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

JOSÉ IGOR DE FARIAS GOMES

EFICIÊNCIA NA BUSCA POR REGIÕES GEOGRÁFICAS SIMILARES: COMPARANDO DIFERENTES MANIPULAÇÕES NOS EMBEDDINGS DE POI E FEIÇÕES GEOGRÁFICAS

CAMPINA GRANDE - PB

2024

JOSÉ IGOR DE FARIAS GOMES

EFICIÊNCIA NA BUSCA POR REGIÕES GEOGRÁFICAS SIMILARES: COMPARANDO DIFERENTES MANIPULAÇÕES NOS EMBEDDINGS DE POI E FEIÇÕES GEOGRÁFICAS

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 em Ciência da Computação.

Orientador : Cláudio Elízio Calazans Campelo

CAMPINA GRANDE - PB

2024

JOSÉ IGOR DE FARIAS GOMES

EFICIÊNCIA NA BUSCA POR REGIÕES GEOGRÁFICAS SIMILARES: COMPARANDO DIFERENTES MANIPULAÇÕES NOS EMBEDDINGS DE POI E FEIÇÕES GEOGRÁFICAS

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 em Ciência da Computação.

BANCA EXAMINADORA:

Cláudio Elízio Calazans Campelo

Orientador – UASC/CEEI/UFCG

Carlos Eduardo Santos Pires

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 15 de maio de 2024.

CAMPINA GRANDE - PB

RESUMO

A representação de regiões geográficas tem sido alvo de pesquisas nos últimos tempos, pois é a peça chave para a realização de diversas tarefas, como a busca por regiões similares. Tal representação, porém, não é tarefa trivial, uma vez que pode envolver inúmeras variáveis no processo. A tendência atual é que essas representações sejam feitas através de vetores de alta dimensão, conhecidos como *embeddings*. Porém, operações de busca por estes costumam ser custosas para a máquina em termos de tempo de processamento e consumo de disco. Neste artigo experimentou-se diferentes manipulações nesses vetores a fim de diminuir o consumo de recursos computacionais no momento da busca sem comprometer significativamente a relevância dos resultados produzidos por ela. Técnicas de redução de dimensionalidade dos vetores e quantização de seus elementos foram executadas, além de comparações entre a busca exata por vizinhos mais próximos e a busca aproximada por estes. Observou-se que a busca aproximada por vizinhos mais próximos reduz o tempo de busca em aproximadamente 42,6%, mantendo uma boa aproximação com os resultados do *baseline*. A técnica de quantização dos *embeddings* apresentou a segunda maior interseção com o *baseline* e reduziu consideravelmente o consumo de disco pelos índices. Técnicas como a redução de dimensionalidades não apresentaram grandes alterações no tempo de busca e tiveram interseções baixíssimas com o *baseline* da pesquisa.

EFFICIENCY IN THE SEARCH FOR SIMILAR GEOGRAPHIC REGIONS: COMPARING DIFFERENT MANIPULATIONS IN POI AND GEOGRAPHIC FEATURES EMBEDDINGS

ABSTRACT

Geographic regions representation has been the main target of several researches in the last years, as it is the key component for performing various tasks, such as searching for similar regions. However, such representation is not a trivial task, as it may involve numerous variables in the process. The current trend is for these representations to be made using high-dimensional vectors, known as embeddings. However, search operations for these tend to be resource-intensive for the machine in terms of processing time and disk usage. In this article we experimented with different kinds of manipulation on these vectors in order to reduce the consumption of computational resources during the search without significantly impacting the relevance of the results produced. Vector dimensionality reduction techniques and the quantization of its elements were performed, in addition to comparing the exact search for nearest neighbors and the approximate search for them. We observed that the approximate search for nearest neighbors reduces the search time by approximately 42,6%, while still maintaining a good approximation with the baseline results. The embeddings quantization technique showed the second-best intersection with the baseline results and significantly reduced disk usage by the indexes. Techniques such as dimensionality reduction did not result in significant changes in the search time and had very low intersection with the research baseline.

Eficiência na busca por regiões geográficas similares: comparando diferentes manipulações nos embeddings de POI e feições geográficas

Trabalho de Conclusão de Curso

José Igor de Farias Gomes (Aluno), Claudio E. C. Campelo (Orientador)
Unidade Acadêmica de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil

RESUMO

A representação de regiões geográficas tem sido alvo de pesquisas nos últimos tempos, pois é a peça chave para a realização de diversas tarefas, como a busca por regiões similares. Tal representação, porém, não é tarefa trivial, uma vez que pode envolver inúmeras variáveis no processo. A tendência atual é que essas representações sejam feitas através de vetores de alta dimensão, conhecidos como *embeddings*. Porém, operações de busca por estes costumam ser custosas para a máquina em termos de tempo de processamento e consumo de disco. Neste artigo experimentou-se diferentes manipulações nesses vetores a fim de diminuir o consumo de recursos computacionais no momento da busca sem comprometer significativamente a relevância dos resultados produzidos por ela. Técnicas de redução de dimensionalidade dos vetores e quantização de seus elementos foram executadas, além de comparações entre a busca exata por vizinhos mais próximos e a busca aproximada por estes. Observou-se que a busca aproximada por vizinhos mais próximos reduz o tempo de busca em aproximadamente 42,6%, mantendo uma boa aproximação com os resultados do *baseline*. A técnica de quantização dos *embeddings* apresentou a segunda maior interseção com o *baseline* e reduziu consideravelmente o consumo de disco pelos índices. Técnicas como a redução de dimensionalidades não apresentaram grandes alterações no tempo de busca e tiveram interseções baixíssimas com o *baseline* da pesquisa.

PALAVRAS-CHAVE

Embeddings; redução de dimensionalidade; quantização; word2vec; KNN; ANN; Pontos de Interesse

1 INTRODUÇÃO

A busca por regiões geográficas similares é de interesse de diversos atores da sociedade, tais como agentes políticos, no planejamento urbano das cidades; turistas, na escolha de seus próximos destinos; ou empresários, na realização de análises estratégicas baseadas em localização.

Para ilustrar, suponha que um empresário do ramo farmacêutico deseja expandir seu negócio para outras cidades. Ele já opera uma farmácia em uma determinada cidade, obtendo um retorno financeiro satisfatório e atribui o sucesso do empreendimento à sua proximidade com grandes hospitais e clínicas. Sendo assim, ele gostaria de replicar esse contexto geográfico ao estabelecer sua nova filial.

