Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Geocodificação de Endereços Brasileiros com Abordagens Lexicais, Indexadas e Semânticas

Sabrina Araújo da Silva e Samantha Miyahira

Monografia Final

mac 499 — Trabalho de Formatura Supervisionado

Supervisora: Prof.ª Dr.ª Kelly Braghetto

São Paulo
2025

Agradecimentos

Agradecimentos de Samantha Miyahira.

Agradeço à Professora Kelly Braghetto pela orientação, disponibilidade e apoio ao longo de todo o desenvolvimento deste trabalho. Suas contribuições e incentivos foram fundamentais para o amadurecimento da pesquisa e para a consolidação dos resultados apresentados.

Agradeço também a Inaê Batistoni e David Macedo pelo acompanhamento e suporte oferecidos durante o desenvolvimento do projeto.

Por fim, agradeço aos meus familiares e a todas as pessoas que fizeram parte da minha trajetória. O apoio, a compreensão e o incentivo foram decisivos não apenas para a conclusão deste trabalho, mas também para meu crescimento pessoal e acadêmico ao longo desse período.

Agradecimentos de Sabrina Araújo.

É um marco muito importante na minha vida estar escrevendo este trabalho final. Ele significa mais do que a conclusão de um ciclo; ele mostra onde todo o meu esforço e dedicação chegaram. Cheguei a este momento graças a todas as pessoas que, de alguma forma, fizeram parte da minha vida e me ajudaram a ser quem sou hoje. Sem vocês, nada disso teria sido possível. Então muitíssimo obrigada.

Agradeço especialmente à minha mãe, que hoje não está mais comigo, mas que me motiva todos os dias a ser alguém melhor e conquistar meus sonhos.

Resumo

Sabrina Araújo da Silva e Samantha Miyahira. **Geocodificação de Endereços Brasileiros com Abordagens Lexicais, Indexadas e Semânticas**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2025.

Este trabalho investiga métodos de correspondência de endereços aplicados à geocodificação de dados públicos brasileiros, com foco na avaliação espacial dos resultados obtidos. Foram implementadas e comparadas três abordagens distintas: lexical, indexada e baseada em modelos de linguagem. Essas abordagens são avaliadas por métricas espaciais como distância em metros, correspondência de setor censitário e coerência territorial dos pontos estimados, com apoio de um procedimento de desempate espacial desenvolvido em PostgreSQL com PostGIS. Os resultados indicam que a abordagem baseada em indexação apresentou melhor desempenho geral, com maior acurácia espacial e eficiência computacional nos conjuntos de dados analisados. O trabalho demonstra a viabilidade de uma solução de geocodificação construída exclusivamente com ferramentas de código aberto, resultando em uma API aberta como alternativa a serviços proprietários e pagos, e discute limitações da abordagem baseada em modelos de linguagem, além de apontar como trabalhos futuros o aprimoramento da precisão da geocodificação, a integração com APIs de dados culturais e o desenvolvimento de dashboards geográficos no contexto do projeto CulturaEduca.

Palavras-chave: geocodificação. correspondência de endereços. dados públicos. avaliação espacial. setores censitários. software livre.

Abstract

Sabrina Araújo da Silva e Samantha Miyahira. . Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2025.

This work investigates address matching methods applied to the geocoding of Brazilian public datasets, with a focus on the spatial evaluation of the results obtained. Three distinct approaches were implemented and compared: lexical, index-based, and language model–based. These approaches are evaluated using spatial metrics such as distance in meters, census tract correspondence, and territorial coherence of the estimated points, supported by a spatial tie-breaking procedure developed in PostgreSQL with PostGIS. The results indicate that the index-based approach achieved the best overall performance, with higher spatial accuracy and computational efficiency across the analyzed datasets. The work demonstrates the feasibility of a geocoding solution built exclusively with open-source tools, resulting in an open API as an alternative to proprietary and paid services, and discusses limitations of the language model–based approach, in addition to outlining future work directions, including improving geocoding precision, automating integration with cultural data APIs, and developing geographic dashboards within the CulturaEduca project.

Keywords: geocoding. address matching. public data. spatial evaluation. census tracts. open-source software.

Lista de abreviaturas

API	Interface de Programação de Aplicações (*Application Programming Interface*)
BM25	Função de ranqueamento baseada em frequência de termos
CNEFE	Cadastro Nacional de Endereços para Fins Estatísticos
CEP	Código de Endereçamento Postal
CNES	Cadastro Nacional de Estabelecimentos de Saúde
CPU	Unidade Central de Processamento (*Central Processing Unit*)
CSV	Comma-Separated Values
HNSW	Hierarchical Navigable Small World
IBGE	Instituto Brasileiro de Geografia e Estatística
IDF	Frequência inversa de documentos (*Inverse Document Frequency*)
IPEA	Instituto de Pesquisa Econômica Aplicada
KDE	Estimativa de densidade por kernel (*Kernel Density Estimation*)
LLM	Modelo de Linguagem de Grande Escala (*Large Language Model*)
RAM	Memória de Acesso Aleatório (*Random Access Memory*)
IME	Instituto de Matemática e Estatística
USP	Universidade de São Paulo

Lista de símbolos

δ	Limiar mínimo de similaridade para considerar correspondência
$E(\cdot)$	Função de codificação que mapeia textos para embeddings vetoriais

Lista de figuras

Sumário

Introdução

Ao lidar com endereços no Brasil, observa-se inevitavelmente uma grande variedade de formas de escrita. Um mesmo logradouro pode surgir abreviado em uma base de dados, aparecer por extenso em outra e, em várias situações, vir acompanhado de erros de digitação ou de informações ausentes. Essa irregularidade parece pequena à primeira vista, mas se torna um obstáculo concreto quando se busca localizar endereços com precisão no mapa.

O processo responsável por essa localização é a geocodificação, que transforma uma descrição textual de endereço em coordenadas geográficas. Em outras palavras, é o momento em que o texto, com todas as suas imperfeições, precisa ser convertido em latitude e longitude. Plataformas que trabalham com análise territorial dependem diretamente dessa conversão. Esse é o caso, por exemplo, da plataforma CulturaEduca,[1] para a qual localizar corretamente escolas, equipamentos culturais e outros serviços públicos do entorno é fundamental para interpretar padrões espaciais, identificar carências do território ou ainda fomentar o planejamento de ações comunitárias e/ou de políticas públicas.

Essa localização não se limita à obtenção das coordenadas. Parte das análises realizadas pelo CulturaEduca depende da associação correta entre o ponto geocodificado e o setor censitário correspondente. O setor censitário é a menor unidade territorial utilizada pelo IBGE para fins estatísticos e organiza o espaço urbano em áreas relativamente homogêneas. Ele funciona como a base primária para a produção de indicadores sociodemográficos e é amplamente utilizado em estudos sobre desigualdade, acesso a serviços públicos e dinâmicas territoriais. Quando um endereço é vinculado ao setor errado, toda a interpretação espacial se desloca, e relações importantes entre território, cultura e escola deixam de aparecer. Garantir essa correspondência correta, portanto, é fortalecer a precisão das análises territoriais produzidas pelo CulturaEduca.

O problema é que o CulturaEduca, apesar de reunir uma grande quantidade de dados públicos e colaborativos, não contava até o momento com uma ferramenta de software livre capaz de realizar essa geocodificação de maneira consistente. Tentativas iniciais de utilizar o pacote R GeocodeBR[2] (desenvolvida pelo IPEA) revelaram limitações importantes. Em vários casos, a ferramenta não conseguia identificar corretamente o setor censitário ou retornava resultados incompatíveis com o endereço informado. Parte dessas falhas ocorre porque o serviço utiliza parâmetros pré-definidos que nem sempre correspondem às necessidades das aplicações que dependem de maior precisão territorial.

[1] https://culturaeduca.cc/

[2] https://ipeagit.github.io/geocodebr/

Essas dificuldades acabaram compondo a motivação central deste trabalho. Buscou-se desenvolver uma solução capaz de lidar com a heterogeneidade natural dos endereços brasileiros e que, ao mesmo tempo, estivesse alinhada ao padrão de referência adotado pelo IBGE. A proposta consiste em criar um sistema que receba um endereço textual, mesmo incompleto ou mal formatado, e seja capaz de encontrar, dentro do Cadastro Nacional de Endereços para Fins Estatísticos (CNEFE),[3] o registro mais provável correspondente. Esse cadastro é publicado pelo Instituto Brasileiro de Geografia e Estatística (IBGE) e inclui endereços georreferenciados de domicílios e estabelecimentos de todo o país.

O CNEFE foi escolhido como base de comparação por reunir o conjunto mais abrangente de endereços oficiais do país e por sustentar grande parte das análises territoriais utilizadas em estudos demográficos e planejamentos públicos. Para aproximar o texto de entrada dos registros dessa base, tornou-se necessário combinar diferentes estratégias computacionais.

Três métodos independentes foram implementados: uma abordagem lexical usando a biblioteca RapidFuzz do Python, uma estratégia baseada em indexação com a ferramenta Elasticsearch e um método semântico baseado em *embeddings* gerados por um modelo de linguagem. Todos operam sobre as mesmas entradas preparadas, o que permite observar com clareza como cada técnica reage a abreviações, divergências de grafia e casos de menor completude.

Mais do que propor um algoritmo pontual, a intenção é contribuir para um conjunto de ferramentas abertas que possam ser reaproveitadas por outras iniciativas públicas e acadêmicas que enfrentam problemas semelhantes. Em campos como educação, cultura e habitação, decisões territoriais dependem de dados georreferenciados confiáveis, e a correspondência correta de endereços desempenha papel decisivo no resultado final dessas análises. O sistema aqui apresentado busca reforçar esse elo entre qualidade da informação e capacidade analítica, oferecendo um ponto de partida mais sólido para a produção de diagnósticos territoriais consistentes.

[3] https://www.ibge.gov.br/estatisticas/sociais/populacao/38734-cadastro-nacional-de-enderecos-para-fins-estatisticos.html

Capítulo 1

Fundamentação teórica

Os desafios discutidos na introdução deixam claro que, ao lidar com dados territoriais reais, é necessário enfrentar uma diversidade de formas de escrita que raramente segue um padrão único. No caso do CulturaEduca, essa variedade aparece de maneira ainda mais evidente, pois os registros vêm de fontes distintas e, muitas vezes, de contribuições manuais feitas por usuários que escrevem o mesmo endereço de modos diferentes. Esse cenário faz com que a identificação precisa de um logradouro na base de dados não seja apenas um detalhe técnico, mas um ponto que afeta diretamente a qualidade das análises territoriais.

Para enfrentar esse problema de modo sistemático, é necessário recorrer ao campo da comparação aproximada de strings, que reúne métodos pensados justamente para reconhecer quando duas expressões diferentes se referem à mesma entidade. Essa área oferece um repertório bastante amplo de técnicas, algumas baseadas em operações sobre caracteres, outras apoiadas em processos de indexação e ranqueamento, e ainda aquelas que tentam capturar relações de significado que não aparecem de forma literal no texto. Cada uma dessas abordagens lida, à sua maneira, com a falta de padronização que caracteriza grandes bases de endereços.

O objetivo deste capítulo é apresentar esse panorama teórico e situar as três abordagens que estruturam a metodologia deste trabalho. Mais do que listar algoritmos, pretende-se mostrar como cada técnica interpreta a ideia de similaridade textual e de que forma elas respondem aos tipos de variação presentes nas bases utilizadas pelo CulturaEduca. A partir disso, torna-se possível compreender porque certas estratégias funcionam bem em casos específicos, porque outras falham e como a combinação entre diferentes métodos acaba oferecendo uma visão mais robusta do problema.

