UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

PAULO MATEUS ALVES MOREIRA

BUSCA ESCALÁVEL DE REGIÃO GEOGRÁFICA SIMILAR UTILIZANDO BM25

CAMPINA GRANDE - PB

2021

PAULO MATEUS ALVES MOREIRA

BUSCA ESCALÁVEL DE REGIÃO GEOGRÁFICA SIMILAR UTILIZANDO BM25

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel ou Bacharela em Ciência da Computação.

Orientador: Professor Dr. Claudio Elízio Calazans Campelo

CAMPINA GRANDE - PB

2021

PAULO MATEUS ALVES MOREIRA

BUSCA ESCALÁVEL DE REGIÃO GEOGRÁFICA SIMILAR UTILIZANDO BM25

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel ou Bacharela em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Claudio Elízio Calazans Campelo

Orientador – UASC/CEEI/UFCG

Professora Dr. Carlos Eduardo Santos Pires

Examinador – UASC/CEEI/UFCG

Professor Tiago Lima Massoni

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 14 de Fevereiro de 2023.

CAMPINA GRANDE - PB

RESUMO (ABSTRACT)

Information retrieval applications are increasingly robust, versatile and serving the most diverse purposes. In the area of Geographic Information Retrieval (GIR), although many approaches have been proposed to deal with several problems, some problems still remain insufficiently explored, such as approaches that allow searching for regions that are similar to a region of interest in a fast and scalable way . Given a spatial region and a query region, a similar region search aims to find the K regions most similar to the query region in the spatial region. This work presents a similarity search approach in maps using a textual similarity algorithm with stored and indexed data. For this, an algorithm was developed based on the use of indexes in ElasticSearch to store geographic data. Queries are textual, using the concept of inverted index together with the BM25 similarity algorithm. In this algorithm, the Points of Interest (POI) information present in the regions of the maps are converted into textual data that are the basis for carrying out queries. The proposed approach is based on a case study using data from large cities obtained through OpenStreetMaps.

Busca escalável de região geográfica similar utilizando BM25

Trabalho de Conclusão de Curso

Paulo Mateus Alves Moreira (Aluno), Cláudio Campelo (Orientador)
Departamento de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Paraíba - Brasil

RESUMO

Aplicações de recuperação de informação estão cada dia mais robustas, versáteis e atendendo aos mais diversos fins. Na área de Recuperação de Informação Geográfica (GIR), embora muitas abordagens tenham sido propostas para lidar com diversos problemas, alguns ainda permanecem insuficientemente explorados, a exemplo de abordagens que permitam buscar regiões que sejam similares a uma região de interesse de forma rápida e escalável. Dada uma região espacial e uma região de consulta, uma pesquisa por região similar visa encontrar as K regiões mais semelhantes à região de consulta na região espacial. Neste trabalho é apresentada uma abordagem de busca por similaridade em mapas utilizando um algoritmo de similaridade textual com dados armazenados e indexados. Para isso, desenvolveu-se um algoritmo baseado no uso de índices no *ElasticSearch*[1] para armazenamento dos dados geográficos. As consultas são textuais, utilizando o conceito de índice invertido junto com o algoritmo de similaridade `BM25`[14]. Nesse algoritmo, as informações de *Points of Interest* (`POI`) presente nas regiões dos mapas são convertidas em dados textuais que são a base para realização de consultas. A abordagem proposta é baseada em um estudo de caso utilizando dados de grandes cidades obtidos através OpenStreetMaps[1].

PALAVRAS-CHAVE

similaridade, regiões similares, elasticsearch, tf-idf, BM25

1 INTRODUÇÃO

A alta disponibilidade de objetos georreferenciados através do aumento do uso e proliferação de dispositivos móveis e serviços baseados em localização fazem com que o gerenciamento, manipulação e aplicações computacionais geográficas avancem naturalmente. Informações como *Points of Interest* (`POI`) estão cada vez mais associadas a informações ricas, como categorias e descrição textual. Isso oferece grandes oportunidades para entendimento das informações de uma determinada região geográfica. Porém, a expansão contínua e a complexidade das cidades dificultam a identificação de possíveis regiões com características até então desconhecidas. Essas características podem ser usadas para diversos fins, sendo um deles a busca por regiões geográficas similares. Por exemplo, no ramo de negócios, empresários que almejam alocar filiais em cidades ou regiões novas poderiam buscar por regiões semelhantes aos locais de seus negócios atuais, isto é, utilizar o conhecimento das características geográficas de suas filiais já existentes comparando-as com possíveis regiões de interesse. Outro exemplo seria de um usuário que recebeu uma promoção no trabalho e vai precisar se mudar para uma outra cidade. Na sua cidade atual ele gosta da região em que mora pois existe mais de um bicicletário próximo, restaurantes de diferentes culinárias e parques para que se possa fazer caminhada e passear com animais. É de interesse deste usuário encontrar em sua nova cidade uma região que seja mais similar a região em que ele morava em sua cidade antiga.

Portanto, é importante pesquisar por regiões similares no espaço geográfico para dar suporte a uma variedade de aplicações do mundo real. Essa busca pode ser aplicada a diversos contextos, como desenvolvimento urbano, recomendação de `POIs`, análise de mercado, administração pública, entre outros. Um dos principais desafios é a forma de capturar uma representação da região considerando as categorias dos seus objetos, palavras chaves e a relação entre esses objetos que permitam que essa região possa ser comparada com uma outra. Um dos primeiros estudos nessa área [15] tenta enfrentar esse desafio propondo medir a similaridade entre duas regiões com a similaridade do cosseno de seus vetores de características espaciais, cada um dos quais é calculado com base nas distâncias dos objetos a cinco pontos de referência predefinidos na região com um algoritmo de busca baseado em *quadtree*. Por outro lado, o estudo conduzido por Yiding et al.[12] tenta capturar as posições relativas dos objetos utilizando um modelo de aprendizagem profunda capaz de comparar duas regiões. As características geográficas de uma região são extraídas usando uma rede neural neural convolucional que pode capturar relacionamentos locais inerentes entre objetos para aprendizado de similaridade.

