Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Computação Aplicada

Identificando Locais Influentes através de Palavras-Chave

Felipe Pains Oliveira Silva

Feira de Santana

2019

Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Computação Aplicada

Felipe Pains Oliveira Silva

Identificando Locais Influentes através de Palavras-Chave

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

Feira de Santana

2019

Ficha Catalográfica - Biblioteca Central Julieta Carteado - UEFS

S58 Silva, Felipe Pains Oliveira
Identificando locais influentes através de palavras-chave / Felipe Pains Oliveira Silva. – 2019.
61 f.

Orientador: João Batista da Rocha Junior
Dissertação (mestrado) – Universidade Estadual de Feira de Santana, Programa de Pós-Graduação em Computação Aplicada, Feira de Santana, 2019.

1. Locais influentes. 2. Identificadores locais. 3. Objetos de referência. 4. Geolocalização. 5. Consulta textual – Internet. 6. Algoritmos. I. Rocha Junior, João Batista da, orient. II. Universidade Estadual de Feira de Santana. III. Título.

CDU: 004.78:528

Luis Ricardo Andrade da Silva - Bibliotecário - CRB-5/1790

Felipe Pains Oliveira Silva

Identificando Locais Influentes através de Palavras-chave

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, 15 de fevereiro de 2019

BANCA EXAMINADORA

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

Dr. Matheus Giovanni Pires
Universidade Estadual de Feira de Santana

Dr. Daniel Gouveia Costa
Universidade Estadual de Feira de Santana

Abstract

Spatial data or spatial objects are used by various applications such as Google Maps or Uber. These applications provide location services for users, such as: finding the nearest restaurant or driver. Similarly, the influential location query aims to find the best place for the installation of a new establishment, according to the weight of the objects in its vicinity. Weight is a constant predefined by applications, such as the number of stars or the number of people in a house. In this research, we study a new type of query where the weight of objects are defined taking into account a set of keywords defined by the user. Therefore, it is possible to select relevant neighbors based on user preference. In this research a new type of query is specified, new algorithms are presented to process this query efficiently and the algorithms proposed in real data sets have been evaluated.

Keywords: spatial objects, influential location, spatio-textual query, preference queries.

Resumo

Dados espaciais, ou objetos espaciais são utilizados por diversas aplicações como Google Maps ou Uber. Estas aplicações fornecem serviços de localização para os usuários, tais como: encontrar o restaurante ou motorista mais próximo. Da mesma forma, a consulta de localização influente visa encontrar o melhor lugar para a instalação de um novo estabelecimento, de acordo com o peso dos objetos em sua proximidade. O peso é uma constante predefinida pelos aplicativos, como o número de estrelas ou o número de pessoas em uma casa. Nesta pesquisa, estudamos um novo tipo de consulta onde o peso dos objetos são definidos levando em consideração um conjunto de palavras-chave definidas pelo usuário. Portanto, é possível selecionar os vizinhos relevantes com base na preferência do usuário. Nesta pesquisa é especificada um novo tipo de consulta, é apresentado novos algoritmos para processar essa consulta de forma eficiente e foram avaliados os algoritmos propostos em conjuntos de dados reais.

Palavras-chave: objetos espaciais, locais influentes, consulta espaço-textual, consultas preferenciais .

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.**

Agradecimentos

Em primeiro lugar, agradeço a Deus pela oportunidade de viver esses momentos, trazendo alegria e orgulho aos meus pais e a todas as pessoas fantásticas que contribuíram para este trabalho.

Agradeço a minha mãe Neire Pains e meu pai José Edson, agradeço os incentivos fundamentais, os conselhos precisos, e ao investimento depositado. Vocês são e sempre serão exemplos em minha jornada. Obrigado, amo muito vocês.

Agradeço a minha namorada Maise Vieira, pelo companheirismo, carinho e por muitas vezes me ajudar em momentos de dificuldade neste trabalho. Obrigado pelos conselhos, que me fizeram renovar o ânimo e tornar meus dias mais felizes. Amo você.

Ao meu orientador João B. Rocha-Junior, agradeço pela incrível paciência que o senhor teve comigo, por todos os conselhos e ensinamentos em cada reunião. Sem sua orientação, este trabalho não seria concluído.

Ao meu irmão João Victor, pelas momentos de descontração que por muitas vezes serviram de escape durante a pesquisa.

Aos amigos que fiz dentro do PGCA e do Grupo ADAM, agradeço pelo incentivo de vocês, por ter o sucesso de vocês como exemplo. Obrigado também pelos momentos de descontrações, as reuniões de estudo que além de produzir, serviam para conversar e relaxar depois de tantas obrigações durante o mestrado. "Quanto conhecimento presente nessas reuniões.

E por fim, mas não menos importante, aos meus familiares pelos conselhos e incentivos durante toda a minha jornada.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Abreviações

Abreviação	**Descrição**
PCs	*Personal Computers*
IF	*Inverted File*
MBR	*Minimum Bounding Rectangle*
BRNN	*Bichromatic Reverse Nearest Neighbor*
aR-Tree	*Aggregated* R-Tree
$Q.D$	Conjunto de palavras-chave da consulta
C	Conjunto de locais candidatos
T	Conjunto de locais existentes
P	Conjunto de objetos de referência
$p.D$	descrição textual de um objeto de referência

Capítulo 1

Introdução

"Só se pode alcançar um grande êxito quando nos mantemos fiéis a nós mesmos."
– Friedrich Nietzsche

É quase impossível imaginar como seriam os dias atuais sem o acesso à internet e a tecnologia. As pessoas se mantém conectadas em casa, no trabalho, até mesmo na escola ou nos momentos de lazer. Dentre os dispositivos que acessam a internet, destacam-se os *Smartphones*, por exemplo, pelo grande número de usuários. Os *Smartphones*, além de acesso à internet e outras funcionalidades, podem produzir dados espaciais, através do GPS (*Global Positioning System*). Dados espaciais, são elementos que apresentam uma localização geográfica (latitude e longitude) e podem estar associados a escolas ou hospitais, por exemplo. Estes dados são utilizados por diversas aplicações como GOOGLE MAPS[1], WAZE[2] e UBER[3], que fornecem serviços de localização para os usuários, tais como, encontrar o restaurante mais próximo ou computar a melhor rota para um determinado local.

O crescimento do uso destas aplicações tem impulsionado o interesse de pesquisadores por novas consultas para localizar objetos espaciais de interesse. Entre estas consultas, destacam-se as voltadas para identificar locais influentes [Du et al. 2005, Zhang et al. 2006, Xiao et al. 2011, Yang et al. 2017, Xu et al. 2017]. Estas consultas tem como objetivo responder à questões como: a) "Qual a localização mais influente para a instalação de um novo estabelecimento que atraia mais clientes?"; b) "Qual o local mais influente para instalar um hospital que fique mais próximo da população atendida?".

[1] www.google.com.br/maps
[2] www.waze.com
[3] www.uber.com

A consulta tradicional para localizar locais influentes requer, como parâmetro, um conjunto de locais candidatos, um conjunto de locais existentes e um conjunto de objetos de referência. Os locais mais influentes são selecionados dentro do conjunto de locais candidatos, levando-se em consideração os locais existentes e os objetos de referência (objetos de referência).

A influência de um local candidato é medida através do número de objetos de referência que são atraídos pelo local candidato, levando-se em consideração que todos os objetos de referência tem o mesmo peso. Diz-se que um cliente (objeto de referência) é atraído por um local candidato, quando a distância entre o cliente e o local candidato é menor que a distância entre o cliente e qualquer outro local existente. O peso é um valor numérico associado a cada cliente estabelecido por cada aplicação, como por exemplo, a renda de uma pessoa. Assim, a consulta por local influente computa a influência de todos os locais candidatos e retorna os que tiverem maior escore.

Por exemplo, a consulta por um local influente pode ajudar um empreendedor que deseja abrir uma nova lanchonete e gostaria de saber qual o melhor local, entre os terrenos que estão disponíveis (locais candidatos). Assim, ele passa como parâmetro para a consulta os locais candidatos (terrenos), os locais existentes (lanchonetes já estabelecidas na cidade) e os objetos de referência (e.g. local das residências através de dados do IBGE). A consulta computa a influência de todos os terrenos e retorna o que tiver maior escore.

A Figura 1.1 mostra objetos de referência (clientes) que possuem pesos diferentes. Os objetos de referência possuem a letra p e os locais existentes (ex. lanchonete) possuem a letra t. Os objetos p_1 e p_3 são atraídos por t_2, porque eles são mais próximos de t_2 do que de t_1 e p_2 é atraído por t_1. Neste caso, o local existente t_2 tem valor de influência $0,7$ devido a soma dos pesos dos objetos de referência p_1 e p_3 atraídos, enquanto o local existente t_1 tem valor de influência $0,6$, referente ao peso do único objeto de referência atraído p_2.

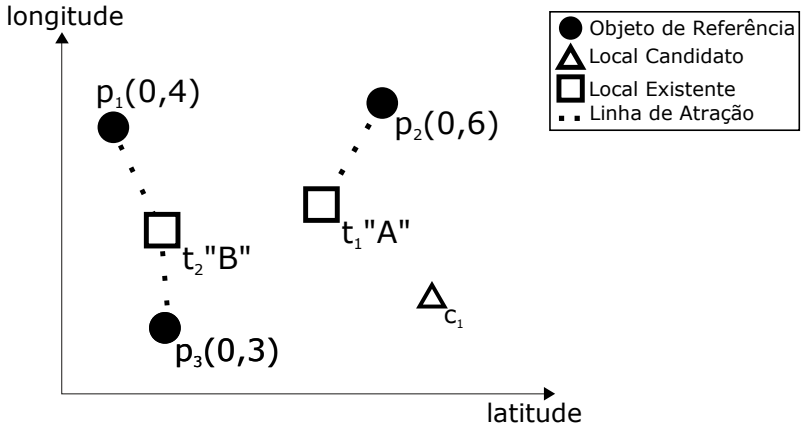
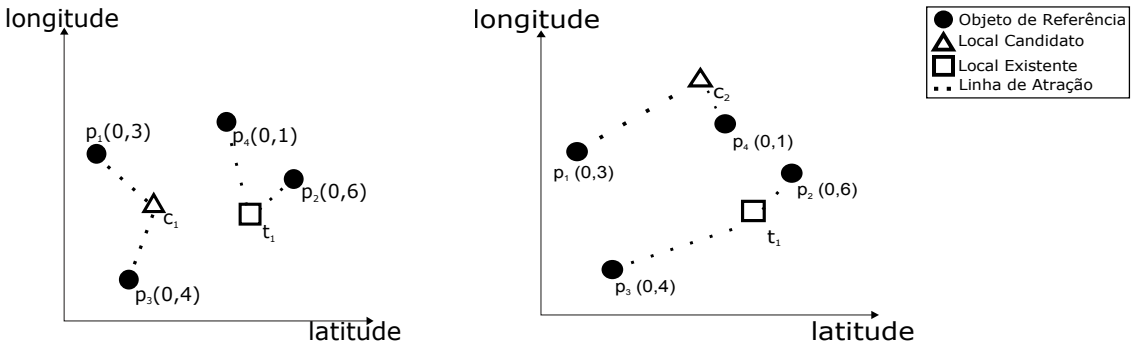


Figura 1.1: Representação de atração entre os objetos de referência e locais existentes

A consulta de localização competitiva é um dos tipos de consulta por localização influente [Xiao et al. 2011, Yao et al. 2014] que busca uma nova instalação que maximize o peso total dos objetos de referência atraídos pelo local candidato. De tal forma que, dado um conjunto de locais candidatos, um conjunto de locais existentes e um conjunto de objetos de referência, é dito que um objeto de referência $p \in P$ é atraído por um local candidato $c \in C$ se, e somente se, não existir um local existente $t \in T$, tal que a distância entre qualquer local existente t para o objeto de referência p seja maior que a distância entre c e p.

Os objetos espaciais podem conter outras informações além da localização geográfica. Uma dessas é a informação textual, que pode dar uma melhor descrição sobre os objetos [de Almeida e Rocha-Junior 2016], como é o caso do *Twitter*, que associa o "tweet" enviado a partir de um *smartphone* a uma localização espacial. Outro exemplo é o *OpenStreetMap*, uma plataforma onde objetos como restaurantes e hospitais possuem um texto descritivo e uma localização espacial. Estes textos são utilizados na identificação de objetos com relevância para algumas consultas [Cao et al. 2012, Rocha-Junior et al. 2011, de Almeida e Rocha-Junior 2016]. Os objetos com informações textuais e espaciais são tratados como objetos espaço-textuais [Vaid et al. 2005].

Um dos problemas da consulta por locais influentes é que ela utiliza um peso pré-definido associado aos objetos de referência, por exemplo, a renda de uma pessoa. A utilização do peso fixo não satisfaz completamente, pois os usuários buscam informações de acordo com o conjunto de palavras-chave consultado. Por exemplo, em uma consulta no Google, em geral os *links* mais satisfatórios são os que estão diretamente relacionados com os caracteres digitados pelo usuário. Assim, ao invés de assumir que todos os objetos de referência atraídos por um local candidato são relevantes para calcular a influência do mesmo, poderia utilizar apenas os objetos de referência que contenham alguma relevância com o interesse do usuário especificado através das palavras-chave.



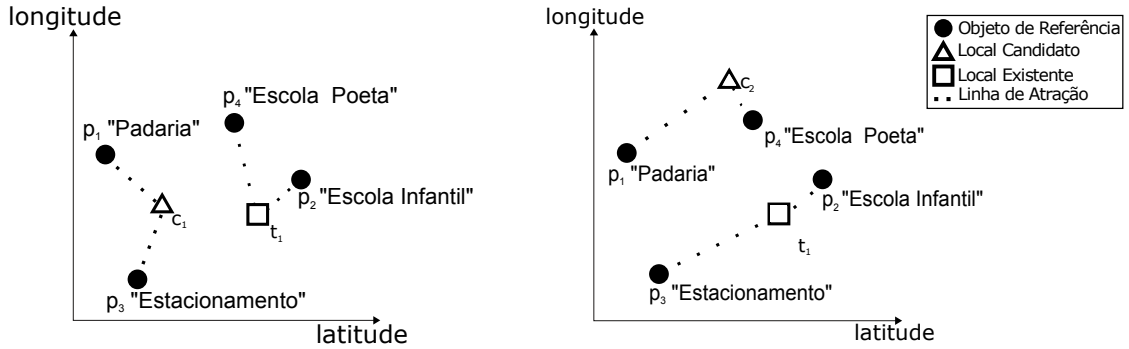
(a) Objetos de referência atraídos por c_1

(b) Objetos de referência atraídos por c_2

Figura 1.2: Representação de atração dos locais candidatos c_1 e c_2 com pesos predefinidos para os objetos de referência

As consultas por locais influentes atuais não utilizam informação textual para computar o peso dos objetos de referência. Por exemplo, na Figura 1.2(a) o local candidato c_1 têm influência igual a $0,7$, de acordo com os pesos predefinidos dos objetos de referência atraídos por ele (p_1 e p_3), enquanto que a Figura 1.2(b), com a mesma disposição dos objetos de referência e local existente presentes na Figura 1.2(a), apresenta o local candidato c_2 que têm influência igual a $0,4$. Assim o local candidato c_1 é retornado em uma consulta pelo local mais influente. Porém, assumindo que um empresário deseja instalar uma nova lanchonete em uma cidade, ele poderia pensar em executar uma consulta de locais influentes para identificar a influência de alguns locais candidatos utilizando a palavra-chave "escola" para selecionar objetos de referência relevantes para o seu negócio.