Note que a presença de Pontos de Interesse (POI) tem alta influência na escolha do novo local. POI são localizações no mapa que podem ser objeto de interesse ou de utilidade das pessoas. Contudo, além da correlação com outros POI, as feições geográficas também devem ser ponderadas na escolha do novo local. Por exemplo, áreas próximas a rodovias tendem a demandar mais oficinas mecânicas e postos de gasolina.

Embora existam ferramentas amplamente utilizadas para dados geográficos, como o Google Maps[1], elas não proveem funcionalidades para busca por regiões similares. Quer seja o critério de similaridade a proximidade entre outros POI, quer sejam as feições geográficas presentes no ambiente.

Um desafio para se realizar esse tipo de busca, porém, é a representação digital dos dados geográficos (tipos de POI, feições geográficas, etc). Uma alternativa que vem sendo bastante explorada por pesquisas recentes é o uso de vetores de alta dimensão (*embeddings*) como meio de se obter essa representação [1, 12, 14].

Os *embeddings* já são uma tecnologia consolidada no universo dos dados textuais por sua eficácia em capturar relações contextuais e semelhança semântica entre palavras [2, 8, 9], sendo utilizados em grandes motores de busca tradicionais.

Ao levar esse tipo de representação para o contexto dos dados geográficos costuma-se obter bons resultados também [13]. Contudo, não foram encontrados na literatura relatos de trabalhos que avaliem a eficiência do uso desses *embeddings* para a busca de regiões similares no mapa.

Além disso, embora os *embeddings* sejam eficazes em representar diversos tipos de dados, sua utilização em buscas pode ser computacionalmente custosa. Um exemplo disso está na deterioração da eficiência de algoritmos de busca por k-vizinhos mais próximos (KNN, sigla em inglês para *K-Nearest Neighbors*). O KNN é uma técnica de busca amplamente utilizada por sua notória eficácia e simplicidade por trás do conceito. Basicamente, o algoritmo consiste em classificar um elemento como sendo da classe X se a maior parte dos k-vizinhos mais próximos a esse elemento também pertencer a essa classe [11].

[1] https://www.google.com/maps

O KNN, porém, é baseado em uma busca exaustiva em todos os elementos da base de dados. Dessa forma, apesar de ser uma estratégia que traz os resultados ótimos para uma busca vetorial, ela é substancialmente prejudicada, à medida que se aumenta o volume de dados e suas dimensões [5].

Motivada por isso, esta pesquisa tem como objetivo principal avaliar diferentes técnicas de busca por regiões similares utilizando *embeddings* e compará-las quanto ao consumo de recursos computacionais e à qualidade dos resultados recuperados. Para atingir esse objetivo, implementamos diversas manipulações nos *embeddings* e comparamos o tempo de busca e a semelhança entre os resultados retornados.

A primeira otimização efetuada foi a redução de dimensionalidade dos vetores. Essa técnica busca simplificar um *embedding* representando-o em uma quantidade menor de dimensões, mantendo o máximo de informações relevantes possível. Duas estratégias foram escolhidas para realização da redução de dimensionalidade: *Principal Component Analysis* (PCA) e *Feature Agglomeration*.

Uma outra otimização realizada nos *embeddings* diz respeito à quantização dos seus componentes. Esta técnica consiste em discretizar os elementos de um vetor, com o objetivo de reduzir o consumo de memória que essas estruturas de dados requerem.

Para cada uma das técnicas de otimização implementadas, foram comparados os tempos de busca utilizando o algoritmo do KNN e o de buscas aproximadas por k-vizinhos mais próximos (ANN, sigla em inglês para *Aproximate Nearest Neighbors*), que são um conjunto de estratégias alternativas ao KNN que não garantem a eficácia ótima na pesquisa mas que são computacionalmente mais baratas [7].

A hipótese inicial é de que as manipulações que serão aplicadas nos vetores e a mudança nos algoritmos de busca podem diminuir o consumo de recursos computacionais sem degradar significativamente os resultados da busca.

O restante deste artigo está organizado da seguinte forma: a Seção 2 demonstra um apanhado da literatura atual no contexto de representações de regiões geográficas através de *embeddings*; a Seção 3 descreve os métodos que foram utilizados para atingir os objetivos de pesquisa; a Seção 4 apresenta um estudo de caso que justifica a escolha pelo *baseline*; a Seção 5 demonstra os resultados observados durante os experimentos e a Seção 6 apresenta as conclusões que foram inferidas a partir da análise dos resultados e sugere possibilidades futuras de pesquisas a serem investigadas.

2 TRABALHOS RELACIONADOS

Nos últimos anos houve um crescente desenvolvimento de redes neurais especializadas no mapeamento de textos em linguagem natural para vetores no espaço vetorial de várias dimensões. Exemplos disso são os modelos de *Word2Vec* [8] e aqueles baseados em *Transformers*, como o *BERT* [2] e o *Sentence-BERT* [10]. Essas diferentes arquiteturas têm sido usadas de forma eficaz em uma vasta quantidade de tarefas, desde a tradução automática de textos [4] até a representação de informações geoespaciais através de *embeddings*.

Vários são os exemplos do último caso encontrados na literatura. Um deles é o *Urban2Vec* [13], em que os autores propõem um modelo de aprendizagem de *embeddings* baseado no *Word2Vec*, treinado com imagens do *Street View* e dados de POI. O *Urban2Vec* se mostra um modelo superior a diversos *baselines* e capaz de aproximar vizinhanças que são comprovadamente semelhantes com base nas suas características físicas e nas características sociais de sua população.

Utilizando o *BERT* como arquitetura-base de seu modelo, é possível encontrar o *GeoBERT* [3]. Descrito pelos autores como o primeiro modelo pré-treinado em larga escala capaz de aprender a representar informações geoespaciais, o *GeoBERT* utilizou dados de mais de 17 milhões de POI de 30 cidades da China como *corpus* de treinamento. O resultado foi um modelo robusto que alcançou boas métricas em tarefas como a classificação de regiões em área de trabalho ou de residência.

O modelo que serviu de base para a geração da representação vetorial das regiões que foram objeto de estudo dessa pesquisa foi proposto por Silva *et al.* [12]. Os autores propuseram o *GeoContext2Vec*, abordagem que combina dados de POI com dados de feições geográficas para representar tipos de POI. A técnica mostrou resultados promissores ao superar modelos estado-da-arte, comprovando que as feições geográficas da região são boas características a serem consideradas na representação de POI.