Os resultados das abordagens citadas são promissores, mas no pior caso, a busca por regiões similares pode ter uma complexidade `O(n)`, onde `n` é a quantidade de regiões possíveis em uma área de consulta. Além disso, consultas em regiões muito grandes podem demorar muito tempo a depender dos recursos computacionais disponíveis pois a similaridade é calculada em tempo de execução para cada região possível. Neste trabalho é proposta uma estratégia com base nos conceitos da recuperação da informação [3] para busca de região similar baseado nas informações de vizinhança entre os pontos e a categoria de cada um deles. A estratégia utiliza índices no *ElasticSearch* com mapeamentos geográficos e textuais, apontando respectivamente para coordenadas e categorias de `POIs`. Os objetos armazenados nesses índices são inicialmente pontos geográficos com latitude/longitude e uma descrição textual que o categoriza. A partir desses dados são indexadas regiões geográficas retangulares (*bounding-box*) junto com as informações textuais de

[1] https://www.openstreetmap.org/

cada ponto interno à esses retângulos. Dessa forma, é possível que algoritmos de similaridade textual, como o BM25 [14], sejam usados sobre essas informações textuais das regiões, permitindo que elas possam ser comparadas com base em suas representações de texto. Assim, esse trabalho garante que a similaridade não é calculada em tempo de execução e sim antes mesmo da consulta ser feita, pois as informações de ranqueamento utilizadas são geradas e armazenadas durante o processo de indexação, fazendo com que o trabalho seja apenas comparar regiões com a região de consulta.

O restante deste trabalho está estruturado da seguinte maneira. A Seção 2 fornece uma breve descrição dos fundamentos teóricos para o entendimento da pesquisa. Em seguida, a Seção 3 discorre sobre trabalhos relacionados. Em seguida, a Seção 4 explica a abordagem proposta junto com as estratégias utilizadas. Posteriormente, a Seção 5 descreve os dados utilizados, experimentos realizados e os resultados alcançados. Por fim, a Seção 6 apresenta conclusões para esse trabalho e aponta para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta uma visão geral dos fundamentos teóricos necessários para o entendimento da pesquisa. De início é explicada uma técnica para calcular distâncias geográficas e em seguida é explicado um algoritmo de similaridade textual.

2.1 Elasticsearch

O *Elasticsearch* é um motor de busca distribuído de código aberto, projetado para lidar com projetos de larga escala que manipulam grandes quantidades de dados em tempo real. Ele é baseado no sistema de índices do *Apache Lucene*[2] e fornece uma interface de busca de alto nível para aplicativos e serviços *web*.

Os dados que são armazenados no *Elasticsearch* são chamados de documentos. Esse documentos por sua vez são agrupados em índices de acordo com o tipo da informação contida em seus campos. O processo de agrupar os documentos em índices, ou seja, a indexação, permite que a busca seja feita de forma rápida, utilizando representação invertida dos dados, que mapeia termos para documentos.

Essa representação invertida é conhecida como índice invertido. Essa técnica é utilizada para indexar e recuperar informações em grandes coleções de documentos, como bibliotecas digitais ou bases de dados de pesquisa. Ela funciona invertendo a relação entre as palavras e os documentos em que elas aparecem. Dessa forma, ao invés de armazenar uma lista de documentos para cada palavra, o índice invertido armazena uma lista de palavras que referenciam os documentos em que aparecem.

Para performar uma busca o *Elasticsearch* funciona da seguinte maneira:

(1) Recebimento de uma consulta: Uma consulta é enviada ao endpoint de busca do Elasticsearch em formato `JSON`. A consulta pode incluir critérios de busca, filtros, agregações, classificações e outros parâmetros.
(2) Verificação do índice: O *Elasticsearch* verifica o índice especificado na consulta para determinar qual conjunto de documentos deve ser pesquisado.
(3) Aplicação de consultas e filtros: A consulta é aplicada aos documentos no índice e filtros adicionais são aplicados, se especificados. Isso resulta em uma lista de documentos que correspondem aos critérios da consulta.
(4) Aplicação de classificações e agregações: A lista de documentos resultantes é classificada de acordo com os parâmetros de classificação especificados na consulta. Além disso, agregações podem ser aplicadas aos dados, se especificadas, para calcular estatísticas e agrupar dados.
(5) Retorno de resultados: Os resultados da busca são retornados ao solicitante em formato `JSON`. O número de resultados retornados pode ser controlado usando parâmetros de paginação.

Outro fator importante sobre o *Elasticsearch* é sua alta escalabilidade, o que significa que ele pode lidar com grandes quantidades de dados e altas demandas de consultas da seguinte forma:

- Clusterização: O *Elasticsearch* é projetado para ser executado em um *cluster*, o que permite dividir os dados em vários nós e distribuir a carga de processamento. Isso significa que é possível adicionar mais nós ao *cluster* conforme a demanda aumenta, permitindo uma escalabilidade horizontal.
- Replicação: Os dados armazenados em um índice do *Elasticsearch* podem ser replicados em múltiplos nós, o que aumenta a disponibilidade dos dados e melhora a tolerância a falhas.
- *Sharding*: Cada índice do *Elasticsearc*h pode ser dividido em vários fragmentos, conhecidos como *shards*. Isso permite que os dados sejam armazenados em múltiplos nós, o que aumenta a capacidade de processamento e permite escalabilidade vertical.