Analisando a Figura 1.3(a), o local candidato c_1 atrai os objetos de referência p_1 e p_3, enquanto que o local candidato c_2 atrai os objetos de referência p_1 e p_4 (Figura 1.3(b)). Se o usuário realizar uma consulta que compare os locais candidatos c_1 e c_2, o local candidato c_2 seria a resposta da consulta, pois possui um objeto de referência (p_4) com similaridade textual com a palavra-chave ("escola") desejada.



(a) Relação de proximidade com o local candidato c_1

(b) Relação de proximidade com o local candidato c_2

Figura 1.3: Representação de atração dos locais candidatos com texto nos objetos de referência

Este trabalho tem como objetivo apresentar um novo método para encontrar locais influentes em consultas espaciais, utilizando o texto dos objetos de referência para calcular a relevância de cada objeto para a consulta. Este cálculo de relevância demonstra ser desafiador, pois a cada consulta o peso dos objetos de referência são computados conforme a sua similaridade com as palavras-chave da consulta, diferente do modelo tradicional que possui pesos fixos para todos os objetos de referência.

1.1 Objetivo

Este trabalho tem como objetivo geral apresentar uma nova abordagem para encontrar locais influentes em consultas espaciais, utilizando o texto dos objetos como referência para calcular a influência dos locais candidatos e selecionar os mais influentes. Sendo assim, o usuário pode refinar melhor quais os objetos de referência estão diretamente relacionados às palavras-chave da consulta.

As principais contribuições deste trabalho são: 1) propor um novo tipo de consulta de localização competitiva que utiliza a informação textual nos objetos de referência para computar o escore dos objetos candidatos; 2) propor algoritmos que processam a consulta de forma eficiente e 3) realizar experimentos para avaliar os algoritmos propostos utilizando bases de dados reais.

1.2 Questões de Pesquisa

A questão de pesquisa geral desta dissertação é: como identificar k locais mais influentes, levando em consideração a similaridade do texto dos objetos de referência com as palavras-chave de interesse do usuário? Esta questão conduz à questões mais específicas. As questões de pesquisa abordadas por esta dissertação são apresentadas a seguir:

Q 1. Como processar a consulta dos k locais mais influentes, levando em consideração a similaridade textual dos objetos de referência com as palavras-chave?

Q 2. É possível utilizar índices textuais para melhorar o desempenho da consulta?

1.3 Aplicações

Empresas, governos e pessoas podem se beneficiar da consulta por localização influente através de palavras-chave. Essa consulta pode fornecer resultados com maior relação com interesse dos usuários, devido ao cálculo de importância dos objetos de referência serem computados através do texto que descrevem cada objeto.

Tomada de decisão para empreendedores

Identificar um local para a abertura de um ponto comercial virou um grande desafio para empreendedores. Um local que atraia o maior número de clientes possíveis é essencial para que a empresa consiga se estabelecer. "O ponto comercial, principalmente quando se trata de comércio varejista, é fundamental para o bom desempenho das vendas", Sebrae[4].

[4]http://www.sebrae.com.br/sites/PortalSebrae/artigos/o-sucesso-do-negocio-depende-de-sua-localizacao,11e89e665b182410VgnVCM100000b272010aRCRD - Acessado em 12 de Abril de

O planejamento estrutural do negócio é importante, mas o ponto comercial ideal pode melhorar a idealização do projeto. “A missão é conseguir o melhor ponto e o menor custo. A ciência de localização de ponto está cada vez mais valorizada”, Luis Henrique Stockler, sócio-diretor da consultoria de varejo e franquias baStockler [5].

O *OpenStreetMap* disponibiliza dados geográficos de regiões ou cidades para serem analisados ao dispor do usuário. Nele é possível identificar a quantidade de lanchonetes em uma cidade e a localização de cada uma, por exemplo. Com isso é possível obter a localização de empreendimentos que podem estar relacionados como concorrentes ao tipo de negócio (ex. lanchonete) que um empreendedor esteja interessado em instalar.

Mensagens do *Twitter* enviadas do *smartphone* que detém GPS integrado possuem uma localização espacial e texto (presente no *tweet*). Uma base de dados de *tweets* pode ser utilizada por uma aplicação para identificar possíveis lacunas de empreendimentos que os usuários do *Twitter* estão com interesse.

Desta forma é possível utilizar o *tweet* para indicar a necessidade de instalação de algum empreendimento em uma rua ou região. A seguir é detalhado cada um dos objetos da consulta para este tipo de aplicação.

Objeto de interesse. Nesta aplicação, o conjunto de interesse é formado por terrenos ou pontos comerciais disponíveis para instalação de um empreendimento (ex. lanchonete).

Objetos de referência. O conjunto de objetos de referência é formado pelos *tweets*.

Objetos existentes. O conjunto de locais existentes é formado pelos empreendimentos presentes identificados como concorrentes do tipo de negócio a ser instalado.

Segurança Pública

Assim como para a instalação de empreendimentos, o *tweet* pode ser analisado também para localizar ruas, ou áreas públicas, com a necessidade de instalação de novo posto policial. O órgão público pode buscar *tweets* contendo termos “assalto” e “crime” em seu conteúdo. Desta forma, as mensagens podem ser utilizadas como indicador de violência em regiões, permitindo que o poder público se organize, e planeje a instalação de um novo posto policial próximo a maior incidência de crimes para melhorar a segurança no local.

A seguir é detalhado cada um dos objetos da consulta para este tipo de aplicação e é descrito um exemplo.

2018

[5] http://exame.abril.com.br/pme/como-escolher-o-melhor-lugar-para-comecar-seu-negocio - Acesso em 28 de setembro de 2017

Objeto de interesse. Nesta aplicação, o conjunto de interesse é formado por locais estratégicos disponíveis para instalação de um posto policial.

Objetos de referência. O conjunto de objetos de referência é formado pelos *tweets* dos usuários informando atos de violência.

Objetos existentes. O conjunto de locais existentes é formado pelos postos policiais presentes na cidade.

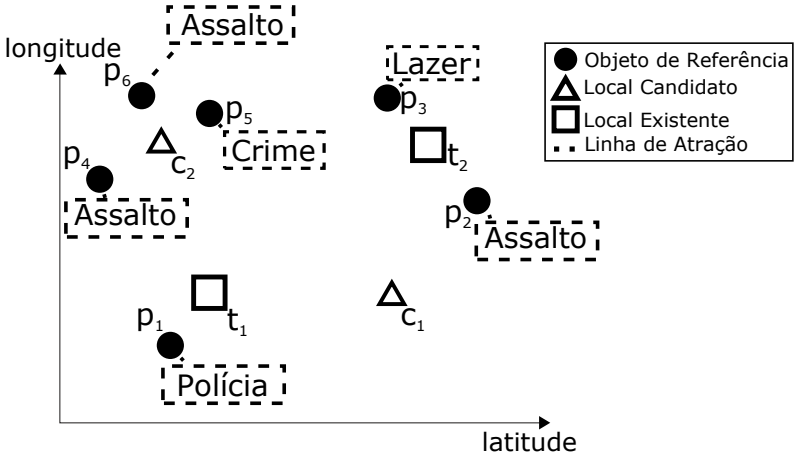


Figura 1.4: Representação de aplicação em segurança pública

Exemplo. A Figura 1.4 ilustra uma região espacial contendo objetos de interesse c (locais para instalação de um posto policial), objetos de referência p (*tweets*) e os objetos existentes t. Uma aplicação pode mostrar o local mais influente para instalação de um posto policial de acordo com palavra-chave "assalto". Neste contexto, a consulta retornaria o objeto de interesse c_2 como local mais influente, visto que em sua proximidade contém objetos de referência mais relevante para a palavra-chave.

1.4 Publicação

Esta seção apresenta a publicação originada desta pesquisa de mestrado, acompanhadas por uma breve descrição.

- Consulta de Locais Influentes Através de Palavras-Chave: uma proposta. Felipe Pains Oliveira Silva, João B. Rocha-Junior. (2017). In WPOS/ERBASE, Bahia, Cruz das Almas, Agosto de 2017.

  Artigo apresentado no *Workshop* de pós-graduação da Escola Regional de Computação Bahia-Alagoas-Sergipe (ERBASE). Neste artigo foi apresentado apenas uma definição da consulta por locais influentes através de palavras-chave e algumas informações relacionadas a proposta da pesquisa de mestrado. Durante o WPOS, o artigo foi analisado por uma banca de doutores que ofereceram sugestões para melhorias do trabalho.

1.5 Sinopse da Dissertação

Capítulo 1: Introdução. Este capítulo apresenta a introdução desta pesquisa. Nele está contido também motivação, o objetivo geral e as contribuições do trabalho.

Capítulo 2: Revisão Bibliográfica. Neste capítulo é apresentado os trabalhos e temas que são relevantes para o desenvolvimento desta pesquisa.

Capítulo 3: Trabalhos Relacionados. Este capítulo apresenta os trabalhos relacionados à pesquisa.

Capítulo 4: Metodologia. Este capítulo apresenta as etapas previstas para a realização da pesquisa.

Capítulo ??: Definição do Problema . Este capítulo apresenta a como a consulta é especificada.

Capítulo 5: Algoritmos. Este capítulo apresenta os algoritmos propostos.

Capítulo 6: Resultados. Este capítulo demonstra os resultados obtidos.

Capítulo 7: Considerações Finais. Este capítulo conclui a dissertação, apresentando as principais contribuições e os trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

"Por vezes sentimos que aquilo que fazemos não é senão uma gota de água no mar. Mas o mar seria menor se lhe faltasse uma gota."
– Madre Teresa de Calcuta

Neste capítulo são apresentados os conceitos fundamentais para contextualizar o trabalho desenvolvido nesta pesquisa. O primeiro tópico discutido refere-se às consultas textuais (Seção 2.1). Em seguida, as consultas espaciais (Seção 2.2), bem como índices espaciais. Posteriormente, é apresentado os conceitos da consulta de bicromática inversa do vizinho mais próximo (Seção 3.1). Finalmente, trabalhos realizados envolvendo o problema de consultas por locais influentes (Seção 3.2).

2.1 Consulta Textual

Tabela 2.1: Identificação das variáveis.

Sigla	**Significado**
$Q.D$	Conjunto de palavras-chave da consulta
C	Conjunto de locais candidatos
T	Conjunto de locais existentes
P	Conjunto de objetos de referência
$p.D$	descrição textual de um objeto de referência

Segundo Zobel e Monffat [Zobel e Moffat 2006], a consulta textual tem um papel fundamental na tecnologia. Estas consultas são utilizadas para recuperar informações de acordo com um conjunto de palavra-chave da consulta

[Salminen e Tompa 1994]. Os motores de busca que estão presentes na *Web* são exemplos de aplicação direta da utilização da consulta com palavra-chave para obtenção de informação.

Uma base de dados textual é uma coleção de dados alfanuméricos (e.g., Páginas *web*, artigos de jornais, publicações acadêmicas). Cada unidade de uma coleção de dados é denominada "Documento" [de Almeida e Rocha-Junior 2016]. O tipo de Documento que o usuário deseja obter pode ser encontrado a partir de uma ou mais palavras-chave digitadas. Em um Sistema de Recuperação de Informação Textual típico, o usuário descreve o Documento que deseja obter através de um conjunto de palavras-chave (*bag of words*) [Zobel e Moffat 2006], e o sistema deve apresentar o conjunto de Documentos relacionados as palavras-chave da consulta do usuário.

Exemplo. Um usuário pode, a partir da base textual apresentada na Tabela 2.2, encontrar a informação que deseja a partir da consulta textual. Cada frase, neste exemplo, é considerada um Documento. O usuário pode identificar o documento de acordo com o seu interesse descrito em uma consulta textual. Assim, quando o usuário realiza uma consulta textual utilizando palavras-chave, esta consulta vai apresentar todas as frases relevantes de acordo com o conjunto de palavras-chave. Como exemplo, caso o usuário realize uma pesquisa com a palavra-chave "horas", a consulta textual retorna os documentos 3 e 5 (Tabela 2.2) como relevantes.

ID	Texto
1.	No trabalho não podemos dormir
2.	Filho.. filho.. É melhor dormir cedo hoje
3.	São três horas de viagem até Salvador
4.	Remédios para dormir podem causar vícios
5.	Descansar oito horas por dia faz bem para a saúde

Tabela 2.2: Lista de mensagens hipotéticas do *Twitter*. Fonte: Adaptada Zobel and Moffat 2006. [Zobel e Moffat 2006]

Para pequenas coleções, a consulta pode ser realizada processando todos os documentos existentes na base, porém, para bases maiores faz se o uso de índices: um tipo de estrutura que mapeia os termos para os documentos que os contêm, para processar essas consultas textuais de forma eficiente [Zobel e Moffat 2006, Cambazoglu et al. 2006]. Para consultas de textos, o índice mais utilizado é o arquivo invertido (*Inverted File - IF*) [Zobel e Moffat 2006, Arroyuelo et al. 2014]: uma estrutura que armazena os termos em listas invertidas, e registra os documentos que estão presentes cada termo do vocabulário (Figura 2.1). Uma etapa de pré-processamento pode ser realizada antes de criar um IF, ela é chamada de *parsing* [Zobel e Moffat 2006, de Almeida e Rocha-Junior 2016].

O índice invertido é um método que indexa palavras de uma coleção de documentos para melhorar a consulta textual. É formado por dois elementos: vocabulário (dicionário), que é o conjunto de termos diferentes nos documentos e a quantidade

de documentos que contém o termo (f_t); e a lista invertida (ou *postings list*), que armazena o Documento (D_{id}) onde o termo (t) ocorre com a frequência ($f_{t,D}$) do termo na descrição textual do Documento [Zobel e Moffat 2006].

O *parsing* é um processo de varredura para identificar sua estrutura gramatical. Este processo pode ser realizado em duas etapas: remoção de *stop words* e a técnica de *casefold*. A remoção de *stop words* consiste em eliminar palavras que são comuns e que agregam pouca informação aos textos, como: $as, at, os, de, para, com, so, foi$ e *etc*. Cada língua possui a sua própria lista de *stop words*. O *casefold* é uma técnica que transforma todas as letras em minúsculas. Aplicando o *parsing* no documento 3 da Tabela 2.2 seria alterado para "três horas viagem salvador". Todo esse pré-processamento resulta na diminuição da quantidade de palavras que serão armazenadas, facilitando o processamento dos dados textuais [Zobel e Moffat 2006].

Exemplo. A Figura 2.1 representa parcialmente a indexação dos termos presentes na Tabela 2.2. No vocabulário estão armazenados os termos, a quantidade de documentos (f_t) que o termos aparecem e um ponteiro (seta) que indica a lista invertida correspondente ao termo. A lista invertida é composta por um identificar do documento que o termo está presente e a frequência deste termo no documento. Tomando o termo "filho" como exemplo, o índice invertido demonstra no vocabulário que ele teve duas ocorrências ($f_t = 2$), sendo que no segundo documento ($D_id = 2$) é apresentado duas vezes ($f_{t,_D} = 2$).