Os trabalhos citados acima se dedicam a representar regiões geográficas de diversas formas e a partir de variados tipos de dados. Contudo, não foram encontrados esforços na literatura que implementem diferentes manipulações de simplificação nesses *embeddings* e atestem a sua eficiência em conjunto com sua eficácia em trazer bons resultados para a busca por regiões geográficas nem por nenhum outro tipo de dado que seja possível de ser representado através de tais vetores.

3 METODOLOGIA

Esta seção apresenta as principais decisões metodológicas para condução da pesquisa: a Subseção 3.1 faz um descritivo dos dados que foram selecionados para a execução da pesquisa e do modelo-base usado para representar as regiões através de *embeddings*; a Subseção 3.2 explica como se deu a representação de cada região do mapa a partir de seus POI e feições geográficas; a Subseção 3.3 narra a definição das métricas do modelo usado como *baseline*; a Subseção 3.4 elucida as manipulações feitas nos *embeddings* originais das regiões; a Subseção 3.5 detalha o algoritmo de busca e ranqueamento utilizado na pesquisa por regiões similares, por fim, a Subseção 3.6 elenca as tecnologias utilizadas durante o desenvolvimento da pesquisa.

3.1 Dados e Modelo-Base

A geração dos vetores de alta dimensão foi feita a partir do GeoContext2Vec [12], uma abordagem que utiliza dados de POI e feições geográficas contextuais para gerar *embeddings* para os tipos de POI. Uma outra vantagem desta técnica é que também é possível se obter *embeddings* das feições geográficas de uma determinada área no mapa. O modelo baseado no GeoContext2Vec foi treinado a partir dos dados do Yelp Challenge[2] (versão de fevereiro de 2021) e dos dados sobre feições geográficas coletados do OpenStreetMap[3] (OSM), uma iniciativa open-source e colaborativa que mapeia informações geoespaciais de todos os lugares do mundo.

[2]https://www.yelp.com/dataset
[3]https://www.openstreetmap.org/

	Coordenadas (Latitude, Longitude)	
Cidade	**Superior Esquerda**	**Inferior Direita**
Austin	30.4214, -97.8204	30.1547, -97.5935
Portland	45.6409, -122.7980	45.4360, -122.4481
Atlanta	33.8892, -84.5518	33.6443, -84.2675
Orlando	28.6451, -81.5055	28.3388, -81.2281
Vancouver	49.3596, -123.3816	49.1554, -122.8666

Tabela 1: Coordenadas geográficas das áreas selecionadas em cada cidade para a pesquisa.

O dataset do Yelp contém informações sobre 160.585 POI em 836 cidades dos Estados Unidos da América (EUA) e Canadá. Para o contexto do treinamento do modelo, apenas dados da cidade de Austin (Texas - EUA) foram usados, por ser a cidade com maior presença de POI no *dataset*.

Para esta pesquisa selecionamos as cinco cidades com maior quantidade de POI registradas no dataset do Yelp, são elas: Austin (Texas), Portland (Oregon), Atlanta (Geórgia), Orlando (Flórida) e Vancouver (British Columbia), sendo as quatro primeiras localizadas nos Estados Unidos e a última localizada no Canadá. As coordenadas da área delimitada em cada cidade para condução dos experimentos estão descritas na Tabela 1.

3.2 Representação das Regiões

Cada uma das áreas escolhidas para fazer parte da pesquisa foi subdividida em regiões, que foram o objeto-alvo das buscas e dos experimentos realizados na pesquisa. A subdivisão ocorreu da seguinte forma: definiu-se que a área total de cada região seria igual a 0.4km², por ser um tamanho razoável e que abrange um espaço que pode conter uma quantidade significativa de feições geográficas e POI. A partir da primeira região definida no canto superior esquerdo do mapa, as demais foram obtidas com o deslizamento de uma janela através da área. Essa janela deslizante suaviza a distância entre uma região e a seguinte e faz com que haja interseção entre diferentes regiões. Esse conceito de janela deslizante auxilia numa quantidade maior de regiões resultantes e ajuda para que a busca seja mais precisa. Essa janela de deslizamento tem tamanho 8, ou seja, divide-se o lado da região por 8 e chega-se ao tamanho do passo que será dado entre uma região e outra. O deslizamento de janela ocorreu nas direções vertical e horizontal, como pode ser visto na Figura 1. A quantidade de regiões resultantes dessa divisão está sumarizada na Tabela 2.

Para cada região definida no mapa, gerou-se a sua representação, utilizando *embeddings* produzidos com o GeoContext2Vec. Os *embeddings* do GeoContext2Vec são gerados a partir de tipos de POI e feições geográficas. Essas feições são classificadas em quatro tipos (linhas, rodovias, polígonos e pontos) e para cada um dos tipos existe um modelo especializado em sua representação vetorial. Cada um desses modelos especializados é capaz de prover a representação das feições daquele tipo e também dos tipos de POI que estão associados àquele tipo de feição. Por exemplo: o modelo de rodovias pode gerar *embeddings* para todas feições do tipo rodovia, mas também para qualquer tipo de POI a partir das informações de rodovias usualmente associadas àquele POI.