Além disso, o *Elasticsearch* fornece ferramentas para ajudar a monitorar e manter a performance do *cluster*, como o *Kibana*[3], que permite visualizar dados de desempenho e configurar alertas.

2.2 Cálculo de distâncias

Em 1880, o geodesista alemão Friedrich Robert Helmert publica um livro [5] onde define teorias e cálculos usados até hoje. Um dos cálculos definidos foi o da distância geodésica da terra. Esse cálculo mede uma distância aproximada entre dois pontos baseado na figura em formato de elipsoide de revolução da terra. O elipsoide de revolução é uma forma geométrica gerada pela rotação de uma semi-elipse em torno de um de seus eixos. A motivação por trás da criação desse modelo geométrico foi a complexidade de realizar cálculos matemáticos considerando o formato geoide da terra[2]. A Equação 1 mede a distância geodésica entre dois pares de coordenadas. O valor de variável *radius* é determinado pela localização de cada coordenada visto que o formato de elipsoide de revolução possui raios variáveis dependendo do referencial.

$$distance(c1, c2) = arccos(c1, c2) * \pi * radius \quad (1)$$

2.3 Algoritmo de ranqueamento

O termo ranqueamento, no contexto de recuperação da informação, diz respeito a quão relevante determinado documento se mostra com relação a uma determinada consulta. O ranqueamento também

[2]https://lucene.apache.org/core/9_1_0/index.html

[3]https://www.elastic.co/pt/kibana/

pode demonstrar o quanto um documento é similar à consulta realizada. Para o cálculo do *ranking* são utilizados algoritmos de similaridade. Esses algoritmos são baseados no conceito de TF-IDF.

TF-IDF é a abreviação para *term frequency-inverse document frequency*. Essa técnica visa calcular um valor que diz o quão relevante uma palavra é para um documento em razão da aparição dela em todos os documentos armazenados em uma coleção utilizando dois valores estatísticos. *Term Frequency* [13] é uma estatística da frequência de um termo em um documento. Já *Inverse Document Frequency* [9] é definida pela divisão entre a quantidade de todos os documentos e a quantidade de documentos que o termo específico aparece. Então essa técnica atribui pesos às palavras de acordo com sua especificidade. Quanto mais comum é uma palavra na coleção de todos os documentos, menor é o peso dela. Por fim, a multiplicação entre essas duas estatísticas gera o valor de TF-IDF de um termo.

Uma das implementações mais usadas de TF-IDF é no algoritmo *Okapi BM25*, desenvolvido por Stephen E. Robertson, Karen Spärck Jones, e outros [14].

2.4 BM25

BM25 (BM25 *Okapi*) é um algoritmo de classificação de informação utilizado para avaliar a relevância de documentos em relação a uma consulta de pesquisa. Foi criado como uma melhoria em relação ao modelo TF-IDF padrão, com o objetivo de fornecer resultados mais precisos em sistemas de pesquisa.

O BM25 é baseado no modelo TF-IDF, mas adiciona alguns ajustes para melhorar a precisão dos resultados. Ele leva em conta não apenas a frequência dos termos em um documento, mas também a frequência dos termos na coleção de documentos inteira, bem como o comprimento do documento. Além disso, o BM25 inclui parâmetros de ajuste que permitem definir a importância relativa desses diferentes fatores. A Equação 2 mostra como é realizado esse cálculo.

$$\sum_{i}^{n} IDF_{(q_i)} = \frac{f(q_i, D) * (k1 + 1)}{f(q_i, D) + k1 * (1 - b + b * \frac{fieldLen}{avgFieldLen})} \quad (2)$$

- q: Representa um termo ao qual se tem interesse calcular um valor de *ranking*
- IDF(q): Frequência inversa do termo q
- *fieldLen/avgFieldLen*: divisão entre o tamanho do documento e a média dos tamanhos de todos os documentos. Podemos pensar nisso como o quão longo é o comprimento de um documento em relação ao comprimento médio do de todos os documentos. Se um documento for maior que a média, o denominador fica maior (diminuindo a pontuação) e se for menor que a média, o denominador fica menor (aumentando a pontuação). Isso é necessário pois, quanto maior a quantidade de termos no documento, menor deve ser a pontuação de termos no documento. O sentido disso é que se, por exemplo, em documento com 10000 palavras o termo do meu interesse só aparece uma vez é bem menos provável que o documento tenha algo a ver com o termo.
- b: Valor que aumenta ou diminui a penalidade aplicada sobre o tamanho de um documento. Esse valor só pode ir de 0 até 1.
- f(q, D): a frequência de um termo q em um documento D
- k1: termo que ajuda a determinar a característica de saturação da frequência do termo em um documento. O k1 funciona como uma espécie de teto para a frequência do termo em um documento. A Partir de uma quantidade X de termos os termos que ainda aparecerem não terão tanto impacto na pontuação.

3 TRABALHOS RELACIONADOS

3.1 Similaridade de regiões geográficas

A busca por similaridade em mapas ainda não é um tópico amplamente discutido, fazendo com que a maioria dos artigos existentes tratem de experimentação, havendo pouco conteúdo literário pontuando o que há de melhor. Entretanto, os artigos observados se mostram promissores, evidenciando estratégias robustas e complexas para poder calcular a similaridade entre regiões. Sheng et al. [15] propuseram uma das primeiras abordagens que visa medir a similaridade entre duas regiões utilizando vetores de características espaciais de cada região. Esse vetor de características é definido pelas distâncias médias entre os objetos e cinco pontos de referência predefinidos em lugares estratégicos na região. Para medir um valor de similaridade utilizou-se a similaridade do cosseno entre os vetores dessas regiões. No trabalho de Yiding et al. [12] a medição de similaridade é realizada sobre os tipos de POIs e as posições relativas deles entre si. Essas informações são extraídas com o uso de uma rede neural convolucional capaz de capturar relacionamentos entre objetos da região e por fim um modelo de aprendizagem profunda tenta classificar um valor de similaridade para as regiões. Uma outra abordagem proposta [7] utiliza bases de dados de grafos de conhecimento espacial (*Spatial Knowledge Graphs*) para gerar vetores espaciais. Os tipos de relações e hierarquias presentes nesses dados contém características que ficam ocultas em outras abordagens.