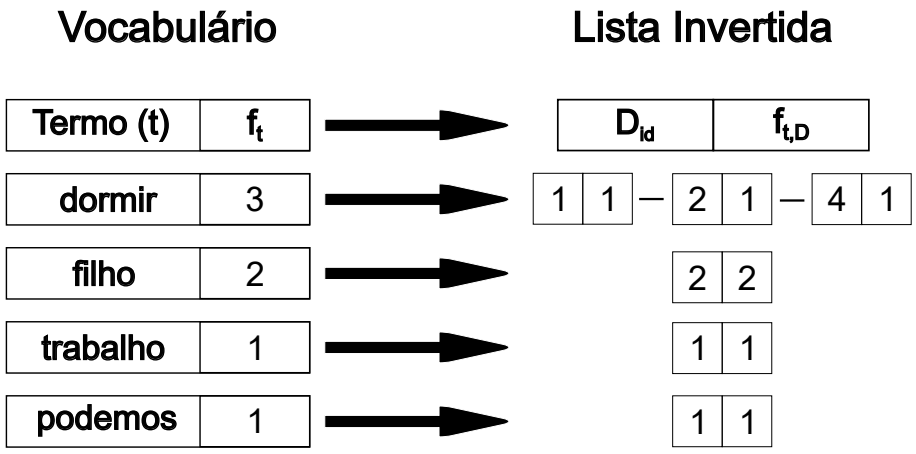


Figura 2.1: Exemplo de Arquivo Invertido utilizando os termos da Tabela 2.1.

Esse sistema de Recuperação de Informação (RI) textual consegue processar os documentos e identificar a relevância dos mesmos em relação à consulta. Assim, um tipo de classificação é utilizado nos documentos para recuperar a melhor resposta possível ao usuário. Para criar essa classificação é necessário quatro etapas: 1) encontrar no vocabulário os termos e suas listas correspondentes; 2) aplicar a medida de similaridade para indicar a proximidade existente entre um documento e as palavras-chave existentes na consulta do usuário; 3) ranquear os documentos de acordo com a relevância textual; e 4) retornar os documentos ordenados pela relevância [Manning et al. 2008].

Uma das formas de calcular a similaridade textual é com a utilização da similaridade cosseno amplamente empregada em trabalho de Recuperação de Informação [Cohen et al. 2003, Zobel e Moffat 2006, Zhu et al. 2011]. A similaridade cosseno $\theta(p.D, Q.D)$, onde $p.D$ é o texto associado aos objeto e $Q.D$ o conjunto de palavras-chave, é representada através de um ângulo entre o vetor da consulta $W_{Q,t}$ (t é o termo presente na base de dados) e o vetor de documento $W_{d,t}$, quanto menor o ângulo, mais similar o documento é considerado. Sendo assim, só será possível indicar um documento como relevante para a consulta, caso ele tenha pelo menos um dos termos da palavra-chave da consulta em sua descrição textual.

Zobel e Moffat [Zobel e Moffat 2006] indicam algumas medidas utilizadas no cálculo do ângulo cosseno entre os vetores, tais como:

- $f_{d,t}$, a frequência do termo t no documento D
- $f_{Q,t}$, a frequência do termo t na consulta
- f_t, número de documentos com ocorrência do termo t
- N, número total de documentos na coleção

A similaridade cosseno pode ser calculada de diferentes formas [Manning et al. 2008, Zobel e Moffat 2006]. Para essa pesquisa, é utilizada a variação apresentada por Zobel e Moffat [Zobel e Moffat 2006] demonstrada nas equações abaixo:

$$W_Q = \sqrt{\sum_t W_{Q,t}^2} \tag{2.1}$$

$$W_{Q,t} = \ln\left(1 + \frac{N}{f_t}\right) \tag{2.2}$$

$$W_d = \sqrt{\sum_t W_{d,t}^2} \tag{2.3}$$

$$W_{d,t} = 1 + \ln f_{d,t} \tag{2.4}$$

$$S_{Q,d} = \frac{\sum W_{Q,t} \cdot W_{d,t}}{W_q \cdot W_d} \tag{2.5}$$

A pontuação é calculada pela equação $S_{Q,d}$. O vetor da consulta ($W_{Q,t}$) calcula o peso do termo t na consulta Q, enquanto o vetor de documento ($W_{d,t}$) calcula o peso do termo t no documento d. Quanto maior o valor de $S_{Q,d}$, maior é a relevância textual entre o documento d e a consulta Q.

O vetor da consulta ($W_{Q,t}$) corresponde a frequência inversa do documento (*inverse document frequency* - IDF), enquanto o vetor do documento descreve a frequência do termo (*term frequecy* - TF). Por isso, a equação $S_{Q,d}$ é conhecida na literatura como TFxIDF [Zobel e Moffat 2006].

2.2 Consulta Espacial

Os bancos de dados se adaptam para conseguir gerenciar diversos tipos de dados [Güting 1994]. A produção de dados espaciais tem crescido bastante, aumentando a aplicação dos mesmos em sistemas de banco de dados espaciais [Rigaux et al. 2001]. Estes sistemas têm suporte para tipos de dados espaciais como ponto, linhas e polígonos. Além disso, devem garantir a recuperação de uma coleção de objetos em um espaço sem a varredura do conjunto completo [Güting 1994].

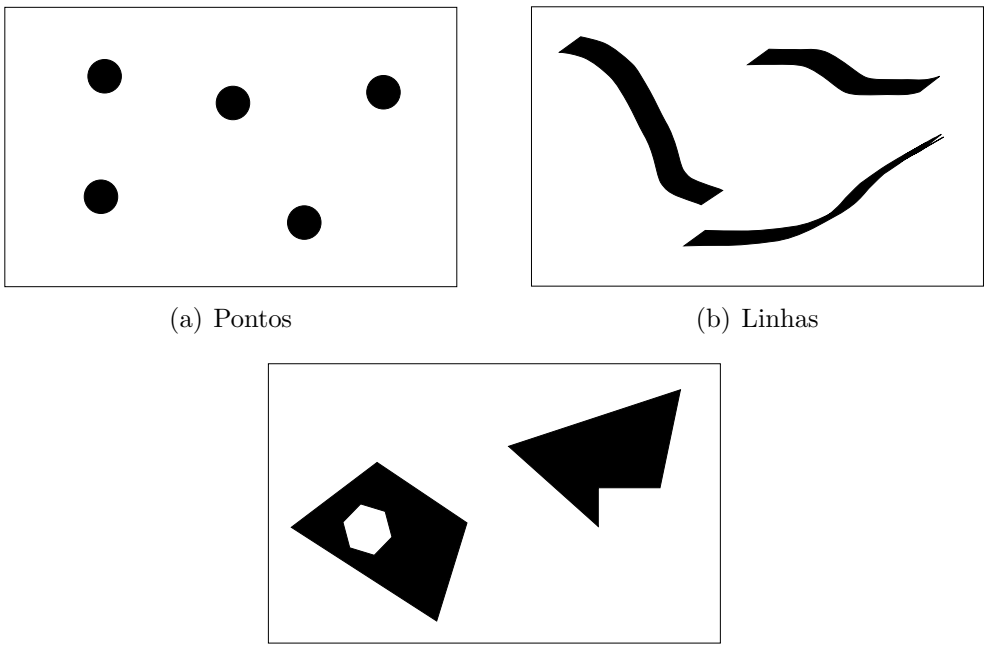
Objetos espaciais. A Figura 2.2 mostra os tipos de dados espaciais mais encontrados: o ponto (Figura 2.2(a)) representa objetos que não têm área relevante, apenas a sua localização espacial. Por exemplo, a localização de um cliente na cidade ou de uma loja (e.g., Filial do McDonald's) pode ser representada por pontos; a linha (Figura 2.2(b)) pode representar estradas ou rios. Linhas podem intersetar outras linhas; a região (Figura 2.2(c)) pode representar objetos com área relevante, como um bairro, uma fazenda ou uma cidade. Regiões são disjuntas, mas podem apresentar partes vazadas [Güting 1994, Rocha-Junior 2012, de Almeida e Rocha-Junior 2016].

Um dos principais tipos de consultas espaciais é a seleção espacial baseado em predicados (*spatial selection*) [Güting 1994, Rocha-Junior 2012]. Uma seleção é uma operação que retorna um conjunto de objetos que cumprem um predicado. Este pode ser representado utilizando uma ou mais relações espaciais (métrica, direcional etc). Um exemplo de uma seleção espacial é "Qual é a lanchonete mais próxima em relação ao ponto 'x'?".

2.2.1 Índices Espaciais

Sistemas de banco de dados espaciais necessitam de mecanismos que forneçam consultas mais rápidas de acordo com a sua localização geoespacial [Güting 1994]. Pesquisadores apresentam diversos índices espaciais para auxiliar na busca eficiente [Beckmann et al. 1990, Papadias et al. 2001].

A *R-tree* é uma estrutura de índice utilizada por parte significativa dos trabalhos com consulta espacial assim como a *B-tree*, a $R-tree$ é balanceada [Comer 1979]. A $R-tree$ contém dois tipos de nós, o nó intermediário e o nó folha. O intermediário apresenta ponteiros para o nó filho, e os índices dos nós filhos apontando para os objetos espaciais, utilizando retângulos mínimos (*Minimum Bounding Rectangle* - MBR) para ordenar os objetos espaciais [Güting 1994]. As entradas da *R-tree* são compostas por MBRs e *ids*. O MBR é o menor retângulo n-dimensional possível capaz de representar o objeto indexado e o *id* é o número que identifica a entrada



(a) Pontos

(b) Linhas

(c) Região (Polígono)

Figura 2.2: Tipos de objetos espaciais mais encontrados Fonte: Adaptado de Rocha-Junior [Rocha-Junior 2012].

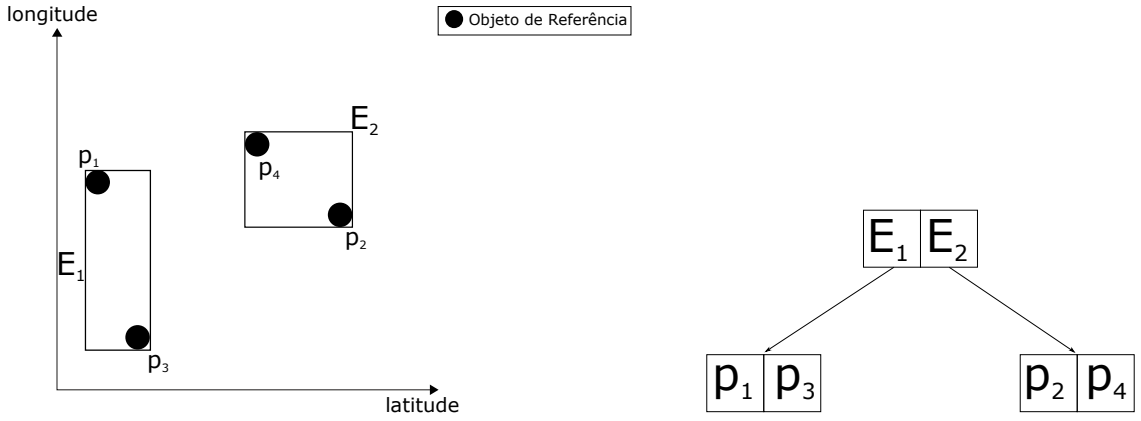
do nó, servindo também como intermediário, como um ponteiro (endereço) para outro nó na árvore [Rocha-Junior 2012, de Almeida e Rocha-Junior 2016].

A *R-tree* é uma árvore de estrutura dinâmica. Em conjunto com a consulta, o operador de remoção e inserção pode ser realizado sem precisar reorganizar a árvore periodicamente. Além disso, a sua estrutura foi planejada para que a consulta percorra um número reduzido de nós, ocorrendo em uma quantidade mínima e máxima de entradas [Güting 1994, Rocha-Junior 2012].

A Figura 2.3 demonstra uma consulta que utiliza uma $R-tree$. Na Figura 2.3(a) é apresentado como os objetos são envolvidos pelos MBRs e na Figura 2.3(b) demonstra a estrutura de como os objetos são armazenados. Essa estrutura evita o acesso às subárvores que não sejam relevantes para a consulta.

A *R-tree* é baseada na minimização da área do MBR em cada nó intermediário. Entretanto, este método não é o melhor a ser utilizado [Beckmann et al. 1990]. Uma das variações mais conhecidas da *R-tree* é a *R*-tree* [Chen et al. 2014, Wu et al. 2012]. A *R*-tree* é superior à *R-tree* em desempenho do processamento da consulta e no algoritmo que define a MBR dos nós [Beckmann et al. 1990].

A *R*-tree* reduz a área de cobertura das MBRs dos nós intermediários. Assim, menos ramos da árvore são acessados durante o processamento da consulta. Além disto, a *R*-tree* reduz a sobreposição entre MBRs, reduzindo a probabilidade de



(a) Delimitação dos MBRs (b) *R-tree*

Figura 2.3: Exemplo de uma *R-tree*

haver mais de uma MBR abrangendo a mesma área e aumentando a eficiência da consulta [Rigaux et al. 2001].

Uma variação muito utilizada da *R-tree* é a *aggreagate R-tree* (*aR-tree*) [Papadias et al. 2001]. A *aR-tree* utiliza os dados não espaciais agregados de forma antecipada para otimizar o processamento da consulta. Ou seja, cada nó contém um dado não espacial agregado (e.i. peso) [de Almeida e Rocha-Junior 2016].

2.2.2 Índices Espaço-Textuais

A consulta espaço-textual recupera o objeto onde seu escore é composto pela combinação entre a distância do objeto e a relevância textual do mesmo através da palavra-chave da consulta [Rocha-Junior et al. 2011, Cao et al. 2012]. Para uma busca mais eficiente em consultas que combinam a localização com a descrição textual, Zhou et. al. [Zhou et al. 2005] criou três estruturas agregando índices espaciais e textuais.

Uma das estruturas foi criada com índices independentes, sendo um índice textual o IF e uma *R*-tree*. A segunda obtém uma dependência entre os índices, tornando o índice textual como base, e cada termo contido no vocabulário indica para uma *R*-tree*. A última estrutura, diferente da segunda, torna a R*tree a base do índice, adicionado um IF em cada nó da R*tree e uma referência com a informação de suas sub-árvores [Zhou et al. 2005]. A segunda estrutura foi a que demonstrou melhor desempenho. Porém, estas consultas realizam buscas booleanas, onde verifica se todos os termos da palavra-chave devem estar contidos na descrição textual do objeto [Zhou et al. 2005].

Índices espaço-textuais foram propostos primeiramente por Cong et al. [Cong et al. 2009] e Li et al. [Li et al. 2011]. A estrutura proposta por eles recebeu o nome de $IR-tree$ (Figura 2.4). Cada nó dessa *IR-tree* possui um ponteiro

para um arquivo invertido IF que indexa as entradas do nó. Os arquivos invertidos mapeiam um termo para uma lista invertida. Portanto, quando a consulta é realizada permite que a *IR-tree* execute uma poda entre a distância espacial e a relevância textual [Rocha-Junior 2012].