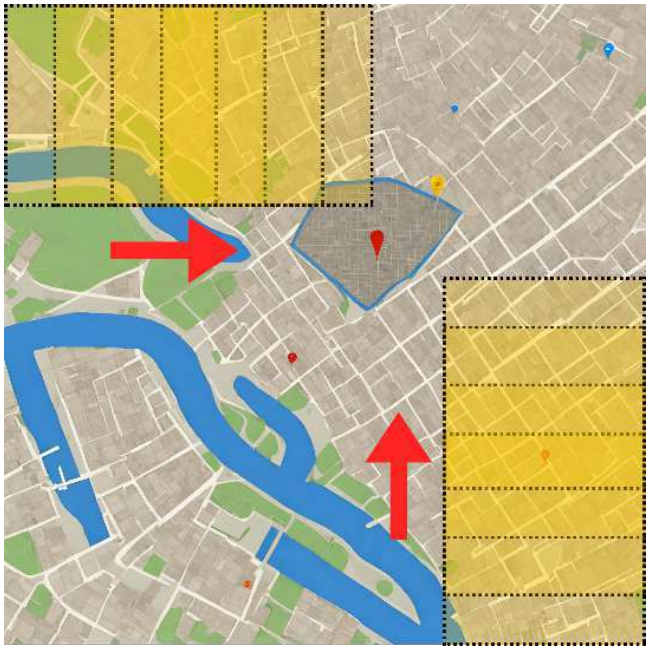

Figura 1: Exemplo de deslizamento vertical e horizontal de janela no mapa

A Figura 2 ilustra o processo para representar cada uma das regiões a partir dos *embeddings* obtidos através de cada um dos modelos especializados. As linhas pretas correspondem aos *embeddings* de feições geográficas e as vermelhas tracejadas são referentes aos dados de POI. Cada região do mapa foi representada por, no máximo, 8 *embeddings*, 4 de feições e 4 de tipos de POI, gerados pelos modelos especializados do GeoContext2Vec. As regiões que não possuíam dados de um dos 4 tipos de feições não tiveram os *embeddings* para aquele tipo armazenados e aquelas que não possuíam POI em seu interior não tiveram nenhum dos 4 *embeddings* de POI armazenados. Dessa forma, algumas regiões tiveram menos de 8 *embeddings* representativos.

A equação 1 descreve como foram calculados os *embeddings* de feições geográficas:

$$vetorFeicao_{type,i} = \frac{\sum_{j=1}^{N_f} model_{type}(feature_{i,j})}{N_f} \quad (1)$$

em que $vetorFeicao_{type,i}$ representa o vetor gerado para a i-ésima região pelo modelo especializado na feição geográfica do tipo $type$; $model_{type}(feature_{i,j})$, representa o vetor obtido pelo modelo especializado na feição geográfica do tipo $type$ para a j-ésima feição da i-ésima região geográfica; e N_f representa o número total de features do tipo $type$ para a i-ésima região.

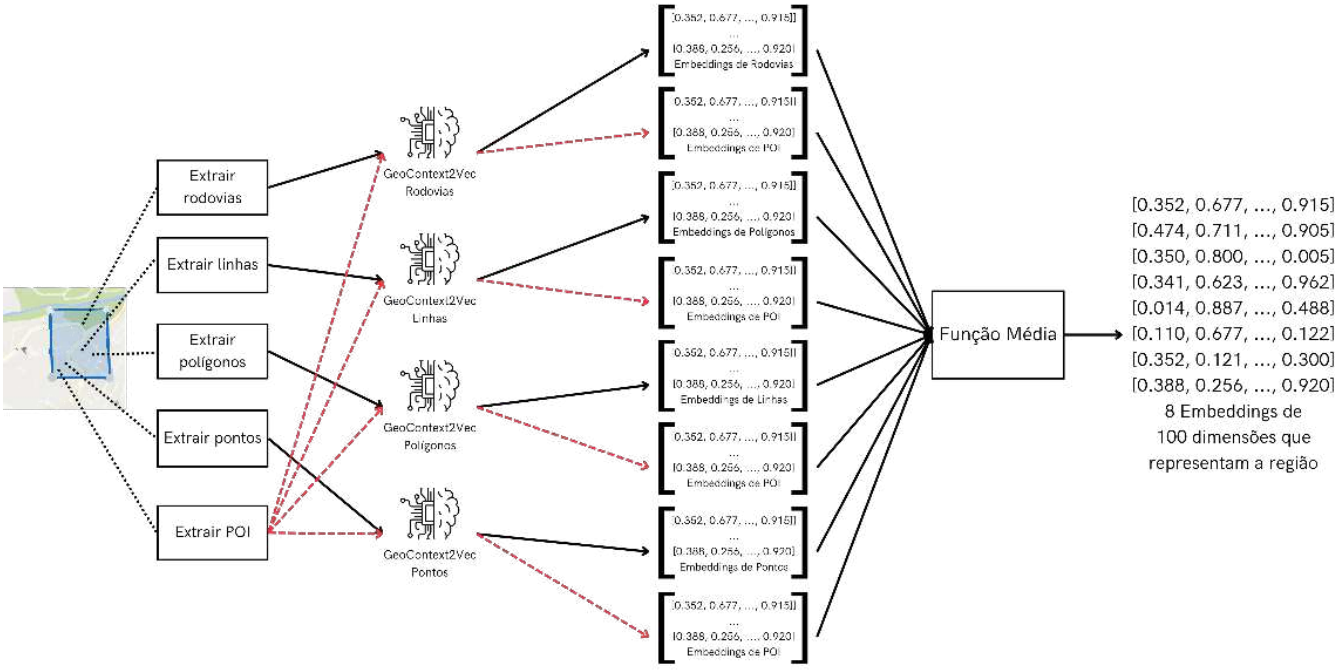

Figura 2: Geração dos *embeddings* das regiões

Cidade	Quantidade de Regiões
Austin	97969
Portland	97969
Atlanta	97969
Orlando	136161
Vancouver	124609
Total	**554677**

Tabela 2: Total de regiões em que cada cidade foi dividida

De maneira análoga, a equação 2 resume o cálculo dos *embeddings* dos tipos de POI para cada uma das regiões geográficas:

$$vetorPOI_{type,i} = \frac{\sum_{k=1}^{N_P} model_{type}(POI_{i,k})}{N_P} \quad (2)$$

em que $vetorPOI_{type,i}$ representa o vetor de POI gerado para a i-ésima região pelo modelo especializado na feição geográfica do tipo $type$; $model_{type}(POI_{i,j})$, representa o vetor gerado pelo modelo especializado na feição geográfica do tipo $type$ para o k-ésimo POI da i-ésima região geográfica; e N_P diz respeito ao número total de POI presentes na i-ésima região.

Após gerar os *embeddings* de todas regiões de cada uma das cidades, foi populado um índice no Elasticsearch[4] com todos os vetores dessas regiões. O Elasticsearch é uma ferramenta especializada em busca frequentemente utilizada por sua eficiência ao lidar com grandes volumes de dados e pela vasta quantidade de funcionalidades de que dispõe. Cada vetor foi associado a seu tipo, de acordo com o modelo especializado que o gerou e o dado que ele representa. Ao fim, obeteve-se um índice com 2.665.575 vetores.

3.3 *Baseline*

Foram selecionadas aleatoriamente dez regiões de cada cidade para compor o conjunto de validação em que seriam realizadas as buscas dos experimentos. Este conjunto serviu como base para que as métricas de eficiência e eficácia dos experimentos fossem aferidas. Para cada uma dessas regiões, foi realizada a busca pelas regiões mais similares através do algoritmo do KNN no Elasticsearch. Antes da realização da busca, foi considerada a restrição de que esta não fosse realizada dentre as regiões pertencentes à mesma cidade da região âncora, para evitar que regiões com interseções em suas áreas dominassem as primeiras posições do *ranking* da busca.