Em comparação com essas abordagens, a proposta apresentada neste artigo calcula a similaridade utilizando apenas as informações textuais de um POI e a proximidade dele com outros dois POIs dentro de um intervalo de distância. Dessa forma é possível identificar uma vizinhança comum entre os POIs de forma simples e rápida.

3.2 Recuperação da informação

Nos trabalhos citados a similaridade é calculada em tempo de execução. Isso significa que em uma consulta por região similar, as regiões candidatas são iteradas de forma a comparar cada uma com a região de consulta. As otimizações realizadas nos algoritmos de busca não conseguem evitar que no pior caso a complexidade seja O(n). Então, embora as abordagem propostas por esses artigos sejam precisos em definir uma similaridade, eles não são tão rápidos quanto, por exemplo, uma consulta textual em um índice com documentos de texto, que tem como complexidade O(log n). Esse trabalho propõe uma abordagem escalável de busca por similaridade baseado nos conceitos base de recuperação da informação [3]. Os estudos na área de recuperação de informação em dados

geográficos [6, 8, 11] discutem atualmente aspectos mais robustos, como por exemplo, busca com frases geográficas complexas que precisam ser interpretadas, como "30 km a oeste Zurique". Por outro lado, o armazenamento e disponibilidade dos dados geográficos em si, atualmente, é realizada em bancos de dados espaciais, que dispõem de um ferramental robusto que permite consultas de distância entre pontos, consultas por polígonos e consultas textuais de forma rápida e eficiente. Um exemplo de banco de dados espacial é o `postGIS`[4] que funciona como uma extensão do banco de dados objeto-relacional `PostgreSQL`[5]. O `postGIS` adiciona suporte para objetos geográficos, permitindo que consultas de localização sejam executadas em `SQL`. Para unir o melhor que existe no quesito de busca textual e o melhor que existe em busca geográfica, esse trabalho faz uso de índices no *ElasticSearch* para armazenar as informações de coordenadas geográficas e seus respectivos textos. Dessa forma, é possível fazer consultas rápidas nos dados geográficos como também consultas rápidas no texto do corpus desses dados.

4 ABORDAGEM PROPOSTA

Essa seção descreve a abordagem proposta para garantir escalabilidade e eficiência das consultas.

4.1 Indexação dos dados

Utilizando dados geográficos de `POIs`, a indexação no *ElasticSearch* é realizada. São criados 3 índices: Índice Raiz, Índice de vizinhanças e Índice de Regiões, os quais possuem pelo menos dois tipos de mapeamento. Esses mapeamentos definem o tipo de informação que será armazenada em cada campo de um documento. Os tipos de mapeamento utilizados nos índices desse trabalho foram os de *text*, *geo_point* e *geo_shape*. O mapeamento do tipo *text* aceita um texto em formato de *string*. Esse tipo de mapeamento permite que a informação do campo seja analisada, ou seja, faz com que ela passe por um analisador para converter a *string* em uma lista de termos individuais antes de serem indexados em uma estrutura de índice invertido. O campo com mapeamento *geo_point*, por sua vez, aceita pares de valores com latitude e longitude. Já o campo com mapeamento *geo_shape* facilita a indexação de informações geográficas em formatos como retângulos, linhas e polígonos.

Uma outra funcionalidade do *Elasticsearch* que foi utilizada neste trabalho é a consulta por *bounding-box*. Esse tipo de consulta retorna todos os *geo_point* e *geo_shape* que fazem intersecção com um retângulo geográfico pré-definido nos parâmetros de busca.

4.1.1 Índice raiz.

O índice raiz é a base de dados fonte neste trabalho. Os documentos nele indexados possuem dois campos: *amenity* e *geometry*, tendo como mapeamentos os tipos *text* e *geo_point* respectivamente. O campo *amenity* guarda a informação que categoriza o `POI` enquanto que o campo *geometry* guarda pares de coordenadas que localizam o `POI`. Esse índice é a base para construção de um outro índice, o Índice de vizinhanças.

4.1.2 Índice de vizinhanças.

O índice de vizinhanças armazena uma informação de vizinhança entre os `POIs`. Os documentos armazenados nesse índice possuem os campos de *geometry* e *pois_neighbor* mapeados para *geo_shape* e *text*. O campo *geometry* guarda linhas geográficas entre no máximo três `POIs`. Essa linha representa uma proximidade entre esses três pontos dentro de um intervalo de distância. O intervalo escolhido neste trabalho foi 200 metros. O algoritmo[6] que realiza a indexação dessas linhas itera sobre todos os `POIs` presentes no Índice raiz criando linhas que ligam os 2 pontos mais próximos dentro do intervalo. Se o `POI` não possui 2 pontos próximos dentro do intervalo é selecionado apenas um ou nenhum. Já no campo de *pois_neighbor* é construída uma informação textual a partir da concatenação dos campos de *amenity* presentes em cada um dos pontos da linha criada anteriormente. A informação de *amenity* é concatenada com um "_" entre cada palavra, gerando textos que representam uma vizinhança sequencial de proximidade. Por exemplo, se estão mais próximos de um `POI` do tipo café, um restaurante e logo em seguida uma farmácia, a informação textual salva no campo *pois_neighbor* fica da forma "café_restaurante_farmácia". Se o `POI` café não estiver próximo a nenhum outro POI no intervalo de 200 metros, a informação do campo de *pois_neighbor* será apenas "café".