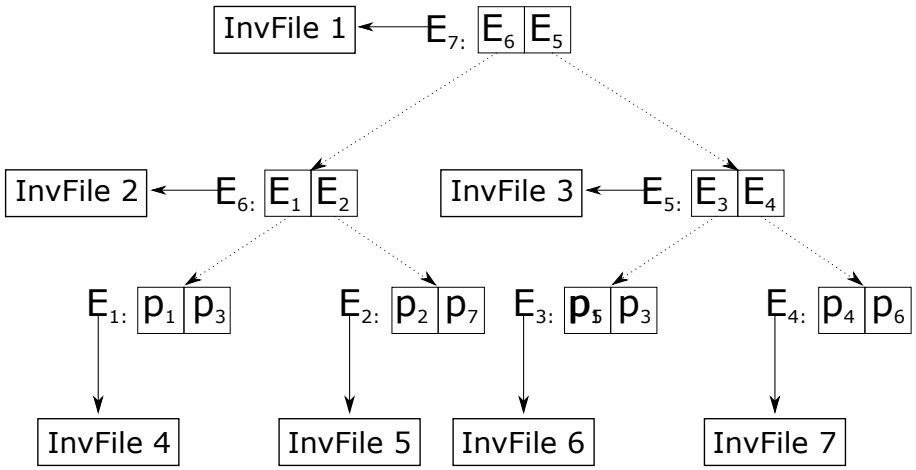
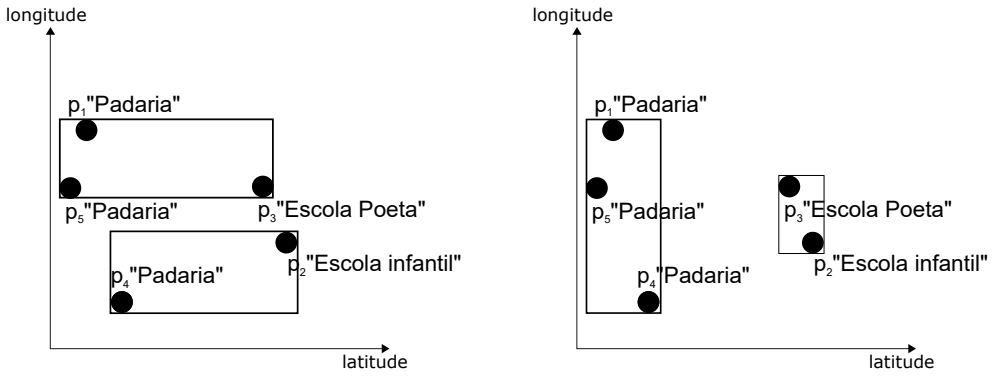


Figura 2.4: Exemplo de uma *IR-tree*.

Outro índice criado por Cong et al. [Cong et al. 2009] foi o DI*R-tree* (*Document similarity enhanced Inverted file R-tree*) demonstrada na Figura 2.5(b), que agrupa objetos de acordo com a sua proximidade espacial e sua similaridade textual. Na inserção de um novo objeto espaço-textual, a descrição textual do objeto é comparada com o arquivo invertido associado aos nós para encontrar o melhor nó para alocar o novo objeto [Rocha-Junior 2012].



(a) MBRs em uma *IR-tree* (b) MBRs em uma *DIR-tree*

Figura 2.5: Diferença entre a delimitação dos MBRs entre a *IR-tree* e a *DIR-tree*

Cong et al. propuseram também a *CDIR-tree* (*Cluster enhanced DIR-tree*). Essa derivação da *IR-tree* agrupa os objetos para melhorar ainda mais o desempenho do processamento das consultas. A ideia é agrupar entradas relacionadas (os objetos), associando um Arquivo Invertido para representar cada grupo. Essa técnica se

mostrou mais eficiente do que as outras, porém exige um processamento extra para o agrupamento dos objetos, exige mais capacidade de armazenamento e obteve uma complexidade maior para ser atualizado.

S2I

Um estudo realizado por Chen et al. [Chen et al. 2013] classificou alguns índices espaço-textuais encontrados na literatura. O estudo indicou que o índice $S2I$ de Rocha-Junior et al. [Rocha-Junior et al. 2011] denominado S2I (*Spatial Inverted Index*), como um dos métodos mais eficiente em relação a desempenho em consultas espaço-textuais.

O S2I é similar ao Arquivo Invertido. Sua estrutura se diferencia por mapear os termos e armazenar em blocos ou em árvores multidimensionais. Os termos mais frequentes são alocados em uma $aR-tree$ [Papadias et al. 2001], cada termo com uma árvore correspondente. Já os termos menos frequentes são armazenados em blocos. Cada bloco armazena os objetos que contém o mesmo termo [Rocha-Junior et al. 2011].

Inicialmente, o S2I armazena todos os termos em blocos e continua preenchendo-os, quando os objetos contidos no bloco de um termo superam a quantidade máxima de 146 entradas (4KB), esse termo passa ser considerado frequente e é armazenado em uma $aR-tree$ [Rocha-Junior et al. 2011].

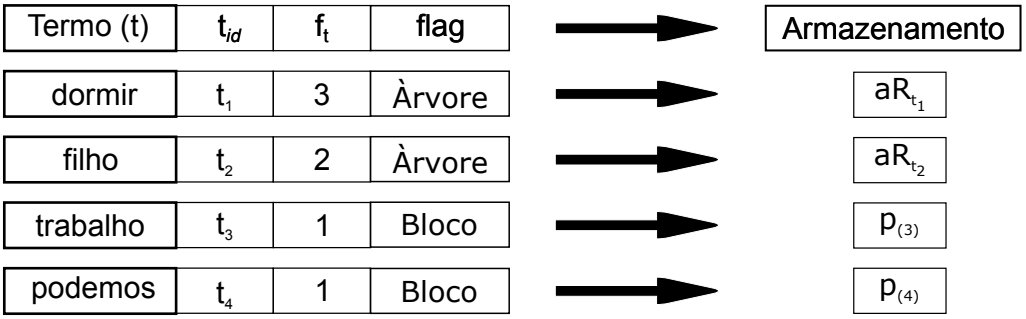


Figura 2.6: Exemplo de armazenamento no *Spartial Inverted Index*.

Capítulo 3

Trabalhos Relacionados

"O sucesso não consiste em não errar, mas em não cometer os mesmos equívocos mais de uma vez."

– George Bernard Shaw

Este capítulo apresenta os trabalhos relacionados com essa pesquisa. A Seção 3.1 apresenta as consultas do vizinho inverso mais próximo (RNN). A Seção 3.2 apresenta os conceitos de localização ideal (localização influente). A Seção 3.3 apresenta a consulta influente através da menor distância. A Seção 3.4 apresenta a consulta influente que minimiza a máxima distância. Por fim, a Seção 3.5 apresenta a consulta dos k melhores locais influentes.

3.1 Bicromática Inversa do Vizinho mais Próximo

A pesquisa do vizinho inverso mais próximo (RNN - *Reverse Nearest Neighbor*) encontra os pontos que têm o ponto de consulta como seu vizinho mais próximo. A pesquisa RNN foi apresentada por Korn e Muthukrishanan [Korn e Muthukrishnan 2000] e tem sido amplamente estudada na comunidade de banco de dados [Stanoi et al. 2001, Xia et al. 2005, Wong et al. 2009].

Existem dois tipos de pesquisa RNN: a monocromática (MRNN - *Monochro- matic Reverse Nearest Neighbor*) e a bicromática (BRNN - *Bichromatic Reverse Nearest Neighbor*). A Monocromática [Lu et al. 2011, Lu et al. 2014] apresenta pontos do mesmo tipo. Um ponto p é considerado como um vizinho mais próximo reverso para um ponto de consulta p se não existir outro objeto de dados t que seja considerado como um vizinho mais próximo para um ponto de consulta t. O BRNN é um conjunto

de objetos em um espaço que estão próximos a um ponto de interesse devido a sua proximidade espacial. Seja P e T dois conjuntos de pontos em um mesmo espaço. Dado um ponto $t \in T$ de locais existentes e um conjunto $p \in P$ de objetos de referência, a consulta BRNN vai encontrar todos os objetos de referência que estão em sua proximidade espacial, desde que não exista outro local existente em T que tenha uma proximidade espacial maior. Assim, o conjunto BRNN de um ponto t será seus objetos de referência mais próximos.

Exemplo. A Figura 3.1 apresenta a distribuição de dois conjuntos em um espaço. O conjunto T referente a 2 (duas) lojas, e um conjunto P com 5 (cinco) objetos de referência. Em um caso hipotético, é necessário saber quais objetos de referência estão interessados em visitar uma das lojas. O resultado do BRNN t_1 *é o conjunto de objetos de referência* $\{p_1,p_2\}$ *e o de* t_2 *é o conjunto* $\{p_3,p_4,p_5\}$.

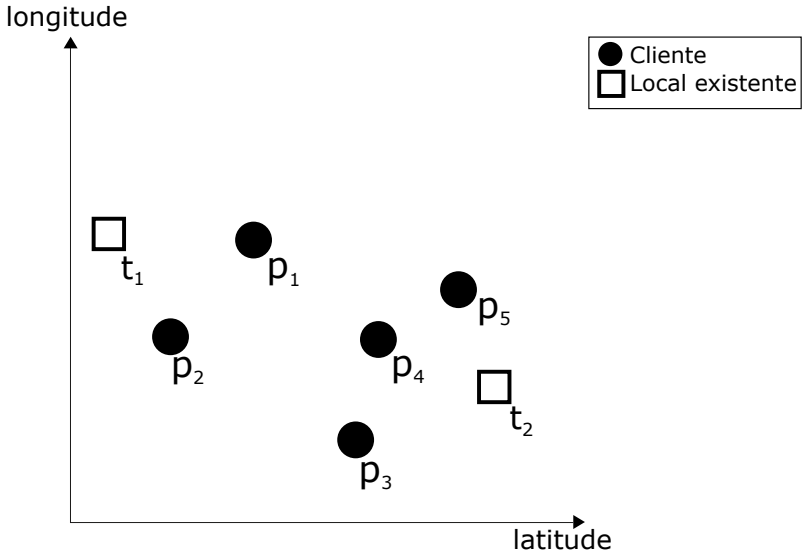


Figura 3.1: Exemplo de aplicação do BRNN.

O algoritmo MaxBRNN é uma variante da abordagem BRNN que encontra o local com maior influência (interesse). Uma grande quantidade de aplicativos que existem na pesquisa BRNN também podem ser aplicadas na busca do MaxBRNN [Liu et al. 2013]. Por exemplo, o planejamento de instalação de um novo estabelecimento é um exemplo comumente apresentado nos trabalhos relacionados ao MaxBRNN. A Figura 3.1 pode ser utilizada na execução de um planejamento de localização para a instalação de uma loja que precisa atrair o maior número possível de objetos de referência. Worm et. al. [Wong et al. 2011], demonstraram também que este problema pode ser utilizado em outras aplicações de emergência, como desastres naturais, grandes eventos repentinos ou até mesmo aplicações militares.

Pesquisas referente a bicromática (BRNN) tem sido estudada de forma extensiva para auxiliar nas operações com base em objetos espaciais [Korn e Muthukrishnan 2000, Kang et al. 2007, Wong et al. 2009]. Além disso,

Choudhury et al. [Choudhury et al. 2016] introduziram o conceito de similaridade textual entre o objeto de referência e a palavra-chave para determinar o escore do objeto a ser instalado.

O estudo de Choudhury et al. [Choudhury et al. 2016] calculam a influencia do local candidato com uma combinação entre a similaridade textual do objeto de referência com a palavra-chave da consulta e a distância espacial dos objetos de referência. O autor compara três métodos diferentes para o cálculo da similaridade textual, sendo que um deste é método de Zobel e Moffat [Zobel e Moffat 2006] utilizado neste trabalho. A grande diferença entre a abordagem proposta neste trabalho e o que foi realizado por Choudhury et al. é a utilização da distância espacial como variável na fórmula do escore, neste trabalho o escore dos objetos de referência são computados através da similaridade textual, uma vez que a distância é utilizada para identificar os objetos que são atraídos pelo possível local candidato e não como variável para calcular o escore do mesmo.

A aplicação mais comum do BRNN é quando existe o interesse de selecionar o "melhor provedor de serviço" [Wong et al. 2009]. Por exemplo, considere que foi solicitado identificar quais são os objetos de referência mais interessados em visitar uma loja de acordo com a sua proximidade espacial.

A Consulta de localização influente é considerada como uma consulta BRNN [Du et al. 2005]. Ela computa o conjunto de objetos em P que estão próximos a um objeto em C do que para qualquer outro ponto em T.

Existem muitos trabalhos sobre consulta por locais influentes, ela pode ser "definida como uma consulta de uma nova instalação que maximize o peso total de objetos que estão mais próximos desta instalação do que para qualquer outro local" [Du et al. 2005]. Nas próximas seções são apresentados alguns tipos de consultas de locais influentes encontrados na literatura.

3.2 Localização Competitiva

A consulta por localização competitiva identifica o local com maior influência, que por conseguinte, se torna a localização mais relevante para a consulta [Xiao et al. 2011]. Por exemplo, a McDonald's está à procura de um local para a construção de uma nova instalação. Dado um conjunto de locais candidatos, a consulta de locais influentes retorna o local para a construção de uma nova loja da McDonald's, que possa atrair o maior número de objetos de referência.

Existem algumas formas de obter o conjunto de locais candidatos. Du et al. [Du et al. 2005] propuseram uma região espacial para obter o melhor local candidato. Xial et al. [Xiao et al. 2011] propuseram um conjunto de locais candidatos finitos para a consulta de localização ideal.

A influência do melhor local candidato c' é computado pela soma dos pesos dos objetos de referência por ele atraído. A Equação 3.1 retorna o local influente com

maior escore, é atribuído ao escore ao local candidato o somatório dos pesos dos objetos de referência para local candidato, tal que a distância do local candidato para o objeto de referência seja menor que a distância do mesmo para qualquer local existente.

$$c.score = \sum \{w(p) \mid p \in P \wedge \forall t \in T : d(c,p) \leq d(t,p)\} \tag{3.1}$$

A consulta de localização ideal foi utilizada nesta pesquisa. A diferença é que este trabalho optou por utilizar o texto associado aos objetos para pontuar cada objeto de referência e posteriormente encontrar os k melhores locais influentes.

3.3 Localização ideal por menor distância

A consulta denominada de *Min-Dist* recebe como parâmetro um conjunto de locais existentes, um conjunto de objetos de referência e uma região espacial "Q". A consulta retorna uma localização c em "Q", no qual ao adicionar em Q uma nova instalação, a distância média total somada dos objetos de referência para o local mais próximo é minimizada [Zhang et al. 2006, Qi et al. 2014, Chung et al. 2018]. Outro método também estudado é quando se define um conjunto de locais candidatos finito, ao invés de limitar uma região espacial Q [Xiao et al. 2011]. A consulta de *MinSum* busca o local ideal que reduza a distância média total do peso (W_{AD}) entre os objetos de referência e os locais de interesse [Chen et al. 2014].

