e $\cup_{1 \leq i \leq n-1} C_i = C$. Todos os servidores M e S_i podem se conectar diretamente com qualquer outro servidor do sistema, no entanto, somente os escravos S_i são capazes de processar a consulta espaço-textual top-k sobre os dados armazenados localmente C_i. O servidor mestre M é responsável por distribuir a consulta entre os escravos S_i e processar o resultado global. O resultado global da consulta espaço-textual top-k é um sub-conjunto da união dos resultados locais processados em cada servidor escravo. Assim, agregando e realizando a junção dos dados locais é possível computar o resultado global da consulta.

Como os escores dos objetos são computados nos escravos e as informações necessárias para calcular o impacto do termo na consulta $\lambda_{q,t}$ (Seção 2.4.2) são globais da coleção, o servidor mestre M armazena o número total de objetos da coleção N e um vocabulário idêntico ao do Arquivo Invertido (Figura 2.2) com todos os termos dessa coleção e a frequência f_t de cada termo. Dessa forma, o valor do impacto $\lambda_{q,t}$ de cada termo pesquisado é calculado no servidor mestre M e esses valores são enviados com a consulta para que os servidores escravos S_i possam calcular os escores dos objetos.

Duas formas de processamento distribuído das consultas são consideradas nesta pesquisa: o paralelo e o sequencial. No processamento paralelo, o servidor mestre M envia a consulta espaço-textual top-k para os servidores escravos S_i, onde cada servidor S_i realiza o processamento da consulta em sua subcoleção C_i e retorna os objetos selecionados para o servidor mestre M. O servidor mestre M realiza um ranqueamento com os objetos recebidos de cada servidor escravo S_i e retorna os k melhores.

No processamento sequencial, o servidor mestre M cria um plano de execução com a ordem dos servidores escravos S_i em que a consulta é processada. Cada servidor escravo S_i processa a consulta na sua subcoleção, mescla com os resultados recebidos da execução anterior, envia os k melhores objetos para o próximo servidor escravo S_i e assim sucessivamente até que o último escravo S_i seja alcançado, retornando o resultado final para o servidor mestre M.

Esta pesquisa avalia somente o processamento distribuído da consulta espaço textual top-k. Sendo assim, questões como segurança da rede, replicação dos dados e confiabilidade do sistema não fazem parte do escopo desta pesquisa. Entretanto, estas questões são importantes e devem ser consideradas em trabalhos futuros.

Capítulo 4

Modelos de Particionamento

Neste Capítulo são apresentados quatro modelos de particionamento de dados para o processamento distribuído da consulta espaço-textual top-k. Dentre os quatro modelos, três são baseados no particionamento por objetos apresentado na Seção 2.1.2 de Consultas Distribuídas e o outro modelo é baseado no particionamento por termos citado na Seção 2.2.1 de Consultas Textuais Distribuídas.

Os modelos de particionamento por objetos consistem em dividir e distribuir a coleção de objetos espaço-textuais de forma disjuntas, ou seja, todas as informações de um objeto são alocadas exclusivamente em um servidor. No entanto, a divisão da coleção pode ocorrer de diversas formas diferentes. Neste estudo optou-se por particionar o conjunto de objetos de três formas: 1) de forma aleatória, 2) pela similaridade textual e 3) pela localização espacial.

No modelo de particionamento por termos a coleção é particionada de acordo com os termos (palavras-chave) do texto presente nos objetos, onde todos os objetos que possuem um determinado termo ficam agrupados no mesmo servidor. Neste modelo, servidores diferentes podem ficar responsáveis por diferentes termos de um único objeto, ou seja, as informações de um objeto podem ficar armazenadas em mais de um servidor.

Em cada modelo são realizadas as seguintes etapas: particionamento, distribuição, construção de estruturas auxiliares e processamento da consulta. A etapa de particionamento consiste em realizar a divisão da coleção em subcoleções de acordo com a quantidade de servidores escravos do sistema distribuído. Na etapa de distribuição, o servidor mestre fica responsável por enviar cada subcoleção, criada na etapa de particionamento, para um servidor escravo do sistema distribuído. Essa etapa é idêntica para todos os modelos.

A etapa de construção de estruturas auxiliares consiste em criar estruturas para auxiliar o processamento da consulta espaço-textual top-k. Para todos os modelos, em cada servidor escravo do sistema distribuído é criado um índice específico para processamento de consultas espaço-textual top-k, permitindo que os objetos sejam

recuperados de forma eficiente. No entanto, outras estruturas podem ser criadas para auxiliar o processamento da consulta.

Na etapa de processamento da consulta é apresentado como a consulta espaço-textual top-k é processada no sistema distribuído tanto de forma paralela como de forma sequencial, utilizando as estruturas auxiliares criadas. O restante deste capítulo aborda em maiores detalhes cada modelo de particionamento e como as consultas espaço-textual top-k são processadas em paralelo e sequencial.

4.1 Particionamento Aleatório

O modelo de particionamento aleatório (MA) consiste em dividir e distribuir a coleção de objetos espaço-textuais de forma aleatória entre os servidores escravos de um sistema distribuído. Neste modelo é realizado uma divisão do conjunto de objetos espaço-textuais sem nenhum critério de similaridade entre os mesmos, onde subconjuntos são criados com objetos selecionados de forma aleatória.

Particionamento. O particionamento da coleção é realizado respeitando um balanceamento entre a quantidade de objetos de cada partição. Os subconjuntos são criados através de uma distribuição uniforme dos objetos entre os servidores do sistema, onde a quantidade de objetos de cada subconjunto pode ser definida através da divisão do número total de objetos da coleção pelo número de servidores escravos do sistema distribuído. Caso a divisão não seja exata, o restante dos objetos são enviados um para cada servidor de forma aleatória. Por exemplo: para uma coleção com 101 objetos e um sistema distribuído com 5 servidores escravos, são gerados 4 subcoleções com 20 objetos e 1 com 21 objetos.

Estruturas auxiliares. Nenhuma estrutura auxiliar é necessária, somente o índice específico para consultas espaço-textuais top-k.

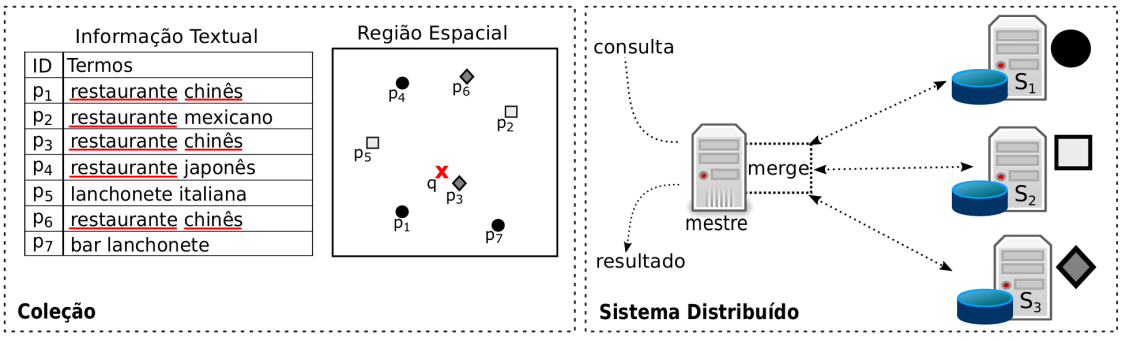


Figura 4.1: Processamento paralelo da consulta espaço-textual top-k no particionamento aleatório.

Processamento da Consulta. O processamento paralelo das consultas é idêntico ao apresentado na Seção 2.1.2, onde a consulta é repassada para todos os servidores

ao mesmo tempo e cada servidor retorna somente os k melhores objetos da sua subcoleção, restando o servidor mestre realizar um ranqueamento dos objetos recebidos e retornar os k melhores.

A Figura 4.1 apresenta o particionamento aleatório em uma região espacial, onde os objetos p_1, p_4 e p_7 (representados através de círculos) são indexados no servidor S_1, os p_2 e p_5 (representados através de quadrados) são indexados no servidor S_2 e os objetos p_3 e p_6 (representados através de losangos) no servidor S_3. Ao realizar uma consulta espaço-textual top-k com as palavras-chave "restaurante chinês", a posição do usuário representado pelo ponto q e $k = 2$, o servidor mestre envia a consulta para todos os servidores (S_1, S_2 e S_3) e aguarda a resposta com os 2 melhores objetos de cada. O servidor S_1 retorna os objetos p_1 e p_4, o S_2 retorna o objeto p_2 e o S_3 retorna os objetos p_3 e p_6. O servidor mestre realiza um ranqueamento entre os 5 objetos recebidos e retorna os objetos p_3 e p_1 como resultado final, por apresentarem descrições textuais com maior número de palavras-chave pesquisadas e uma proximidade espacial maior para o local de busca.

No processamento sequencial (Seção 2.1), o plano de execução da consulta é realizado de forma aleatória, ou seja, a ordem definida pode ser qualquer combinação dos servidores escravos.

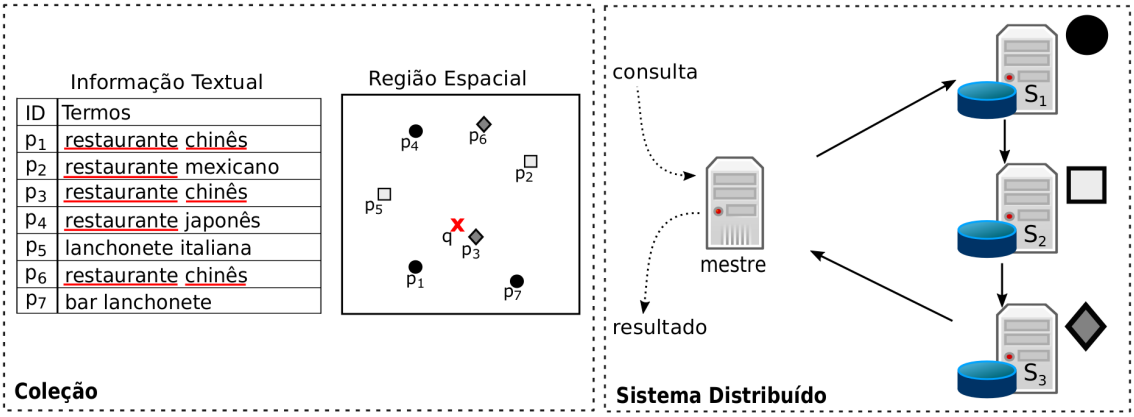


Figura 4.2: Processamento sequencial da consulta espaço-textual top-k no particionamento aleatório.

A Figura 4.2 apresenta o processamento de uma consulta de forma sequencial. Ao realizar uma consulta espaço-textual top-k, com as palavras-chave "restaurante chinês", a posição do usuário representado pelo ponto q e $k = 2$, o servidor mestre cria um plano de execução com a seguinte ordem de acesso aos servidores S_1 -> S_2 -> S_3. O processamento inicia no servidor S_1 que seleciona os objetos p_1 e p_4, repassando para o servidor S_2 que seleciona o objeto p_2 e compara o escore com os objetos recebidos (p_1 e p_4), seleciona os 2 melhores e envia para o S_3. O S_3 seleciona os objetos p_3 e p_6, compara com os 2 melhores da execução anterior e envia para o servidor mestre os objetos p_3 e p_1 como resultado final.

4.2 Particionamento por Similaridade Textual

O modelo de particionamento por similaridade textual (MST) consiste em distribuir os objetos entre os servidores escravos a partir da similaridade entre as informações textuais dos objetos. Neste modelo, é realizado uma divisão do conjunto de objetos espaço-textuais considerando as informações textuais dos objetos para encontrar grupos de objetos que abordem conteúdos textuais similares.

Particionamento. Um dos algoritmos da técnica de agrupamento (*clustering*) da mineração de dados pode ser utilizado para encontrar grupos de objetos com similaridade textual. Por exemplo, o *k-means* [Witten e Frank 2005] é um algoritmo de agrupamento que identifica n grupos em uma coleção considerando a similaridade textual dos objetos. O valor de n é um parâmetro informado no momento da consulta para determinar a quantidade de agrupamentos desejados. Portanto, o *k-means* pode ser utilizado com valor de n igual a quantidade de servidores escravos do sistema distribuído para criar as subcoleções necessárias para o modelo.

Estruturas auxiliares. No momento em que o servidor mestre realiza a distribuição dos objetos, um vocabulário é criado com todos os termos presentes na coleção de objetos espaço-textuais e para cada termo são armazenados os servidores que possuem os objetos com esse termo. Com o auxílio do vocabulário é possível eliminar servidores que não possuem as palavras-chave pesquisadas, diminuindo a quantidade de servidores acessados tanto no processamento paralelo como sequencial da consulta espaço-textual top-k.