1.1 Visão geral da comparação aproximada de strings

A comparação aproximada de strings é um problema clássico em Ciência da Computação e surgiu associado a tarefas práticas, como correção ortográfica, recuperação de

informações e verificação de similaridade textual (Navarro, 2001). Contudo, sua relevância tornou-se ainda mais evidente com o crescimento de sistemas digitais que armazenam grandes volumes de dados não padronizados, produzidos por diferentes fontes, usuários ou instituições. Trata-se de um tema central sempre que se deseja identificar, de forma automatizada, correspondências entre textos que representam a mesma entidade, mas que não compartilham a mesma grafia, seja por erros ortográficos, abreviações, variações regionais, inconsistências de digitação ou ausência de padronização formal.

Esse desafio aparece de maneira bastante evidente no caso dos endereços brasileiros. A forma como um endereço é registrado pode variar desde pequenas alterações, como a falta de acentuação em "Rua Sao Joao", até diferenças estruturais mais expressivas, como a abreviação de "Avenida" para "Av.", a mudança na ordem dos termos ou a omissão de elementos como "Travessa", o número do imóvel ou o CEP. Essas inconsistências impactam diretamente processos que dependem da comparação ou integração entre bases distintas e acabam dificultando desde a vinculação automática de registros até etapas posteriores, como a geocodificação e a produção de indicadores territoriais confiáveis.

Diante desse cenário, o foco deste estudo concentra-se na aplicação de técnicas de comparação aproximada para realizar a correspondência entre endereços registrados de formas divergentes, assegurando a integração entre diferentes bases de dados e a correta associação desses registros à sua localização geográfica.

1.2 Definição formal da correspondência aproximada

A comparação aproximada de strings, frequentemente referida na literatura como *approximate string matching* pode ser formalmente descrita como o processo de determinar se uma sequência de caracteres S corresponde a um padrão de referência P, mesmo quando não há coincidência literal entre as duas (Navarro, 2001). No modelo clássico de correspondência exata, a decisão é estritamente binária e depende da igualdade perfeita de todos os símbolos. Contudo, em aplicações reais, especialmente aquelas que lidam com dados produzidos por diferentes fontes ou preenchidos manualmente, essa condição raramente é satisfeita.

Para lidar com esse cenário, a correspondência aproximada adota uma função de similaridade $f(S, P)$, que mede o quanto S e P se aproximam. A decisão passa a depender de um limiar δ, de modo que considera-se haver correspondência quando

$$f(S, P) \geq \delta.$$

A função f pode assumir vários formatos e costuma ser normalizada em escalas como $[0, 1]$ ou $[0, 100]$, conforme o algoritmo e o tipo de interpretação buscada pelo usuário. Navarro (2001) discute esse enquadramento conceitual ao definir o problema de correspondência aproximada como a busca, em um texto T, por todas as ocorrências de um determinado padrão P que se mantenham abaixo de um limite máximo de diferença tolerável. Essa formulação é deliberadamente ampla, pois não impõe um método específico,

mas abre espaço para uma variedade de abordagens, desde as baseadas na distância de edição até técnicas de indexação e ranqueamento, ou ainda modelos semânticos que operam com representações vetoriais.

Essa flexibilidade ajuda a entender por que o tema aparece em diferentes contextos práticos. Entre os mais conhecidos estão a correção ortográfica automática e os sistemas de recuperação de informação, nos quais é necessário identificar textos semelhantes mesmo quando a grafia varia. Esses exemplos ilustram como a ideia de correspondência aproximada pode ser aplicada em situações diversas sem exigir que o texto original esteja perfeitamente padronizado.

1.3 Distância de edição: a métrica de Levenshtein

Entre as várias funções de similaridade usadas na correspondência aproximada de strings, a distância de edição, e em particular a métrica proposta por Levenshtein em 1965, acabou se tornando uma das mais conhecidas. Essa popularidade se explica tanto pela formulação bastante direta quanto pela capacidade de representar, de modo explícito, o custo necessário para transformar uma sequência de caracteres em outra por meio de operações simples de edição. Essa abordagem, sistematizada em estudos de referência como o levantamento de Navarro (2001), modela a comparação entre duas strings s_1 e s_2 como um problema de minimização: busca-se o número mínimo de operações necessárias para converter uma na outra, admitindo inserções, deleções e substituições de caracteres.

Sejam s_1 e s_2 duas strings de comprimentos m e n, respectivamente. Define-se uma matriz d de dimensão $(m+1)\times(n+1)$, na qual cada elemento $d(i,j)$ representa a distância de edição entre o prefixo de comprimento i de s_1 e o prefixo de comprimento j de s_2. A matriz é preenchida de forma incremental, considerando os custos mínimos associados às operações de edição permitidas.

A definição recursiva clássica da distância de Levenshtein pode ser expressa como:

$$d(i,j) = \min \begin{cases} d(i-1,j)+1, & \text{(Remoção)} \\ d(i,j-1)+1, & \text{(Inserção)} \\ d(i-1,j-1)+[s_1[i] \neq s_2[j]], & \text{(Substituição)} \end{cases} \tag{1.1}$$

com as condições iniciais $d(0,j) = j$ e $d(i,0) = i$. O valor final obtido corresponde ao custo mínimo de edição entre as duas sequências. Em muitas aplicações, porém, não se utiliza diretamente esse valor absoluto. Para facilitar a comparação entre registros, é comum empregar versões normalizadas da métrica, que convertem esse custo em um índice contínuo, geralmente em intervalos como $[0,1]$ ou $[0,100]$, nos quais 1 ou 100 indicam equivalência total. A Figura 1.1 ilustra esse processo para um caso simples, destacando as operações mínimas necessárias para converter uma string na outra.

Embora a formulação mais simples atribua custo igual para todas as operações, existem extensões que permitem ajustar esses pesos, por exemplo dando uma penalidade maior às substituições do que às inserções. Também é possível incluir uma operação adicional, a transposição de caracteres adjacentes, o que leva à chamada distância de Damerau–

		a	v	e	n	i	d	a
	0	1	2	3	4	5	6	7
a	1	0	1	2	3	4	5	6
v	2	1	0	1	2	3	4	5
.	3	2	1	1	2	3	4	**5**

Figura 1.1: *Exemplo de cálculo da distância de edição de Levenshtein entre as palavras "av." e "avenida". Cada célula da matriz representa o custo mínimo acumulado para transformar um prefixo da primeira string no prefixo correspondente da segunda. O valor final, destacado em vermelho (**5**), indica que são necessárias cinco operações de edição (uma substituição e quatro inserções) para converter "av." em "avenida".*

Levenshtein. Essa variante costuma ser especialmente útil quando os erros de digitação envolvem justamente a troca de posição entre duas letras consecutivas.

1.3.1 Otimizações da distância de edição

A implementação direta do algoritmo de Levenshtein, que consiste em preencher integralmente uma matriz dinâmica de tamanho $|s_1| \times |s_2|$, envolve um custo de tempo e memória proporcional ao produto dos comprimentos das strings, ou seja, $O(|s_1| \cdot |s_2|)$. Esse custo é administrável quando a comparação é pontual, mas rapidamente se torna inviável em aplicações que precisam lidar com milhões de registros. Para contornar essa limitação, ao longo do tempo surgiram implementações otimizadas que exploram propriedades estruturais do problema e recursos modernos de processamento para reduzir o número efetivo de operações executadas (Navarro, 2001).

Entre essas implementações, destaca-se a biblioteca *RapidFuzz*, que oferece uma versão altamente otimizada das métricas de distância e similaridade derivadas de Levenshtein (Bachmann, 2024). Além de fornecer versões normalizadas e suporte a pesos personalizados, a biblioteca incorpora variantes como Damerau–Levenshtein. Além disso, incorpora mecanismos de interrupção antecipada (*score_cutoff*), que encerram o cálculo assim que fica claro que o valor final não atingirá o limiar mínimo de similaridade desejado.

O ganho de desempenho proporcionado pelo *RapidFuzz* resulta da combinação de várias estratégias. Uma delas é justamente a interrupção antecipada, que evita o preenchimento completo da matriz quando a similaridade entre as strings é muito baixa, prevenindo o preenchimento desnecessário da matriz dinâmica. Outra é o uso de algoritmos bit-paralelos, em que blocos de caracteres são representados como vetores de bits, permitindo que operações de máquina processem até 64 posições ao mesmo tempo. Na prática, isso reduz o custo para algo proporcional a $\left\lceil \frac{|s_1|}{w} \right\rceil \cdot |s_2|$, em que w representa o tamanho da palavra da CPU. Há ainda uma seleção adaptativa de rotinas, na qual o algoritmo escolhe automaticamente a estratégia mais eficiente de acordo com o tamanho e as características das strings comparadas.

Além dessas otimizações estruturais, o *RapidFuzz* incorpora mecanismos auxiliares, como pré-processadores (*processors*) destinados à padronização e limpeza textual, incluindo a normalização de maiúsculas e minúsculas, a remoção de acentos e caracteres não alfabéticos, o que contribui para aumentar a precisão e reduzir o ruído nas comparações.

O resultado de todas essas otimizações é que a distância de Levenshtein passa a ser utilizada de forma viável mesmo em cenários com grandes volumes de dados. Na prática, ela acaba servindo como base matemática de muitos métodos modernos de correspondência aproximada de strings, especialmente em sistemas que precisam equilibrar precisão, escalabilidade e desempenho.

1.3.2 Similaridade baseada em conjuntos de tokens

Embora a distância de edição seja uma métrica eficaz para comparar palavras curtas ou sequências com pequenas variações de caracteres, seu desempenho e sua capacidade de captura semântica diminuem quando o objeto de comparação deixa de ser uma palavra isolada e passa a ser uma expressão composta, como ocorre no caso de endereços completos.

Nesse tipo de situação, diferenças na ordem dos termos, na presença de elementos adicionais ou supérfluos, ou ainda na abreviação de componentes específicos podem gerar um aumento artificial da distância, mesmo quando as duas expressões se referem ao mesmo local no mundo real.

Para lidar com esse tipo de situação, muitas bibliotecas atuais passam a tratar a string não mais como uma sequência contínua de caracteres, mas como um conjunto de unidades léxicas obtidas por tokenização.

O processo de tokenizar consiste em segmentar o texto em palavras ou termos relevantes, eliminando elementos que pouco influenciam a identificação, como pontuação, variações de espaçamento ou diferenças entre maiúsculas e minúsculas. Depois dessa etapa, a comparação entre duas strings pode ser feita avaliando a interseção entre os conjuntos de tokens, sem se preocupar com a ordem em que esses termos apareciam originalmente.

Entre essas métricas, destaca-se a *token set ratio*, amplamente utilizada em bibliotecas como o *RapidFuzz* (Bachmann, 2024). Ela mede a similaridade com base apenas nos tokens que aparecem em ambas as expressões, descartando repetições e removendo termos irrelevantes para a interseção. Em termos resumidos, a pontuação pode ser calculada por

$$\text{score} = \frac{2 \cdot |A \cap B|}{|A| + |B|} \times 100,$$

em que A e B representam os conjuntos de tokens extraídos de cada string. Essa formulação privilegia a proporção de termos coincidentes, o que explica por que expressões como “Rua das Flores”, “R. das Flores” e “Rua das Flores 123” costumam obter valores elevados mesmo apresentando diferenças estruturais.