Exemplo. Uma prefeitura pode buscar um local para a construção de um novo hospital para atender a população. Este hospital tem que ser adicionado em uma região estratégica, que possa diminuir a distância média que cada cidadão tenha que percorrer até o hospital mais próximo. A consulta retorna uma localização para uma nova instalação no qual a distância média total somada dos objetos de referência para o local mais próximo é minimizada [Zhang et al. 2006, Qi et al. 2014]. Um método estudado é quando se define um conjunto de locais candidatos finito, ao invés de limitar uma região espacial Q [Xiao et al. 2011], mas existem pesquisas que estudam a delimitação de regiões para buscar o local ideal [Zhang et al. 2006, Qi et al. 2014].

O W_{AD} dos objetos de referência é encontrado através do produto entre o peso do objeto de referência ($w(p)$), e a distância ($a(p)$) do local existente mais próximo. A Equação 3.2 retorna o local ideal que consiga minimizar a soma total dos pesos ponderados com a distância, multiplicando o peso do objeto de referência com a distância do local existente mais próximo.

$$c'.score = \arg\min_{c \in C} \sum w(p) \cdot \min\{d(t,p) \mid t \in T\} \tag{3.2}$$

A consulta *Min-Dist* se preocupa em minimizar a distância média entre os objetos de referência e os locais existentes, ou seja, diminui a distância que os "objetos de

referência" precisam percorrer até o local existente mais próximo. O local candidato escolhido é aquele que tem a menor soma das distâncias entre os objetos de referência e os locais.

3.4 Minimização da Máxima Distância

A consulta de *MinMax* realiza uma consulta que minimize a distância máxima total do peso (W_{AD}) entre os objetos objetos de referência e o local de interesse mais próximo. Por exemplo, o departamento de segurança da cidade de Feira de Santana quer adicionar um novo posto policial. A consulta deve encontrar o local influente para a instalação de um posto policial, que minimize a distância máxima entre esta nova unidade de segurança, e as regiões de atendimento da equipe de segurança. A Equação 3.3 apresenta o modelo do problema, no qual é encontrado o local mais influente que diminua o peso ponderado entre o peso do objeto de referência e a distância do mesmo para o local existente mais próximo.

$$c'.score = \arg\min_{c \in C}(\max\{w(p) \cdot \min\{d(t,p) \mid t \in T\}\}) \tag{3.3}$$

Esta consulta é diferente da estudada nesta pesquisa. A consulta *MinMax* tem como objetivo minimizar a distância máxima entre o objeto de referência e o local existente mais próximo, independente do total de objetos de referência atraídos. No *MinMax* é levado em conta apenas o mais distante e não a soma total do escore como acontece na consulta de localização competitiva.

3.5 *K* melhores locais ideais

Segundo Huang et al. [Huang et al. 2011], a proposta de Du et al. [Du et al. 2005] não responde problemas em que uma quantidade maior de locais candidatos é requerida (k). Com isso, Du et al. foi estudado a indicação da quantidade de locais candidatos que devem ser retornados pela consulta [Xia et al. 2005].

Nesta pesquisa, é definida a quantidade de locais candidatos que a consulta deve retornar, para que em uma tomada de decisão, o usuário possa escolher entre os locais candidatos definidos, qual dos locais candidatos retornados é mais apropriado ou estratégico, por exemplo.

Um levantamento do estado da arte foi realizado, e atualmente não existe uma consulta de locais influentes onde o peso dos objetos de referência atraídos seja definido pela similaridade textual entre a palavra-chave da consulta e o texto dos objetos de referência. Por isso, as técnicas mencionadas acima não podem ser aplicadas diretamente para processar a consulta proposta neste trabalho.

Capítulo 4

Metodologia

"Talvez não tenha conseguido fazer o melhor, mas lutei para que o melhor fosse feito. Não sou o que deveria ser, mas Graças a Deus, não sou o que era antes."

– Marthin Luther King

O trabalho está dividido nas seguintes etapas: i) revisão bibliográfica; ii) proposta de algoritmos e iii) avaliação dos resultados. O processo metodológico praticado é conhecido como método experimental, que submete um estudo a alguma variação ou adição de fator para avaliar como o resultado pode ser alterado [Prodanov e de Freitas 2013].

4.1 Fazer revisão bibliográfica

A Revisão bibliográfica consistiu em investigar a consulta por locais influentes, para aumentar o conhecimento da área pesquisada. Após a definição do problema da pesquisa, outros tópicos foram estudados para investigar os métodos utilizados para obtenção dos resultados. Estes tópicos foram relacionados a ferramentas necessárias para o desenvolvimento da proposta.

Foram revisados trabalhos relacionados com indexação textuais presentes em conferências e jornais importantes para essas temáticas, tais como: ACM, IEEE, VLDB etc. A relevância dos trabalhos presentes na revisão foram analisados de acordo com o número de citações, a originalidade das técnicas e relação direta com o tema proposto nesta pesquisa. A consulta influente tem trabalhos publicados a partir de 2005 [Du et al. 2005, Xia et al. 2005], portanto, o intervalo dos trabalhos se deu basicamente entre 2005 e 2018. As abordagens presentes nos trabalhos são necessárias

para a criação de algoritmos eficientes para a consulta de locais influentes que é proposto neste trabalho.

4.2 Identificar e preparar os dados

Duas bases de dados foram utilizadas: o *OpenStreetMap* para referenciar a base de dados dos **locais existentes** e do *Twitter* para representar a base de dados dos **objetos de referência**.

O *OpenStreetMap*[1] (OSM), é um projeto de mapeamento colaborativo com dados geoespaciais, onde uma comunidade voluntária é responsável por manter com rigor os dados. Qualquer pessoa pode editar os dados, e os usuários mais antigos confirmam a veracidade das informações editadas a partir de receptores GPS portáteis, fotografias aéreas, imagens de satélite e outras fontes livres. A utilização dos dados é livre e tem aplicação em vários temas, tais como: navegação GPS; estudo de geografia e mapeamento para escolas e faculdades; projetos em ferramentas CAD; estudos sobre objetos geoespaciais. Estes dados do OSM contém as informações geográficas necessárias para representar os locais existentes. Os dados são facilmente coletados através da plataforma do OSM, onde é possível baixar o arquivo em formato *.xml* de uma determinada região ou cidade; foi necessário preparar esse arquivo para ser lido pelo algoritmo de forma correta, eliminando informações que não são utilizadas na consulta.

A outra base utilizada é a do *Twitter*[2], que associa a mensagem enviada pelo usuário através de um *smartphone* a sua localização espacial. Estes dados representam locais reais que as pessoas costumam frequentar. O *Twitter* é uma rede social que disponibiliza parte dos seus dados de forma gratuita. Os dados do *Twitter* foram extraídos através da API (*Application Programming Interface*), sendo que os dados dos usuários são anônimos. Os arquivos passam por um processamento para manter apenas as informações necessárias para a consulta, tais como: localização e texto associado.

4.3 Definir problema

A consulta por locais influentes através de palavra-chave recebe como parâmetro um conjunto de palavras-chave de busca $Q = (Q.D)$, um conjunto de locais candidatos $c \in C$, um conjunto de locais existentes $t \in T$, um conjunto de objetos de referência $p \in P$, e pela quantidade k de objetos de referência que se deseja obter como resposta, a consulta retorna os k melhores locais $c \in C$ mais influentes. Onde a influência é computada a partir da função que calcula o escore.

[1] www.openstreetmap.com
[2] www.twitter.com

Os locais candidatos $c \in C$ possuem apenas a localização espacial e são definidos como $c = (c.x, c.y)$, onde x e y representam latitude e longitude, respectivamente. Assim, como os locais candidatos, os locais existentes $t \in T$ possuem apenas a localização espacial $t = (t.x, t.y)$. Os objetos de referência $p \in P$ possuem localização espacial e um texto descritivo $p = (p.x, p.y, p.D)$, onde $p.x$ e $p.y$ representam a coordenada geográfica e $p.D$ a descrição textual do objeto.

A localização espacial dos objetos de referência $p \in P$, dos locais candidatos $c \in C$ e existentes $t \in T$ é requerida para buscar o local atrator de um objeto de referência p. Essa atração é definida pela menor distância, utilizando a distância Euclidiana, entre o objeto e o local mais próximo.

O processamento da consulta por locais influentes à partir de palavras-chave, utiliza a similaridade textual θ entre as palavras-chave da consulta $Q.D$ e a descrição dos objetos de referência $p.D$ para computar um peso para estes objetos ($w(p)$). A similaridade textual é obtida utilizando as equações da Seção 2.1. Cada local candidato c tem sua pontuação computada de acordo com a proximidade espacial para um objeto de referência p, onde p está mais próximo do local candidato c do que para qualquer outro local existente em T e a similaridade textual (θ) for mair que zero.

A representação da consulta de localização competitiva é definida como

$$c'.score = \sum \{w(p) \mid p \in P \wedge \forall t \in T : d(c,p) \leq d(t,p), c \neq t\} \tag{4.1}$$

onde $w(p) = \theta(p.D, Q.D)$.

Essa consulta nunca foi proposta anteriormente, um vez que o grande desafio, comparada a outras consultas similares, são os pesos dos objetos de referência. Os pesos são valores dinâmicos determinados no momento da consulta através da similaridade textual entre as palavras-chave da consulta e o texto do objetos de referência.

4.4 Desenvolver algoritmos

Existem, no estado da arte, pesquisas sobre consulta de locais influentes. Essas pesquisas são utilizadas neste trabalho para a construção do algoritmo *Baseline*, que computa o escore dos locais candidatos. Estes métodos realizam uma varredura com todos os dados presentes nas bases e identifica o escore dos locais candidatos de acordo com a similaridade textual dos objetos de referência atraídos. Para avaliar se o algoritmo estava computando o escore exato dos locais candidatos, testes de mesa foram executados inicialmente na elaboração dos algoritmos, para provar que o resultado obtido pelos algoritmos seriam fiéis a resposta correta.

Algoritmos eficientes utilizando índices textuais foram estudados para indexar os dados, a fim de melhorar o desempenho da consulta. Essa técnica diminuiu a varredura dos dados consultados em uma busca, pois faz um filtro dos objetos a partir das palavras-chave, tornando o processamento eficiente.

Os algoritmos foram codificados na linguagem de programação JAVA, pois além de fácil implementação, é utilizada por uma quantidade significativa dos trabalhos estudados na revisão bibliográfica.

4.5 Avaliar resultados

A avaliação dos algoritmos foi realizada medindo o tempo de resposta e o número de páginas lidas. O tempo de resposta é a duração que o algoritmo executa desde a solicitação até a resposta dos k locais mais influentes, enquanto o número de páginas lidas mede a entrada e saída de dados realizados pela consulta. Estes valores são obtidos ao variar o número de palavras-chave da consulta, a cardinalidade (tamanho) das bases de dados dos locais existentes, a cardinalidade dos objetos de referência e o número de locais candidatos.

Estas variáveis foram utilizadas, pois os trabalhos encontrados sobre consulta por local influente utilizam desses parâmetros para avaliar suas respectivas pesquisas [Chen et al. 2015].

Capítulo 5

Algoritmos

"Se um dia tiver que escolher entre o mundo e o amor lembre-se: se escolher o mundo ficará sem o amor, mas se escolher o amor com ele você conquistará o mundo. "

– Albert Einstein

Neste capítulo são apresentados os algoritmos elaborados nessa pesquisa. Na Seção 5.1, é apresentado o algoritmo *baseline*, para demonstrar a técnica presente no estado da arte. Na Seção 5.2 é apresentado o algoritmo que utiliza índice textual. Na Seção 5.3 o algoritmo com índice textual aprimorado e na Seção 5.4 os algoritmos com índice espacial e espaço-textual.

5.1 Algoritmo Baseline

O Algoritmo *Baseline* foi criado para servir de parâmetro de comparação com os algoritmos mais avançados propostos nas Seções Seção 5.2 e Seção 5.3. O *Baseline* não utiliza nenhum tipo de índice para processar a consulta.

O Algoritmo 1 processa a consulta de locais influentes, retornando os k melhores locais candidatos com seus respectivos escores. Para calcular a influência destes locais, é verificado qual o local existente mais próximo para cada objeto de referência. Em seguida, é identificado quais são o objetos de referência atraídos pelos locais candidatos, comparando a distância do candidato com o local existente mais próximo ($p.distSmall$). Após esta etapa, a similaridade textual é computada e incrementada ao escore do local candidato. Finalizando no armazenamento dos k melhores locais candidatos mantidos em uma $Max\text{-}Heap$.

Algorithm 1: Baseline

Input : $Q.D$(Conjunto de Palavras-Chave), C(Locais Candidatos), T(Locais existentes), P(Objetos de Referência) , k (Quantidade dos melhores locais candidatos retornados)
Output: M //maxHeap dos k candidatos

```
1  M ← ∅
2  foreach c ∈ C do
3  |  foreach p ∈ P do
4  |  |  distSmall ← ∞
5  |  |  foreach t ∈ T do
6  |  |  |  if d(t, p) < distSmall then
7  |  |  |  |  distSmall ← d(t, p)
8  |  |  if d(c, p) ≤ distSmall then
9  |  |  |  c.score ← c.score + θ(p.D, Q.D)
10 |  M.add(c)
11 |  if |M| > k then
12 |  |  M.removeLast()
13 return M
```

O *Baseline* é apresentado no Algoritmo 1. O algoritmo percorre todos os locais candidatos (linha 2), em seguida realiza a varredura da base de dados dos objetos de referência (linha 3) e os locais existentes (linha 5). Uma condição é utilizada para verificar o local existente mais próximo dentre todos presentes na base e armazenado (linhas 6-7). Uma comparação é utilizada para verificar se a distância do local candidato é menor que o local existente mais próximo do objeto de referência (linha 8), e incrementa ao escore do local candidato o peso do objeto de referência de acordo com a similaridade textual com as palavras-chave (linha 9). Os locais candidatos são armazenados em uma $Max\text{-}Heap$ (linha 10). Em seguida verifica se o tamanho da $Max\text{-}Heap$ é maior que a quantidade k requisitada, eliminando o menor, caso a condição seja verdadeira. Por fim, retornando a $Max\text{-}Heap$ (linha 13).

Este algoritmo foi criado utilizando técnicas presente no estado da arte, o passo seguinte foi criar novos algoritmos utilizando índices textuais e índices espaciais para tornar o algoritmo eficiente.

5.2 Algoritmo com Índice Textual

O Algoritmo *Baseline* (Seção 5.1) requer acessar todos os objetos de referência presentes na base de dados e verificar qual o local existente mais próximo. Em seguida, para cada local candidato é comparado também a distância para todos os objetos de referência presentes na base. Se ele atrair o objeto de referência, o cômputo da relevância textual de cada um destes objetos de referência para as

palavras-chave da consulta é atribuído ao escore do local candidato.

O índice textual dos objetos de referência foi processado antes da execução das consulta dos k melhores locais influentes. Pois, uma vez realizado a indexação dos termos, não é mais necessário a execução do índice. Os arquivos invertidos e vocabulários se mantém os mesmos para qualquer execução e são atualizados se, somente se, a base de dados modificar.