O Índice de vizinhanças serve tanto como base para criação do texto utilizado na consulta, como também para a criação do Índice de regiões.

4.1.3 Índice de regiões.

O índice de regiões guarda informações de regiões geradas a partir de consultas por *bounding-box* no índice de vizinhanças. Os documentos indexados no índice de regiões possuem os campos de *region*, *neighbor_pois*, *x_distance_m* e *y_distance_m* sendo mapeados respectivamente para *"geo_shape"*, *"text"*, *"integer"* e *"interger"*. No campo *region* são guardados 5 pares de coordenadas que representam os vértices do polígono retangular que delimitam uma região candidata (R_c). Os campos *x_distance_m* e y_distance_m guardam a informação do comprimento em metros da R_c nos eixos X e Y no plano cartesiano (longitude e latitude). E por fim, o campo *neighbor_pois* guarda a informação textual da R_c.

Para realização da indexação das R_c's, são realizadas consultas por *bounding-box* no índice de vizinhanças baseado na ideia de janela deslizante com tamanhos variados. Na Figura 1 é mostrado como funciona, de forma parcial, a janela deslizante. É considerado um polígono retangular ao redor de uma cidade e dentro desse retângulo a janela deslizante percorre todas as R_c's possíveis realizando consultas por meio de *bounding-box*.

Para garantir que informação não seja perdida durante o deslizamento da janela, é definido um parâmetro que determina qual a distância que a janela deve percorrer para realizar uma nova consulta. A distância de deslocamento da janela em algum eixo precisa ser menor que o comprimento da janela nesse mesmo eixo para garantir que haja uma sobreposição da R_c, como mostrado na Figura 2. Essa abordagem visa diminuir a perda de informação de uma R_c que possa existir entre duas R_c's disjuntas. A distancia de deslocamento é equivalente a 20% do comprimento da janela em qualquer um dos eixos.

Como forma de aumentar ainda mais a quantidade de R_c's, também são indexados R_c's com tamanhos de área diferentes. Para esse projeto foram escolhidos os comprimentos de 1 Km, 2 Km e 3Km em cada eixo, gerando R_c's com áreas de 1 Km², 4 Km² e 9 Km². Para

[4] https://postgis.net/
[5] https://www.postgresql.org/
[6] https://github.com/paulomaateus/TCC/tree/main

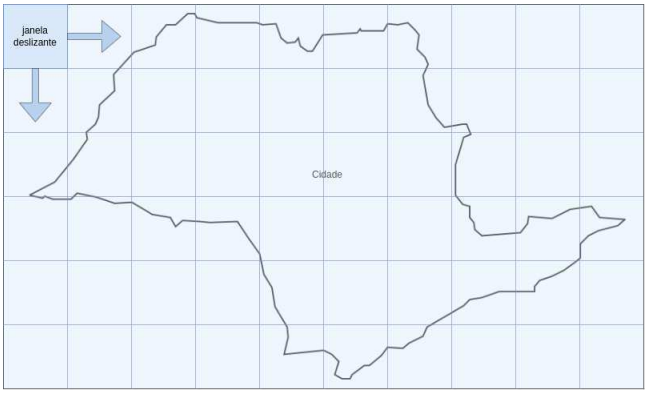

Figura 1: Conceito de janela deslizante.

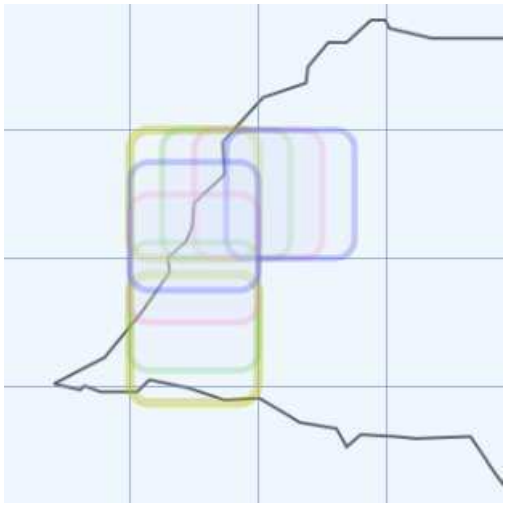

Figura 2: Sobreposição de regiões candidatas.

realização do cálculo desses comprimentos, foi utilizada a biblioteca *geoPy*[4] que implementa o cálculo da distância geodésica proposto por Helmert [5] e adaptado para computadores por Karney [10].

Com os dados das consultas por *bounding-box* no índice de vizinhanças devidamente coletados, é possível usar a informação textual do resultados dessas consultas para criar uma outra informação textual que será armazenada no campo de *"pois_neighbor"* do índice de regiões. Essa informação textual é a união dos textos de todas as vizinhanças encontradas na consulta por *bounding-box*. Supondo que uma consulta por *bounding-box* retornou 3 resultados, sendo seus campos textuais "café_restaurante_farmácia", "shopping_farmacia" e "escola", o algoritmo transforma essas 3 strings em uma só, separando cada uma por espaço: "café_restaurante_farmácia shopping_farmacia escola".