Os resultados retornados por essa busca foram estabelecidos como *baseline* da pesquisa, por serem os *embeddings* mais robustos disponíveis e pelo fato da busca ter utilizado o algoritmo mais completo possível (o KNN), em que há a garantia de se encontrar os vizinhos mais próximos dentre toda a base de dados. A partir do *baseline* foram propostas soluções de melhoria no desempenho da

[4]https://www.elastic.co/elasticsearch

busca através de otimizações nos dados e variações no algoritmo de buscas.

3.4 Manipulações nos *Embeddings*

Diferentes estratégias de manipulação foram implementadas nos *embeddings* a fim de testar o desempenho da busca em cada um dos cenários. Foram empregadas técnicas de redução da dimensionalidade dos *embeddings* e quantização dos seus elementos.

As técnicas de redução de dimensionalidade de *embeddings* têm o objetivo de alcançar uma representação dos vetores de alta dimensão em um espaço latente menor com o mínimo de perda de informação. Em particular, foram comparadas as técnicas de *Principal Component Analysis* (PCA) e *Feature Agglomeration*.

As técnicas acima foram aplicadas em cada um dos *embeddings* que representam uma região para gerar vetores de 90, 70 e 50 dimensões. As implementações utilizadas foram aquelas disponibilizadas pela biblioteca scikit-learn[5] com seus hiperparâmetros padrão.

Para a quantização dos elementos dos vetores, foi utilizada uma técnica de mapeamento dos valores contínuos dos *embeddings* para inteiros de 8 bits (entre −128 e 127).

3.5 Algoritmo de Busca e Ranqueamento

Quanto aos algoritmos de busca, foi comparada a eficiência do KNN em relação ao ANN. O algoritmo de ANN escolhido para esta pesquisa foi o *Hierarchical Navigable Small World graphs* (HNSW) [7] implementado de forma nativa pela API do Elasticsearch.

Para cada região utilizada como entrada da busca, foram selecionados seus 8 *embeddings* representativos na base de dados. Buscou-se assegurar que todas as regiões do conjunto de validação possuíam todos os 8 tipos de *embeddings*, ou seja, são regiões que possuem em seu interior ao menos um POI e ao menos um exemplo de cada um dos tipos de feições geográficas.

Para cada um dos 8 *embeddings* de cada região, foi executada a busca pelos seus 100 *embeddings* mais similares de mesmo tipo do vetor de entrada usando o KNN ou ANN. Ou seja, a partir do vetor de rodovias foram recuperados os 100 *embeddings* de rodovias mais similares a ele; a partir do vetor de polígonos foram recuperados os 100 mais similares a ele; e assim por diante para cada um dos tipos de *embedding* daquela região. Ao fim do processo, houve um total de 800 regiões candidatas a serem definidas como similares à região de entrada.

Após essas buscas, as 800 regiões retornadas foram ranqueadas a fim de se definir dentre todas elas quais eram mais e menos similares em relação à região de entrada, pois cada uma das buscas anteriores aconteceu de maneira independente e neste momento seus resultados deviam ser unificados. Esse ranqueamento ocorreu do seguinte modo: primeiramente, foram removidas as regiões duplicadas, ou seja, aquelas que, na etapa anterior, foram retornadas em mais de uma consulta por tipo de vetor.

Para fins de exemplificação, suponha que em determinada busca, após essa remoção de duplicatas, obtiveram-se 600 regiões. Cada uma delas foi então comparada com a região de entrada da busca através da similaridade do cosseno entre cada um de seus *embeddings* de mesmo tipo. Por exemplo, considere a região A como sendo a entrada da busca e a região B como sendo uma das 600 regiões obtidas na fase anterior. Os *embeddings* de rodovia da região A foram comparados aos *embeddings* de rodovia da região B através da similaridade do cosseno, gerando um *score* de 0 a 1. O mesmo aconteceu para as *embeddings* de linhas, pontos, polígonos e POI. Nos casos em que a região B não possuía *embeddings* de certo tipo por ausência de dados, o *score* foi definido para um valor padrão de 0.5. Este valor foi escolhido pois, nesses casos, não era possível apontar que tal par de regiões era similar ou dissimilar, portanto o valor de 0.5 passa a semântica de uma similaridade neutra entre as regiões. Por fim, obteve-se um total de 8 *scores* de similaridade entre os *embeddings* da região A e da região B. Foi então calculada a média harmônica entre esses 8 *scores* para que fosse definida a similaridade final entre as regiões A e B.

Após a realização desse cálculo para cada uma das regiões retornadas na busca, elas foram ordenadas em ordem decrescente de similaridade e retornou-se as 200 primeiras como resultado final da busca.

Os resultados das consultas, por sua vez, foram comparados em termos de sua interseção com os mais similares obtidos utilizando o *baseline* original e do tempo total de execução da busca. Para diminuir a influência de fatores randômicos no tempo aferido, cada consulta foi realizada 5 vezes e aferiu-se a média do tempo para cada consulta e o desvio padrão observado.

3.6 Tecnologias Utilizadas

A linguagem de programação *Python* foi utilizada durante o desenvolvimento da pesquisa por possuir um vasto arsenal de ferramentas para manipulação de dados, dentre elas algumas bibliotecas como numpy[6] e pandas[7], que serviram para leitura de *datasets*, tratamento e operações com vetores, etc.

4 ESTUDO DE CASO

Como mencionado na Seção 3.3, para avaliar o *trade-off* entre a simplificação dos *embeddings* e eficácia de busca, definimos um *baseline* ótimo para comparação. Para isto, assumimos que a busca com KNN pelos embeddings originais de 100 dimensões oferece a melhor base de comparação, uma vez que são os embeddings mais robustos que temos, originalmente produzidos pelo GeoContext2Vec.

Para atestar que o modelo *baseline* realmente traz resultados coerentes na sua busca por similares, realizou-se um estudo de caso em algumas amostras de regiões. Foi analisado se os resultados fornecidos por este modelo faziam sentido com a realidade, ou seja, se as regiões que o modelo aponta como similares de fato o são.