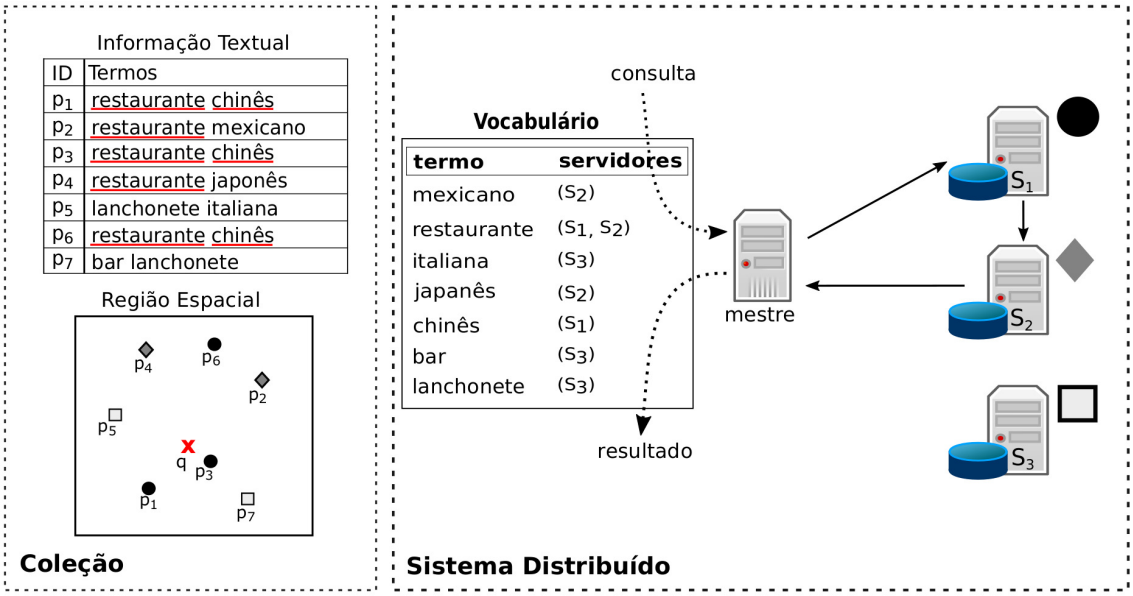


Figura 4.3: Distribuição por similaridade textual.

A Figura 4.3 apresenta uma distribuição dos objetos através do particionamento por similaridade textual, onde os objetos p_1, p_3 e p_6 são indexados no servidor S_1, os p_2 e p_4 no servidor S_2 e os objetos p_5 e p_7 no servidor S_3. O servidor mestre possui um vocabulário com todos os termos da coleção ("mexicano", "restaurante", "italiana", "japonês", "chines", "bar"e "lanchonete") e para cada termo, a identificação do servidor que armazena os objetos com esse termo. Neste modelo, um termo pode ter seus objetos em mais de um servidor, o que pode ser observado com o termo "restaurante".

Processamento da Consulta. Neste modelo, o processamento paralelo das consultas é idêntico ao apresentado no modelo aleatório (Seção 4.1), sendo que nem todos os escravos podem ser utilizados no processamento, a consulta é enviada somente para os servidores que possuem os termos pesquisados.

No processamento sequencial, o plano de execução é criado eliminando os servidores que não possuem os termos pesquisados, iniciando no servidor escravo que possui mais palavras-chave de busca e finalizando no escravo que possui a quantidade menor dos termos.

Ao realizar uma consulta espaço-textual top-k, na região espacial da Figura 4.3, com as palavras-chave "restaurante chinês", a posição do usuário representado pelo ponto q e $k = 2$. O servidor mestre consulta o vocabulário pelas palavras-chave de busca e seleciona somente os servidores S_1 e S_2 que possuem pelos menos uma das palavras-chave de busca. No processamento paralelo, a consulta é repassada para os escravos S_1 e S_2 processarem concorrentemente, enquanto que no processamento sequencial o processamento inicia no escravo S_1 porque possui as duas palavras-chave e termina no S_2. O servidor S_1 seleciona os objetos p_1, p_3 e p_6 e envia para o S_2 os 2 melhores objetos (p_3 e p_1), S_2 seleciona p_2 e p_4, compara os escores com os objetos recebidos e envia os 2 melhores (p_3 e p_1) para o servidor mestre.

4.3 Particionamento por Localização Espacial

O modelo de particionamento por localização espacial (MLE) distribui os objetos entre os servidores escravos do sistema distribuído através da sua localização espacial. Este modelo realiza uma divisão da coleção de objetos espaço-textuais considerando a localização espacial de cada objeto, para criar subcoleções com objetos próximos (espacialmente) uns dos outros.

Particionamento. Assim como no modelo por similaridade textual (Seção 4.2), um algoritmo de agrupamento pode ser utilizado para identificar grupos de objetos considerando a distância entre os objetos. O algoritmo *k-means* pode ser utilizado para particionar a coleção em subcoleções de objetos proximos uns dos outros. Neste estudo, a quantidade de grupos é definida com base na quantidade de servidores escravos.

Neste modelo, o processamento paralelo é idêntico ao apresentado na Seção 4.1 para o particionamento aleatório. No entanto, o processamento sequencial pode ser executado de duas formas diferentes: uma utilizando somente a informação espacial (puro) e outra de forma híbrida, utilizando a informação espacial e textual. As duas formas de processamento sequencial e as estruturas auxiliares necessárias são apresentadas nas seções 4.3.1 e 4.3.2.

4.3.1 Processamento sequencial por localização espacial puro

Nesta forma de processamento, somente a informação espacial é utilizada para definir o plano de execução da consulta. O processamento da consulta inicia no escravo que armazena o conjunto de objetos mais próximos ao local de interesse informado e termina com o conjunto de objetos mais distante.

Estruturas auxiliares. No momento da distribuição das subcoleções para os servidores escravos, o servidor mestre cria e armazena um MBR (menor retângulo que envolve todos os objetos da subcoleção) para cada subcoleção, onde esse MBR representa a região espacial de cada escravo do sistema distribuído. Com o auxílio dessa estrutura é possível calcular a distância mínima de cada região espacial para o local de interesse.

A Figura 4.4 apresenta uma distribuição dos objetos através do particionamento por localização, onde os objetos p_1, p_3 e p_7 são indexados no servidor S_1, os objetos p_2 e p_6 no servidor S_2 e os objetos p_4 e p_5 no servidor S_3. Para cada servidor escravo é possível observar o MBR que representa sua região espacial e engloba os objetos de sua subcoleção.

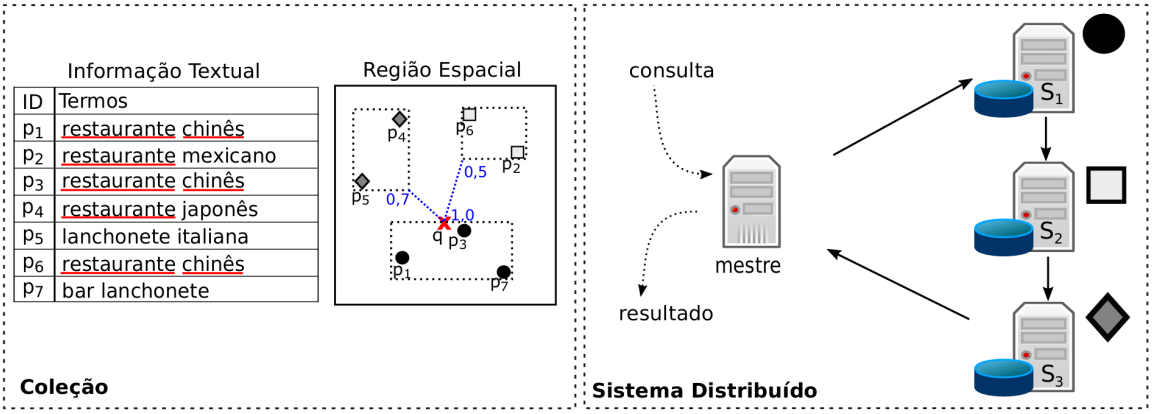


Figura 4.4: Modelo de Particionamento por Localização com MBRs.

Processamento da Consulta. Com a estrutura de MBRs, o servidor mestre cria um plano de execução que leva em consideração a distância entre o local da consulta e os grupos de objetos. O processamento inicia no escravo que armazena o grupo de objetos com menor distância e termina no escravo com maior distância.

Ao realizar uma consulta espaço-textual top-k, na região espacial da Figura 4.4, com as palavras-chave "restaurante chinês", a posição do usuário representado pelo ponto q e $k = 2$. O servidor mestre calcula a distância do ponto q para cada MBR dos servidores escravos e cria um plano de execução com a ordem dos servidores S_1 -> S_2 -> S_3. Neste caso o processamento da consulta segue os seguintes passos: 1) inicia no servidor S_1 que seleciona os objetos p_1 e p_3 e envia para o servidor S_2, 2) o servidor S_2 recebe os objetos p_1 e p_3, seleciona os objetos p_2 e p_6 em sua subcoleção e compara os escores dos 4 objetos, selecionando os 2 melhores (p_3 e p_1) e repassando para o servidor S_3, 3) o S_3 seleciona o objeto p_4, compara com os objetos recebidos e envia os 2 melhores (p_3 e p_1) para o servidor mestre.

4.3.2 Processamento sequencial por localização espacial híbrido

Nesta forma de processamento, a informação textual também é utilizada para definir o plano de execução do processamento da consulta. Além da distância mínima de cada região espacial para o local de interesse, é calculado o melhor escore textual das palavras-chave pesquisadas para os objetos armazenados em cada região espacial. Assim é possível estimar para cada região espacial um escore global que represente o melhor escore que um objeto possa ter na sua região espacial. O processamento da consulta inicia no escravo que possui maior escore global e termina no que possui o menor escore global.

Estruturas auxiliares. Além dos MBRs representando regiões espaciais, o servidor mestre mantém um vocabulário com todos os termos presentes na coleção. Para cada termo é armazenado o endereço do servidor escravo que possui esse termo e o maior valor de impacto λ (Seção 2.4.2) desse termo entre os objetos da sua subcoleção.

A Figura 4.5 apresenta o processamento sequencial por localização espacial híbrido. O vocabulário armazena os termos da coleção ("mexicano", "restaurante", "italiana", "japonês", "chines", "bar"e "lanchonete") com a identificação dos servidores que possuem esse termo e o maior valor de impacto de cada termo.

Processamento da Consulta. No processamento sequencial da consulta utilizando as duas estruturas auxiliares, a equação de ranqueamento (Capítulo 3) é utilizada para calcular um escore global de cada escravo do sistema. A menor distância do local de busca para cada região espacial permite calcular o escore espacial, enquanto os valores dos maiores impactos de cada termo de busca em cada região

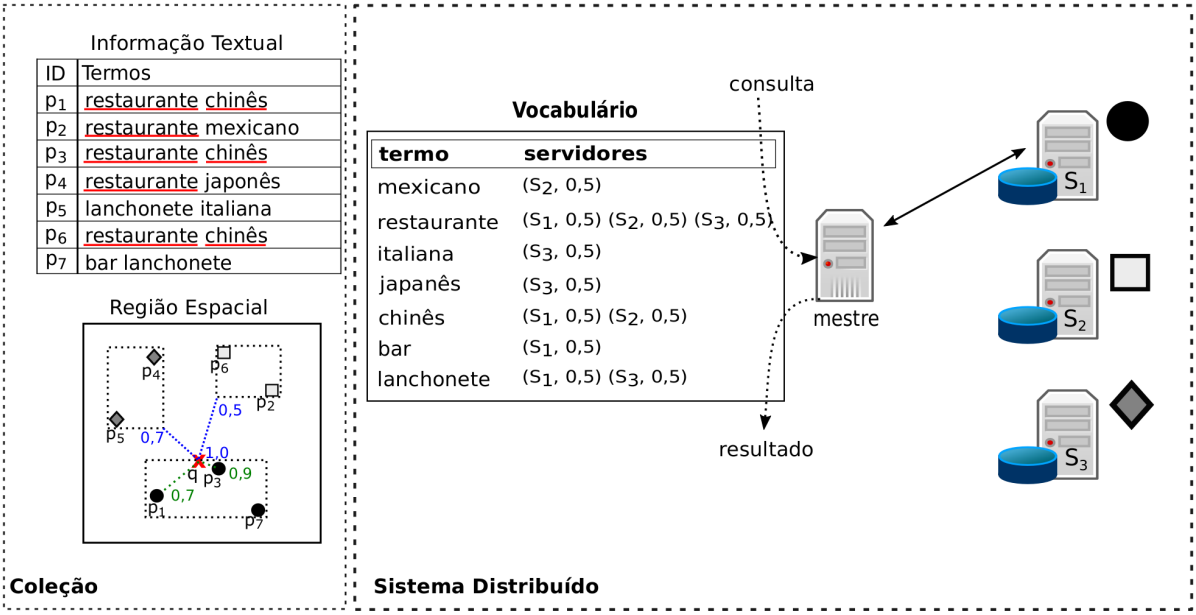


Figura 4.5: Modelo de Particionamento por Localização com duas estruturas auxiliares.

permite calcular o escore textual. Dessa forma, é possível estimar o melhor escore que um objeto possa ter em cada servidor do sistema distribuído e evitar que a consulta seja executada em todas os servidores. Neste processamento, o plano de execução da consulta é criado iniciando no servidor que possui maior escore global e finaliza no que possui menor escore global, podendo eliminar servidores do plano se o escore global for menor que o pior escore dos objetos já encontrados.