A principal vantagem dessa abordagem está justamente na capacidade de lidar com variações de forma sem penalizar excessivamente a similaridade, algo que o cálculo posi-

cional baseado em edição de caracteres não oferece. Além disso, a tokenização contribui para a escalabilidade dos algoritmos de comparação textual, pois permite a aplicação de filtros iniciais que reduzem o número de candidatos a serem avaliados por métricas mais custosas, como as baseadas em distância de edição.

Dessa forma, métodos baseados em conjuntos de tokens mostram-se especialmente adequados para cenários em que o texto é redundante, pouco padronizado ou sujeito à variação humana, como ocorre em registros de endereços brasileiros. Entre as implementações mais difundidas desse tipo de abordagem destaca-se o módulo *fuzz.token_set_ratio*, da biblioteca *RapidFuzz*, amplamente utilizado em tarefas de correspondência textual aproximada.

1.4 Busca e ranqueamento por recuperação de informação

A formulação apresentada na Seção 1.2 descreve o problema da correspondência aproximada como a aplicação de uma função de similaridade $f(S, P)$, responsável por estimar o quão próximas duas expressões realmente estão. Essa forma de enxergar o problema é relativamente simples, mas é o suficiente para abranger métodos muito diferentes entre si, desde métricas clássicas, como a distância de edição, até técnicas recentes que recorrem a representações vetoriais e embeddings para capturar nuances semânticas que não aparecem apenas pela comparação caractere a caractere.

Nessa linha de raciocínio, o texto deixa de ser entendido como uma simples sequência linear de caracteres e passa a assumir uma forma mais estruturada, apoiada em mecanismos de indexação. O principal deles é o índice invertido, apresentado originalmente por Manning, Raghavan e Schütze (Manning *et al.*, 2008). Nesse tipo de estrutura, cada termo que aparece na coleção é associado a uma lista com todas as suas ocorrências, o que permite executar consultas sem percorrer o conjunto de dados. Com isso, encontrar itens textuais semelhantes deixa de depender de uma comparação direta entre strings e passa a se apoiar em um processo de recuperação bem mais eficiente, ainda que baseado nos mesmos elementos fundamentais do texto.

Uma vez estruturada a coleção, a relevância dos documentos é calculada por meio de funções de ranqueamento, que combinam diferentes sinais para estimar o quanto cada item é pertinente à consulta. Entre os modelos mais utilizados está o *BM25*, uma função clássica baseada em frequência de termos. Ele foi proposto por Robertson e Zaragoza (2009) e segue a lógica de aproveitar informações simples, como a recorrência dos termos, mas ajustando esses valores de forma a evitar distorções. Sua formulação é dada por:

$$\mathrm{BM25}(q, d) = \sum_{t \in q} \mathrm{IDF}(t) \cdot \frac{f(t, d) \cdot (k_1 + 1)}{f(t, d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{\mathrm{avgdl}}\right)},$$

em que $f(t, d)$ representa a frequência do termo t no documento d, $|d|$ é o tamanho do documento e avgdl é o comprimento médio da coleção. Os parâmetros k_1 e b controlam a saturação da frequência e a normalização por comprimento, respectivamente. A pre-

sença do fator IDF(t) (*inverse document frequency*), atua como um contraponto ao peso dos termos muito comuns, reduzindo sua influência e destacando aqueles que são mais informativos para a busca.

O ponto central desse tipo de abordagem é que a similaridade deixa depender exclusivamente de operações sobre caracteres. Ela passa a ser interpretada como um problema de relevância, em grande parte orientado por modelos probabilísticos. Com isso, abrem-se caminhos para incorporar certa tolerância a variações linguísticas e erros ortográficos. Em muitos sistemas, essa flexibilidade aparece na forma de operadores de busca *fuzzy*, que se apoiam na própria distância de edição (Navarro, 2001) para identificar documentos cujo conteúdo apresenta pequenas divergências do termo consultado.

Esse conjunto de conceitos, isto é, o uso de índices invertidos, funções de ranqueamento e mecanismos de tolerância a ruídos, fundamenta várias das ferramentas contemporâneas de busca textual. Entre elas, destaca-se o *Elasticsearch*, que implementa o modelo do Lucene (Gormley e Tong, 2015) e oferece um ecossistema robusto para indexação, análise e recuperação eficiente de texto em larga escala.

O *Elasticsearch* utiliza índices invertidos para organizar o texto, aplica diferentes analisadores para normalizar e tokenizar os termos, e emprega o BM25 como função de ranqueamento padrão. Além disso, oferece recursos que aumentam a tolerância a imperfeições comuns em textos reais. Entre eles estão as buscas *fuzzy*, consultas por frase com algum grau de variação permitido, conhecidas como *slop*, e estratégias como *multi-match* e *cross-fields*. Essas últimas ajudam a combinar diferentes campos textuais, por exemplo logradouro e bairro, dentro de uma única consulta, o que costuma melhorar bastante a recuperação de dados heterogêneos. Por essa combinação de técnicas, o *Elasticsearch* acaba se ajustando bem a contextos em que há abreviações, redundâncias e pequenas inconsistências de grafia, um cenário comum nos registros de endereços brasileiros.

Com isso, os mecanismos de busca e ranqueamento oferecem uma alternativa eficiente às abordagens baseadas apenas em comparação direta de strings. Em vez de comparar cada string com todas as demais, explora-se uma estrutura de indexação para reduzir o conjunto de candidatos e concentrar o esforço apenas onde é necessário. A similaridade, então, é calculada de maneira mais seletiva e, sobretudo, mais escalável. Essa perspectiva se mostra especialmente útil para sistemas que precisam lidar com grandes volumes de dados e equilibrar a precisão com o desempenho.

A Figura 1.2 ilustra, de maneira simplificada, como um índice invertido pode organizar uma coleção de endereços. Cada endereço é previamente tokenizado e normalizado, gerando termos como *rua*, *das*, *flores* ou números de porta. O índice então associa cada termo a uma lista de documentos onde ele aparece. Assim, quando uma consulta como "rua da flor 123" é realizada, o sistema converte a consulta em tokens (*rua*, *da*, *flor*, *123*) e recupera rapidamente os documentos que contêm cada um desses termos. Por exemplo, o token "rua" recupera D1 e D2, enquanto o número "123" recupera D1.

Embora alguns termos da consulta não apareçam exatamente nos documentos, como "flor", que não coincide literalmente com "flores", mecanismos tolerantes a variações, como o *fuzzy matching* do Elasticsearch, permitem localizar termos próximos com base em diferenças pequenas de grafia. Esse comportamento amplia o conjunto de candidatos

relevantes e ajuda a identificar registros mesmo diante de variações lexicais. Com essa combinação de indexação e tolerância a ruídos, o sistema consegue chegar aos endereços mais prováveis de maneira muito mais rápida, sem precisar comparar a string da consulta com todas as strings armazenadas.

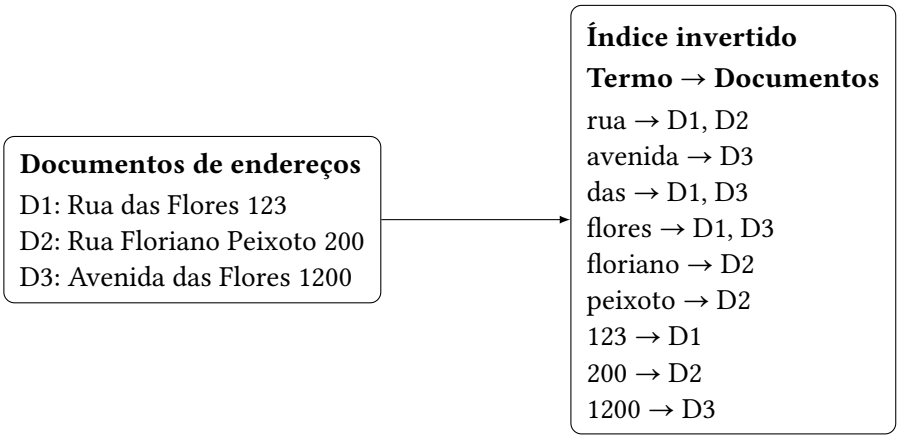

Figura 1.2: *Exemplo simplificado de índice invertido construído a partir de uma coleção de endereços. Cada termo (como* rua, flores *ou números de porta) é associado aos documentos em que aparece, permitindo que consultas por termos recuperem rapidamente apenas os endereços candidatos.*

1.5 Similaridade semântica por embeddings

A formulação geral apresentada na Seção 1.2 descreve a correspondência aproximada como a aplicação de uma função de similaridade $f(S, P)$ sobre duas strings S e P. A decisão final depende de um limiar δ, que define se a similaridade obtida é suficiente ou não. Nos métodos tradicionais, essa função opera sobre o texto em sua forma literal, isto é, a comparação é realizada no nível dos caracteres ou dos tokens sem recorrer a nenhuma camada adicional de interpretação. Esse tipo de procedimento costuma funcionar bem para identificar erros ortográficos simples ou pequenas alterações de forma.

Apesar disso, a análise permanece presa à superfície do texto e não leva em conta o significado das expressões. Situações bastante comuns, como "Av. Paulista", "Avenida Paulista" ou "Paulista, Av.", seguem sendo tratadas como sequências distintas, embora, para qualquer pessoa, seja claro que todas apontam para o mesmo endereço. É justamente nesse ponto que as limitações das abordagens puramente lexicais se tornam mais evidentes.

A possibilidade de ultrapassar esse limite surge com o uso de *embeddings*, representações numéricas contínuas que projetam palavras, sentenças ou documentos em um espaço vetorial de alta dimensionalidade. Nessa forma de representação, elementos que carregam significados próximos acabam ocupando regiões vizinhas do espaço, enquanto expressões de sentidos diferentes tendem a se afastar naturalmente. Essa mudança de perspectiva desloca o foco do texto literal para a estrutura geométrica subjacente às relações semânticas.

A Figura 1.3 oferece uma visualização intuitiva dessa ideia. Expressões que se referem ao mesmo endereço, ainda que escritas de maneiras diferentes, aparecem agrupadas em uma mesma região, enquanto termos semanticamente não relacionados ocupam regiões

mais distantes no plano vetorial. Essa visualização reforça a ideia de que a similaridade deixa de ser determinada pela forma exata das palavras e passa a ser avaliada geometricamente. Comparar duas sentenças, nesse contexto, significa medir a proximidade entre seus vetores dentro desse espaço contínuo.

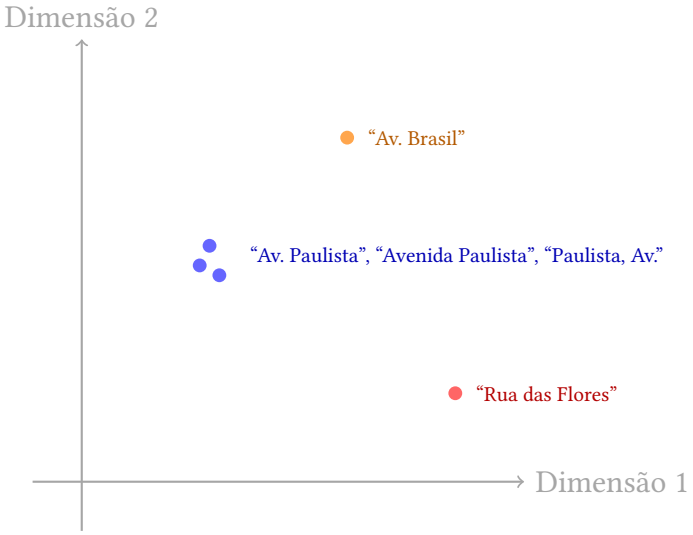

Figura 1.3: *Representação conceitual de embeddings de sentenças no espaço vetorial. Expressões semanticamente equivalentes ocupam regiões próximas, enquanto expressões não relacionadas aparecem distantes.*

Esses vetores são produzidos por modelos de linguagem que passam por um pré-treinamento extenso em grandes coleções de textos. Ao longo desse processo, o modelo aprende regularidades da língua, desde relações de similaridade semântica até padrões sintáticos e variações lexicais que aparecem no uso real. A ideia central é construir uma função de codificação $E(\cdot)$ capaz de transformar cada sentença em um vetor que represente seu conteúdo semântico como um todo, e não apenas a sua forma textual.