O exemplo abordado nesta Seção trata da busca realizada a partir de uma região na cidade de Vancouver. A região tem como latitude e longitude superiores esquerdas 49.25979502816142 e −123.02714020875631 e inferiores direitas 49.255258438532366 e −123.01572916266714. A região retornada como mais similar a esta após realizada a busca fica em Portland e tem como latitude e longitude superiores esquerdas 45.455921680792684 e −122.7848748530676 e inferiores direitas 45.45080117368662 e −122.77615350338439. É possível visualizar ambas na Figura 3.

Percebe-se através das figuras que há diversos elementos em comum que indicam que as regiões são, de fato, semelhantes. Em

[5] https://scikit-learn.org/stable/

[6] https://numpy.org/

[7] https://pandas.pydata.org/

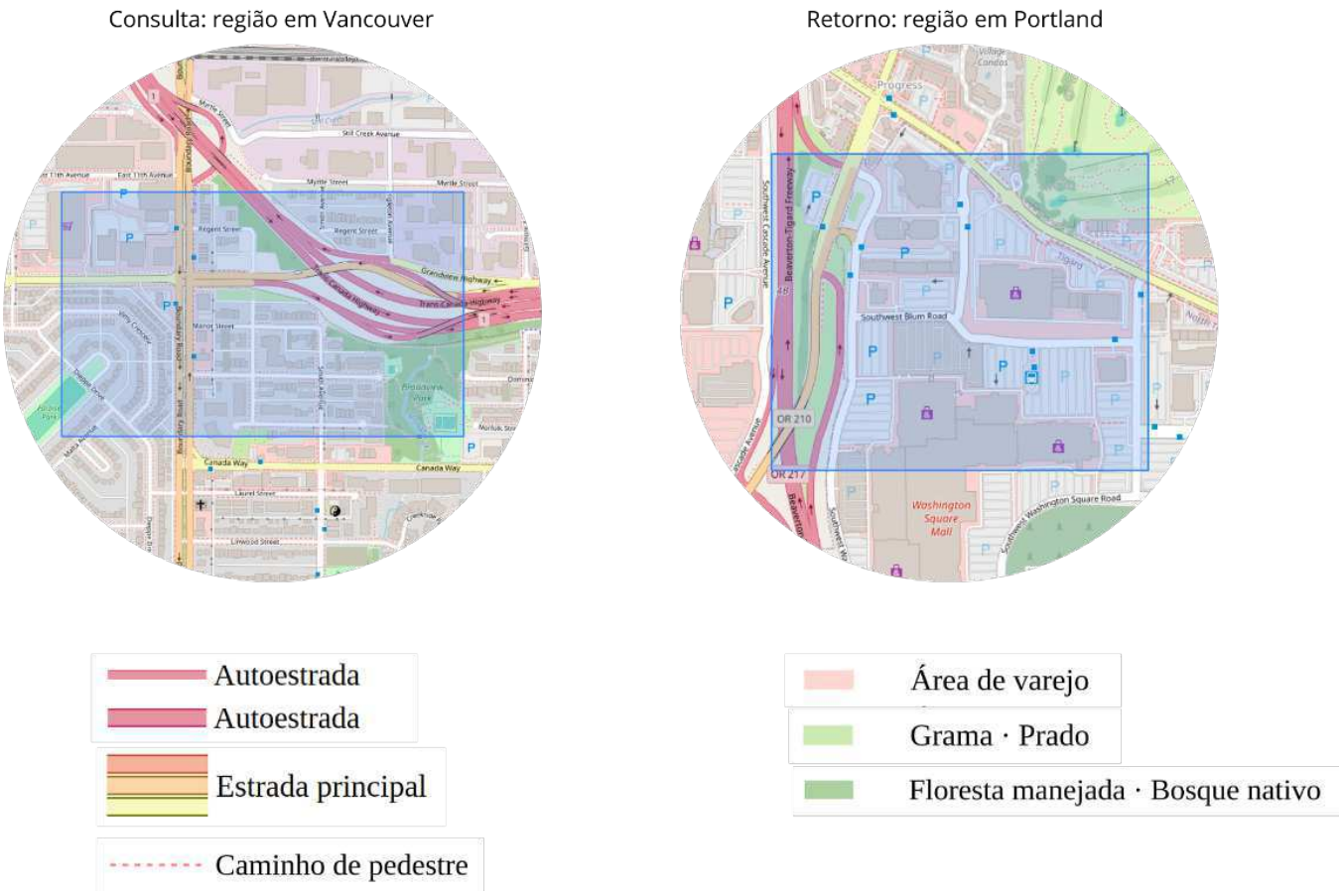

Figura 3: Comparação entre regiões similares em Vancouver e Portland

ambas as regiões há a presença de autoestradas marcadas em vermelho e estradas principais, em laranja e amarelo. Além disso, é possível perceber áreas verdes com a presença de gramados (marcados em verde-claro) e bosques nativos (marcados em verde-escuro) em ambas as imagens. Outros elementos que devem ter influenciado na semelhança apontada pelo algoritmo são os vias que servem de caminho para pedestres (identificadas por linhas tracejadas vermelhas), áreas de comércio varejista e a presença de estacionamentos para bicicletas (marcados por quadrados azuis) e para automóveis (marcados pela letra P). A legenda completa para o mapa pode ser consultada no *site* do OpenStreetMap[8].

Com base nas evidências de que o modelo *baseline* trazia resultados coerentes providas por este estudo de caso, seguiu-se com a pesquisa.

5 RESULTADOS E DISCUSSÃO

A implementação das técnicas de redução de dimensionalidade resultou em uma diminuição do espaço ocupado em disco pelos índices no Elasticsearch. As Tabelas 3, 4 e 5 comparam o tamanho dos índices resultantes de cada uma das abordagens com o tamanho dos índices construídos a partir dos *embeddings* originais.

[8]https://www.openstreetmap.org/key

	Original	**PCA**		
Dimensões	100	90	70	50
Tamanho do Índice	4.4GB	4.1GB	3.1GB	2.2GB

Tabela 3: Comparação entre o tamanho do índice original e dos índices após aplicação do PCA nos *embeddings*.

A diminuição mais expressiva aconteceu com a aplicação da quantização, cujos *embeddings* de mesma quantidade de dimensões que a abordagem original ocuparam um espaço aproximadamente 70,4% menor do que o índice original.

As técnicas de redução de dimensionalidade resultaram em índices de tamanhos semelhantes. A redução no espaço em disco ocupado pelos índices após a aplicação dessa otimização variou entre 6,8% e 50%. A diminuição observada é aproximadamente igual à diminuição no tamanho dos índices em termos de proporção.