Assumindo que o escore textual é o valor do impacto armazenado no vocabulário, ao realizar uma consulta espaço-textual top-k, na região espacial da Figura 4.5, com as palavras-chave "restaurante chinês", a posição do usuário representado pelo ponto q e $k = 2$. O servidor mestre cria um plano de execução (S_1 -> S_2 -> S_3) iniciando em S_1 por ter escore global igual a 2,0 (1,0 de escore espacial e 1,0 de escore textual), depois no servidor S_2 por ter escore global igual a 1,5 (0,5 de escore espacial e 1,0 de escore textual) e finalizando no escravo S_3 por ter escore global igual a 1,2 (0,7 de escore espacial e 0,5 escore textual). No servidor S_1 os objetos p_1 e p_3 que possuem escores 1,9 e 1,6 respectivamente são selecionados e antes de enviar para o servidor S_2 é verificado se o escore global do servidor S_2 é menor que o escore do objeto p_1 que é o pior entre os 2 selecionados, neste exemplo, tanto os servidores S_2 e S_3 são eliminados do processamento por possuirem escores globais menores, sendo o resultado (p_3 e p_1) repassado para o servidor mestre.

4.4 Particionamento por Termos

O modelo de particionamento por termos (MT) é idêntico ao modelo de particionamento horizontal do Arquivo Invertido apresentado na Seção 2.2.1. O particionamento é realizado através da distribuição dos objetos considerando os termos presentes na coleção, onde cada servidor fica responsável por processar as consultas para somente alguns termos da coleção. Este modelo permite que uma consulta seja realizada por somente alguns servidores do sistema distribuído.

Particionamento. Um vocabulário global é criado pelo servidor mestre com todos os termos encontrados na coleção de objetos espaço-textuais e para cada termo do vocabulário é armazenado a quantidade de vezes que esse termo aparece na coleção (frequência). Os termos do vocabulário são ordenados de acordo com sua frequência, permitindo manter um equilíbrio entre termos frequentes e infrequentes no momento do particionamento, onde os termos são acessados um por vez e enviado um para cada subcoleção. A quantidade de termos em cada servidor pode ser calculada através da divisão do número de termos presentes no vocabulário pelo número de servidores do sistema distribuído. Caso a divisão não seja exata, o restante dos termos são distribuídos entre os servidores de forma aleatória.

Cada servidor escravo do sistema distribuído recebe uma subcoleção de termos e cria um vocabulário local com os termos que ficou responsável. Toda a coleção de objetos espaço-textuais é repassada para os servidores escravos e em cada servidor é criado um índice contendo somente os objetos que possuem os termos do seu vocabulário.

Estruturas Auxiliares. No momento em que o servidor mestre realiza a distribuição dos termos, é registrado no vocabulário global para cada termo, o servidor que ficou responsável por processar consultas para esse termo. Com o auxílio do vocabulário é possível eliminar servidores que não possuam as palavras-chave pesquisadas, diminuindo a quantidade de servidores acessados no processamento das consultas espaço-textual top-k.

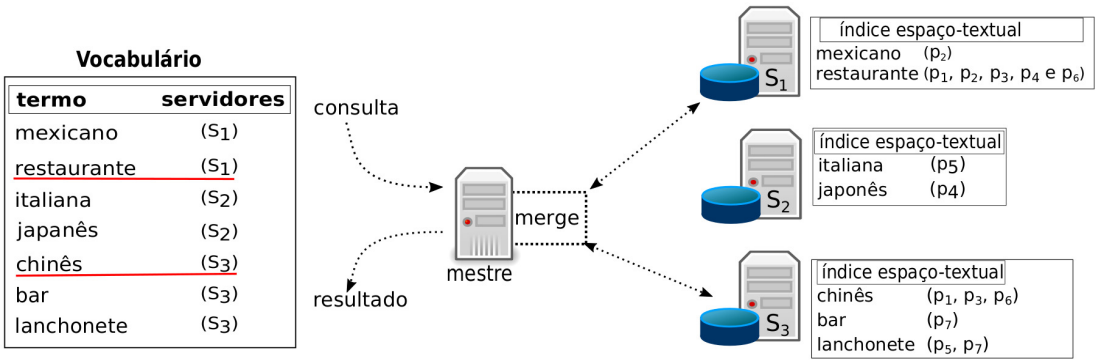


Figura 4.6: Particionamento por termos.

A Figura 4.6 apresenta um particionamento através dos termos, onde o servidor mestre cria um vocabulário global com os termos "mexicano", "restaurante", "japonês", "italiana", "chinês", "bar"e "lanchonete"e para cada termo a referência do servidor que possui os objetos com esse termo. O servidor S_1 é responsável por processar as consultas pelos termos "mexicano" e "restaurante", o S_2 pelos termos "japonês" e "italiana" e o S_3 pelos termos "chinês", "bar"e "lanchonete".

Processamento da Consulta. Neste modelo, quando uma consulta for por termos que estão armazenados em um mesmo servidor, o processamento da consulta espaço-textual top-k é idêntico ao modelo aleatório citado na Seção 4.1, seja ele paralelo ou sequencial. Isso acontece porque todas as informações necessárias estão no mesmo servidor, sendo a consulta processada e os k melhores objetos retornados para o servidor mestre.

Para o processamento das consultas com termos que envolvam mais de um servidor é necessário enviar a lista com todos os objetos indexados para que um servidor realize o *merge* das listas e o ranqueamento dos objetos. Em testes preliminares foi possível observar que a migração de toda a lista de objetos dos termos para um servidor realizar o *merge* era muito lento se comparado com os modelos por objetos que só migram a quantidade k de objetos solicitados. Por isso, foi adotado uma recuperação incremental dos objetos, onde cada servidor escravo recupera uma quantidade fixa de objetos (definida como parâmetro no momento da criação do modelo) e envia para o servidor mestre realizar o processamento. Os objetos são recuperados considerando a proximidade espacial para o local de busca informado, onde os mais próximos são recuperados primeiro. À medida que o *merge* é realizado, o servidor mestre compara o escore do objeto o_k com o último objeto processado para decidir se é necessário solicitar mais objetos dos servidores escravos ou se já possui os k melhores.

O processamento paralelo passa pelos seguintes passos: 1) o servidor mestre recebe uma consulta espaço-textual top-k, 2) separa a informação textual em termos distintos e localiza no vocabulário os servidores que são responsáveis pelo processamento de cada termo, 3) distribui a consulta entre os servidores responsáveis informando os termos e localização espacial da consulta, 4) cada servidor processa a consulta em sua coleção, 5) retorna uma quantidade fixa de objetos ordenados pelo escore espacial, 6) o servidor mestre realiza o ranqueamento dos objetos recebidos e solicita mais até que o escore dos objetos selecionados não possa ser superados pelos próximos objetos, devido a distância espacial ser muito grande, e por fim 7) retorna os k melhores.

Ao realizar uma consulta espaço-textual top-k, na região espacial da Figura 4.6, com as palavras-chave "restaurante chinês", a posição do usuário representado pelo ponto q e $k = 2$, o servidor mestre envia a consulta para os servidores S_1 e S_3, onde S_1 devolve os objetos p_1, p_2, p_3, p_4 e p_6 e S_2 os objetos p_1, p_3 e p_6 para o servidor mestre realizar o processamento da consulta e selecionar os 2 melhores objetos (p_3 e p_1).

O processamento sequencial não foi adotado neste modelo devido à necessidade de transferir grandes quantidades de dados entre os servidores e a complexidade de uma abordagem de recuperação incremental dos objetos entre os servidores.

Capítulo 5

Avaliação Experimental

"Algo só é impossível até que alguém duvide e resolva provar ao contrário."
– Albert Einstein

Neste capítulo, uma avaliação experimental é realizada entre as abordagens que utilizam os modelos propostos. Este capítulo esta organizado da seguinte forma: a Seção 5.1 apresenta as bases de dados utilizadas, a Seção 5.2 a configuração utilizada para o desenvolvimento dos sistemas que utilizam os modelos propostos, bem como, os valores dos parâmetros utilizados em cada experimento. A Seção 5.3 apresenta os experimentos para construção de cada sistema e finalmente as seções 5.4 e 5.5 os experimentos com processamento paralelo e sequencial da consulta espaço-textual top-k.

5.1 Base de Dados

Para execução dos experimentos foram utilizadas três bases de dados, sendo duas criadas com objetos espaço-textual reais, coletados nos sites OpenStreetMap e Twitter. A terceira base de dados foi criada através da junção de um conjunto de dados textuais com uma base de dados espacial disponível na internet[1,2]. Todas as bases de dados são armazenadas em arquivos no formato .txt contendo um objeto espaço-textual por linha com as seguintes informações separadas por espaço em branco: identificação (ID), latitude, longitude e texto.

[1] snap.stanford.edu/data/twitter7.html
[2] www.dis.uniroma1.it/challenge9/download.shtml

Base	**#objetos**	**avg_termos**	**#termos_unicos**	**#termos**
Tweet	1.570.850	7	583.654	12.722.677
OSM	7.783.286	2	1.939.904	16.465.511
Rotas	8.000.000	12	2.173.256	85.320.790

Tabela 5.1: Informações das Bases de Dados

O primeiro conjunto de dados (*Tweet*) é formado por 1.570.850 mensagens com localização espacial de usuários do Twitter. O Twitter é uma rede social que disponibiliza parte dos seus dados de forma gratuita, no entanto para realizar a coleta é necessário implementar uma aplicação utilizando a API *Twitter Search API* disponibilizada pelo próprio Twitter. Com essa API foi implementada uma aplicação para coletar mensagens (*tweets*) de diversos usuários. Para as mensagens coletadas, foi realizado um preprocessamento eliminando instâncias repetidas e que não possuíam informações de localização espacial (latitude e longitude).

A segunda base de dados OSM é formada por 7.783.286 objetos espaço-textuais extraídos do site OpenStreetMap. O OpenStreetMap é um projeto de mapeamento colaborativo que possui uma base de dados com objetos que cobrem todo o mundo e é disponibilizada para download de forma gratuita. No site *Planet OSM*[3] foi adquirido a base de dados completa do OpenStreetMap em formato XML e posteriormente transformado em formato .txt utilizando o programa *Travel Assistant* desenvolvido pelo projeto PerTA[4] (*Personal Travel Assistant*). Com a base de dados no formato .txt foi realizado um preprocessamento para eliminar as instâncias que não possuíam informação textual associada, gerando finalmente a base de dados OSM com 7.783.286 objetos.

A terceira base de dados (Rotas) é formada pela junção de 8 milhões de mensagens de texto com 8 milhões de objetos espaciais. As mensagens de texto utilizadas foram extraídas de uma base de dados do SNAP[5] que é formada por 467 milhões de mensagens de texto coletadas na rede social Twitter. Os objetos espaciais foram extraídos da base de dados do DIMACS[6] formada por quase 24 milhões de objetos que representam locais (latitude e longitude) nos Estados Unidos. Os primeiros 8 milhões de objetos de cada base foram coletados e associados para formarem a base de dados Rotas com 8 milhões de objetos.

A Tabela 5.1 apresenta características de cada base de dados utilizada nos experimentos. Na primeira coluna são listados os nomes das bases de dados e nas demais colunas a quantidade de objetos (#objetos), a média de termos únicos por objeto (avg_termos), a quantidade total de termos únicos em toda a coleção (#termos_unicos) e o número total de termos existentes na coleção (#termos).

[3] planet.osm.org/

[4] sites.ecomp.uefs.br/perta/

[5] snap.stanford.edu/data/twitter7.html

[6] www.dis.uniroma1.it/challenge9/download.shtml

5.2 Configuração

Para avaliar a influência do particionamento de dados no processamento das consultas espaço-textual top-k, foi desenvolvido um sistema para cada modelo proposto. Esses sistemas foram batizados com as siglas SMAleatório, SMTextual, SMEspacial e SMTermo, onde SMAleatório utiliza o modelo de particionamento aleatório, SMTextual o modelo de similaridade textual, SMEspacial o modelo de particionamento por localização espacial e por fim, SMTermo o modelo de particionamento por termos.