Entre os modelos que realizam esse tipo de codificação, aqueles baseados na arquitetura *Transformer* ganharam destaque e acabaram se tornando predominantes no processamento de linguagem natural. Isso ocorre porque eles analisam cada palavra considerando o contexto em que aparece, o que contrasta bastante com abordagens anteriores, nas quais os termos eram tratados isoladamente. Os *Transformers* consideram todas as relações de dependência dentro da sentença, permitindo construir representações que refletem o significado global do texto e não apenas a sequência de palavras que o compõe.

Reimers e Gurevych (2019) apresentam o Sentence-BERT como um modelo representativo dessa abordagem e amplamente utilizado para geração de embeddings de sentenças. Em vez de gerar um embedding separado para cada palavra, ele produz vetores densos de dimensão fixa para sentenças inteiras. O resultado é que duas expressões equivalentes, mesmo quando escritas de formas diferentes, acabam ocupando posições próximas no espaço contínuo, o que facilita a comparação semântica.

Além do modelo completo, há também versões mais leves baseadas na mesma arquitetura. Um exemplo é o MiniLM (Wang *et al.*, 2021), que aplica técnicas de compactação e otimização para reduzir o custo computacional sem abandonar o princípio central da

representação vetorial de significado. Esses modelos menores preservam a ideia fundamental da comparação semântica, mas tornam sua aplicação viável em contextos em que há restrições de memória ou de tempo de processamento, algo comum quando se trabalha com um grande volume de dados.

Com o uso de embeddings, a função de similaridade f deixa de ser entendida como uma comparação literal entre caracteres e passa a assumir a forma de uma medida de proximidade entre vetores, tipicamente expressa pela similaridade do cosseno:

$$f_{\text{emb}}(S, P) = \cos\big(E(S), E(P)\big) = \frac{\langle E(S), E(P)\rangle}{\|E(S)\| \cdot \|E(P)\|}.$$

Essa mudança de perspectiva permite capturar relações de sentido que vão além da correspondência exata de palavras ou símbolos. Por isso, ela se mostra especialmente valiosa em tarefas de entity matching, nas quais o desafio é reconhecer que dois registros descrevem a mesma entidade do mundo real, mesmo que venham de bases diferentes ou contenham variações de escrita.

Quando o contexto inclui ruídos textuais como abreviações, erros de digitação ou simplesmente a falta de padronização, métodos puramente lexicais costumam apresentar desempenho limitado. Trabalhos recentes, como o de Brunner e Stockinger (2020), indicam que modelos baseados em Transformers superam as abordagens tradicionais, oferecendo um equilíbrio mais consistente entre precisão e cobertura. Esses resultados reforçam o papel das representações vetoriais como alternativa teórica e prática às métricas clássicas de similaridade, sobretudo em domínios textuais mais complexos, como o dos endereços brasileiros.

1.6 Síntese das abordagens de correspondência

As seções anteriores apresentaram um conjunto amplo de técnicas voltadas à correspondência aproximada de strings. Inicialmente, são apresentadas métricas clássicas baseadas em distância, em seguida, métodos apoiados em indexação e ranqueamento e, na sequência, abordagens recentes que recorrem a representações semânticas em espaços vetoriais. Cada uma dessas linhas propõe uma forma distinta de medir similaridade textual, refletindo diferentes níveis de abstração linguística, custos computacionais variados e capacidades próprias de generalização.

De maneira geral, essas técnicas podem ser organizadas em três vertentes que estruturam o campo. A abordagem lexical opera diretamente sobre caracteres ou tokens e mostra boa eficiência quando lida com erros ortográficos e pequenas mudanças de grafia. Já os métodos baseados em busca e ranqueamento introduzem mecanismos de indexação que permitem reduzir o espaço de comparação e acelerar a recuperação dos candidatos mais prováveis, sobretudo em bases extensas. Por fim, as abordagens semânticas deslocam o foco para o significado das expressões, representando sentenças em vetores contínuos que preservam relações de proximidade mesmo quando as palavras usadas são distintas.

Embora partam de princípios diferentes, todas compartilham o objetivo central de reconhecer que duas expressões textuais podem se referir à mesma entidade. Divergem ape-

nas nas estratégias para alcançar isso, seja enfatizando a forma literal do texto, a estrutura da busca ou a interpretação semântica. Quando observadas juntas, revelam a amplitude de interpretações possíveis sobre o que significa medir similaridade entre strings, algo que não é trivial quando se trabalha com dados heterogêneos.

Essa diversidade metodológica é especialmente relevante para o problema que motivou este estudo: a geocodificação de endereços brasileiros. Endereços incompletos, abreviados ou registrados com grafias incompatíveis com o padrão do IBGE dificultam a correspondência direta com bases de referência como o CNEFE. Em contextos como o do CulturaEduca, em que a correta localização de escolas, equipamentos culturais e estruturas do entorno influencia a interpretação de padrões territoriais e a identificação de setores censitários, a escolha do método de correspondência afeta diretamente a fidelidade das análises subsequentes.

As três abordagens apresentadas respondem, cada uma à sua maneira, às formas de variação mais comuns nesse tipo de dado. Algumas lidam melhor com abreviações, outras com divergências estruturais no texto e outras ainda conseguem capturar equivalências mais profundas de significado. Essa complementaridade oferece uma base sólida para o desenvolvimento de uma ferramenta que consiga aproximar os registros da entrada textual ao endereço oficial mais provável no CNEFE.

Dessa forma, o panorama teórico discutido aqui é uma base sólida para o capítulo seguinte, em que essas abordagens são incorporadas ao sistema de correspondência proposto e avaliadas com base nas exigências práticas do processo de geocodificação.

Capítulo 2

Metodologia

A discussão apresentada no capítulo anterior mostrou que a correspondência aproximada de strings não pode ser reduzida a uma única técnica. O problema admite diferentes caminhos, cada qual enfatizando um aspecto particular da linguagem: a comparação mais literal entre caracteres, as estratégias de busca estruturada baseadas em indexação ou, ainda, métodos que tentam capturar alguma noção de significado. Essa diversidade acabou influenciando diretamente a escolha dos procedimentos adotados neste trabalho. A partir das três vertentes identificadas, a abordagem lexical, a indexação com ranqueamento e a análise semântica, buscou-se construir um sistema que permitisse observar, de maneira consistente, como cada abordagem responde às irregularidades comuns nos endereços brasileiros.

Em vez de selecionar apenas uma técnica e explorá-la isoladamente, optou-se por desenvolver o sistema em uma arquitetura modular. Um núcleo central concentra as etapas de preparação e normalização dos dados, enquanto três módulos independentes implementam as estratégias de correspondência propriamente ditas. Essa separação facilita a comparação entre os métodos, já que todos recebem as mesmas entradas e passam pelo mesmo pré-processamento.

Nos tópicos seguintes, são apresentados os componentes dessa arquitetura: o processo de normalização textual dos endereços, as abordagens específicas utilizadas para a correspondência e o procedimento adotado para avaliar o desempenho obtido por cada método.

2.1 Contexto inicial e motivação do sistema

A motivação inicial deste trabalho nasceu diretamente das demandas do CulturaEduca, plataforma que integra dados públicos e informações colaborativas para apoiar análises territoriais no entorno das escolas. Uma parte importante do funcionamento da plataforma depende da possibilidade de localizar corretamente diferentes tipos de equipamentos, culturais, sociais, educacionais, utilizando diferentes bases de dados que raramente seguem o mesmo padrão. Cada fonte registra seus endereços de um modo próprio, e essa heterogeneidade acaba se tornando um obstáculo na hora de aproximar esses dados do território real.

Foi nesse contexto que surgiu a necessidade de desenvolver uma ferramenta capaz de geocodificar endereços brasileiros de forma consistente com o padrão do CNEFE. A intenção não era apenas resolver um problema pontual dentro do CulturaEduca, mas caminhar para a construção de uma solução aberta, que pudesse ser reutilizada por outros projetos públicos e acadêmicos. Em termos práticos, buscava-se estabelecer a base para uma ferramenta em software livre que viabilizasse a vinculação de endereços, vindos de fontes distintas, aos registros do CNEFE.

À primeira vista, a forma mais direta de realizar essa vinculação seria comparar cada endereço de entrada com os registros do CNEFE e selecionar o mais provável. No entanto, logo se tornou evidente que essa tarefa está longe de ser trivial. Os endereços usados por diferentes instituições nem sempre seguem a mesma convenção, muitos vêm com abreviações, erros de digitação ou pequenas mudanças de grafia. Além disso, certos campos, como o tipo de logradouro ou o bairro, variam bastante entre bases, mesmo quando apontam para o mesmo ponto no mapa.

Diante dessas dificuldades, definiu-se o objetivo central da pesquisa: desenvolver um algoritmo flexível, capaz de lidar com essa diversidade de escrita e produzir, para cada endereço textual, uma resposta consistente sobre o registro mais provável no CNEFE. Para isso, foi estruturado um sistema modular, escrito em Python, organizado em etapas de normalização dos dados, chamada aos algoritmos de correspondência e análise final dos resultados. A implementação do sistema foi disponibilizada em repositório público, reunindo os principais componentes do trabalho desenvolvido..[1]

2.2 Processo de construção do algoritmo

O primeiro passo do desenvolvimento do sistema foi trabalhar com um cenário controlado. Optou-se por trabalhar inicialmente com um recorte do CNEFE correspondente ao município de Diadema, que oferece um volume de registros suficientemente representativo, mas ainda manejável para experimentação.

Como conjunto de teste, foram utilizados os endereços dos estabelecimentos de saúde do CNES, previamente validados. A escolha desse conjunto se deve ao fato de que, por já serem endereços corretos e geocodificados, permitiam avaliar com precisão se o sistema seria capaz de encontrar, no CNEFE, o mesmo ponto que serviria de referência.

Com essas bases definidas, foi implementada a primeira versão do algoritmo, ainda bastante direta. Essa versão inicial realizava comparações textuais entre os endereços utilizando duas bibliotecas: *RapidFuzz*, representando as abordagens lexicais fundamentadas em distância e operações sobre tokens, e *Elasticsearch*, que explora mecanismos de indexação e ranqueamento para consultas eficientes em conjuntos extensos de documentos.

2.2.1 Organização da implementação e estrutura do código

A ferramenta foi estruturada em módulos independentes, cada um responsável por uma parte específica do processo de correspondência, mas todos articulados em torno de

[1] https://github.com/sabrizzs/tcc-culturaeduca

um mesmo fluxo de execução.

O ponto de entrada do sistema é o arquivo `run_comparador.py`. Ele cumpre o papel de conduzir a ferramenta: carrega os conjuntos de dados, define quais colunas serão interpretadas como logradouro, número e bairro, e escolhe o algoritmo que deve ser acionado em cada rodada. Ao final, o próprio script exporta os resultados consolidados, o que facilita a inspeção manual dos casos corretos, ambíguos ou incorretos.