O Algoritmo 2 apresentado nesta seção realiza um filtro utilizando arquivos invertidos para diminuir a quantidade de objetos de referência acessados. Esse mecanismo foi utilizado para diminuir o número de objetos processados, uma vez que as bases de dados são formadas por milhares de objetos e fica custoso processar tudo. O arquivo invertido serve para retornar os objetos que tem relevância textual com as palavras-chave da consulta.

O algoritmo IT (Índice Textual) é apresentado no Algoritmo 2. O algoritmo realiza uma varredura nas palavras-chave para buscar apenas os objetos de referência com similaridade textual (linha 3). O algoritmo recupera na variável L, todos os objetos da lista invertida do termo q (linha 4). Então o algoritmo armazena em "P" todos os objetos de referência recuperados em "L" (linhas 6-7), que são os objetos com similaridade textual. O algoritmo percorre todos os locais candidatos (linha 8), em seguida realiza a varredura da base de dados dos objetos de referência (linha 9) recuperados pelo filtro executado antes (linhas 3-7) e percorre todos os locais existentes (linha 11). Uma condição é utilizada para verificar o local existente mais próximo dentre todos presentes na base e armazenado (linhas 12-13). Uma comparação é utilizada para verificar se a distância do local candidato é menor que o local existente mais próximo do objeto de referência identificado (linhas 14), e incrementa ao escore do local candidato o peso do objeto de referência de acordo com a similaridade textual com as palavras-chave (linha 15). Os locais candidatos são armazenados em uma $Max\text{-}Heap$ (linha 16) que ordena pelo escore. Em seguida verifica se o tamanho da $Max\text{-}Heap$ é maior que a quantidade k requisitada, eliminando o menor, caso a

condição seja verdadeira. Por fim, retornando a $Max\text{-}Heap$ (linha 19).

Algorithm 2: IT (Índice Textual)

Input : $Q.D$(Conjunto de Palavras-Chave), C(Locais Candidatos), T(Locais existentes), k (Quantidade dos melhores locais candidatos retornados)
Output: M //maxHeap dos k candidatos

```
1  P ← ∅
2  M ← ∅
3  foreach q ∈ Q.D do
4  |  L ← IF.list(q)
5  |  if L ≠ isEmpty() then
6  |  |  while l ∈ L do
7  |  |  |  P.put(l)
8  foreach c ∈ C do
9  |  foreach p ∈ P do
10 |  |  distSmall ← ∞
11 |  |  foreach t ∈ T do
12 |  |  |  if d(t, p) < distSmall then
13 |  |  |  |  distSmall ← d(t, p)
14 |  |  if d(c, p) ≤ distSmall then
15 |  |  |  c.score ← c.score + θ(p.D, Q.D)
16 |  M.add(c)
17 |  if |M| > k then
18 |  |  M.removeLast()
19 return M
```

Este algoritmo recupera inicialmente apenas os objetos de referência que contém as palavras-chave da consulta. Em seguida, realiza o processo de busca pelos k melhores locais candidatos similar ao $Baseline$, a diferença fica por conta da quantidade de objetos de referência menor no Algoritmo 2 em relação so Algoritmo 1.

5.3 Algoritmo com Índice Textual Aprimorado

O Algoritmo 1 e o Algoritmo 2 buscam para cada objeto de referência o local existente mais próximo, e em seguida verifica se o local candidato é atrator em relação ao local existente mais próximo dos objetos de referência. Este procedimento pode ser melhorado, pois o processo de busca pelo local existente mais próximo é realizado para todos os locais candidatos, mas as distâncias entre os locais existentes e objetos de referência são fixas, e não é necessário realizar essa verificação de distância para todos os locais candidatos. No Algoritmo 3 foi realizado essa modificação, para encontrar o local existente mais próximo dos objetos de referência apenas uma vez, e armazenar em um dos atributos do objeto de referência. Porém, este processo usa

mais memória, justamente por utilizar mais um atributo.

Algorithm 3: ITA (Índice Textual Aprimorado)

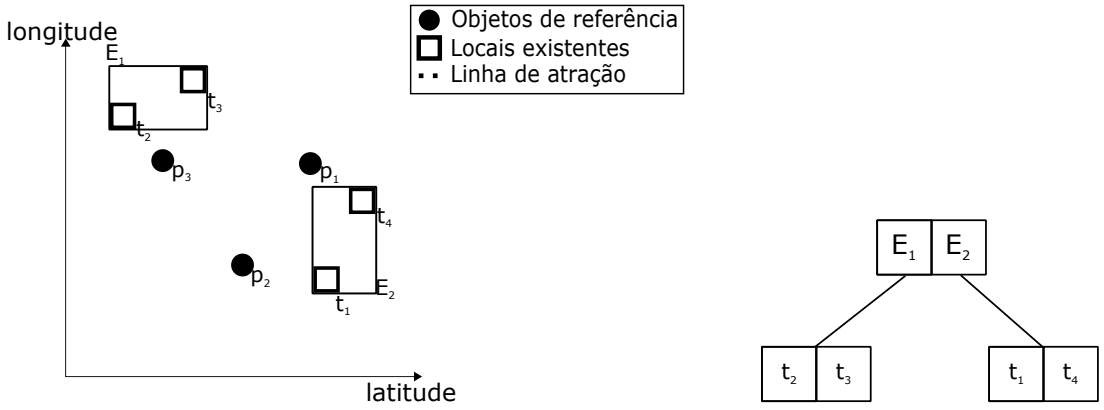
Input : $Q.D$(Conjunto de Palavras-Chave), C(Locais Candidatos), T(Locais existentes), k (Quantidade dos melhores locais candidatos retornados)
Output: M //maxHeap dos k candidatos

```
1  P ← ∅
2  M ← ∅
3  foreach q ∈ Q.D do
4  |  L ← IF.list(q)
5  |  if L ≠ isEmpty() then
6  |  |  while l ∈ L do
7  |  |  |  P.put(l)
8  foreach p ∈ P do
9  |  p.distSmall ← ∞
10 |  while t ∈ T do
11 |  |  if d(t, p) < p, distSmall then
12 |  |  |  p.distSmall ← d(t, p)
13 foreach c ∈ C do
14 |  while p ∈ P do
15 |  |  if d(c, p) ≤ p.distSmall then
16 |  |  |  c.score ← c.score + θ(p.D, Q.D)
17 |  linhas 16-18 do Algoritmo 2
18 return M
```

O algoritmo ITA (Índice Textual Aprimorrado) é apresentado no Algoritmo 3. Assim como acontece no Algoritmo 2, o algoritmo realiza o filtro dos objetos de referência utilizando o índice textual (linha 3-7). O algoritmo realiza a varredura dos objetos de referência (linha 8), a variável de distância do objeto p é inicializada como infinito (linha 9) e em seguida percorre todos os locais existentes (linha 10). Uma condição é utilizada para verificar o local existente mais próximo dentre todos presentes na base de dados, e armazenado no objeto de referência (linha 11-12). Um laço de repetição faz a varredura dos locais candidatos (linha 13). O algoritmo percorre os objetos de referência dentro do laço dos locais candidatos (linha 14). Uma condição é utilizada para identificar se o objeto de referência é atraído pelo local candidato (linha 15) e incrementa ao escore do local candidato o valor da similaridade textual do objeto de referência com as palavras-chave (linha 16). O algoritmo armazena em uma $Max\text{-}Heap$ os locais candidatos, assim como acontece no Algoritmo 2 (linha 17). Por fim, retornando a $Max\text{-}Heap$ (linha 18).

5.4 Algoritmo com Índice Textual e Espacial

O índice espacial pode ser utilizado para diminuir o número de acessos à base de dados dos locais existentes. Para isso seria necessário indexar os locais existentes em um índice espacial, por exemplo, *R-Tree* (Subseção 2.2.1). O algoritmo, a partir da localização do objeto de referência, percorre uma *R-Tree* e encontra o local existente mais próximo.



(a) Exemplo da aplicação do índice espacial (b) *R-Tree*

Figura 5.1: Representação da *R-Tree*

Exemplo. A Figura 5.1(a) mostra dois MBRs (E_1 e E_2), quatro locais existentes (t_1, t_2, t_3 e t_4) e três objetos de referência (p_1, p_2 e p_3). Com a utilização da *R-Tree*, é possível encontrar para p_1 o local existente mais próximo (t_4) de forma eficiente.

O algoritmo ITE (Índice Textual Espacial) é apresentado no Algoritmo 4. Inicialmente, foi criado o índice textual para os objetos de referência e um índice espacial para os locais existentes. Os Algoritmos 1, 2 e 3 percorrem para cada objeto de referência, todo o conjunto dos locais existentes para encontrar o mais próximo. O Algoritmo 4 utiliza a indexação dos locais existentes para encontrar o mais próximo, tornando a consulta eficiente.

Assim como acontece no Algoritmo 2 e 3, o ITE realiza o filtro dos objetos de referência utilizando o índice textual (linha 3-7). O algoritmo indexa os locais existentes em uma *R-Tree* (linha 8). Um laço de repetição faz a varredura dos locais candidatos (linha 9). O algoritmo percorre os objetos de referência dentro do laço dos locais candidatos (linha 10). Um condição é utilizada para identificar se o objeto de referência é atraído pelo local candidato (linha 11) e incrementa ao escore do local candidato o valor da similaridade textual do objeto de referência com as palavras-chave (linha 12). O algoritmo armazena em uma *Max-Heap* os locais candidatos, assim como acontece no Algoritmo 2 (linha 13). Por fim, retornando a *Max-Heap* (linha 8).

Os locais candidatos têm a mesma estrutura dos locais existentes, e podem ser inde-

Algorithm 4: ITE (Índice Textual e Espacial)

Input : $Q.D$(Conjunto de Palavras-Chave), C(Locais Candidatos), T(Locais existentes), k (Quantidade dos melhores locais candidatos retornados)
Output: M //maxHeap dos k candidatos

```
1  P ← ∅
2  M ← ∅
3  foreach q ∈ Q.D do
4  |   L ← IF.list(q)
5  |   if L ≠ isEmpty() then
6  |   |   while l ∈ L do
7  |   |   |   P.put(l)
8  R ← R-tree(T)
9  foreach c ∈ C do
10 |   while p ∈ P do
11 |   |   if d(c, p) ≤ R.minDist(p) then
12 |   |   |   c.score ← c.score + θ(p.D, Q.D)
13 |   linhas 16-18 do Algoritmo 2
14 return M
```

xados em um índice espacial, porém o conjunto de locais candidatos normalmente é pequeno (entre 3 à 5) e aleatório, portanto, seria criado para cada conjunto de locais candidatos um índice espacial para este conjunto pequeno, e não seria relevante para tornar mais eficiente o algoritmo.

Após pesquisar diversas alternativas para utilizar índices híbridos, não se identificou uma alternativa que deixasse claro a efetividade do uso de índices híbridos para processar a consulta dos k locais mais influentes a partir de palavras-chave. Uma das razões para não ter encontrado uma alternativa é que a consulta requer duas funcionalidades que não podem ser atendidas por um índice híbrido que são: atração e escore textual. A atração só requer a distância espacial e não faz sentido utilizar um índice híbrido para obtê-la, enquanto que o escore textual só requer um índice textual e não faz sentido utilizar um índice híbrido para obtê-lo.

Capítulo 6

Resultados

"Eu acredito demais na sorte. E tenho constatado que, quanto mais duro eu trabalho, mais sorte eu tenho. "

– Coleman Cox

Neste Capítulo é apresentado os resultados dos algoritmos propostos. Está organizado da seguinte forma: a Seção 6.1 apresenta as bases de dado dos objetos utilizados na pesquisa. A Seção 6.2 apresenta a configuração utilizada para o desenvolvimento dos modelos propostos, bem como os valores dos parâmetros utilizados em cada experimento. A Seção 6.3 apresenta os experimentos dos algoritmos com os parâmetros variados e os resultados obtidos de acordo com o tempo de resposta e o número de páginas lidas.

6.1 Base de dados

Os elementos que representam a **base de dados dos locais existentes** foram extraídos do OSM no mês de abril de 2018. Um total de 7.783.286 objetos estão presentes na base extraída. O arquivo que originalmente está no formato *.xml* foi transformado para *.txt* com as informações necessárias para a pesquisa, tais como: localização espacial e texto associado. Informações como hora e data foram descartadas por não ter relação com a pesquisa.

O *Twitter* foi utilizado para representar a **base de dados dos objetos de referência** coletados via API (*Application Programming Interface*) da plataforma, que possibilita baixar dados de um certo tempo ou ainda recuperar os dados em tempo real e são representados pelos objetos de referência. O número de objetos extraídos do *Twitter* foi de 1.570.850, todos com a localização e uma descrição textual (*tweet*)

associada. Para representar a **base de dados dos locais candidatos**, foi extraída uma amostra dos locais existentes, para representar locais reais.

6.2 Configuração

Esta seção apresenta os cenários que os algoritmos foram submetidos. Os experimentos foram executados em um computador com processador Intel core i5 2.2GHz, 8 GB de memória RAM, 1 TB de HD com o sistema operacional Windows 7.

Cada experimento avalia o impacto de uma única variável, enquanto as outras são mantidas fixas. Em cada experimento, o tempo de resposta e o número de páginas lidas são coletados. O resultado obtido leva em consideração a obtenção dos objetos relevantes para a consulta dos algoritmos 2, 3 e 4. Para o algoritmo 4 é considerado também a criação da *R-tree* na consulta em cada conjunto de locais existentes. Todos os algoritmos foram implementados para ler páginas de $4KB$ devido ao tamanho máximo de *clusters* do formato NTFS adotado pelo sistema *Windows*. Nessas páginas estão armazenadas os objetos com suas informações e a cada acesso a uma página para realizar uma leitura é computado como **uma página lida**. O tempo de resposta é medido em segundos.

Tabela 6.1: Parâmetros utilizados com os valores padrões destacados em negrito.

Parâmetros	**Valores**
Número de objetos de referência (*Twitter*)	4000, 8000, **12000**, 16000, 20000
Número de locais existentes	1500, 2000, **2500**, 3000, 3500
k locais candidatos	1, 2, **3**, 4, 5
Número de palavra chave	1, 2, **3**, 4, 5

Em cada experimento são executadas 10 consultas de aquecimento, para que o sistema operacional carregue os dados necessários no seu *buffer*, e coleta a média dos resultados das próximas 100 consultas dos k locais mais influentes. Às palavras-chaves da consulta são obtidas coletando palavras aleatórias e sem repetição entre os 100 termos mais frequentes presentes na descrição textual da base de dados dos objetos de referência. A Tabela 6.1 apresenta os parâmetros utilizados para a consulta, os valores destacados em negrito foram escolhidos como padrões, pois as pesquisas encontradas no estado da arte utilizam destes intervalos para medir o desempenho de suas abordagens.

6.3 Experimentos

Esta seção apresenta o experimento dos algoritmos presentes no Capítulo 6. Na Subseção 6.3.1 é apresentado o experimento que varia o tamanho da base de dados dos locais existentes. A Subseção 6.3.2 é apresentado o experimento que avalia o

desempenho de acordo com a variação da base de dados dos objetos de referência. A Subseção 6.3.3 apresenta a variação da quantidade de locais candidatos. Por fim, a Subseção 6.3.4 avalia a variação da quantidade de palavras-chave. Os algoritmos foram nomeados com as siglas **IT**, **ITA** e o **ITE**, onde IT (índice textual) representa o Algoritmo 2, ITA (índice textual aprimorado) o Algoritmo 3 e o ITE (índice textual e espacial) o Algoritmo 4.