Todos os sistemas foram implementados em linguagem de programação Java, utilizando a API de *Sockets* para realizar a comunicação entre os servidores do sistema distribuído. Em cada sistema foi utilizado o índice híbrido S2I+ para indexar os objetos e processar as consultas espaço-textual top-k nos servidores escravos. No armazenamento do S2I+, assumiu-se blocos com 4KB de tamanho, tanto para a estrutura de blocos como para os nós das árvores aR-tree.

Para os sistemas que utilizam os modelos de proximidade espacial e similaridade textual, o algoritmo *k-means* foi utilizado para agrupar objetos similares em uma mesma partição. O algoritmo *k-means* apresenta um custo de processamento muito elevado para dados textuais quando comparado com o processamento para dados espaciais. Por isso a sua execução foi limitada a 10 iterações no sistema SMTextual.

No sistema SMTermo foi definido o valor 876 como a quantidade de objetos recuperadas e transferidos pela rede em cada requisição do servidor mestre a um escravo. Esse valor é a quantidade de objetos suportados por 6 blocos de 4KB e foi utilizado porque o S2I+ recupera e ordena todos os objetos de um termo infrequente antes de iniciar o processamento da consulta. Os termos infrequentes são recuperados de uma vez só e migrados para o servidor mestre enquanto que os termos frequentes são recuperados de forma incremental.

Os experimentos foram executados em um sistema distribuído composto por 7 computadores com hardware homogêneo formado por processador Intel 3.4 GHz, 4 GB de memória e interligados por uma rede 100 Mbps. Para todos os modelos um servidor (mestre) ficou responsável por receber as consultas, enviar para os outras 6 servidores escravos do sistema distribuído e retornar o resultado final.

Em cada experimento são executadas 50 consultas de aquecimento, para que o sistema operacional carregue os dados necessários no seu *buffer*, e coletada a média dos resultados das próximas 900 consultas espaço-textual top-k. A localização espacial utilizada em cada consulta é gerada de forma aleatória e as palavras-chave de busca são retiradas de uma base de dados formada por consultas textuais reais realizadas no engenho de busca AOL[7]. Para cada experimento é considerado um valor diferente para uma variável enquanto as outras continuam com valores fixos, sendo coletado a

[7] search.aol.com

Paramêtros	**Valores**
Número de palavras-chave	1, 2, 3, **4**, 5
Número de resultados	1, 5, 10, **15**, 20
Número de alfa	0,1, 0,3, 0,5, 0,7, **0,9**
Base de Dados	Tweet, OSM, **Rotas**

Tabela 5.2: Variáveis estudadas, onde os valores padrão estão em negrito

média do tempo de resposta, do número de páginas lidas no disco, da quantidade de dados trafegados na rede e da quantidade de servidores utilizados no processamento.

A Tabela 5.2 apresenta as variáveis estudadas, sendo que os valores em negrito são os valores utilizados como padrão nas consultas espaço-textual top-k. Algumas figuras desta seção são apresentadas em escala logarítmica devido a grande diferença entre os resultados apresentados pelos sistemas.

Para as execuções das consultas, foi adotado a seguinte metodologia:

1. **Variação no valor de α:** a equação utilizada pela consulta espaço-textual top-k para ranquear os objetos utiliza uma variável α que possui valor entre 0 e 1 para permitir que o usuário defina uma relevância maior entre à medida textual e a espacial. Neste experimento, o valor da variável α é modificado progressivamente entre um conjunto de valores formado por 0,1, 0,3, 0,5, 0,7, 0,9. Os valores menores que 0,5 definem um peso maior para a similaridade textual e acima de 0,5 um peso maior para a proximidade espacial.
2. **Variação da quantidade de palavras-chave da consulta:** o conjunto de palavras-chave para o processamento da consulta espaço-textual top-k é definido pelo usuário no momento em que a consulta é representada, não havendo limite para a quantidade de palavras-chave. Neste experimento, a quantidade de palavras-chave utilizadas na consulta é incrementado progressivamente, onde inicia com 1 (uma) palavra-chave e é incrementado a cada novo experimento até chegar em 5, sendo realizados experimentos com um conjunto de palavras-chave formado por 1, 2, 3, 4 e 5.
3. **Variação da quantidade de objetos k na resposta da consulta:** a consulta espaço-textual top-k retorna a quantidade k de objetos desejados pelo usuário. Neste experimento, a quantidade de objetos desejados k é modificada entre os valores 1, 5, 10, 15, 20.
4. **Variação da base de dados:** neste experimento são realizadas consultas espaço-textuais top-k em diferentes bases de dados. Os experimentos são realizados na base Tweets, depois na base do OSM e por fim, na base Rotas.

5.3 Construção

Esta seção apresenta experimentos para construção das estruturas de dados utilizadas nos sistemas (SMAleatório, SMTermo, SMEspacial e SMTextual). Em cada experimento foi utilizado uma base de dados diferente (Tweet, OSM, Rotas) e coletado o tempo gasto para construção.

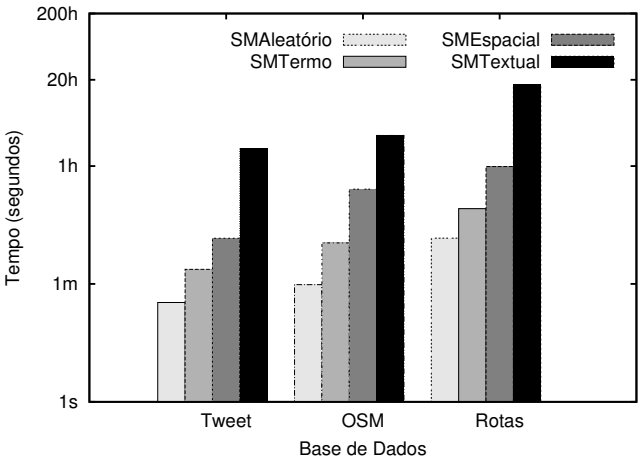


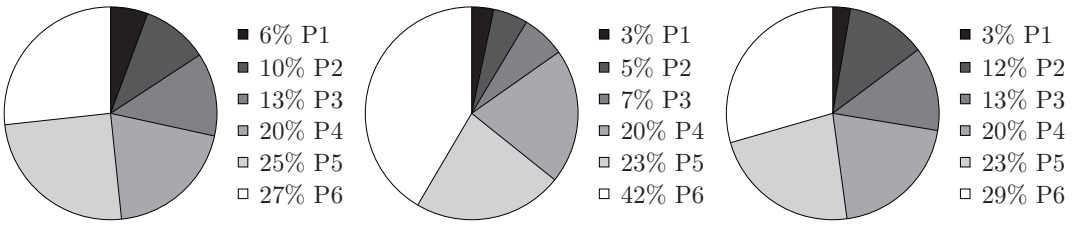
Figura 5.1: Gráfico com o tempo para construção dos modelos.

A Figura 5.1 apresenta o tempo utilizado para construção das estruturas de dados de cada sistema utilizando diferentes bases de dados. Dentre os quatro sistemas, SMAleatório utiliza menor tempo para construir sua estrutura. Isso pode ser explicado porque este sistema não realiza nenhum preprocessamento nos dados para criar suas partições, os objetos são simplesmente distribuídos de forma aleatória entre o servidores escravos do sistema distribuído.

Os sistemas SMEspacial e SMTextual que utilizam o algoritmo *k-means* para realizar o particionamento dos dados, gastaram um tempo muito superior aos demais (SMTermo e SMAleatório). Isso ocorre porque o *k-means* acessa e compara todos os objetos da coleção mais de uma vez (iteração) antes de definir os particionamentos, sendo que a comparação dos objetos através das informações textuais tem um custo ainda maior de processamento, o que justifica o sistema SMTextual ser o pior em tempo gasto para construção.

Neste experimento, SMAleatório criou 6 partições contendo cerca de 17% dos objetos da coleção em cada, mantendo um bom balanceamento na quantidade de objetos indexados por cada escravo do sistema distribuído. No entanto, os sistema SMEspacial e SMTextual criaram partições muito diferentes para todos as bases de dados.

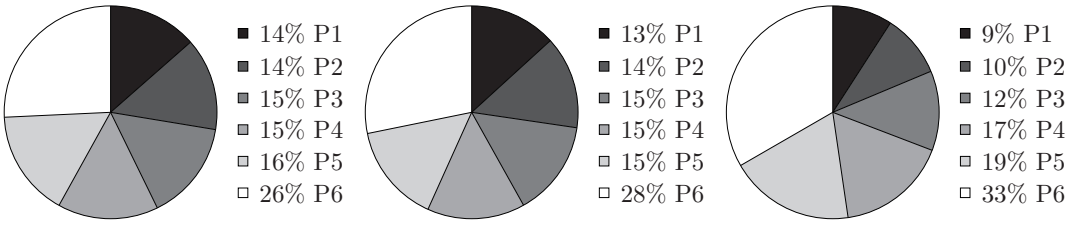
A Figura 5.2 apresenta os particionamentos criados pelo sistema SMEspacial através da proximidade espacial dos objetos para as três bases de dados. Na base OSM



(a) Tweet (b) OSM (c) Rotas

Figura 5.2: Particionamentos criado pelo sistema SMEspacial nas bases de dados.

(Figura 5.2(a)), as partições P4, P5 e P6 agrupam 72% dos objetos, sendo os restantes distribuídos entre P1, P2, P3. Para a base Tweet (Figura 5.2(b)), 42% dos objetos são agrupados na partição P6 e 43% nas partições P4 e P5, os 15% restante são distribuídos entre P1, P2 e P3. Na base Rotas (Figura 5.2(c)), três partições (P4, P5 e P6) agrupam 72% dos objetos, enquanto os 28% restante são agrupados em outras 3 partições (P1, P2 e P3). Isso acontece porque o algoritmo utilizado (*k-means*) para criar as partições pode gerar grupos com quantidade diferentes de objetos, não mantendo um balanceamento nas quantidades.

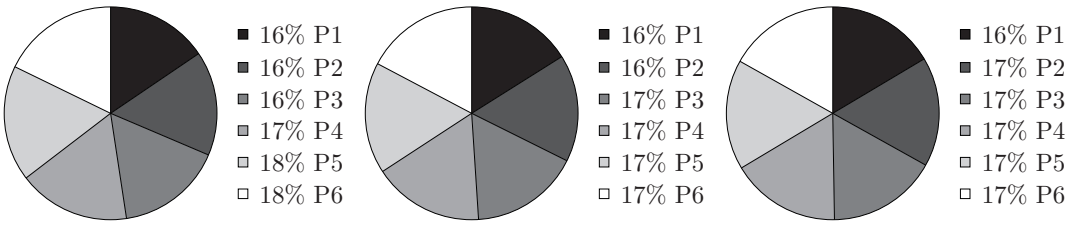


(a) Tweet (b) OSM (c) Rotas

Figura 5.3: Distribuição dos objetos através do *K-means* por similaridade textual

A Figura 5.3 apresenta os particionamentos criados pelo sistema SMTextual através da similaridade textual dos objetos. Nas bases OSM e Tweet, SMTextual cria uma partição com mais de 25% dos objetos e distribui o restante de forma similar entre as outras partições (Figuras 5.3(a) e 5.3(b)). Para a base Rotas 5.3(c)), 33% dos objetos são agrupados na partição P6 e o restante distribuído entre as outras 5 partições. Assim como SMEspacial, o sistema SMTextual utiliza o algoritmo *k-means* e cria partições com quantidade diferentes de objetos. No SMTextual, as partições criadas para as bases de dados OSM e Tweet ficaram melhor balanceadas se comparadas com as partições criadas por SMEspacial.

A Figura 5.4 apresenta os particionamentos criados pelo sistema SMTermo para as três bases de dados. SMTermo cria partições com quantidades de objetos bastante similares para todas as bases de dados. Isso acontece porque as partições são criadas agrupando termos frequentes e infrequentes para manter um balanceamento na quantidade de objetos de cada partição. Apesar de SMTermo criar partições balanceadas em quantidade de objetos, o número de objetos indexados por cada servidor



(a) Tweet (b) OSM (c) Rotas

Figura 5.4: Particionamentos criado pelo SMTermo para as três bases de dados.