4.2 Procedimento de busca

Para a realização da busca foi criada uma `API` utilizando *fastAPI* com um *endpoint*. Esse *endpoint* recebe no corpo um `JSON` com 3 campos. O `JSON` é descrito na Figura 3. No campo *query* é esperado um objeto

```
{
    "query": {"north": 48.86055924490544,
              "west" : 2.467686650717086,
              "south": 48.83358247099506,
              "east": 2.508638103987778 }

    "regions": [{"north":-7.109221832876315,
              "west": -34.83089598008695,
              "south": -7.11961742983047,
              "east":-34.820419724326314},
             {"north":-7.146853924509486,
              "west" : -34.806822639565326,
              "south": -7.161795742755956,
              "east":-34.790918799173454}]

    "size_m": 1000000
}
```

Figura 3: *Endpoint* de consulta.

com coordenadas norte, sul, leste e oeste de modo que formem um polígono retangular delimitador de uma região de consulta (R_q). No campo *regions* o valor esperado é uma lista de regiões. Essas regiões são no mesmo formato da R_q e podem ser uma ou mais. São regiões de interesse (R_i) do usuário, ou seja, regiões que ele tem interesse em buscar por uma sub-região similar a R_q. Por sua vez, o campo de "size_m" é opcional. Ele define quais os tamanhos das áreas em metros que o usuário quer que sejam considerados no momento da busca. Caso o valor não seja informado, o algoritmo irá considerar todas as R_c's existentes no índice de regiões.

O algoritmo que realiza a busca no *Elasticsearch* é puramente textual. Primeiramente, as áreas de busca são reduzidas às R_i's informadas pelo usuário. Depois disso, a R_q é usada em uma consulta por *bounding-box* no índice de vizinhanças, fazendo com que sejam coletados os dados referentes a vizinhança de `POIs` na R_q. Com esses dados é construída a *string* usada para buscar regiões similares a partir do algoritmo `BM25` no índice de regiões.

5 RESULTADOS E DISCUSSÃO

5.1 Conjunto de dados

Os dados geográficos escolhidos para realização dos experimentos estão disponíveis no OpenStreetMap[7]. O OpenStreetMap é um projeto de mapeamento colaborativo para criar um mapa livre e editável do mundo, inspirado por sites como a *Wikipédia*. Essa base de dados é disponibilizada em dados abertos, sob a licença *Open Data Commons Open Database License* (`ODbL`) pela Fundação *OpenStreetMap*. Com o uso da biblioteca `OSMnx`[8] é possível consultar a base de dados aberta do OpenStreetMap de várias formas. Com a ajuda dessa biblioteca foram buscadas as informações dos POIs de 10 grandes cidades, tendo o resultado consulta descrito na Tabela 1. Os dados retornados nessas consultas por cidade retornam diversas

[7]https://www.openstreetmap.org/
[8]https://osmnx.readthedocs.io/en/stable/

informações, como também vários tipos de objetos geográficos. Neste trabalho os objetos geográficos explorados foram apenas as coordenadas e a categoria dos POIs que estão nos campos de *geometry* e *amenity*, respectivamente.

Tabela 1: Quantidade de POIs por cidade.

Cidades	Quantidade de POIs
Austin - Estados Unidos	5781
Belo Horizonte - Brasil	2365
Blumenau - Brasil	1549
João Pessoa - Brasil	845
Las Vegas - Estados Unidos	1614
Nova Iorque - Estados Unidos	42963
Paris - França	53430
Recife - Brasil	1657
Rio de Janeiro - Brasil	11005
Salvador - Brasil	4023

5.2 Experimentos

Os experimentos conduzidos nesse estudo pretendem demonstrar o potencial de escalabilidade do método de busca similar utilizando BM25. Os resultados são comparativos, utilizando características visuais e intuitivas de uma região. Uma região pode ser considerada similar a outra se elas possuem quantidades aproximadas de cada tipo de POI e/ou quando seus POIs possuem distâncias relativas similares.

5.2.1 *Experimento 1.*

O primeiro experimento tem o objetivo de demonstrar o potencial de escalabilidade da abordagem proposta. Como as consultas textuais são realizadas no Índice de regiões, a Tabela 2 mostra a quantidade de R_c totais, por área. A soma de todas as quantidades de R_cs é igual a 100545, ocupando espaço de 22.89mb de armazenamento. Para realização desse experimento foi considerada toda a base de dados indexada. Um conjunto de cinco R_q foram definidas de forma randômica para realização de diversas consultas sequenciais. A Tabela 3 mostra os tempos alcançados por cada montante de consultas. É possível observar que o crescimento do tempo de consultas é diretamente proporcional ao aumento no número de requisições, no entanto, esse aumento é linear, sendo facilmente escalável para cenários em que há necessidade de curtos tempos de resposta.

5.2.2 *Experimento 2.*

O segundo experimento busca observar a qualidade do ranqueamento da busca. A R_q escolhida para realizar a busca foi selecionada arbitrariamente de acordo com a concentração de POIs que demonstrassem alguma característica visual. Foi delimitada uma região no Rio de Janeiro, como mostrado na Figura 4. A consulta foi realizada delimitando a região da cidade de Salvador como R_i e definindo o tamanho de 1km² para regiões candidatas. A Figura 5 mostra qual foi a região encontrada como resultado. É possível notar de forma intuitiva que o resultado da consulta é no litoral, tal qual a R_q. Na Tabela 4 é mostrada a quantidade dos cinco primeiros POIs que

Tabela 2: Quantidade de R_c por cidade.

Cidades	1 Km²	4 Km²	9 Km²
Austin - Estados Unidos	8356	3429	1857
Belo Horizonte - Brasil	4174	1727	926
Blumenau - Brasil	2033	925	587
João Pessoa - Brasil	1917	876	486
Las Vegas - Estados Unidos	4450	1910	1082
Nova Iorque - Estados Unidos	18623	6044	3078
Paris - França	3042	882	454
Recife - Brasil	2836	1143	640
Rio de Janeiro - Brasil	13833	5511	3091
Salvador - Brasil	4263	1662	929

Tabela 3: Tempo em segundos para consultas.