A lógica central do processo encontra-se no arquivo `comparador.py`. Esse módulo funciona como um núcleo comum que padroniza os dados de entrada, aplica as funções de normalização desenvolvidas ao longo do projeto e distribui cada consulta para o algoritmo selecionado. É nessa etapa que ocorrem as transformações textuais mais importantes que garantem que todas as abordagens trabalhem sob as mesmas condições e possam ser comparadas de maneira justa.

As estratégias de correspondência, por sua vez, foram encapsuladas em módulos distintos. O arquivo `rapidfuzz_module.py` reúne a lógica baseada em similaridade léxica, utilizando operações sobre tokens e métricas derivadas da distância de edição. E o `elasticsearch_module.py` concentra a abordagem orientada à indexação e ao ranqueamento, que permite consultas mais flexíveis e eficientes em bases extensas.

Essa organização tornou possível alternar rapidamente entre as abordagens, ajustar parâmetros, introduzir novas funções de normalização e observar, de maneira sistemática, como cada técnica reagia ao mesmo conjunto de entradas. Ao longo do desenvolvimento, vários ajustes se mostraram necessários e essa estrutura flexível permitiu incorporá-los sem comprometer a coerência do sistema como um todo.

2.3 Limitações observadas e ajustes necessários

Embora funcional, essa primeira implementação evidenciou rapidamente uma série de limitações. Diferenças aparentemente pequenas na forma de escrita, como abreviações, tipos distintos de logradouro ou variações numéricas, eram suficientes para alterar o resultado esperado. Em diversos casos, o endereço correto não surgia como primeira opção, apesar de estar presente na base de referência.

Grande parte dessas divergências não estava vinculada ao algoritmo em si, mas às múltiplas maneiras pelas quais um mesmo endereço pode ser registrado em bases distintas, muitas vezes de forma pouco previsível. A partir dessa constatação, o desenvolvimento da ferramenta passou a seguir um processo essencialmente iterativo. Implementava-se uma versão inicial, o conjunto de testes era executado, os erros eram examinados cuidadosamente e, com base neles, novas funções eram introduzidas ou decisões anteriores eram revisadas. Esse ciclo de testar, interpretar e ajustar se repetiu diversas vezes e acabou moldando a organização final da implementação. Com isso, foi desenvolvido um conjunto de funções de normalização no módulo `normalizar.py`.

A função `normalizar_abreviacoes` surgiu como uma das primeiras intervenções necessárias no processo de preparação dos dados. Observou-se que grafias como “Av. Paulista” e “Avenida Paulista”, embora equivalentes do ponto de vista semântico, produziam

pontuações bastante diferentes nos algoritmos mais sensíveis à forma textual. No RapidFuzz, por exemplo, a comparação baseada em tokens penalizava a abreviação de forma acentuada, o que fazia com que endereços corretos fossem trocados por alternativas claramente equivocadas. A expansão das abreviações para suas versões completas eliminou um conjunto significativo de erros.

Outra modificação relevante envolveu o tratamento do tipo de logradouro. Era comum encontrar, em bases distintas, registros como "Rua Ana Neri" e "Avenida Ana Neri", apesar de ambos se referirem ao mesmo ponto no território. Essa variação, aparentemente trivial, gerava inconsistências nos scores e, em diversas situações, levava à escolha de correspondências incorretas. A função `remover_tipo_logradouro` passou a excluir esse elemento antes da comparação, preservando apenas a parte substantiva do nome da via. Esse ajuste reduziu de maneira expressiva o número de falsos negativos nas primeiras rodadas de experimentação.

As divergências relativas ao bairro também se mostraram frequentes. Observou-se que diferentes instituições utilizam recortes próprios de bairros, como resultado, dois registros que apontavam para o mesmo endereço podiam apresentar bairros distintos. Para evitar que esse tipo de variação influenciasse indevidamente a comparação textual do logradouro, optou-se por tratar o bairro como um componente separado, com peso próprio na composição do score. A normalização desse campo ficou a cargo da função `normalize_bairro`.

O número do imóvel exigiu uma atenção específica. A comparação direta entre dígitos tratados como strings gerava interpretações pouco adequadas, pois, sob a ótica textual, "1" acaba mais próximo de "10" do que de "2", ainda que numericamente a relação seja exatamente o oposto. Nos testes iniciais, foram observadas situações em que o RapidFuzz sugeria o número 10 como correspondência para um endereço cujo número correto era 1, simplesmente porque a distância lexical entre as grafias era menor do que entre "1" e "2". Para evitar esse tipo de distorção, o componente numérico passou a ser separado do restante do logradouro. Cada módulo de correspondência aplica então a sua própria lógica de comparação numérica, baseada na proximidade real entre os valores, enquanto o logradouro permanece sujeito apenas às métricas textuais. Essa combinação de comparação lexical e comparação numérica independente reduziu de maneira significativa os erros associados a imóveis com numeração sequencial.

Outra situação recorrente envolvia números presentes no próprio logradouro, e não apenas no campo de numeração do imóvel. Exemplos típicos incluem expressões como "Rua Vinte e Dois de Abril" e "Rua 22 de Abril". Nessas situações, a comparação puramente textual tratava "vinte e dois" e "22" como sequências completamente distintas, produzindo uma penalização artificial. Para mitigar esse problema, foi criada a função `numeros_para_texto`, que converte números escritos por extenso para sua forma numérica (ou vice-versa, dependendo da convenção adotada). Esse tratamento se mostrou importante para reduzir discrepâncias na comparação do logradouro, especialmente em vias cujos nomes incorporam datas, números ordinais ou referências históricas.

Além dessas funções específicas, o módulo `normalizar.py` também reúne um conjunto de operações de normalização mais gerais: remoção de acentos, conversão para minúsculas, eliminação de múltiplos espaços, entre outras. Embora sejam procedimen-

tos simples, eles desempenham um papel importante na redução de ruídos textuais e na garantia de que diferenças superficiais não prejudiquem a comparação.

Por fim, todas essas transformações são organizadas de modo que cada endereço seja decomposto em três componentes fundamentais: *logradouro*, *bairro* e *número*. Assim, cada parte é comparada de forma independente e com pesos distintos na composição do score. O módulo de comparação envia esses três elementos já normalizados para o algoritmo selecionado, que utiliza a ponderação definida para estimar o candidato mais provável dentro do CNEFE.

A etapa de normalização acabou se mostrando uma parte realmente central do processo. Ela não só organiza o texto de entrada, como também influencia de maneira direta a precisão que se alcança nos métodos de correspondência apresentados nas próximas seções. Sem esse preparo inicial, muitos dos resultados posteriores simplesmente não se sustentariam com a mesma consistência.

2.3.1 Inclusão de um terceiro método: correspondência semântica por LLM

Até esse ponto do desenvolvimento, todo o processo havia sido conduzido com base nas duas abordagens originais: RapidFuzz e Elasticsearch. Ambas permitiram compreender de forma clara os limites práticos das técnicas lexicais e baseadas em indexação quando aplicadas a endereços brasileiros. No entanto, à medida que a análise dos erros remanescentes avançava, tornou-se evidente que parte das divergências não se explicava apenas por ruídos superficiais de escrita, mas por variações mais profundas na forma como os textos representavam um mesmo lugar.

Diante disso, optou-se por incorporar um terceiro método de correspondência ao sistema: uma abordagem de caráter semântico baseada em embeddings gerados por um modelo de linguagem. A ideia era ampliar o escopo da comparação, permitindo observar como um modelo treinado para capturar relações de significado lida com expressões que, embora não coincidam literalmente, descrevem o mesmo endereço.

Com a inclusão desse terceiro método, o sistema passou a reunir três formas distintas de comparar textos: a aproximação lexical, a busca indexada e a análise semântica. Todas trabalham a partir da mesma entrada normalizada, mas cada uma trata o problema por um ângulo diferente. Cada abordagem será discutida separadamente, com atenção às escolhas de implementação e aos ajustes que se mostraram necessários para lidar com as peculiaridades dos endereços brasileiros. Essa divisão ajuda a perceber não só onde cada método funciona melhor, mas também como eles se complementam dentro do processo geral de correspondência.

2.4 Métodos de correspondência

A etapa de correspondência propriamente dita inicia-se após a normalização dos endereços. Cada registro é encaminhado para um dos três métodos implementados no sistema, todos organizados em módulos independentes. Essa separação modular permitiu testar e

ajustar cada abordagem de forma isolada, sem comprometer a estrutura central do código. Ela também favoreceu a transparência do processo, já que cada módulo se dedica exclusivamente à lógica de seu algoritmo, enquanto o núcleo do sistema permanece responsável pelo pré-processamento e pela coordenação das execuções.

As três estratégias adotadas refletem diferentes maneiras de interpretar a noção de similaridade textual. A primeira é uma abordagem léxica, construída sobre a biblioteca RapidFuzz, que opera com métricas baseadas em distância de edição e manipulação de tokens. A segunda utiliza o Elasticsearch, um mecanismo de busca que combina indexação, tokenização e ranqueamento probabilístico para recuperar candidatos relevantes em grandes coleções textuais. A terceira explora um modelo de linguagem para produzir embeddings vetoriais, permitindo comparar endereços pelo seu conteúdo semântico e não apenas por sua forma literal.

Embora partam de princípios distintos, todas recebem a mesma entrada normalizada e retornam estruturas de resultado comparáveis, o que facilita a análise cruzada entre métodos. Nas subseções seguintes, cada abordagem é detalhada individualmente, com atenção às decisões de implementação, aos parâmetros ajustados e às particularidades que surgiram ao aplicá-las ao domínio dos endereços brasileiros.

2.4.1 Abordagem lexical: RapidFuzz

O RapidFuzz foi uma das primeiras ferramentas avaliadas para lidar com a comparação entre endereços. Ele oferecia uma forma simples e rápida de aproximar duas expressões escritas de maneiras distintas, o que ajudou a iniciar os testes com mais segurança. A biblioteca serviu como um ponto de partida concreto para explorar a ideia de similaridade textual e observar, na prática, como uma técnica lexical reage às variações presentes nos registros brasileiros.

A biblioteca reúne implementações otimizadas das métricas baseadas em distância de edição e das variações construídas a partir de conjuntos de tokens, o que facilita explorar exatamente esse nível mais superficial da comparação textual. Em vez de operar diretamente sobre caracteres um a um, como ocorre na distância de Levenshtein, foram utilizados principalmente os métodos baseados em tokens fornecidos pelo módulo *fuzz*, em especial o *token_set_ratio*, que reduz a influência da ordem das palavras e lida razoavelmente bem com abreviações e redundâncias presentes em endereços reais.

Dentro do módulo rapidfuzz_module.py, o funcionamento segue uma lógica que mistura essa teoria com demandas muito concretas do problema. Assim que o endereço entra normalizado, a biblioteca é acionada por meio da função process.extract, que executa uma busca aproximada apenas entre os registros mais promissores do CNEFE. Esse comportamento, denominado fuzzy apenas nos melhores candidatos, teve impacto direto no desempenho. Em vez de comparar cada consulta com todo o conjunto de endereços, a análise é limitada a um número reduzido de candidatos recuperados por esse primeiro filtro lexical. Na prática, esse corte inicial não apenas reduz o tempo de execução, mas também ajuda a isolar casos realmente próximos, algo que facilita o cálculo posterior do score final.