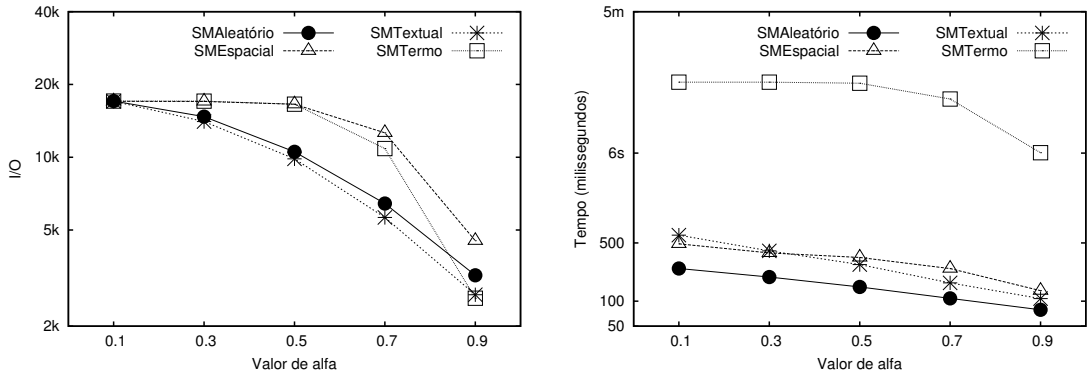
escravo é superior aos demais sistemas por permitir que um objeto seja indexado por servidores diferentes.

5.4 Processamento Paralelo

Esta seção contém experimentos com os quatro modelos propostos (MA, MT, MLE, MST). Em todos os experimentos são coletados a quantidade de páginas lidas em disco, o tempo de resposta, a quantidade de servidores utilizados e a quantidade de dados que trafegou na rede durante o processamento da consulta.

5.4.1 Variando o valor de α

Esta seção estuda o impacto nos modelos propostos ao variar o valor de α no processamento da consulta espaço-textual top-k. A Figura 5.5 apresenta o número de páginas lidas (I/O) e o tempo de resposta obtido por cada sistema ao variar o valor de α.



(a) I/O (b) Tempo de resposta

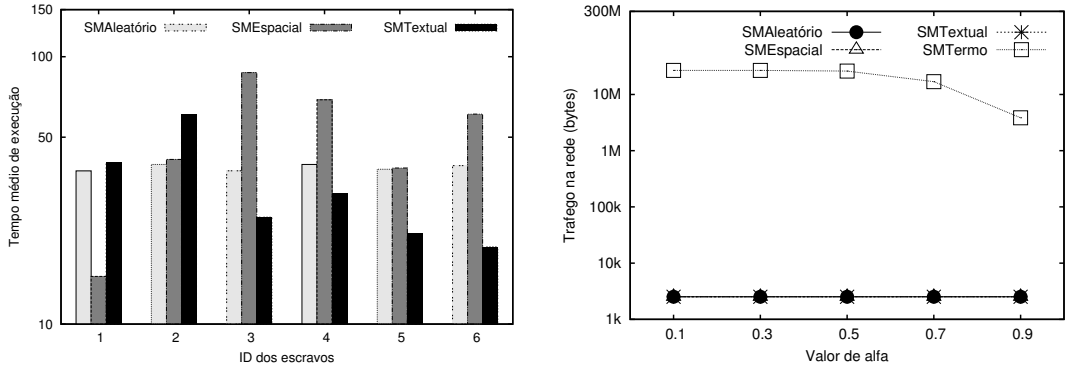
Figura 5.5: I/O e Tempo de resposta ao variar o valor de α no processamento paralelo.

A Figura 5.5(a) apresenta o número de páginas lidas (I/O) de cada sistema ao variar o valor de α. Todos os sistemas apresentam valores menores de I/O com o aumento do valor de α. Para SMAleatório, SMTextual e SMEspacial que processam as consultas nos escravos, isso pode ser explicado porque todos utilizam uma variação do índice S2I nos servidores escravos, onde o S2I acessa uma quantidade menor de objetos ao processar uma consulta com valores mais altos de α [Rocha-Junior et al. 2011].

No sistema SMTermo, os escravos transferem para o mestre os objetos ordenados pela distância espacial, permitindo que o mestre processe uma quantidade menor de objetos com valores maiores de α porque aumenta o peso do escore espacial.

A Figura 5.5(b) apresenta o tempo de resposta de cada sistema ao variar o valor de α. SMAleatório, SMTextual e SMEspacial apresentam tempos de resposta bastante próximos, enquanto SMTermo apresenta um tempo de resposta muito superior, mesmo acessando um número de páginas muito similar ao SMTextual quando o valor de α é 0.9. A similaridade nos resultados entre os sistemas SMAleatório, SMEspacial e SMTextual pode ser explicada porque a consulta em paralelo é realizada de forma idêntica para os três sistemas.

O número de acessos a páginas de SMEspacial refletiu no tempo de resposta, onde SMEspacial obteve tempo de resposta superior aos modelos SMAleatório e SMTextual. Já o sistema SMAleatório obteve um tempo de resposta um pouco melhor que SMTextual, mesmo acessando uma quantidade de páginas superior (Figuras 5.5(a) e 5.5(b)).



(a) Tempo de execução local

(b) Tráfego na Rede

Figura 5.6: Tempo de execução local e tráfego na rede ao variar o valor de α no processamento paralelo

Para avaliar o melhor desempenho de SMAleatório, foi coletado o tempo local de processamento das consultas, com valor padrão das variáveis, em cada servidor escravo do sistema distribuído. A Figura 5.6(a) apresenta a média do tempo de resposta local em cada escravo. SMAleatório consegue manter uma carga de trabalho linear entre os servidores do sistema distribuído, enquanto os demais sistemas não conseguem. O tempo de resposta é calculado entre o momento da submissão da consulta

e o seu retorno. Assim, um único servidor sobrecarregado pode afetar o tempo de resposta como um todo.

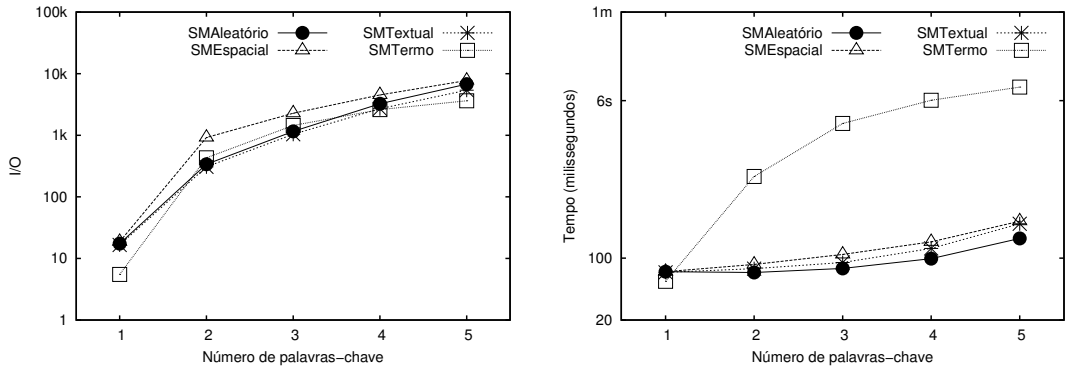
A Figura 5.6(b) apresenta a quantidade de dados trafegados na rede ao variar o valor de α. SMTermo trafega uma quantidade de dados na rede muito superior aos outros sistemas. Isso acontece porque SMTermo recupera e transfere muitos objetos para o servidor mestre realizar *merge* e ranqueamento dos objetos, enquanto os demais sistemas enviam somente os k melhores de sua subcoleção.

A quantidade de servidores acessados no processamento da consulta foi constante durante a variação do valor de α. Os sistemas SMAleatório, SMTextual e SMEspacial utilizaram todos os servidores para realizar o processamento das consultas, enquanto SMTermo utilizou cerca de 3 servidores por consulta.

Neste experimento, todos os sistemas são impactados pela variação do valor de alfa (Figura 5.5). Os sistemas apresentam um desempenho melhor com valor de α definindo um peso maior para a informação espacial em detrimento da informação textual.

5.4.2 Variando a quantidade de palavras-chave

Esta seção estuda o impacto nos modelos propostos ao variar a quantidade de palavras-chave no processamento da consulta espaço-textual top-k. A Figura 5.7 apresenta o número de páginas lidas (I/O) e o tempo de resposta obtido por cada sistema ao variar o número de palavras-chave.



(a) I/O

(b) Tempo de resposta

Figura 5.7: I/O e Tempo de resposta ao variar o número de palavras-chave no processamento paralelo

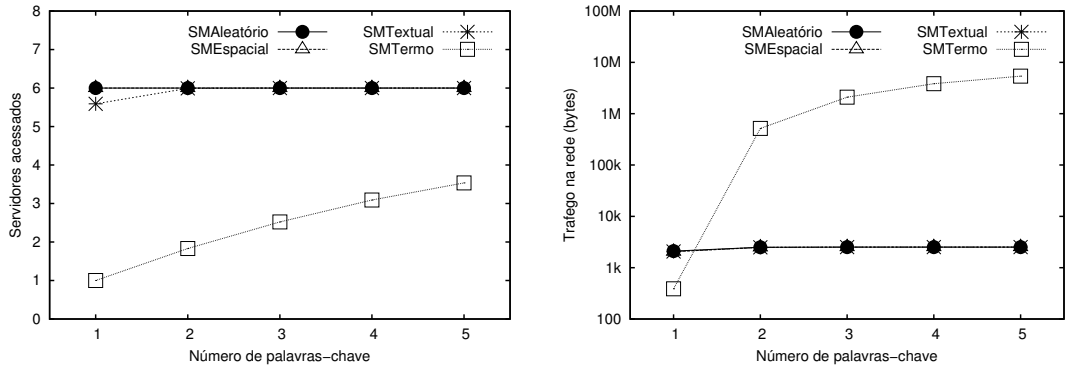
A Figura 5.7(a) apresenta o número de páginas acessadas (I/O) por cada sistema ao variar o número de palavras-chave. Todos os sistemas apresentam valores maiores de I/O com o aumento do número de palavras-chave. Isso acontece porque ao aumentar o número de palavras-chave consequentemente aumenta o número de objetos acessados para o processamento da consulta.

SMTermo acessa o menor número de páginas para uma e cinco palavras-chave, enquanto os demais sistemas acessam um número de páginas bastante similar. sendo o SMEspacial o sistema que acessa maior número de páginas em todos os experimentos.

A Figura 5.7(b) apresenta o tempo de resposta de cada sistema ao variar o número de palavras-chave. A maioria dos sistemas obtêm tempo de resposta muito similares ao número de páginas acessadas do disco (I/O), exceto para SMTermo que obtêm o menor tempo de resposta para uma palavra-chave e o maior para os demais experimentos.

O melhor desempenho de SMTermo para uma palavra-chave ocorre porque todas as informações necessárias para processar a consulta estão em somente um servidor, permitindo que a consulta seja processada sem a necessidade de transportar dados pela rede, sem esperar por processamento em outros servidores e sem realizar o ranqueamento no servidor mestre. No entanto, com números maiores de palavras-chave, o sistema SMTermo apresenta um aumento considerável no tempo de resposta.

Com mais de duas palavras-chave, SMAleatório apresenta o menor tempo de resposta para processar as consultas, mesmo acessando mais páginas do disco que SMTextual. SMAleatório consegue um tempo de resposta menor por possuir melhor balanceamento de carga no processamento da consulta, conforme apresentado na Seção 5.4.1.



(a) Servidores utilizados (b) Tráfego na Rede

Figura 5.8: Número de servidores utilizados e tráfego na rede ao variar o número de palavras-chave no processamento paralelo

A Figura 5.8(a) contém o número de servidores acessados ao variar o número de palavras-chave. O SMTermo acessa um número menor de servidores que os demais sistemas, ao aumentar o número de palavras-chave a quantidade de servidores utilizados também aumenta. O restante dos sistemas acessam uma quantidade constante de servidores, onde todos os servidores são acessados, exceto para SMTextual que acessa um número de servidores menor para uma palavra-chave. Isso acontece por-

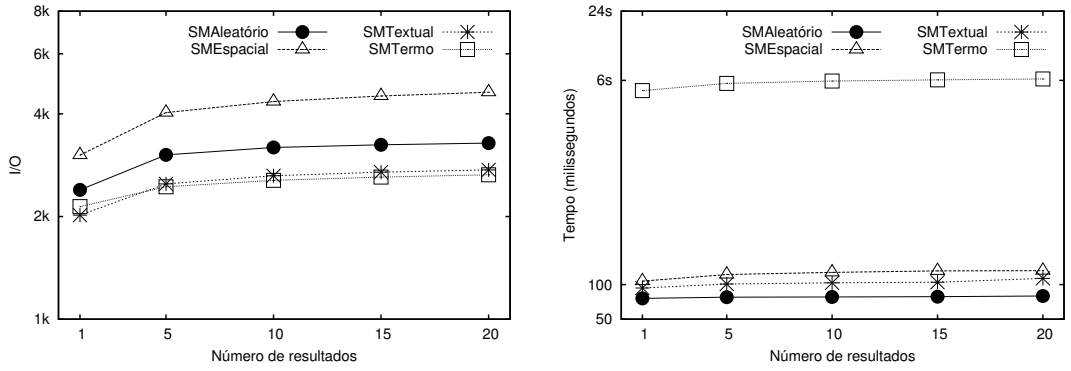
que SMTextual elimina servidores do processamento quando não possuí pelo menos uma das palavras-chave pesquisadas.