Nas Tabelas 6, 7 e 8, é possível comparar os tempos de busca aferidos após as diferentes manipulações nos dados. Elas fazem um comparativo entre os tempos da busca com KNN usando os

	Original	*Feature Agglomeration*		
Dimensões	100	90	70	50
Tamanho do *Índice*	4.4GB	3.9GB	3GB	2.2GB

Tabela 4: Comparação entre o tamanho do índice original e dos índices após aplicação do *Feature Agglomeration* nos *embeddings*.

	Original	Quantização
Dimensões	100	100
Tamanho do *Índice*	4.4GB	1.3GB

Tabela 5: Comparação entre o tamanho do índice original e dos índices após aplicação da Quantização nos *embeddings*.

embeddings originais e aqueles alcançados a partir do uso de *embeddings* de dimensionalidade reduzida através da estratégia de PCA. Nestas tabelas também se comparam os dois algoritmos usados para a busca dos *embeddings*: o KNN e o ANN.

Percebe-se que os tempos de busca ao aplicar o PCA não sofreram variações significativas ao se utilizar o KNN como algoritmo de busca, estando todos os tempos aferidos dentro da margem do desvio padrão. Ao utilizar o ANN, há uma certa diminuição no tempo de busca médio observado, mas ainda há interseção entre esses valores e os do *baseline* considerando-se a margem do desvio padrão.

No caso da redução de dimensionalidade com *Feature Agglomeration*, houve uma redução no tempo de busca para a busca com KNN com *embeddings* de 50 dimensões. Entretanto, para os outros cenários de busca com KNN a variação foi praticamente nula em relação ao *baseline* e a margem do seu desvio padrão. Enquanto isso, para a busca feita com ANN, houve inclusive um aumento no tempo de busca médio, havendo interseção apenas entre o tempo de busca para os *embeddings* de 50 dimensões e o *baseline*, ao se considerar a margem do desvio padrão.

Por sua vez, a Tabela 8 faz a comparação entre o tempo de busca pelos *embeddings* quantizados e o *baseline*. Mais uma vez não observa-se mudança significativa entre os tempos de busca após a aplicação da quantização e o *baseline*. Apesar de haver uma queda no tempo médio ao se utilizar de ambas as técnicas de busca, há interseção entre a margem do desvio padrão nas duas abordagens.

Uma hipótese explicativa para não haver tanta diferença nos tempos de busca antes e depois da aplicação das reduções de dimensionalidade é a de que, em números absolutos, a diminuição da quantidade de dimensões tenha sido muito pequena. Dessa forma, a busca pelos *embeddings* não sofre daquilo que a literatura define como "praga da dimensionalidade", ou seja, a crescente necessidade por recursos computacionais à medida que os *embeddings* têm uma quantidade extremamente alta de dimensões [6].

Por fim, a Tabela 9 compara o tempo de busca pelos mesmos *embeddings* originais de 100 dimensões através de dois algoritmos de busca diferentes: o KNN e o ANN. Percebe-se que a simples mudança

	Algoritmo de Busca	Dimensões	Tempo Médio (em s)	Desvio Padrão
PCA	**KNN**	**90**	0.197776	0.030471
		70	0.172140	0.010504
		50	0.150741	0.019554
	ANN	**90**	0.135739	0.027985
		70	0.133145	0.030405
		50	0.138603	0.029780
Baseline	**KNN**	**100**	0.171767	0.019032

Tabela 6: Comparação entre o tempo de busca com os *embeddings* reduzidos através de PCA e com os *embeddings* originais.

	Algoritmo de Busca	Dimensões	Tempo Médio (em s)	Desvio Padrão
Feat. Agg.	**KNN**	**90**	0.186680	0.014094
		70	0.143829	0.010361
		50	0.116841	0.007826
	ANN	**90**	0.277182	0.036526
		70	0.282050	0.021683
		50	0.208434	0.027667
Baseline	**KNN**	**100**	0.171767	0.019032

Tabela 7: Comparação entre o tempo de busca com os *embeddings* reduzidos através de Feature Agglomeration e com os *embeddings* originais.

	Algoritmo de Busca	Dimensões	Tempo Médio (em s)	Desvio Padrão
Quantização	**KNN**	**100**	0.158186	0.011650
	ANN	**100**	0.129520	0.026959
Baseline	**KNN**	**100**	0.171767	0.019032

Tabela 8: Comparação entre o tempo de busca com os *embeddings* quantizados e com os *embeddings* originais.

	Algoritmo de Busca	Dimensões	Tempo Médio (em s)	Desvio Padrão
Baseline	**KNN**	**100**	0.171767	0.019032
	ANN	**100**	0.098633	0.033683

Tabela 9: Comparação entre o tempo de busca para os *embeddings* originais a partir do uso dos algoritmos de KNN e ANN.

Embeddings Utilizados	Algoritmo de Busca	Dimensões	Interseção Média (em %)
Quantizados	KNN	100	25,0
	ANN	100	21,9
Originais	ANN	100	93,5

Tabela 10: Interseção entre o resultado das buscas com quantização e ANN e o *baseline*

no algoritmo de busca já é suficiente para provocar uma redução significativa no tempo, que tem uma queda de aproximadamente 42,6%.

Finalmente, foi feita a comparação entre os resultados retornados pelas buscas por *embeddings* após as manipulações para definir o quão semelhantes elas foram em relação ao *baseline* da pesquisa.

No caso das reduções de dimensionalidade, com qualquer das técnicas, a interseção entre suas regiões retornadas e aquelas retornadas pelo *baseline* foi baixissima, não chegando nem a 0,01 % em média.

Já no caso da quantização e da troca de algoritmo de busca de KNN para ANN, os resultados das interseções estão descritos na Tabela 10. Percebe-se uma semelhança expressiva entre os resultados da busca com ANN e KNN, apontando que a troca de algoritmo seja uma estratégia útil na melhora do desempenho da busca. No caso da quantização, percebe-se uma certa interseção que não chega ser suficiente para se afirmar que seus resultados são tão bons quanto os do *baseline*.