Vários testes foram realizados nos algoritmos com modificações na forma de processar a distância entre os objetos, com a finalidade de obter um resultado melhor do que apenas com o índice textual. Assim, foi descoberto que ao processar a obtenção da distância do local existente mais próximo de todos os objetos de referência inicialmente, e em seguida, ao identificar se o local candidato é mais próximo que o local existente já processado, verificou-se uma melhora no desempenho do algoritmo conforme pode ser visto nos experimentos deste capítulo.

Os algoritmos 2, 3 e 4 alocam todos os objetos de referência em memória, e isso é um problema encontrado na abordagem desta pesquisa. Os arquivos são armazenados em disco, e os algoritmos executam estes arquivos por inteiro para serem comparados. Para pesquisas futuras, pretende-se melhorar esse quesito e realizar todo o processo em disco, aliviando a memória do sistema computacional que executa a consulta dos k melhores locais influentes através de palavras-chave.

6.3.1 Variando a quantidade de locais existentes

Neste experimento, pretende-se avaliar o impacto do tamanho da base de dados dos locais existentes na consulta por locais influentes. Para avaliar este impacto foi coletado o tempo de resposta e o número de páginas lidas.

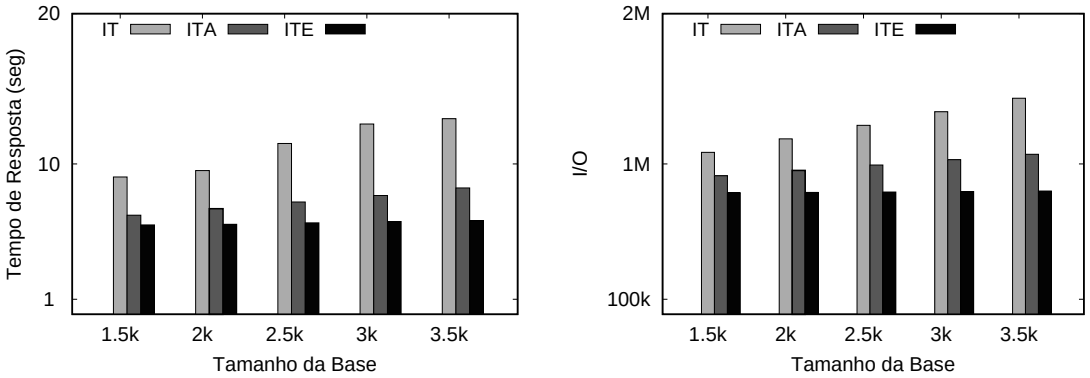
A Tabela 6.2 apresenta a quantidade de objetos de referência filtrados através das palavras-chave (valor padrão três) para cada conjunto da base de dados dos objetos de referência. Os algoritmos utilizaram destes objetos filtrados para executar a consulta, ou seja, reduziu consideravelmente o número de objetos que seriam necessários para o processamento.

O algoritmo *Baseline*, como o próprio nome sugere, foi criado apenas para servir de base de comparação com os algoritmos mais avançados que utilizam índice. Entretanto, os resultados obtidos foram insatisfatórios, levando entre 9 e 17 minutos para processar a consulta. Assim, optou-se por não exibir os resultados dos experimentos para o *Baseline* no gráfico, eles são apenas descritos no texto. O desempenho ruim deste algoritmo se dá pela varredura de todos os itens da base de locais existentes para computar o escore de cada local candidato, assim o tempo de resposta cresce proporcionalmente ao tamanho da base dos locais existentes.

Observa-se na Figura 6.1(a) que o tempo de resposta aumenta à medida que a quantidade de locais existentes aumenta, ou seja, a cada variação dos locais existentes para ser processado, a diferença de tempo entre as variações é similar de acordo

Tabela 6.2: Número médio de objetos de referência encontrados depois da utilização do índice para filtrar três palavras-chave.

Conjunto inicial	**Quantidade média dos objetos de referência relevantes para as palavras-chave da consulta.**
4000	27
8000	52
12000	74
16000	97
20000	123



(a) Tempo de resposta ao variar o tamanho da base de locais existentes.

(b) Páginas lidas ao variar o tamanho da base de locais existentes.

Figura 6.1: Resultado ao variar a base dos locais existentes.

com o acréscimo de locais existentes nos algoritmos. Os algoritmos com índice textual tiveram um desempenho melhor em relação ao *Baseline*, isso ocorreu devido à amostra de objetos de referência processados ser menor em relação ao *Baseline*. O algoritmo processa apenas os objetos de referência que contém similaridade textual com as palavras-chave escolhidas pelo algoritmo.

A Figura 6.1(a) apresenta a redução no tempo de resposta do ITE em relação ao IT e ITA. O parâmetro $1.5k$, por exemplo, o ITE obteve um tempo aproximado em 5 segundos, o ITA obteve um tempo de resposta aproximado de $6,5$ segundos, o IT obteve aproximadamente 9 segundos de tempo de resposta.

O número de páginas lidas cresceu proporcionalmente ao tamanho da base de dados dos locais existentes. O algoritmo *Baseline* leu cerca de 50 vezes mais páginas do que os algoritmos com índice textual, o qual reduz significativamente a quantidade de objetos processados, passando de 12 mil (base original) para 74 objetos após a seleção dos objetos textualmente relevantes (Tabela 7.2). Logo, os algoritmos que utilizam índices acessam um número bem menor de páginas.

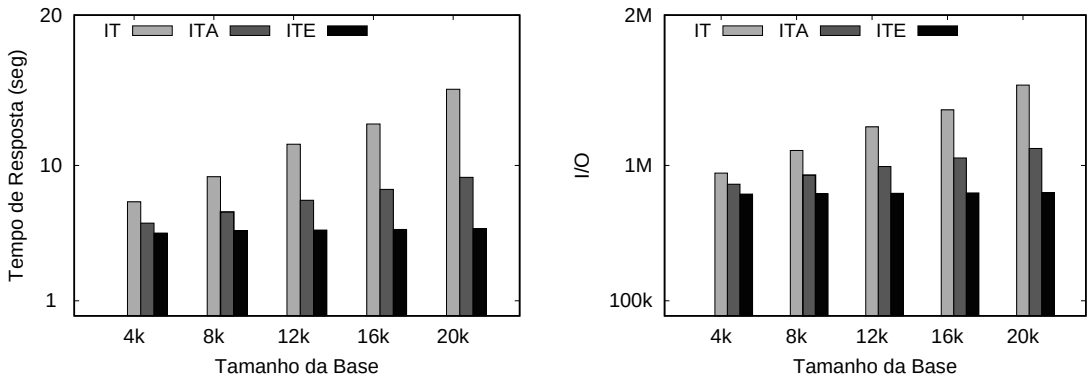
A Figura 6.1(b) representa o número de páginas lidas comparando os algoritmos, provando que o ITE reduziu também a quantidade de páginas lidas pelo algoritmo. Isso ocorre devido ao uso da *R-tree* presente no algoritmo, reduzindo a busca pelo local existente mais próximo do objeto de referência. O ITA armazena o local existente mais próximo de cada objeto de referência, e o IT realiza essa consulta do local existente mais próximo para cada local candidato.

6.3.2 Variando a quantidade dos objetos de referência

Neste experimento, pretendeu-se avaliar o impacto do tamanho da base de dados dos objetos de referência na consulta por locais influentes. Para mensurar este impacto foi coletado o tempo de resposta e o número de páginas lidas.

O Algoritmo *Baseline*, assim como na variação dos locais existentes, obteve tempo elevado em relação aos algoritmos com índice textual. O tempo de resposta para a consulta utilizando o algoritmos *Baseline* variou entre 5 minutos (4 mil objetos) e aproximadamente 20 minutos (20 mil objetos).

Para a variação do tamanho da base de dados dos objetos de referência, o tempo cresce proporcional ao passo que o número de objetos é aumentado. O ITE usa a busca com a *R-tree*, mantendo seu tempo de consulta na casa dos 5 segundos. O ITA não tem esse fator crítico na sua consulta, pois ele só realiza a varredura nos locais existentes uma vez, e avança para realizar a comparação nos locais candidatos. O IT executa para cada objeto de referência a proximidade espacial com todos os locais existentes da base para cada local candidato, por isso tem seu tempo elevado em relação às outras duas técnicas.



(a) Tempo de resposta ao variar o número de objetos de referência.

(b) Número de páginas lidas ao variar o número de objetos de referência.

Figura 6.2: Resultado ao variar a base dos objetos de referência.

A Figura 6.2(a) apresenta o tempo de resposta dos algoritmos. O comportamento deles é linear e crescente. No entanto, o ITE eleva seu tempo de resposta a cada variação em intervalos menores em relação ao IT e ITA. Nota-se que a distância

entre as barras vai aumentando ao passo que o tamanho da base é aumentado. Vale lembrar que para os três algoritmos (IT, ITA e ITE), o filtro dos objetos de referência está sendo aplicado Tabela 6.2, diminuindo a quantidade de objetos processados pela consulta. Sendo assim os algoritmos com índice textual reduziram o tempo de resposta da consulta, que inicialmente era executada em 20 minutos do *Baseline* para menos de 5 segundos, como o caso do ITE.

Como apresentado no tempo de resposta, o número de páginas lidas cresceu proporcional ao número de registros processados. A utilização de índices textuais ajudou a diminuir essa leitura de páginas lidas, pois torna eficiente a consulta dos registros em disco. O algoritmo *Baseline* variou entre 45 milhões ($4k$) a 155 milhões ($20k$) de páginas lidas.

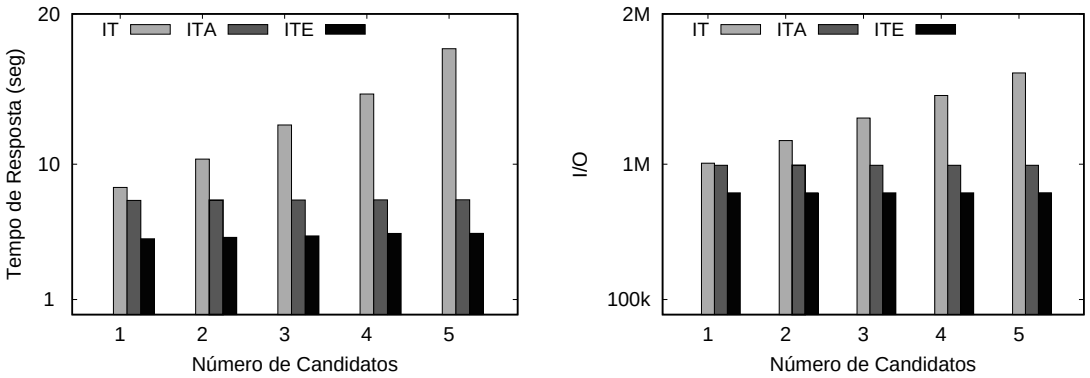
A Figura 6.2(b) apresenta o número de páginas lidas pelos algoritmos. Assim como acontece com o *Baseline*, os algoritmos com índice textual aumentam o número de páginas lidas proporcionalmente ao número de objetos de referência presentes na base de dados, porém o ITE mantém ssua leitura em um valor sempre menor que os demais. O ITE variou seu número de páginas lidas entre 809 mil a 820 mil. O algoritmo IT variou o número de páginas lidas entre 900 mil a $1,5$ milhões. O algoritmo ITA variou entre 854 mil a 1 milhão de páginas lidas aproximadamente. Portanto, com a variação dos objetos de referência, o ITE melhorou tanto em tempo de resposta quanto em número de páginas lidas à consulta.

6.3.3 Variando o número de locais candidatos

Neste experimento, pretendeu-se estudar o impacto do tamanho da base de dados dos locais candidatos na consulta por locais influentes. Para avaliar este impacto, foi coletado o tempo de resposta e o número de páginas lidas.

O tempo entre cada execução aumenta devido ao processo de comparação que cada candidato exige para todos os objetos de referência da base, ou seja, com uma base de 12 mil objetos de referência, o algoritmo *Baseline* compara para cada candidato a sua distância com os 12 mil objetos de referência, caso o objeto de referência seja atraído pelo candidato, é incrementado o valor da similaridade textual ao local candidato. O tempo de resposta do algoritmo *Baseline* variou entre 5 minutos (1 candidato) a 21 minutos (5 candidatos), demonstrando o péssimo desempenho do algoritmo em relação ao tempo de resposta.

A Figura 6.3(a) apresenta o tempo de resposta dos algoritmos 2, 3 e 4. Novamente o ITE têm a menor variação de tempo de execução, na faixa dos 5 segundos. O IT variou entre 9 e 18 segundos, e o algoritmo ITA alternou dentro dos 7 segundos. A diferença entre os algoritmos com índice textual acontece devido a modificação apresentada no Capítulo 6, que provoca uma diminuição na comparação dos objetos de referência com os locais existentes. Já a diferença do ITA e do ITE é a utilização de uma consulta na *R-tree* para os locais existentes.



(a) Tempo de resposta ao variar tamanho da base de locais candidatos.

(b) Número de páginas lidas ao variar o número de locais candidatos.

Figura 6.3: Resultado ao variar a base dos locais candidatos.

Similar ao tempo de resposta, o número de páginas lidas aumenta para todos os algoritmos, ao passo que o número de locais candidatos aumenta, pois mais objetos de referência são consultados para calcular a proximidade e a sua similaridade textual. O algoritmo *Baseline* manteve o comportamento similar às variações dos locais existentes e objetos de referência, linear e crescente. o número de páginas lidas variou entre 31 milhões (1 local candidato) a 155 milhões (5 locais candidatos).

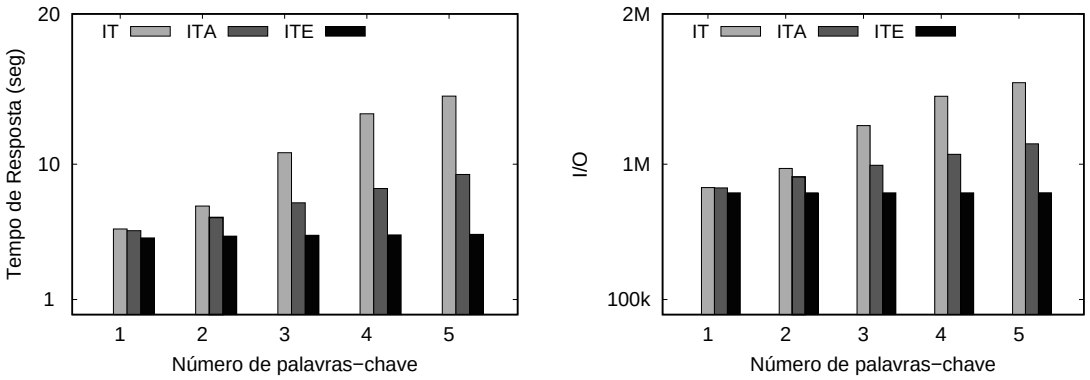
A Figura 6.3(b) apresenta o número de páginas lidas, ao variar o número de locais candidatos para os algoritmos que utilizam índice textual. O comportamento é linear e crescente nas duas abordagens, porém o algoritmo ITE e ITA tem uma variação mínima entre cada quantidade de locais candidatos. Isso acontece devido ao processamento dos locais existentes com os objetos de referência ser realizado apenas uma vez por execução no ITA e utilização da *R-tree* para buscar o locais existente mais próximo do objeto de referência, ao contrário do IT que faz o processamento dos locais existentes com os objetos de referência para cada local candidato.