A Figura 5.8(b) apresenta a quantidade de dados trafegados na rede ao variar o número de palavras-chave. SMTermo trafegou o menor número de dados para uma palavra-chave porque a consulta é processada por somente um servidor, onde somente os k melhores objetos são migrados para o servidor mestre, no entanto à medida que o número de palavras-chave aumenta, mais servidores são utilizadas e o tráfego na rede aumenta consideravelmente.

Neste experimento, todos os sistemas são impactados pela variação do número de palavras-chave (Figura 5.7). O maior número de palavras-chave aumenta a quantidade de objetos processados e consequentemente o tempo de resposta no processamento da consulta espaço-textual top-k para todos os sistemas.

5.4.3 Variando o valor de k

Esta seção estuda o impacto nos modelos ao variar a quantidade de objetos desejados k no processamento da consulta espaço-textual top-k. A Figura 5.9 apresenta o número de páginas lidas (I/O) e o tempo de resposta obtido por cada sistema ao variar a quantidade de k.



(a) I/O (b) Tempo de resposta

Figura 5.9: I/O e Tempo de resposta ao variar o valor de k no processamento paralelo

A Figura 5.9(a) apresenta a quantidade de páginas acessadas (I/O) por cada sistema ao variar a quantidade de objetos desejados. Todos os sistemas apresentam valores maiores de I/O com o aumento do valor de k. O aumento no número de objetos desejados, aumenta o número de objetos processados e consequentemente aumenta o número de páginas acessadas. SMTermo e SMTextual acessam uma quantidade menor de páginas do disco, enquanto SMAleatório e SMEspacial acessam uma quantidade maior, sendo SMEspacial o sistema que acessa a maior quantidade de páginas do disco.

A Figura 5.9(b) apresenta o tempo de resposta de cada sistema ao variar a quantidade de objetos desejados. Assim como na quantidade de páginas lidas, o tempo de resposta também aumenta à medida que o número de objetos desejados aumenta. Dentre os 4 sistemas, SMTermo obteve o maior tempo de resposta, enquanto o sistema SMAleatório obteve o melhor em todas as varições. Esse desempenho ruim de SMTermo se deve a necessidade de trafegar muitos objetos na rede para encontrar o resultado final.

A quantidade de páginas lidas não refletiu no tempo de resposta de SMAleatório. Este sistema apresenta valores de I/O maiores que o sistema SMTextual e mesmo assim, consegue processar as consultas em menor tempo de resposta. Isso se deve ao bom balanceamento de carga do sistema SMAleatório já discutido na Seção 5.4.1.

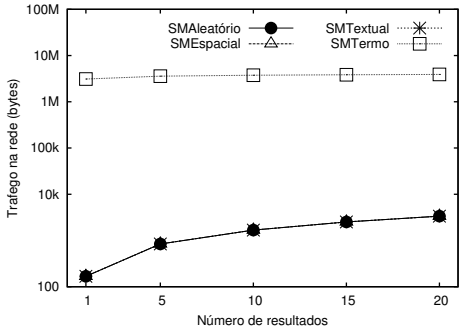


Figura 5.10: Tráfego na rede ao variar o valor de k no processamento paralelo

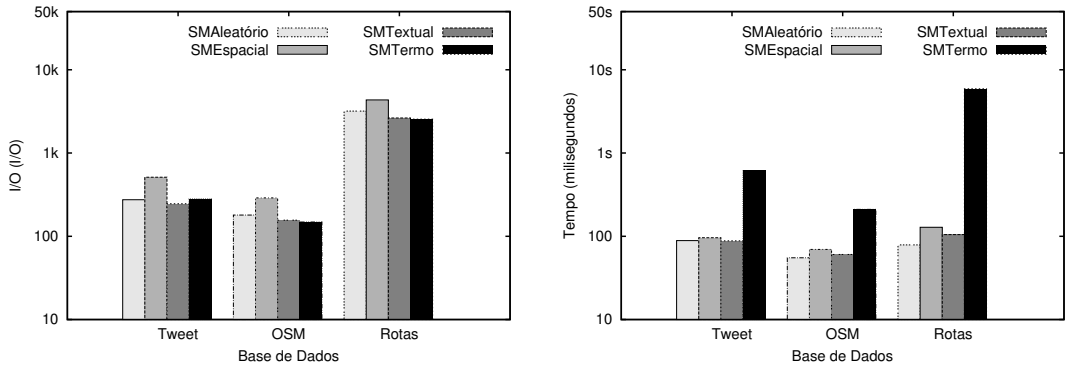
A Figura 5.10 apresenta a quantidade de dados trafegados na rede ao variar o número de objetos desejados. SMTermo trafega uma quantidade de dados na rede muito superior aos outros sistemas. Para maiores valores de k, a quantidade de dados trafegados na rede aumenta para todos os sistemas.

A quantidade de servidores acessados no processamento da consulta foi constante durante a variação do valor de k. Os sistemas SMAleatório, SMTextual e SMEspacial utilizaram todos os servidores para realizar o processamento das consultas, enquanto SMTermo utilizou cerca de 3 servidores por consulta.

Neste experimento, todos os sistemas são impactados pela variação do número de objetos desejados k (Figura 5.9). Um maior número de k aumenta a quantidade de objetos processados e consequentemente diminui o desempenho de todos os sistemas.

5.4.4 Variando a base de dados

Esta seção estuda o impacto nos modelos ao variar a base de dados no processamento da consulta espaço-textual top-k. A Figura 5.11 apresenta o número de páginas lidas (I/O) e o tempo de resposta obtido por cada sistema em diferentes bases de dados.



(a) I/O (b) Tempo de Resposta

Figura 5.11: I/O e Tempo de resposta ao variar a base de dados no processamento paralelo

A Figura 5.11(a) apresenta a quantidade de páginas acessadas (I/O) por cada sistema ao variar a base de dados. Os sistemas SMEspacial e SMAleatório acessam o maior número de páginas do disco em todas as bases de dados, sendo que SMEspacial é o sistema com maior número de acessos a disco. SMTermo acessa a menor quantidade de páginas do disco para as bases de dados OSM e Rotas, entretanto para a base Tweets o mesmo é superado pelo sistema SMTextual.

A Figura 5.11(b) apresenta o tempo de resposta obtido por cada sistema ao variar a base de dados. Dentre os 4 sistemas, SMTermo apresenta o maior tempo de resposta enquanto SMAleatório apresenta o menor tempo de resposta em todas as bases de dados.

Os resultados apresentados com diferentes bases de dados são muito parecidos, confirmando que os resultados obtidos para avaliação dos sistemas são consistentes. O sistema SMAleatório realiza as consultas em menor tempo de resposta, para todas as bases de dados, mesmo acessando uma quantidade de páginas superior a outros sistemas. Isso acontece porque o SMAleatório divide muito bem a carga de trabalho entre os servidores escravos. O sistema SMTermo utiliza o maior tempo de resposta, para todas as bases de dados, por recuperar e transferir muitos objetos pela rede para que o servidor mestre realize o processamento da consulta.

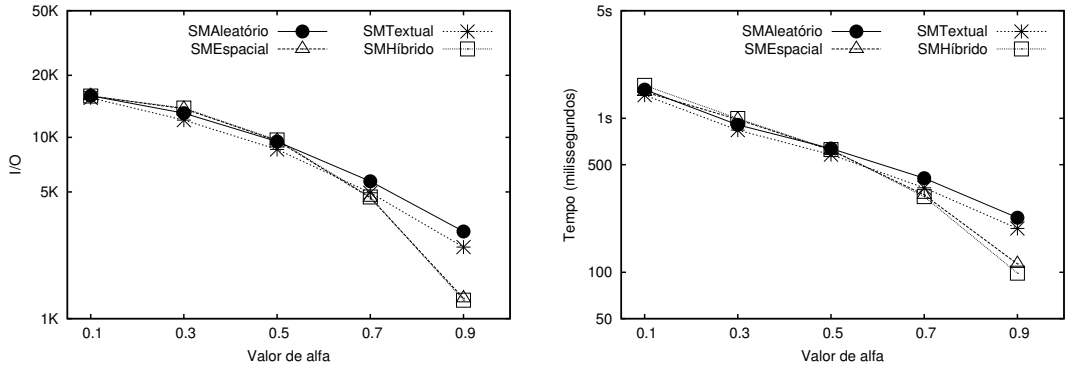
5.5 Processamento Sequencial

Esta seção contém experimentos com os sistemas SMAleatório, SMEspacial e SMTextual. O sistema SMTermo não é utilizado nesse experimento porque não realiza o processamento sequencial. No entanto, SMEspacial é utilizado duas vezes, a primeira utilizando o processamento por localização espacial puro (SMEspacial) e a segunda utilizando o processamento por localização espacial híbrido (SMHíbrido).

Em todos os experimentos são coletados o número de páginas lidas (I/O), o tempo de resposta, a quantidade de dados trafegados na rede e a quantidade de servidores utilizados no processamento da consulta.

5.5.1 Variando o valor de α

Esta seção estuda o impacto nos modelos propostos ao variar o valor de α no processamento da consulta espaço-textual top-k. A Figura 5.12 apresenta o número de páginas lidas (I/O) e o tempo de resposta obtido por cada sistema ao variar o valor de α.



(a) I/O (b) Tempo de resposta

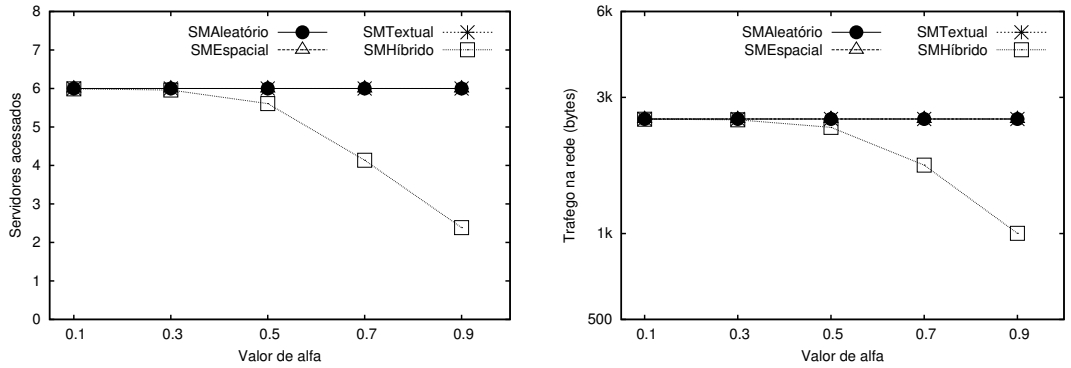
Figura 5.12: I/O e Tempo de resposta ao variar o valor de α no processamento sequencial

A Figura 5.12(a) apresenta o número de páginas lidas (I/O) de cada sistema ao variar o valor de α. Todos os sistemas apresentam valores menores de I/O com valores maior de α. Isso acontece porque todos utilizam uma variação do índice S2I nos servidores escravos. O S2I acessa uma quantidade menor de objetos ao processar uma consulta com valores mais altos de α [Rocha-Junior et al. 2011].

O sistema SMTextual acessa uma quantidade de páginas menor que os outros sistemas com valor α até 0.5, quando o valor é maior que 0.5, os sistemas baseados em localização espacial SMEspacial e SMHíbrido apresentam valores menores de I/O. Essa melhora no número de páginas lidas de SMEspacial e SMHíbrido acontece porque o experimento inicia atribuindo uma relevância maior para a informação textual e termina com uma relevância menor.

A Figura 5.12(b) apresenta o tempo de resposta de cada sistema ao variar o valor de α. Assim como no número de páginas acessadas, o tempo de resposta também diminui para valores mais altos de α. Em todos os experimentos, os sistemas apresentam tempos de resposta bastante similares, onde somente no valor de α igual a 0.9 os sistemas que utilizam o modelo baseado em localização espacial apresentam uma redução significativa do tempo de resposta em comparação aos outros sistemas.

SMTextual apresenta um desempenho levemente melhor que os demais sistemas para valores de α que definem um peso maior para a informação textual e os sistemas SMEspacial e SMHíbrido apresentam um melhor desempenho quando o valor de alfa define um peso maior para a informação espacial.