6 CONCLUSÕES E TRABALHOS FUTUROS

Inicialmente, formulou-se a hipótese de que as técnicas de redução de dimensionalidade e a quantização dos elementos dos *embeddings*, bem como a troca no algoritmo de busca de KNN para ANN diminuiriam o consumo de recursos computacionais necessários para a busca por regiões geográficas sem afetar significativamente a relevância dos resultados obtidos. Essa hipótese foi parcialmente rejeitada. De fato, após serem aplicadas as técnicas de redução de dimensionalidade e quantização, o consumo de disco requerido pelos índices dos *embeddings* foi reduzido. No entanto, observou-se uma baixa interseção entre os resultados obtidos com os *embeddings* modificados e aqueles produzidos com os *embeddings* originais do *baseline*. Além disso, o tempo de busca pelas regiões não foi consideravelmente afetado ao se trabalhar com os *embeddings* modificados.

Por sua vez, a troca no algoritmo de busca, como esperado, ocasionou uma redução no tempo médio de procura pelos *embeddings* ao se tratar dos *embeddings* originais e teve uma boa interseção com os resultados da busca pelos *embeddings* originais.

Em trabalhos futuros, pretende-se avaliar a aplicação dessas mesmas técnicas de redução de dimensionalidade e quantização em *embeddings* de dimensionalidade maior, a fim de que seja possível observar uma mitigação da "praga da dimensionalidade" e, consequentemente, uma diminuição no tempo de busca pelos *embeddings*.

AGRADECIMENTOS

Esta pesquisa - e toda minha graduação - só teve sucesso porque muita gente boa esteve presente em minha jornada e a alguns dos quais eu agradeço aqui:

A Dona Cristina e Seu Abraão: não houve um único momento desses 5 anos que eu não tenha me sentido a pessoa mais sortuda do mundo por ter vocês como pais. Prometo que recompensarei cada esforço que vocês investiram.

A Tatá, cuja cumplicidade inúmeras vezes foi meu sustento e cuja casa foi minha também sempre que eu sentisse necessidade.

Aos meus colegas de laboratório que me ensinaram tanto e fizeram parte dos meus dias durante quase toda a graduação. Em especial à Salatiel Dantas e Eduardo Cavalcanti: as sugestões e orientações de vocês foram essenciais para que eu conseguisse desenvolver este trabalho.

Ao professor Cláudio Campelo, que, com seu arcabouço técnico incomparável, me orientou com a presença e a cobrança necessárias e me deu a oportunidade de aprender tanto no LACINA.

A Bia, Drica, Emilly, Helen, Henrique, Matias, Natália, Neto, Ricardo e Tuza: uma vez nos questionaram "quem tem tantos amigos?" e, sinceramente, eu não sei se é possível manter tantas amizades mas sei que guardo boas memórias com cada um de vocês e agradeço por tê-los encontrado na minha graduação.

A Hemilly, Gustavo, Isabella, Prissila e Almir: eu não conseguiria ter apenas amigos de computação. Com vocês eu me senti a vontade para rir do humor mais quebrado e para desabafar sobre aquilo que mais me angustiava.

A Taize Kobayashi, que tem me ajudado a me tornar um ser humano maduro emocionalmente e a buscar o meu próprio desejo.

A todos os amigos que fiz no mundo da música, em especial a Lemuel Guerra: sua presença me inspira todos os dias a ser um questionador e um músico melhor.

A Deus, que agiu através de toda essa gente massa.

REFERÊNCIAS

[1] Alessandro Crivellari and Euro Beinat. 2019. From Motion Activity to Geo-Embeddings: Generating and Exploring Vector Representations of Locations, Traces and Visitors through Large-Scale Mobility Data. *ISPRS International Journal of Geo-Information* 8, 3 (Mar 2019), 134. https://doi.org/10.3390/ijgi8030134

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/V1/N19-1423

[3] Yunfan Gao, Yun Xiong, Siqi Wang, and Haofen Wang. 2022. GeoBERT: Pre-Training Geospatial Representation Learning on Point-of-Interest. *Applied Sciences* 12, 24 (2022). https://doi.org/10.3390/app122412942

[4] Zhiyu Guo and Minh Le Nguyen. 2020. Document-Level Neural Machine Translation Using BERT as Context Encoder. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop*, Boaz Shmueli and Yin Jou Huang (Eds.). Association for Computational Linguistics, Suzhou, China, 101–107. https://aclanthology.org/2020.aacl-srw.15

[5] Yoonho Hwang, Bohyung Han, and Hee-Kap Ahn. 2012. A fast nearest neighbor search algorithm by nonlinear embedding. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 3053–3060. https://doi.org/10.1109/CVPR.2012.6248036

[6] Xiaoyu Jiang, Xiangyin Kong, and Zhiqiang Ge. 2023. Augmented Industrial Data-Driven Modeling Under the Curse of Dimensionality. *IEEE/CAA Journal of Automatica Sinica* 10, 6 (2023), 1445–1461. https://doi.org/10.1109/JAS.2023.123396

[7] Yu A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (apr 2020), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473

[8] Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of Workshop at ICLR* 2013 (01 2013).

[9] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Marilyn Walker, Heng Ji, and Amanda Stent (Eds.). Association for Computational Linguistics, New Orleans, Louisiana, 2227–2237. https://doi.org/10.18653/v1/N18-1202

[10] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 3982–3992. https://doi.org/10.18653/v1/D19-1410

[11] R. Rithesh. 2017. SVM-KNN: A Novel Approach to Classification Based on SVM and KNN. *International Research Journal of Computer Science* 4 (08 2017). https://doi.org/10.26562/IRJCS.2017.AUCS10088

[12] Salatiel Dantas Silva, Claudio Elízio Calazans Campelo, and Maxwell Guimarães De Oliveira. 2023. POI Types Characterization Based on Geographic Feature Embeddings. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23)*. Association for Computing Machinery, New York, NY, USA, 507–514. https://doi.org/10.1145/3555776.3577659

[13] Zhecheng Wang, Haoyuan Li, and Ram Rajagopal. 2020. Urban2Vec: Incorporating Street View Imagery and POIs for Multi-Modal Urban Neighborhood Embedding. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 01 (Apr. 2020), 1013–1020. https://doi.org/10.1609/aaai.v34i01.5450

[14] Bo Yan, Krzysztof Janowicz, Gengchen Mai, and Song Gao. 2017. From ITDL to Place2Vec: Reasoning About Place Type Similarity and Relatedness by Learning Embeddings From Augmented Spatial Contexts. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '17)*. Association for Computing Machinery, New York, NY, USA, Article 35, 10 pages. https://doi.org/10.1145/3139958.3140054