Quantidade de requisições	Segundos
1	0.01785135
10	0.53966164
100	5.88703536
1000	58.9677422

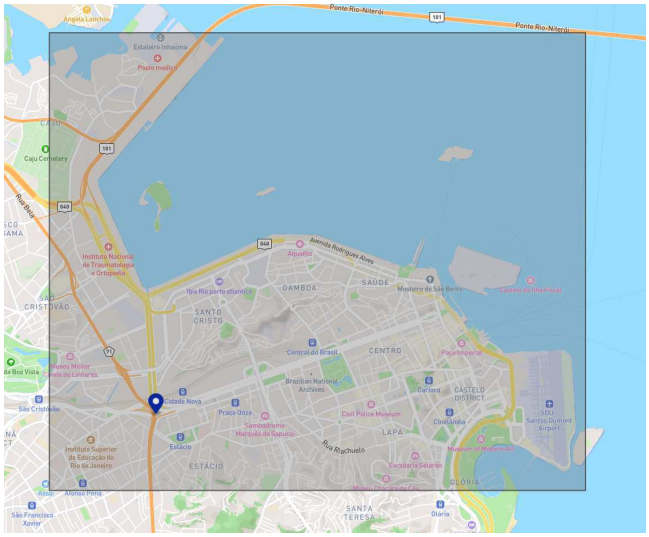

Figura 4: Região de consulta no Rio de Janeiro.

mais aparecem na R_q. Por outro lado, na Tabela 5 é mostrada a quantidade dos cinco primeiros POIs da região resultante.

Uma segunda consulta utilizando a mesma R_q da consulta anterior foi realizada, dessa vez utilizando a cidade de Recife como R_i e o tamanho de 1km² para as R_c. A Figura 6 mostra a região mais similar encontrada. Diferente da última consulta, esse resultado não foi na orla. A Tabela 6 mostra a quantidade dos cinco primeiros POIs nessa região.

Por fim, outra consulta foi realizada utilizando a mesma R_q utilizadas nas consultas anteriores, com a *Ri* na cidade de Nova Iorque com R_c de 9km². A Figura 7 mostra o resultado que teve a maior

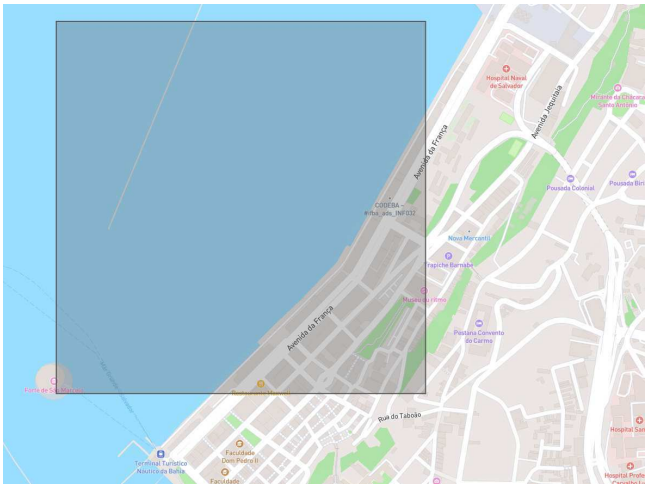

Figura 5: Resultado da consulta em Salvador.

Tabela 4: Quantidade de POIs na R_q.

POI	Quantidade
restaurante	246
fast food	125
banco	122
bicicletário	117
pub	68

Tabela 5: Quantidade de POIs na resposta da consulta.

POI	Quantidade
banco	5
restaurante	4
parada de ônibus	1
posto de gasolina	1
fast foot	1

Tabela 6: Quantidade de POIs na resposta da consulta.

POI	Quantidade
restaurante	3
taxi	2
bicicletário	1
pub	1
clínica	1

pontuação de *ranking*, enquanto que a Tabela 7 mostra a quantidade dos cinco POIs que mais aparecem nessa região.

Considerando que Nova Iorque é uma das cidades com a maior concentração de POIs como mostrado na Tabela 1, o resultado obtido ao procurar por uma região similar a R_q não foi satisfatório. Comparando as Tabelas 4 e 7 é possível notar que nenhum dos cinco pontos que mais aparecem na Tabela 7 aparecem na tabela 4.

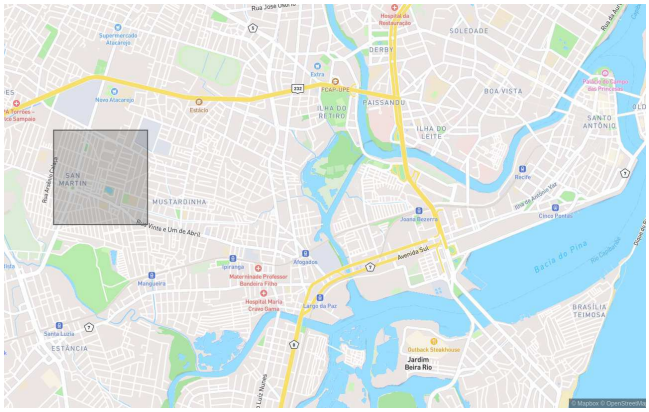

Figura 6: Resultado de consulta em Recife.