O RapidFuzz retorna, para cada correspondência, a similaridade lexical do logradouro. A partir daí, o sistema calcula os demais componentes da comparação e combina tudo em

um score único. Essa composição ponderada, que atribui pesos diferentes ao logradouro, ao número e ao bairro, foi uma forma de fazer com que a comparação lexical dialogasse melhor com as particularidades dos endereços brasileiros. A biblioteca oferece a métrica bruta, e o sistema ajusta o comportamento final considerando o que é mais relevante para definir se dois registros podem, de fato, apontar para o mesmo ponto no território.

Depois de calculados todos os scores, o conjunto de candidatos passa por um ordenamento simples e, em seguida, por dois procedimentos que se mostraram bastante úteis ao longo do desenvolvimento. O primeiro foi o tratamento dos empates. Em várias situações, dois ou mais registros obtinham exatamente o mesmo score final, devido a diferenças em colunas que não são relevantes para o objetivo do CulturaEduca, como o complemento do endereço ou outras informações auxiliares. Nesses casos, em vez de escolher um candidato automaticamente, optou-se por registrar todos os registros que obtiveram o score máximo para um dado endereço de entrada. Isso significa que, quando múltiplos registros (candidatos) possuem o mesmo score máximo, eles serão registrados para um processo subsequente de desempate, o qual será detalhado na Seção 2.5. Para os endereços que não apresentaram empate, o processo segue normalmente. Essa decisão teve um efeito positivo na análise, pois permitiu observar se o método não apenas acertava o endereço correto, mas também se era capaz de incluir registros incorretos entre os escolhidos. Isso ampliou bastante nossa compreensão dos limites da abordagem lexical.

O segundo procedimento foi o *override* associado ao número exato. Em algumas situações, a similaridade lexical do logradouro fazia um candidato se destacar, mas outro apresentava o número correto do imóvel. Como o número é uma informação muito específica, esse candidato passou a ser priorizado sempre que sua pontuação estivesse muito próxima da melhor pontuação geral. Esse mecanismo de ajuste fino trouxe previsibilidade para casos em que a métrica lexical, isoladamente, não fornecia o melhor resultado.

Por fim, o módulo utiliza paralelização por *threads* para distribuir as consultas entre vários núcleos de processamento. Essa estratégia não altera a lógica da comparação, mas torna o uso da biblioteca mais compatível com o volume de dados a ser analisado.

A atuação do RapidFuzz, no conjunto, reflete exatamente o que caracteriza a abordagem lexical apresentada na fundamentação teórica. O método se concentra na forma das palavras, na presença ou ausência de tokens, no grau de sobreposição entre expressões. Ele funciona bem sempre que a divergência entre dois registros se limita à grafia ou à estrutura superficial. Ao mesmo tempo, a biblioteca deixa claro onde esse tipo de comparação começa a perder precisão, o que reforça a necessidade das demais abordagens incorporadas ao sistema. Ainda assim, o RapidFuzz desempenhou um papel central na fase de experimentação, e boa parte da arquitetura posterior foi moldada a partir das observações realizadas sobre o comportamento desse primeiro módulo.

2.4.2 Abordagem baseada em indexação: Elasticsearch

O Elasticsearch passou a integrar o sistema quando ficou claro que parte das limitações observadas na comparação puramente lexical estava relacionada ao tamanho da base e à forma como os dados são estruturados. A ferramenta oferece uma lógica bem diferente da adotada pelo RapidFuzz. Em vez de avaliar cada endereço diretamente contra toda a

coleção, o Elasticsearch se apoia em um índice invertido que reduz o conjunto de candidatos logo no início da consulta. De certa forma, isso cumpre um papel semelhante ao filtro inicial que foi usado no módulo lexical, embora aqui ele seja realizado pela própria infraestrutura de busca da ferramenta.

Antes mesmo de realizar a primeira consulta, o módulo responsável pela indexação percorre toda a base do CNEFE, aplica os analisadores configurados e cria um índice invertido que organiza cada logradouro e cada bairro em unidades menores. Esse processo, realizado apenas uma vez, deixa a base preparada para consultas rápidas e direcionadas.

A partir disso, a busca não opera sobre os textos brutos, mas sobre o índice já construído. Os analisadores textuais quebram os endereços em tokens que se tornam as chaves de acesso ao índice. A recuperação dos documentos passa a depender dessa representação reduzida e da forma como esses tokens aparecem distribuídos nos registros. O ranqueamento BM25 entra exatamente nesse ponto. Ele estima a relevância de cada documento com base na frequência dos termos, na raridade relativa de cada token e no equilíbrio entre campos curtos e longos. Com o uso desse indíce, o Elasticsearch conseguiu manter um bom desempenho mesmo com um volume mais extenso de dados.

Com o índice criado e o modelo de ranqueamento operando sobre ele, as consultas seguem uma lógica progressiva voltada a identificar, com alguma eficiência, o melhor candidato para cada endereço. A primeira rodada aciona buscas do tipo *match* com *fuzziness* automático e *match phrase* com um valor reduzido de *slop*. Essas consultas procuram capturar rapidamente registros que já apresentam uma sobreposição razoável de tokens, tolerando pequenas divergências ortográficas ou variações limitadas na ordem dos termos. Quando essas tentativas iniciais não retornam resultados satisfatórios, o módulo expande o escopo da busca utilizando estratégias como *cross fields* e *most fields*. Essas modalidades exploram a tokenização aplicada durante a indexação e permitem combinar informações provenientes de múltiplos campos textuais, como logradouro e bairro, de forma mais flexível, ampliando o conjunto de candidatos recuperados.

As principais estratégias de consulta utilizadas podem ser resumidas da seguinte forma:

- ***Match***: consulta padrão do Elasticsearch baseada no modelo vetorial, na qual o texto de entrada é analisado e comparado com os termos indexados em um campo específico. O uso de *fuzziness* automático permite a recuperação de termos com pequenas variações de grafia, o que é especialmente relevante em dados de endereços, frequentemente sujeitos a erros de digitação ou abreviações.
- ***Match phrase***: variante da consulta *match* que exige que os termos apareçam na mesma ordem em que foram informados na consulta. O parâmetro *slop* controla o número máximo de deslocamentos permitidos entre os termos, possibilitando pequenas variações na sequência das palavras sem perder completamente a noção de frase.
- ***Cross fields***: estratégia de consulta voltada à busca em múltiplos campos simultaneamente, tratando-os como se fossem um único campo lógico. Essa abordagem é útil quando a informação relevante pode estar distribuída entre diferentes atributos textuais, como logradouro e bairro, permitindo que os termos da consulta sejam

combinados entre esses campos.

- ***Most fields***: modalidade de consulta que executa a busca separadamente em vários campos e combina os resultados, somando os scores obtidos em cada um deles. Essa estratégia tende a favorecer registros que apresentam correspondência em mais de um campo, mesmo que de forma parcial, ampliando a recuperação de candidatos em cenários de maior variação textual.

O número do imóvel também precisou de um tratamento próprio. Esse campo não se comporta como um termo textual comum, algo que já havia ficado claro nos testes com o RapidFuzz. No Elasticsearch, essa informação foi incorporada ao ranqueamento por meio de uma ponderação gaussiana, que reforça a proximidade entre números e dá ainda mais peso quando há coincidência exata. A lógica geral lembra o que foi feito no módulo lexical, embora a implementação siga caminhos diferentes. Na prática, quando o número coincide ou está muito próximo, ele tende a influenciar de maneira perceptível a ordenação final dos candidatos.

Outra etapa importante foi lidar com registros repetidos que apareciam no índice. O uso do *collapse_on_code* ajudou a enxugar as listas de retorno, evitando que versões equivalentes do mesmo endereço ocupassem espaço entre os resultados. Após a recuperação, os scores foram normalizados tomando como referência o melhor candidato, seguindo a mesma ideia aplicada no RapidFuzz para facilitar comparações internas.

Os empates, que já eram tratados explicitamente no módulo lexical, também aparecem aqui. Sempre que vários registros atingem pontuações equivalentes após a combinação dos sinais textuais e numéricos, mantemos todos no resultado, o que ajuda a identificar situações ambíguas.

De modo geral, o Elasticsearch oferece uma forma distinta de abordar o problema. Ele se apoia na estrutura do índice para recuperar rapidamente os candidatos mais prováveis e combina diferentes indícios extraídos do texto, não apenas a forma literal das palavras. Sua atuação complementa a comparação lexical. Enquanto o RapidFuzz examina a escrita de perto, caractere a caractere ou token a token, o Elasticsearch reorganiza o texto para tornar a busca mais eficiente e recuperar um conjunto consistente de possibilidades, mesmo em coleções extensas.

2.4.3 Abordagem semântica: LLM

A introdução de um método semântico ocorreu quando ficou claro que as abordagens lexical e baseada em indexação já tinham alcançado o limite do que conseguiam resolver. Mesmo depois da normalização e de vários ajustes, ainda persistiam casos em que o endereço correto não aparecia entre os melhores candidatos. Nessas situações, as diferenças não vinham apenas de abreviações, erros de grafia ou trocas simples de termos. Muitas vezes, a forma como o endereço era construído variava de maneira mais profunda entre bases distintas, embora apontassem para o mesmo ponto no território. Esse tipo de variação não se expressa apenas em caracteres, tokens ou pela estrutura do índice invertido. Diante desse cenário, fez sentido incluir uma estratégia que olhasse para o significado das expressões.

O método semântico foi implementado usando a biblioteca *SentenceTransformers*, que disponibiliza modelos pré-treinados para geração de *embeddings*. Entre as alternativas disponíveis, foi selecionado um modelo da família MiniLM, especificamente o paraphrase-MiniLM-L3-v2. Ele produz vetores densos de 384 dimensões que procuram representar o conteúdo semântico completo de cada logradouro, bairro e número. A lógica se distancia das técnicas anteriores, porque o modelo não compara palavras isoladas, mas tenta representar a relação entre elas dentro de uma frase completa. Com isso, diferentes maneiras de escrever o mesmo lugar acabam projetadas em regiões próximas no espaço vetorial. Um nome abreviado, reordenado ou incompleto tende a permanecer semanticamente associado ao original, algo que não aparecia com a mesma clareza nas abordagens anteriores.

A forma como o número do imóvel era tratado também exigiu uma mudança de abordagem. Nos métodos baseados em comparação textual direta, a análise se concentrava sobretudo no logradouro e no bairro, enquanto o número precisava ser avaliado por uma função numérica separada, já que RapidFuzz e Elasticsearch interpretam esse campo apenas como texto. Sem essa função numérica, os métodos apresentavam uma limitação que gerava distorções, como a falsa semelhança entre valores que apenas compartilham grafia, caso de “1” e “10”.

No método semântico, o tratamento do número passou a funcionar de outro modo. Como o modelo MiniLM produz embeddings que capturam relações numéricas, tornou-se viável incorporar o número diretamente ao algoritmo, sem recorrer a cálculos externos. Números próximos tendem a se concentrar em regiões semelhantes do espaço vetorial, enquanto valores mais distantes se separam naturalmente. Dessa forma, o próprio modelo consegue distinguir corretamente proximidades reais, como entre “1” e “2”, e afastar combinações como “1” e “10”. Isso eliminou a necessidade de uma função adicional para comparar números, exigida nas abordagens lexical e indexada, já que ali o valor era interpretado apenas como texto.