6.3.4 Variando o número de palavras-chave

Neste experimento, pretendeu-se estudar o impacto da quantidade de palavras-chave na consulta por locais influentes. Para avaliar este impacto foi coletado o tempo de resposta e o número de páginas lidas.

O tempo entre cada execução não teve mudanças perceptíveis no algoritmo *Baseline*, pois as palavras-chave são utilizadas para computar a similaridade textual dos objetos de referência. Uma vez que os objetos são mantidos no valor padrão de 12.000 (doze mil), o cálculo de similaridade é realizado para cada um destes objetos, independente do número de palavras-chave. Os algoritmos com índice textual estão presentes na Figura 6.4(a).

É possível visualizar na Figura 6.4(a), que para os algoritmos o tempo de resposta



(a) Tempo de resposta ao variar a quantidade de palavras-chave.

(b) Número de páginas lidas ao variar a quantidade de palavras-chave.

Figura 6.4: Resultado ao variar a quantidade de palavras-chave.

é crescente entre os parâmetros. Uma vez que a quantidade de objetos de referência aumenta de acordo com a quantidade de palavras-chave selecionadas. A Tabela 6.3 apresenta o número de objetos de referência de acordo com a quantidade de palavras-chave.

Tabela 6.3: Número médio de objetos de acordo com a quantidade de palavras-chave

Palavras-Chave	**Quantidade média dos objetos de referência relevantes para a quantidade de palavras-chave da consulta.**
1	12
2	43
3	74
4	105
5	134

Comparando as abordagens na Figura 6.4(a), novamente o ITE obteve melhor desempenho em relação ao tempo de resposta do que os outros dois. O ITE variou dentro dos 5 segundos, o ITA variou entre 5 e 9 segundos dentro dos parâmetros, enquanto que o IT alternou entre 5 e 14 segundos.

Levando em consideração que os objetos foram processados anteriormente para obter o número de ocorrências de cada termo, o número de páginas lidas não foi alterado ao variar o número de palavras-chaves no algoritmo *Baseline*, visto que para processar a similaridade textual entre as palavras-chave da consulta e o texto dos objetos de referência, não é necessário acessar dados armazenados em disco, pois as informações necessárias para processar essa similaridade são previamente pré-processadas e armazenadas em memória. Os número de páginas lidas se manteve em aproximadamente 100 milhões, para todos os parâmetros.

A Figura 6.4(a) apresenta o tempo de resposta dos algoritmos após a execução do filtro dos objetos de referência através do índice textual. O tempo de resposta variou de acordo com o acréscimo de palavras-chave. Ambos mantiveram o comportamento de páginas lidas ao passo que cresce a quantidade de palavras-chave, tal como aconteceu com os outros parâmetros.

O IT obteve uma variação entre 840 mil até $1,55$ milhão de páginas lidas. O algoritmo ITA obteve uma variação entre 840 mil até $1,1$ milhão de páginas lidas. O algoritmo ITE obteve uma variação menor de 809 mil até 511 mil páginas lidas.

Capítulo 7

Considerações Finais

"A ciência nunca resolve um problema sem criar pelo menos outros dez."
– George Bernard Shaw

Essa pesquisa apresentou uma nova proposta para a consulta de locais influentes através de palavra-chave. Este trabalho baseou-se em estudos de consultas de locais influentes presentes na literatura. A investigação na literatura apresentou consultas de locais influentes com diversidade em algoritmos para processa-las.

A consulta de locais influentes pode ser modificada para utilizar o texto presente na base de dados para avaliar a relevância dos objetos de referência. Sendo assim, propôs-se uma nova consulta de locais influentes à partir de palavra-chave que permite o usuário utilizar um conjunto palavras-chave que esteja diretamente relacionado ao seu interesse. Além disso, utilizou-se da técnica de índice textual para melhorar o desempenho da consulta, obtendo tempos de resposta satisfatórios e com menos paginas lidas.

7.1 Contribuições

As principais contribuições desta pesquisa são os algoritmos propostos que permitem o processamento da consulta de locais influentes através de palavras-chave de forma eficiente. Sistemas que utilizam consulta através de texto podem se beneficiar da abordagem proposta para realizar a consulta por locais influentes.

As bases de dados utilizadas nos experimentos foram disponibilizadas para que outros pesquisadores possam utilizá-las em suas pesquisas ou até mesmo compara-las com os resultados desta pesquisa. A fundamentação teórica também é um contribui-

ção por apresentar conceitos e estruturas utilizadas no processamento das consulta por locais influentes.

7.2 Pesquisas futuras

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

Índice Espacial. A utilização de dados geoespaciais possibilita a utilização de índices espaciais para melhorar o desempenho da consulta.

Base de Dados. Os experimentos foram realizados em um conjunto de 1,5 milhão de objetos. No entanto, a necessidade dos sistemas funcionarem com um volume maior de dados torna interessante aprofundar a pesquisa com bases maiores e de diferentes fonte como *Instagram*, por exemplo.

Sistema com Análise Semântica. A análise semântica do texto pode auxiliar na obtenção mais precisa do interesse de um determinado objeto, desta forma a consulta pode filtrar os objetos pelo seu desejo real, e não por ter escrito uma palavra com similaridade, mas que não tem relação com o produto final.

Índices Aplicados nas Abordagens *MinSum* e *MinMax*. O *MinSum* e o *MinMax* que se diferem pelo método de computar o escore dos locais influentes desta pesquisa, mas o cálculo de similaridade textual pode ser o mesmo utilizado nesta pesquisa, e portanto, o índice textual pode diminuir a quantidade de objetos processados pela consulta.

Referências Bibliográficas

[Arroyuelo et al. 2014] Arroyuelo, D., Bonacic, C., Gil-Costa, V., Marin, M., e Distributed text search using suffix arrays. *Parallel Computing*, 40(9):471–495.

[Athayde-Novaes 2017] Athayde-Novaes, T. F. (2017). Processamento distribuído da consulta espaço textual top-k. 2017. 80p. Dissertação, Universidade Estadual de Feira de Santana, Programa de Pós-Graduação em Computação Aplicada.

[Beckmann et al. 1990] Beckmann, N., Kriegel, H.-P., Schneider, R., e Seeger, B. (1990). The r*-tree: an efficient and robust access method for points and rectangles. In *Acm Sigmod Record*, volume 19, pp. 322–331. ACM.

[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. Springer.

[Cao et al. 2012] Cao, X., Chen, L., Cong, G., Jensen, C. S., Qu, Q., Skovsgaard, A., Wu, D., e Yiu, M. L. (2012). Spatial keyword querying. In *International Conference on Conceptual Modeling*, pp. 16–29. Springer.

[Chen et al. 2013] Chen, L., Cong, G., Jensen, C. S., e Wu, D. (2013). Spatial keyword query processing: an experimental evaluation. In *Proceedings of the VLDB Endowment*, volume 6, pp. 217–228. VLDB Endowment.

[Chen et al. 2014] Chen, Z., Liu, Y., Wong, R. C.-W., Xiong, J., Mai, G., e Long, C. (2014). Efficient algorithms for optimal location queries in road networks. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 123–134. ACM.

[Chen et al. 2015] Chen, Z., Liu, Y., Wong, R. C.-W., Xiong, J., Mai, G., e Long, C. (2015). Optimal location queries in road networks. *ACM Transactions on Database Systems (TODS)*, 40(3):17.

[Choudhury et al. 2016] Choudhury, F. M., Culpepper, J. S., Sellis, T., e Cao, X. (2016). Maximizing bichromatic reverse spatial and textual k nearest neighbor queries. *Proceedings of the VLDB Endowment*, 9(6):456–467.

[Chung et al. 2018] Chung, Y.-C., Su, I.-F., e Lee, C. (2018). k-most suitable locations selection. *GeoInformatica*, pp. 1–32.

[Cohen et al. 2003] Cohen, W., Ravikumar, P., e Fienberg, S. (2003). A comparison of string metrics for matching names and records.

[Comer 1979] Comer, D. (1979). Ubiquitous b-tree. *ACM Computing Surveys (CSUR)*, 11(2):121–137.

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

[de Almeida e Rocha-Junior 2016] de Almeida, J. P. D. e Rocha-Junior, J. B. (2016). Top-k spatial keyword preference query. *Journal of Information and Data Management*, 6(3):162.

[Du et al. 2005] Du, Y., Zhang, D., e Xia, T. (2005). The optimal-location query. In *International Symposium on Spatial and Temporal Databases*, pp. 163–180. Springer.

[Güting 1994] Güting, R. H. (1994). An introduction to spatial database systems. *The VLDB Journal-The International Journal on Very Large Data Bases*, 3(4):357–399.

[Huang et al. 2011] Huang, J., Wen, Z., Qi, J., Zhang, R., Chen, J., e He, Z. (2011). Top-k most influential locations selection. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pp. 2377–2380. ACM.

[Kang et al. 2007] Kang, J. M., Mokbel, M. F., Shekhar, S., Xia, T., e Zhang, D. (2007). Continuous evaluation of monochromatic and bichromatic reverse nearest neighbors. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pp. 806–815. IEEE.

[Korn e Muthukrishnan 2000] Korn, F. e Muthukrishnan, S. (2000). Influence sets based on reverse nearest neighbor queries. In *ACM Sigmod Record*, volume 29, pp. 201–212. ACM.

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

[Liu et al. 2013] Liu, Y., Wong, R. C.-W., Wang, K., Li, Z., Chen, C., e Chen, Z. (2013). A new approach for maximizing bichromatic reverse nearest neighbor search. *Knowledge and information systems*, 36(1):23–58.

[Lu et al. 2011] Lu, J., Lu, Y., e Cong, G. (2011). Reverse spatial and textual k nearest neighbor search. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pp. 349–360. ACM.

[Lu et al. 2014] Lu, Y., Lu, J., Cong, G., Wu, W., e Shahabi, C. (2014). Efficient algorithms and cost models for reverse spatial-keyword k-nearest neighbor search. *ACM Transactions on Database Systems (TODS)*, 39(2):13.

[Manning et al. 2008] Manning, C. D., Raghavan, P., Schütze, H., et al. (2008). *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

[Papadias et al. 2001] Papadias, D., Kalnis, P., Zhang, J., e Tao, Y. (2001). Efficient olap operations in spatial data warehouses. *Advances in spatial and temporal databases*, pp. 443–459.

[Prodanov e de Freitas 2013] Prodanov, C. C. e de Freitas, E. C. (2013). *Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico-2ª Edição*. Editora Feevale.

[Qi et al. 2014] Qi, J., Zhang, R., Wang, Y., Xue, A. Y., Yu, G., e Kulik, L. (2014). The min-dist location selection and facility replacement queries. *World Wide Web*, 17(6):1261–1293.

[Rigaux et al. 2001] Rigaux, P., Scholl, M., e Voisard, A. (2001). *Spatial databases: with application to GIS*. Morgan Kaufmann.

[Rocha-Junior 2012] Rocha-Junior, J. B. (2012). *Efficient Processing of Preference Queries in Distributed and Spatial Databases*. PhD thesis, Norges teknisk-naturvitenskapelige universitet, Fakultet for informasjonsteknologi, matematikk og elektroteknikk, Institutt for datateknikk og informasjonsvitenskap.

[Rocha-Junior et al. 2011] Rocha-Junior, J. B., Gkorgkas, O., Jonassen, S., e Nørvåg, K. (2011). Efficient processing of top-k spatial keyword queries. In *SSTD*, volume 6849, pp. 205–222. Springer.

[Salminen e Tompa 1994] Salminen, A. e Tompa, F. W. (1994). Pat expressions: an algebra for text search. *Acta Linguistica Hungarica*, 41(1):277–306.

[Stanoi et al. 2001] Stanoi, I., Riedewald, M., e Agrawal, D. (2001). Discovery of influence sets in frequently updated databases. In *VLDB*, volume 2001, pp. 99–108.

[Vaid et al. 2005] Vaid, S., Jones, C. B., Joho, H., e Sanderson, M. (2005). Spatio-textual indexing for geographical search on the web. In *International Symposium on Spatial and Temporal Databases*, pp. 218–235. Springer.

[Wong et al. 2011] Wong, R. C.-W., Özsu, M. T., Fu, A. W.-C., Yu, P. S., Liu, L., e Liu, Y. (2011). Maximizing bichromatic reverse nearest neighbor for lp-norm in two-and three-dimensional spaces. *The VLDB Journal-The International Journal on Very Large Data Bases*, 20(6):893–919.

[Wong et al. 2009] Wong, R. C.-W., Özsu, M. T., Yu, P. S., Fu, A. W.-C., e Liu, L. (2009). Efficient method for maximizing bichromatic reverse nearest neighbor. *Proceedings of the VLDB Endowment*, 2(1):1126–1137.

[Wu et al. 2012] Wu, D., Yiu, M. L., Cong, G., e Jensen, C. S. (2012). Joint top-k spatial keyword query processing. *IEEE Transactions on Knowledge and Data Engineering*, 24(10):1889–1903.

[Xia et al. 2005] Xia, T., Zhang, D., Kanoulas, E., e Du, Y. (2005). On computing top-t most influential spatial sites. In *Proceedings of the international conference on Very large data bases*, pp. 946–957. VLDB Endowment.

[Xiao et al. 2011] Xiao, X., Yao, B., e Li, F. (2011). Optimal location queries in road network databases. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pp. 804–815. IEEE.

[Xu et al. 2017] Xu, L., Mai, G., Chen, Z., Liu, Y., e Dai, G. (2017). Minsum based optimal location query in road networks. In *International Conference on Database Systems for Advanced Applications*, pp. 441–457. Springer.

[Yang et al. 2017] Yang, J., Zhang, W., Zhang, Y., Wang, X., e Lin, X. (2017). Categorical top-k spatial influence query. *World Wide Web*, 20(2):175–203.

[Yao et al. 2014] Yao, B., Xiao, X., Li, F., e Wu, Y. (2014). Dynamic monitoring of optimal locations in road network databases. *The VLDB Journal*, 23(5):697–720.

[Zhang et al. 2006] Zhang, D., Du, Y., Xia, T., e Tao, Y. (2006). Progressive computation of the min-dist optimal-location query. In *Proceedings of the 32nd international conference on Very large data bases*, pp. 643–654. VLDB Endowment.

[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*, pp. 155–162. ACM.

[Zhu et al. 2011] Zhu, S., Wu, J., Xiong, H., e Xia, G. (2011). Scaling up top-k cosine similarity search. *Data & Knowledge Engineering*, 70(1):60–83.

[Zobel e Moffat 2006] Zobel, J. e Moffat, A. (2006). Inverted files for text search engines. *ACM computing surveys (CSUR)*, 38(2):6.