(a) Servidores utilizados (b) Tráfego na Rede

Figura 5.13: Número de servidores utilizados e tráfego na rede ao variar o valor de α no processamento sequencial

A Figura 5.13(a) apresenta a quantidade de servidores acessados ao variar o valor de α. Os sistemas SMAleatório, SMTextual e SMEspacial acessam todos os servidores do sistema distribuído, enquanto SMHíbrido diminui a quantidade de servidores acessados para valores menores de α. Isso acontece porque à medida que o peso da informação textual vai diminuindo, SMHíbrido encontra os melhores objetos nos primeiros servidores acessados e elimina os outros do processamento da consulta.

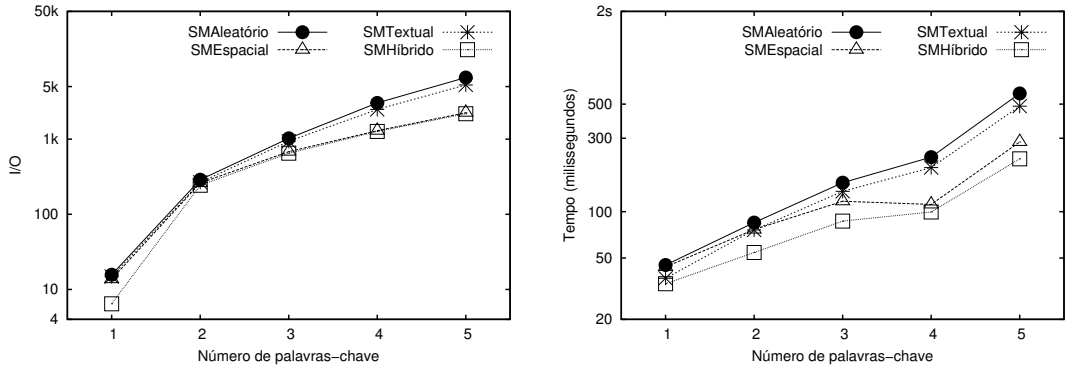
SMEspacial apresenta desempenho similar a SMHíbrido mesmo acessando todos os servidores do sistema distribuído. No processamento sequencial, cada escravo inicia seu processamento com os melhores objetos de um processamento anterior, onde só interessa objetos com escore maior que o pior escore da execução anterior. Isso permite que SMEspacial encontre os k melhores objetos nos primeiros servidores processados e nos servidores seguintes processe somente uma quantidade mínima de objetos.

A Figura 5.13(b) apresenta a quantidade de dados trafegados na rede ao variar o valor de α. Assim como na quantidade de servidores acessados, SMAleatório, SMTextual e SMEspacial trafegam uma quantidade constante de dados, enquanto SMHíbrido reduz a quantidade de servidores com menores valores de α.

5.5.2 Variando a quantidade de palavras-chave

Esta seção estuda o impacto nos modelos propostos ao variar a quantidade de palavras-chave no processamento da consulta espaço-textual top-k. A Figura 5.14

apresenta o número de páginas lidas (I/O) e o tempo de resposta obtido por cada sistema ao variar o número de palavras-chave.



(a) I/O (b) Tempo de resposta

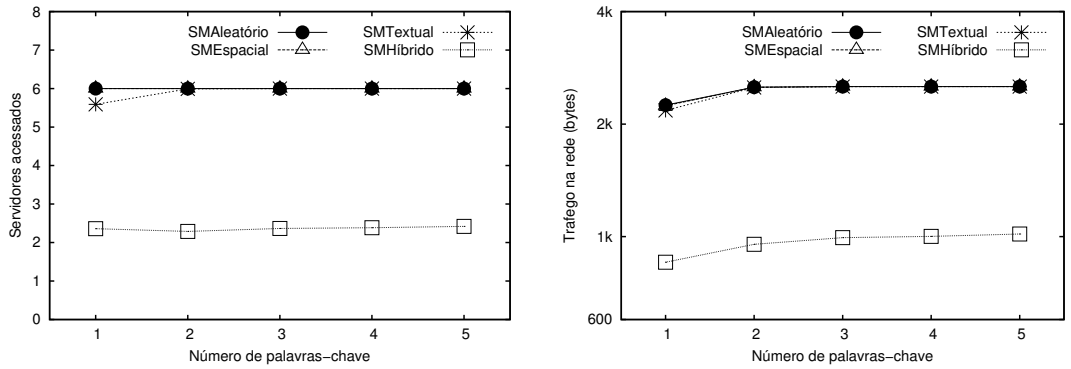
Figura 5.14: I/O e Tempo de resposta ao variar o número de palavras-chave no processamento sequencial

A Figura 5.14(a) apresenta o número de páginas acessada por cada sistema ao variar o número de palavras-chave. A quantidade de páginas acessadas em disco aumenta para todos os sistemas quando o número de palavras-chave aumenta. Ao aumentar o número de palavras-chave, aumenta o número de objetos acessados e consequentemente o número de páginas lidas também.

Os sistemas baseados em localização espacial (SMEspacial e SMHíbrido) acessam uma quantidade de páginas menor que os demais (SMAleatório e SMTextual). Isso pode ser explicado pelo valor de 0.9 no parâmetro alfa, onde é definido um peso maior para localização espacial. Entre os quatro sistemas, SMAleatório acessa uma quantidade maior de páginas em todas as variações no número de palavras-chave, enquanto o SMHíbrido acessa a quantidade menor de páginas.

A Figura 5.14(b) apresenta o tempo de resposta de cada sistema ao variar o número de palavras-chave. Assim como nos acessos a páginas do disco, todos os sistemas apresentam maiores tempo de resposta para número de palavras-chave maiores. A diferença no número de páginas acessadas refletiu no tempo de resposta para os sistemas SMAleatório e SMHíbrido, onde SMAleatório apresenta o pior desempenho enquanto SMHíbrido apresenta o melhor.

Para os sistemas SMTextual e SMEspacial, o número de páginas lidas não refletiu no tempo de resposta para consultas com até duas palavras-chave. SMTextual apresenta menor tempo de resposta que SMEspacial para consultas com uma palavra-chave, mesmo acessando uma quantidade de páginas um pouco maior. Isso acontece porque com uma quantidade de palavras-chave menor, SMTextual consegue eliminar servidores do processamento da consulta que não possuam as palavras-chave pesquisadas.



(a) Servidores utilizados (b) Tráfego na Rede

Figura 5.15: Número de servidores utilizados e o tráfego na rede ao variar o número de palavras-chave no processamento sequencial

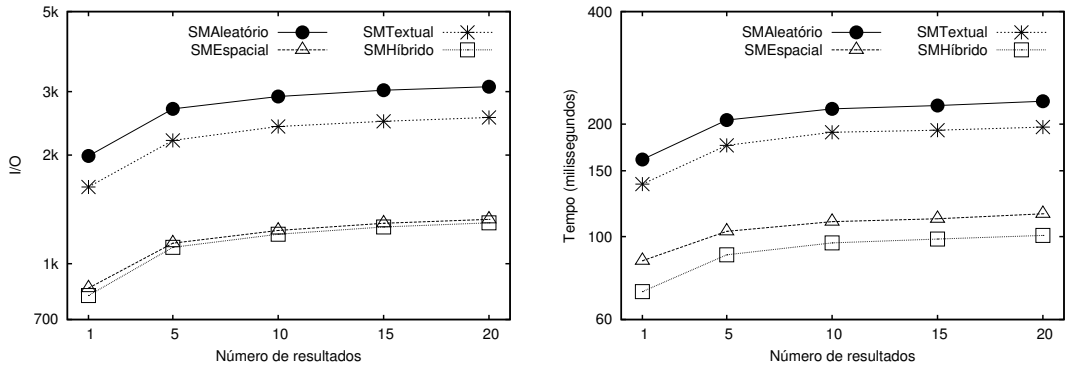
A Figura 5.15(a) apresenta o número de servidores utilizados ao variar o número de palavras-chave. SMHíbrido processa a consulta acessando somente 2 ou 3 servidores do sistema distribuído, o que justifica o melhor desempenho desse sistema. O restante dos sistemas acessam uma quantidade constante de servidores, onde todos os servidores são acessados, exceto para SMTextual que acessa um número de servidores menor para uma palavra-chave. Isso acontece porque SMTextual elimina servidores do processamento quando não possuem a palavra-chave pesquisada.

A Figura 5.15(b) apresenta a quantidade de dados trafegados na rede ao variar o número de palavras-chave. Os sistemas SMAleatório, SMEspacial e SMTextual que utilizam todos os servidores no processamento apresentam um número constante de dados trafegados na rede, exceto para uma palavra-chave que apresenta uma leve redução. Isso acontece porque a consulta com uma palavra-chave envolve uma quantidade menor de objetos e os primeiros servidores acessados podem não possuir a quantidade k de objetos desejados, ocasionando em uma transferência de dados menor. SMHíbrido apresenta um leve aumento no número de dados trafegados com a variação do número de palavras-chave.

Neste experimento, todos os sistemas são impactados pela variação do número de palavras-chave (Figura 5.14). O maior número de palavras-chave aumenta a quantidade de objetos processados e consequentemente o tempo de resposta no processamento da consulta espaço-textual top-k para todos os sistemas.

5.5.3 Variando o valor de k

Esta seção estuda o impacto nos modelos ao variar a quantidade de objetos desejados k no processamento da consulta espaço-textual top-k. A Figura 5.16 apresenta o número de páginas lidas (I/O) e o tempo de resposta obtido por cada sistema ao variar a quantidade de k.



(a) I/O (b) Tempo de resposta

Figura 5.16: I/O e Tempo de resposta ao variar o valor de k no processamento sequencial

A Figura 5.16(a) apresenta a quantidade de páginas acessadas (I/O) por cada sistema ao variar a quantidade de objetos desejados. Todos os sistemas apresentam maiores valores de I/O com valores maiores de k. O aumento no número de objetos desejados, aumenta o número de objetos processados e consequentemente aumenta o número de páginas acessadas. SMTextual e SMAleatório acessam a maior quantidade de páginas do disco, enquanto SMEspacial e SMHíbrido acessam uma quantidade menor, sendo SMHíbrido o sistema que acessa a menor quantidade de páginas do disco. Isso acontece porque o valor padrão de α define um peso maior para localização espacial.

A Figura 5.16(b) apresenta o tempo de resposta de cada sistema ao variar a quantidade de objetos desejados. A quantidade de páginas lidas refletiu no tempo de resposta de todos os sistemas. Dentre os 4 sistemas, SMAleatório apresenta o maior tempo de resposta, enquanto SMHíbrido apresenta o melhor em todas as variações. Isso acontece porque SMHíbrido elimina servidores do processamento da consulta e consequentemente trafega uma quantidade menor de dados na rede.

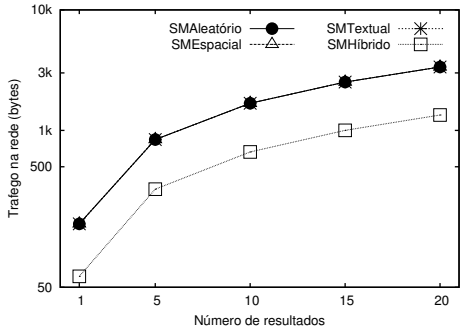


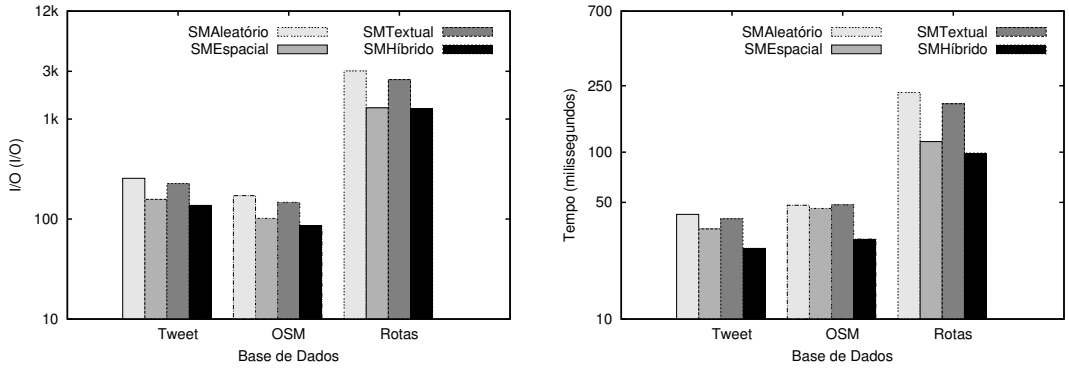
Figura 5.17: Tráfego na rede ao variar o valor de k no processamento sequencial

A Figura 5.17 apresenta a quantidade de dados trafegados na rede ao variar o número de objetos desejados. Todos os sistemas aumentam o tráfego de dados na rede com o aumento da quantidade de objetos. Enquanto os sistemas SMAleatório, SMTextual e SMEspacial trafegam quantidades iguais de dados na rede, SMHíbrido trafega uma quantidade menor. Isso acontece porque SMHíbrido acessa uma quantidade de servidores menor que os demais sistemas e consequentemente trafega uma quantidade menor de dados na rede. SMHíbrido utiliza menos de 3 servidores escravos para encontrar os melhores objetos enquanto os outros sistemas acessam todos os servidores escravos do sistema distribuído.