A inclusão dos embeddings numéricos deixou claro que o método semântico exigiria um mecanismo capaz de lidar com milhões de vetores de forma eficiente. A primeira tentativa foi utilizar a biblioteca HNSWlib, que organiza os embeddings em grafos hierárquicos e permite buscas aproximadas muito rápidas. O desempenho era bom enquanto a base permanecia pequena, porém o índice completo precisava ficar inteiramente na memória. Como cada registro do CNEFE passou a armazenar três embeddings independentes, referentes ao logradouro, ao número e ao bairro, o volume total rapidamente ultrapassou a capacidade de RAM disponível. O processo chegava a iniciar, mas o tempo de construção do índice aumentava de forma desproporcional e, em vários testes, a execução simplesmente não se completava. Diante dessa inviabilidade prática, tornou-se necessário adotar uma solução mais estável, o que levou à migração para o PostgreSQL com a extensão pgvector.

Nessa nova configuração, os embeddings produzidos em Python passam a ser armazenados diretamente no PostgreSQL, que passa a concentrar a parte pesada das buscas. A extensão pgvector adiciona ao banco um tipo próprio para vetores e operadores de distância, incluindo o `<->`, utilizado para medir a proximidade entre embeddings. Assim, o Postgres fica encarregado de guardar os vetores e executar as consultas de vizinhança, devolvendo ao módulo apenas os candidatos mais próximos.

Mesmo lidando com uma quantidade grande de dados, o desempenho se tornou muito

mais estável do que na solução baseada em HNSWlib. O banco não precisa manter todo o índice em memória e consegue responder às consultas de forma consistente, ainda que com algum custo de tempo. Com isso, o sistema consegue recuperar de forma mais eficiente os registros relevantes, o que tornou o método escalável para bases maiores.

Depois que o banco retorna os primeiros candidatos, o módulo segue para o cálculo das similaridades detalhadas. Esses candidatos iniciais são simplesmente os endereços do CNEFE que aparecem mais próximos no espaço vetorial, segundo a distância aplicada pelo pgvector. Eles funcionam como um filtro inicial, reduzindo a comparação apenas aos registros que o modelo considera mais plausíveis para aquela entrada.

Com esse conjunto reduzido, o sistema calcula a similaridade do logradouro usando a distância de cosseno entre os embeddings, convertida depois para uma escala de zero a cem. O mesmo é feito para o bairro e o número. Em seguida, as três similaridades são combinadas por uma ponderação simples. O logradouro recebe o maior peso, o número ajusta a decisão quando está bem definido e o bairro funciona como apoio contextual.

Depois de obtida a pontuação final, o módulo seleciona o candidato com a maior similaridade. Se vários registros atingem exatamente esse valor máximo, o sistema mantém todos os empates. Entre eles, o primeiro retorno do banco é adotado como correspondência principal, e os demais só são mantidos quando apresentam o mesmo logradouro, número e bairro. Empates desse tipo são comuns em situações nas quais um mesmo endereço corresponde a mais de um registro no CNEFE, como ocorre em edifícios ou conjuntos habitacionais. Quando há diferenças nesses campos, apenas o candidato principal permanece. Dessa forma, o método registra o melhor resultado sem descartar empates que correspondem ao mesmo endereço.

Com isso, a abordagem semântica passa a integrar efetivamente o conjunto de métodos avaliados ao longo do sistema. Enquanto o RapidFuzz trabalha essencialmente sobre a forma das palavras e o Elasticsearch explora a estrutura de indexação para recuperar candidatos prováveis, o modelo de linguagem acrescenta uma camada de interpretação que aproxima expressões mesmo quando a escrita diverge de maneira mais profunda. Isso permite observar, lado a lado, como representações lexicais, estruturais e semânticas se comportam diante do mesmo conjunto de endereços e em quais cenários cada abordagem se mostra mais adequada para vincular registros ao CNEFE.

2.5 Avaliação espacial dos resultados no PostgreSQL

Depois que os três métodos de correspondência são executados, cada um produz um arquivo `CSV` que reúne, para cada registro da base de entrada, o endereço original e as informações do candidato encontrado no CNEFE. Esses arquivos já representam o resultado final de cada algoritmo, pois trazem tanto o endereço localizado quanto as pontuações calculadas e as coordenadas associadas ao ponto estimado.

A partir desse ponto, o objetivo deixa de ser comparar textos. Agora a questão é analisar se esses resultados são de fato bons em termos espaciais. Para isso, foi utilizado o

PostgreSQL com a extensão PostGIS, que oferece funções geoespaciais capazes de identificar em que setor censitário cada ponto caiu, medir distâncias em metros e avaliar a coerência entre o ponto estimado e o setor considerado correto na base de entrada.

Essa etapa de análise espacial exige informações que não fazem parte do funcionamento interno dos algoritmos de correspondência. Enquanto os módulos trabalham exclusivamente com logradouro, número e bairro, a avaliação posterior depende de dados adicionais que acompanham a base de entrada, como as coordenadas verdadeiras e o código do setor censitário considerado correto para cada endereço. O objetivo é verificar se o ponto estimado por cada método se aproxima do ponto de referência e se cai no setor adequado.

O PostgreSQL passa então a operar sobre dois conjuntos de informações. De um lado estão os arquivos `CSV` gerados por cada abordagem, que trazem o candidato encontrado no CNEFE e suas coordenadas estimadas. Do outro está a base geográfica dos setores censitários, carregada no PostGIS, que permite identificar em qual setor cada ponto realmente caiu por meio das funções de interseção espacial.

Com os resultados carregados no banco, o primeiro passo consiste em criar uma referência geométrica para cada endereço da base de entrada. Como alguns métodos podem produzir mais de um candidato para o mesmo `idx_df1`, o PostgreSQL agrupa esses registros e calcula a mediana da latitude e da longitude usando `percentile_cont(0.5)`.

Depois de calculado o ponto mediano de cada grupo de candidatos, o PostgreSQL computa duas distâncias essenciais para cada registro. A primeira mede o quanto o ponto estimado, isto é, o ponto do CNEFE retornado pelo método, se afasta dessa mediana, usando `ST_DistanceSphere`. Essa medida serve para identificar qual candidato é espacialmente mais coerente dentro do grupo, então quanto menor a distância à mediana, maior a consistência em relação aos demais pontos gerados para aquele endereço. Em seguida, calcula-se a distância entre o ponto estimado e o ponto verdadeiro da base de entrada, que representa a localização real utilizada como referência. Essa segunda medida quantifica diretamente o erro espacial associado ao candidato. Ambas as distâncias são utilizadas na etapa seguinte de ordenação e escolha do melhor resultado.

Depois das distâncias internas serem calculadas, o PostgreSQL passa a avaliar a relação entre o ponto estimado e o setor censitário considerado correto. O ponto estimado corresponde às coordenadas do registro do CNEFE escolhido por cada método, enquanto o ponto verdadeiro é o ponto que acompanha a base de entrada. A comparação começa verificando se o setor associado ao ponto estimado (`cd_setor_resultante`) coincide com o setor verdadeiro informado na entrada (`cd_setor_verdadeiro`). Quando ambos coincidem, considera-se que o método acertou exatamente o setor. Caso contrário, calcula-se a distância em metros entre o ponto estimado e o polígono do setor verdadeiro por meio de `ST_Distance`, o que quantifica o desvio espacial do candidato em relação à área correta. Além disso, o sistema verifica se o setor retornado é vizinho do setor verdadeiro utilizando `ST_Touches`. Essa verificação distingue situações em que o método errou apenas por uma fronteira, caindo em um setor imediatamente adjacente, de casos em que o ponto estimado se encontra mais distante da região esperada. Essas informações complementam a análise ao indicar não apenas se houve acerto, mas também a qualidade geométrica do erro quando ele ocorre.

Com todas as informações calculadas, o banco aplica uma lógica de seleção para escolher, entre todos os candidatos associados a cada entrada, aquele considerado mais consistente. Isso é feito por meio de uma cláusula `ROW_NUMBER()` combinada a uma ordenação que segue uma sequência de critérios. Primeiro, ganha prioridade o candidato cuja posição está mais próxima da mediana do grupo, já que esse ponto tende a representar melhor o conjunto de resultados gerados para aquele endereço. Em seguida, o sistema favorece registros que efetivamente caíram em algum setor censitário. Só depois entram os indicadores textuais produzidos pelos métodos de correspondência, como a similaridade final e, em menor peso, as similaridades individuais de logradouro, número e bairro. Caso ainda permaneçam empates, o banco utiliza o identificador interno `ctid` apenas para garantir uma escolha consistente entre registros que, do ponto de vista espacial e textual, são indistinguíveis.

O processo encerra com a criação de uma tabela consolidada que reúne, para cada endereço da base de entrada, o candidato escolhido segundo os critérios espaciais definidos no PostgreSQL, além de todas as métricas calculadas ao longo da análise. Essa tabela funciona como um resumo organizado do comportamento dos métodos, reunindo desde as distâncias medidas até as verificações de setor. É esse conjunto final de informações que alimenta o capítulo de resultados, no qual as três abordagens passam a ser comparadas em termos de precisão geográfica e coerência territorial.

Capítulo 3

Resultados

Os resultados apresentados neste capítulo partem das saídas produzidas pelos diferentes métodos de correspondência e posteriormente analisadas no PostgreSQL. Para cada abordagem, esse processo resultou em uma tabela consolidada que reúne não apenas as informações originalmente retornadas pelos algoritmos, como o endereço encontrado, as coordenadas geográficas e o setor censitário associado, mas também métricas adicionais obtidas na etapa de análise espacial, como distâncias entre pontos e a verificação de correspondência com o setor censitário considerado correto na base de entrada.

Esse conjunto de informações permite avaliar o desempenho das abordagens a partir de critérios que vão além da correspondência textual. Mais do que verificar se um endereço foi corretamente identificado do ponto de vista textual, é fundamental analisar se o ponto retornado se encontra no setor censitário correto. No contexto do CulturaEduca, essa informação é essencial, pois as análises territoriais da plataforma dependem diretamente da correta associação entre endereços e setores censitários. Por esse motivo, o desempenho dos algoritmos foi avaliado principalmente pela capacidade de localizar os endereços na região correta.

Para fazer essa análise, os resultados foram explorados por meio de gráficos gerados em Python, utilizando as bibliotecas `pandas`, `matplotlib` e `seaborn`. Esses gráficos construídos a partir das análises realizadas com o PostGIS, permitem observar padrões que não são evidentes a partir das tabelas resultantes dos algoritmos apenas.

A distribuição do erro de distância entre o ponto verdadeiro e o ponto estimado foi analisada por meio de gráficos de densidade (*KDE*), que ajudam a entender não apenas se houve acerto, mas também o quão distante o resultado ficou quando o erro ocorre. Para analisar a relação entre os pontos estimados e os setores censitários, foram utilizados gráficos de rosca. Esse tipo de visualização permite observar a proporção de registros que caem no setor censitário correto, em um setor vizinho ou em um setor incorreto. Dessa maneira, é possível avaliar a qualidade espacial das correspondências produzidas por cada método.

Para avaliar os resultados obtidos, utilizou-se três bases de dados provenientes de diferentes regiões do Brasil: o município de Diadema, o estado de Rondônia e o município

de São Paulo. Essas bases foram escolhidas por apresentarem diferenças significativas na quantidade de endereços avaliados, o que permite observar como as abordagens se comportam tanto em conjuntos menores quanto em bases extensas e mais complexas.