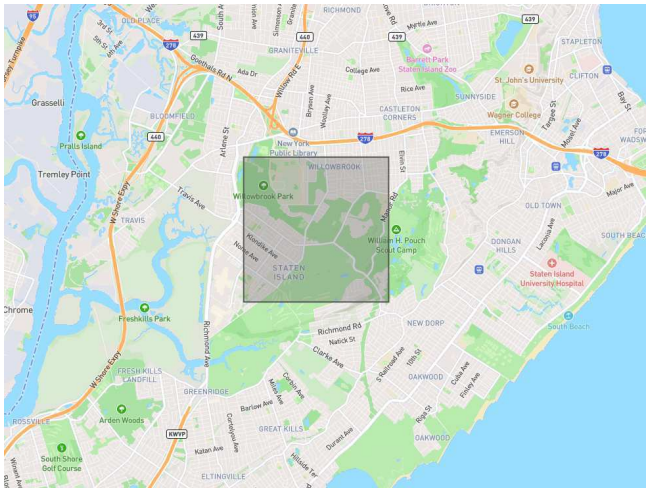

Figura 7: Resultado da consulta em Nova Iorque.

Tabela 7: Quantidade de POIs na resposta da consulta.

POI	Quantidade
bancos de praça	17
templo religioso	2
quadras de basquete	2
escola	1
instalação social	1

6 CONCLUSÕES E TRABALHOS FUTUROS

Nesse trabalho foi implementada uma abordagem escalável de busca por similaridade em mapas utilizando o algoritmo BM25. Os dados utilizados foram indexados em índices no *Elasticsearch* de forma a garantir melhor escalabilidade e eficiência durante as consultas.

Observou-se que a abordagem avaliada não obteve um bom desempenho em termos de similaridade entre regiões. Uma possível razão para isso está na justificativa de que o algoritmo BM25 atribui pontuações mais altas a consultas que possuem termos mais raros.

Dessa forma, as consultas realizadas com termos comuns de `POIs` podem não encontrar regiões realmente relevantes. Além disso, a abordagem proposta não considera outros tipos de objetos a não ser `POI`, fazendo com que os resultados possam não alcançar as métricas de eficiência de outros trabalhos que utilizam modelos de aprendizagem de máquina.

O algoritmo de busca se mostra eficiente e escalável, visto que o *Elasticsearch* permite que sejam criados algoritmos de ranqueamento customizáveis, dessa forma, em trabalhos futuros, pretende-se desenvolver algoritmos de ranqueamento customizados que ofereçam maior cobertura às limitações identificadas, possibilitando uma busca por regiões similares mais eficaz utilizando apenas informação textual.

Por fim, os resultados mostraram que a busca por região geográfica similar ainda é um desafio e, portanto, trata-se de um campo potencialmente prolífico para trabalhos futuros.

AGRADECIMENTOS

Inicialmente agradeço ao meu orientador, professor Cláudio Campelo, pelo suporte, orientação e paciência durante o desenvolvimento dessa pesquisa. Estendo o agradecimento ao professor Maxwell que também fez parte do processo de orientação, junto com o aluno de doutorado Salatiel e o aluno de mestrado Carlos Vinícius. Além desses, quero agradecer também aos demais professores do curso de Ciência da Computação, que contribuíram ativamente na minha formação, e à Universidade Federal de Campina Grande, pela oferta de um curso tão brilhante.

Sou muito grato também às amizades cultivadas ainda no início da graduação, e fortalecidas a cada projeto em grupo. Brener, Diego, Flávio, Iago, Raiany e Cilas. Sem vocês não teria como, obrigado!

Finalmente, obrigado família, que mesmo de longe acredita e torce por mim, me dando apoio sempre que necessário. À minha irmã, Jéssica Daniele, à minha mãe Aurenice, e aos meus Tios Gesualdo e Nalva, devo isso a vocês. Seu suporte e incentivo durante toda minha graduação foi substancial, muito obrigado!

REFERÊNCIAS

[1] Shay Banon. 2021. What is ElasticSearch? https://www.elastic.co/pt/what-is/elasticsearch
[2] S DOMINICH. 1990. *The solid earth: an introduction to global geophysics.* Cambridge University Press.
[3] S DOMINICH. 2008. *The Modern Algebra of Information Retrieval, 1st ed.* Springer.
[4] Kostya Esmukov. 2015. GeoPy Client. https://github.com/geopy/geopy
[5] Friedrich Robert Helmet. 1880. *Die Mathematischen und Physikalischen Theorieen der Höheren Geodäsie, (vol 1).* B. G. Teubner Leipzig.
[6] Martin Raubal Janowicz, Krzysztof and Werner Kuhn. 2011. The semantics of similarity in geographic information retrieval. *Journal of Spatial Information Science.* 2 (2011), 29–57.
[7] et al. Jin, Xiongnan. 2019. Learning region similarity over spatial knowledge graphs with hierarchical types and semantic relations. *Proceedings of the 28th ACM international conference on information and knowledge management.* (2019).
[8] Christopher B. Jones and Ross S. Purves. 2008. Geographical information retrieval. *International Journal of Geographical Information Science* 22.3 (2008), 219–228.
[9] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* (1972).
[10] Charles FF Karney. 2013. Algorithms for geodesics. *Journal of Geodesy* 87.1 (2013), 36–44.
[11] Ray R. Larson. 1995. Geographic information retrieval and spatial browsing. *Geographic information systems and libraries: patrons, maps, and spatial information.* (1995), 10–12.
[12] Kaiqi Zhao Liu, Yiding and Gao Cong. 2018. Efficient similar region search with deep metric learning. *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery data Mining.* (2018).
[13] Hans Peter Luhn. 1957. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development* 1.4 (1957), 309–317.
[14] Stephen E. Robertson and K. Sparck Jones. 1976. Relevance weighting of search terms. *Journal of the American Society for Information science* 27.3 (1976), 129–146.
[15] et al. Sheng, Chang. 2010. Answering top-k similar region queries. *International conference on database systems for advanced applications.* (2010).