Neste experimento, todos os sistemas são impactados pela variação do número de objetos desejados k (Figura 5.16). Um maior número de k aumenta a quantidade de objetos processados e consequentemente diminui o desempenho de todos os sistemas.

5.5.4 Variando a base de dados

Esta seção estuda o impacto nos modelos ao variar a base de dados no processamento da consulta espaço-textual top-k. A Figura 5.18 apresenta o número de páginas lidas e o tempo de resposta obtido por cada sistema em diferentes bases de dados.



(a) I/O (b) Tempo de Resposta

Figura 5.18: I/O e Tempo de resposta ao variar a base de dados no processamento sequencial

A Figura 5.18(a) apresenta a quantidade de páginas acessadas (I/O) por cada sistema ao variar a base de dados. Os sistemas SMTextual e SMAleatório acessam o maior número de páginas do disco em todas as bases de dados, sendo que SMAleatório é o sistema com maior número de acessos a disco. Isso acontece porque SMAleatório particiona a base de dados sem nenhum critério de similaridade entre os objetos, o que dificulta o processamento local de cada escravo, sendo necessário acessar uma quantidade maior de objetos para encontrar os k melhores. SMHíbrido acessa a menor quantidade de páginas do disco, para todas as bases de dados, porque encontra

os k melhores objetos nos primeiros servidores acessados e elimina o restante do processamento.

A Figura 5.18(b) apresenta o tempo de resposta obtido por cada sistema ao variar a base de dados. A quantidade de páginas lidas refletiu no tempo de resposta de cada sistemas, onde o sistema que acessa mais páginas apresenta um tempo de resposta maior. Dentre os 4 sistemas, SMAleatório apresenta o maior tempo de resposta enquanto SMHíbrido apresenta o menor tempo de resposta em todas as bases de dados.

Neste experimento, todos os sistemas apresentam resultados similares em diferentes bases de dados, confirmando que os resultados obtidos para avaliação dos sistemas são consistentes.

Capítulo 6

Considerações Finais

"Não cruze os braços diante de uma dificuldade, pois o maior homem do mundo morreu de braços abertos."
– Bob Marley

Esta dissertação explorou o processamento distribuído da consulta espaço-textual top-k. Para permitir o processamento distribuído, quatro modelos com formas diferentes de particionar uma coleção de objetos espaço-textuais foram propostos: modelo de particionamento aleatório, modelo de particionamento por localização espacial, modelo particionamento por similaridade textual e modelo de particionamento por termos. A definição dos modelos utilizaram técnicas e algoritmos existentes nas áreas de recuperação da informação e mineração de dados, bem como, um índice específico para processar consultas espaço-textuais top-k (S2I+).

Os modelos propostos foram avaliados em um sistema distribuído real com diferentes bases de dados preparadas no decorrer desta pesquisa. O sistema que utilizou o modelo de particionamento aleatório apresentou o melhor desempenho para o processamento em paralelo das consultas, por permitir uma melhor distribuição da carga de trabalho entre os servidores do sistema distribuído, enquanto o sistema que utilizou particionamento por localização espacial apresentou o melhor desempenho no processamento sequencial, por utilizar uma quantidade menor tanto de servidores como de objetos para realizar o processamento das consultas.

6.1 Contribuições

As principais contribuições desta pesquisa são os modelos formalizados que permitem o processamento distribuído da consulta espaço-textual top-k. Sistemas de informação que utilizam sistemas distribuídos e possuem objetos espaço-textuais podem

se beneficiar de um dos modelos propostos para realizar a consulta espaço-textual top-k. Aplicações que realizam a consulta espaço-textual top-k e ainda não utilizam sistemas distribuídos também podem escolher um dos modelos para permitir o processamento distribuído da consulta.

As bases de dados utilizadas nos experimentos foram disponibilizadas para que outros pesquisadores possam utilizá-las em suas pesquisas ou até mesmo na comparação com os resultados desta pesquisa. A fundamentação teórica também é uma contribuição por apresentar conceitos e estruturas utilizadas no processamento de consultas distribuídas, textuais, espaciais e espaço-textuais, podendo auxiliar outros pesquisadores em estudos semelhantes.

6.2 Pesquisas Futuras

Esta seção apresenta algumas possibilidades de estender o trabalho explorado nesta dissertação.

Bases de Dados Nesta dissertação, as abordagens que utilizam os modelos foram avaliadas em bases de dados com no máximo 8 milhões de objetos. No entanto, a necessidade dos sistemas de informação em funcionarem com um enorme volume de dados torna interessante um aprofundamento desta pesquisa com bases maiores e de diferentes fontes de dados. O Wikipédia é um exemplo de site que armazena objetos espaço-textuais com informações textuais maiores que as dos objetos utilizados nesta pesquisa.

Sistema Distribuído Os experimentos foram realizados em um sistema distribuído real de pequeno porte com somente 7 (sete) servidores. Aplicações do mundo real utilizam sistemas distribuído com milhares de computadores, inclusive com conjuntos de servidores em localizações geográficas diferentes. As abordagens que utilizaram os modelos propostos poderiam ser avaliadas em um sistema distribuído maior, permitindo criar mais partições e podendo influenciar no desempenho dos sistemas de cada modelo, principalmente no modelo MST que pode eliminar servidores tanto no processamento paralelo como sequencial.

Índice específico Para todos os modelos foi adotado o S2I+ para processar a consulta espaço-textual top-k nos servidores escravos. O impacto da variável α no desempenho das abordagens de cada modelo apresentou resultados similares ao desempenho do índice S2I com essa variável. Experimentos podem ser realizados utilizando os mesmos modelos propostos entretanto com a um índice especifico para o processamento de consulta espaço-textual top-k diferente.

Algoritmo de agrupamento Os modelos por similaridade textual e espacial foram avaliados utilizando o *k-means* para gerar as partições. Este algoritmo apresentou um custo muito alto para criar as partições, onde as duas abordagens que utilizaram esse algoritmo criaram suas estruturas em maior tempo, principalmente para dados

textuais. Outros algoritmos ou até mesmo estruturas de dados podem ser utilizadas para criar as partições. Por exemplo, o algoritmo em grade pode ser utilizado para criar partições com regiões espaciais diferentes.

Referências Bibliográficas

[Athayde-Novaes et al. 2016] Athayde-Novaes, T. F., Fonseca, F. L., e Rocha-Junior, J. B. (2016). S2i+: Armazenamento eficiente emíndice espaço-textual. In *XXXI Simpósio Brasileiro de Banco de Dados - Short Papers, Salvador, Bahia, Brasil, October 04-07, 2016.*

[Cambazoglu et al. 2006] Cambazoglu, B. B., Catal, A., e Aykanat, C. (2006). Effect of inverted index partitioning schemes on performance of query processing in parallel text retrieval systems. In *ISCIS*, pp. 717–725.

[Cambazoglu et al. 2013] Cambazoglu, B. B., Kayaaslan, E., Jonassen, S., e Aykanat, C. (2013). A term-based inverted index partitioning model for efficient distributed query processing. *ACM Trans. Web*, 7(3):15:1–15:23.

[Cao et al. 2012] Cao, X., Chen, L., Cong, G., Jensen, C., Qu, Q., Skovsgaard, A., Wu, D., e Yiu, M. (2012). Spatial keyword querying. In Atzeni, P., Cheung, D., e Ram, S., editors, *Conceptual Modeling*, volume 7532 of *Lecture Notes in Computer Science*, pp. 16–29. Springer Berlin Heidelberg.

[Chen et al. 2013] Chen, L., Cong, G., Jensen, C. S., e Wu, D. (2013). Spatial keyword query processing: An experimental evaluation. *Proc. VLDB Endow.*, 6(3):217–228.

[Cong et al. 2009] Cong, G., Jensen, C. S., e Wu, D. (2009). Efficient retrieval of the top-k most relevant spatial web objects. *Proc. VLDB Endow.*, 2(1):337–348.

[Coulouris et al. 2011] Coulouris, G., Dollimore, J., Kindberg, T., e Blair, G. (2011). *Distributed Systems: Concepts and Design*. Addison-Wesley Publishing Company, USA, 5th edition.

[De Felipe et al. 2008] De Felipe, I., Hristidis, V., e Rishe, N. (2008). Keyword search on spatial databases. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pp. 656–665, Washington, DC, USA. IEEE Computer Society.

[Doulkeridis et al. 2017] Doulkeridis, C., Vlachou, A., Mpestas, D., e Mamoulis, N. (2017). Parallel and distributed processing of spatial preference queries using keywords. In *Proceedings of the 20th International Conference on Extending Database Technology (EDBT)*. EDBT.

[Güting 1994] Güting, R. H. (1994). An introduction to spatial database systems. *The VLDB Journal*, 3(4):357–399.

[Guttman 1984] Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. *SIGMOD Rec.*, 14(2):47–57.

[He et al. 2015] He, P., Xu, H., Zhao, X., e Shen, Z. (2015). Scalable collective spatial keyword query. In *Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference on*, pp. 182–189.

[Jonassen e Bratsberg 2010] Jonassen, S. e Bratsberg, S. (2010). A combined semi-pipelined query processing architecture for distributed full-text retrieval. In Chen, L., Triantafillou, P., e Suel, T., editors, *Web Information Systems Engineering ? WISE 2010*, volume 6488 of *Lecture Notes in Computer Science*, pp. 587–601. Springer Berlin Heidelberg.

[Kwon et al. 2015] Kwon, H.-Y., Wang, H., e Whang, K.-Y. (2015). G-index model: A generic model of index schemes for top-k spatial-keyword queries. *World Wide Web*, 18(4):969–995.

[Kwon et al. 2013] Kwon, H.-Y., Whang, K.-Y., Song, I.-Y., e Wang, H. (2013). Rasim: a rank-aware separate index method for answering top-k spatial keyword queries. *World Wide Web*, 16(2):111–139.

[Li 2011] Li, H. (2011). A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 94(10):1854–1862.

[Li et al. 2011] Li, Z., Lee, K., Zheng, B., Lee, W.-C., Lee, D. L., e Wang, X. (2011). Ir-tree: An efficient index for geographic document search. *Knowledge and Data Engineering, IEEE Transactions on*, 23(4):585–599.

[Moffat et al. 2007] Moffat, A., Webber, W., Zobel, J., e Baeza-Yates, R. (2007). A pipelined architecture for distributed text query evaluation. *Information Retrieval*, 10(3):205–231.

[Papadias et al. 2001] Papadias, D., Kalnis, P., Zhang, J., e Tao, Y. (2001). Efficient olap operations in spatial data warehouses. In *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, SSTD '01, pp. 443–459, London, UK, UK. Springer-Verlag.

[Porwal e Shiwani 2015] Porwal, A. e Shiwani, S. (2015). Efficient processing of top-k spatial boolean queries for distributed system. *Suresh Gyan Vihar University Journal of Engineering and Technology*, 1(2):38–43.

[Rocha-Junior et al. 2011] Rocha-Junior, J. a. B., Gkorgkas, O., Jonassen, S., e Nørvåg, K. (2011). Efficient processing of top-k spatial keyword queries. In *Proceedings of the 12th International Conference on Advances in Spatial and Temporal Databases*, SSTD'11, pp. 205–222, Berlin, Heidelberg. Springer-Verlag.

[Samet 1984] Samet, H. (1984). The quadtree and related hierarchical data structures. *ACM Comput. Surv.*, 16(2):187–260.

[Tanenbaum e Steen 2006] Tanenbaum, A. S. e Steen, M. v. (2006). *Distributed Systems: Principles and Paradigms (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Witten e Frank 2005] Witten, I. H. e Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Zhang et al. 2013] Zhang, D., Tan, K.-L., e Tung, A. K. H. (2013). Scalable top-k spatial keyword search. In *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13, pp. 359–370, New York, NY, USA. ACM.

[Zhou et al. 2005] Zhou, Y., Xie, X., Wang, C., Gong, Y., e Ma, W.-Y. (2005). Hybrid index structures for location-based web search. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pp. 155–162, New York, NY, USA. ACM.

[Zimmermann et al. 2004] Zimmermann, R., Ku, W.-S., e Chu, W.-C. (2004). Efficient query routing in distributed spatial databases. In *Proceedings of the 12th Annual ACM International Workshop on Geographic Information Systems*, GIS '04, pp. 176–183, New York, NY, USA. ACM.

[Zobel e Moffat 2006] Zobel, J. e Moffat, A. (2006). Inverted files for text search engines. *ACM Comput. Surv.*, 38(2).