Além das abordagens desenvolvidas, os resultados do GeoCodeBR também foram incluídos nas análises. O GeoCodeBR havia sido utilizado anteriormente no contexto do CulturaEduca, mas apresentou limitações práticas, tanto na configuração quanto na qualidade dos resultados em alguns cenários. A comparação permite, portanto, avaliar se a solução proposta consegue superar essas limitações e se mostrar uma alternativa mais adequada para a geocodificação de endereços brasileiros. Para garantir uma avaliação consistente, os resultados do GeoCodeBR foram convertidos para o mesmo formato das demais abordagens e submetidos às mesmas métricas e validações espaciais no PostgreSQL.

Nas seções seguintes, os resultados são analisados separadamente para cada município, com a discussão dos padrões observados, das diferenças entre as abordagens e do que esses comportamentos indicam para o problema de vincular endereços brasileiros ao CNEFE.

3.1 Resultados para Diadema

Os resultados apresentados para Diadema utilizam como base de entrada os endereços de estabelecimentos de saúde do município, provenientes do CNES. Esses registros foram verificados manualmente, o que garante a confiabilidade do conjunto como referência de validação.

Essa base teve um papel fundamental no desenvolvimento do sistema, pois foi a partir dela que as primeiras versões do código foram testadas, permitindo observar se o algoritmo indicava o endereço correto. Sempre que surgiam inconsistências, a lógica de comparação era revista e aprimorada, implementando novos tratamentos até que os endereços fossem corretamente associados.

Nesse cenário, os endereços do CNES de Diadema foram utilizados como base de entrada, enquanto o CNEFE do próprio município serviu como conjunto de referência para a correspondência.

O gráfico de densidade para Diadema indica que RapidFuzz, Elasticsearch e o método semântico apresentam comportamentos muito semelhantes, com curvas estreitas e fortemente concentradas próximas de erro zero. Esse padrão sugere que, quando esses métodos retornam um ponto, a estimativa tende a ser espacialmente precisa, com pouca variação entre os resultados.

O GeoCodeBR também concentra parte de sua distribuição próxima de zero, mas exibe um espalhamento maior, com uma cauda longa associada a erros mais elevados. Isso evidencia um comportamento menos estável, no qual o método acerta em diversos casos, mas produz desvios significativos com maior frequência em comparação aos demais.

De forma geral, Diadema configura um cenário no qual as abordagens baseadas em similaridade textual e o método semântico alcançam níveis de precisão bastante próximos, enquanto o GeoCodeBR apresenta maior variabilidade na localização dos pontos

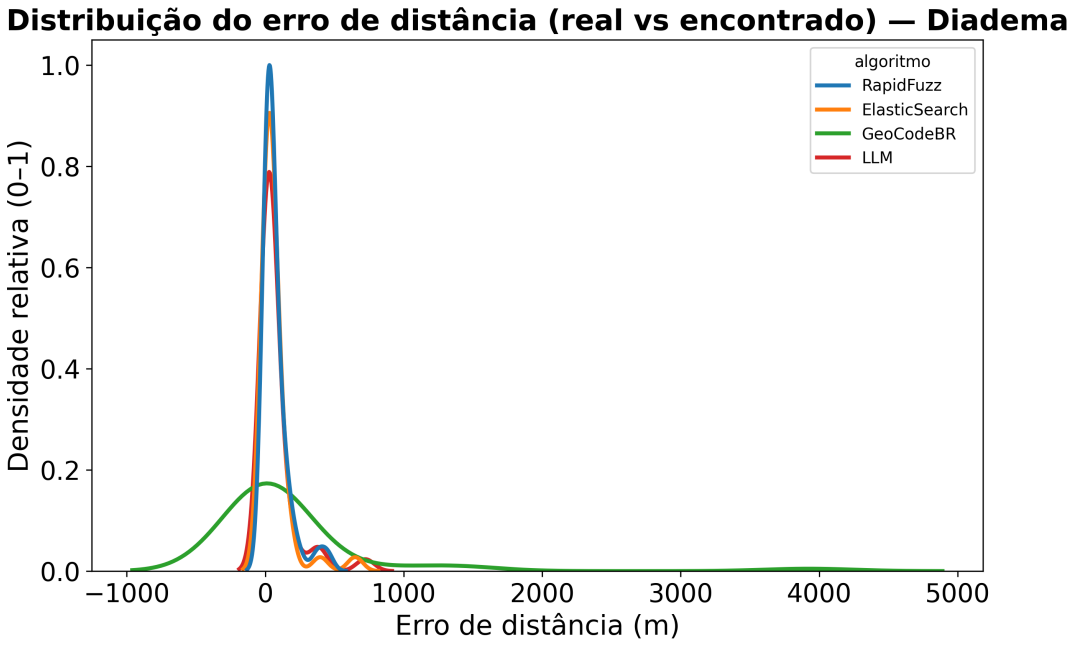

Figura 3.1: *Distribuição do erro de distância entre o ponto verdadeiro e o ponto estimado pelos quatro métodos — Gráfico de densidade: Diadema.*

estimados.

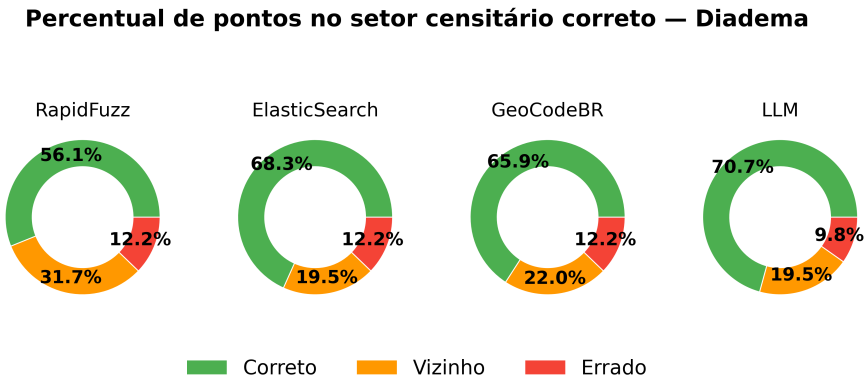

Figura 3.2: *Proporção de pontos localizados no setor correto, em setor vizinho ou setor incorreto — Gráfico de rosca: Diadema.*

Os gráficos de rosca para Diadema mostram que os métodos apresentam desempenhos bastante próximos na identificação do setor censitário correto. As taxas de acerto direto variam dentro de um intervalo pequeno, o que indica que, nesse conjunto de dados menor, não há diferenças muito acentuadas entre as abordagens.

O LLM se destaca por apresentar o maior percentual de registros localizados exatamente no setor correto e, ao mesmo tempo, a menor taxa de erros. Enquanto RapidFuzz, Elasticsearch e GeoCodeBR registram 12,2% de casos classificados como incorretos, o LLM reduz esse valor para 9,8%, indicando maior consistência na escolha do ponto final.

Entre os métodos textuais, o Elasticsearch alcança um desempenho mais próximo ao do LLM, seguido pelo GeoCodeBR. O RapidFuzz apresenta a menor taxa de acertos, mas concentra uma parcela relevante dos resultados na categoria de setores vizinhos, sugerindo que, mesmo quando não acerta exatamente o setor, tende a posicionar o ponto em regiões próximas à correta.

De forma geral, os resultados de Diadema indicam que todas as abordagens conseguem localizar os endereços com boa precisão. As diferenças observadas são pontuais e menos expressivas do que em bases maiores, o que mostra o papel desse município como um cenário adequado para avaliação inicial dos métodos.

3.2 Resultados para Rondônia

Os resultados para Rondônia utilizam uma base de endereços da área da educação, utilizada pelo projeto CulturaEduca. Trata-se de um conjunto maior do que o utilizado em Diadema, mas ainda consideravelmente menor do que bases extensas como a do município de São Paulo. Por esse motivo, Rondônia ocupa uma posição intermediária na avaliação, permitindo observar como os algoritmos se comportam quando aplicados a volumes mais elevados de dados, sem atingir ainda a complexidade de grandes centros urbanos.

Nesse caso, os endereços da base educacional de Rondônia foram utilizados como entrada, enquanto o CNEFE do próprio estado serviu como conjunto de referência para a correspondência.

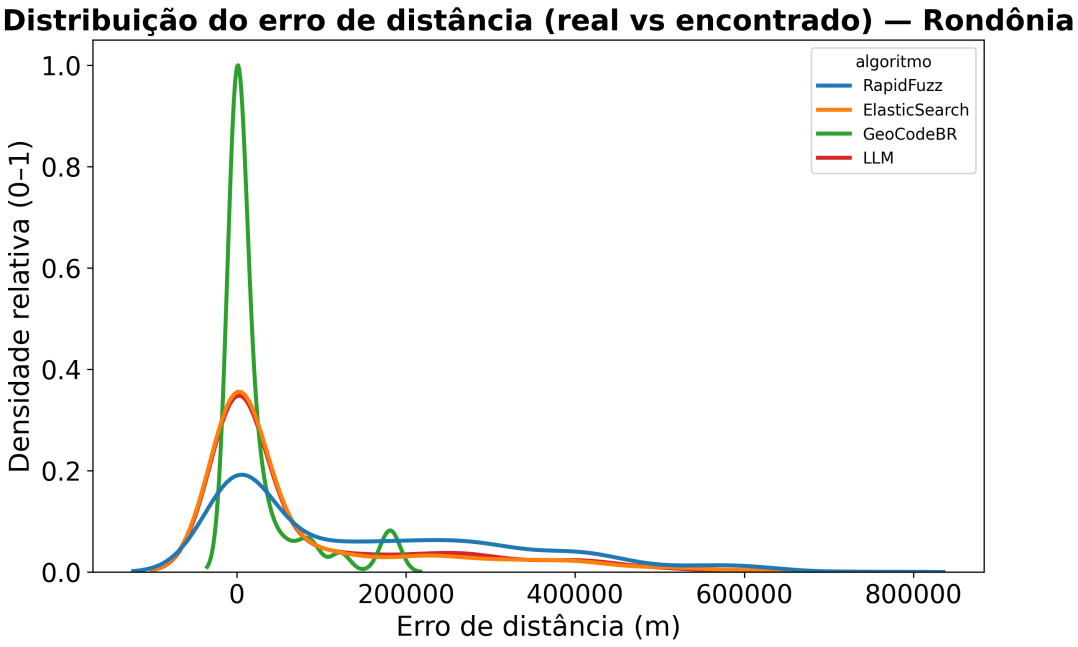

Figura 3.3: *Distribuição do erro de distância entre o ponto verdadeiro e o ponto estimado pelos quatro métodos — Gráfico de densidade: Rondônia.*

O gráfico de densidade para Rondônia revela um cenário mais heterogêneo do que o observado em Diadema. As distribuições são mais dispersas e apresentam caudas longas, o

que indica a ocorrência de erros espaciais mais elevados. Esse comportamento é coerente com a maior extensão territorial do estado e com a diversidade dos endereços presentes na base educacional utilizada.

O GeoCodeBR mostra uma concentração bastante acentuada próxima de erro zero, indicando que, nos casos em que a correspondência ocorre corretamente, os pontos retornados tendem a estar muito próximos da localização verdadeira. Ainda assim, a distribuição apresenta mudanças mais irregulares ao longo do eixo de distância, o que indica instabilidade em parte dos registros e a ocorrência de erros mais elevados em alguns casos.

RapidFuzz, Elasticsearch e LLM apresentam distribuições mais suaves e espalhadas ao longo do eixo de distância. Esses métodos concentram boa parte dos resultados em erros menores, mas também exibem uma variedade maior de desvios espaciais, o que indica dificuldades adicionais na correspondência de endereços em bases mais complexas.

De forma geral, Rondônia configura um cenário mais desafiador, no qual as diferenças entre os métodos se tornam mais evidentes devido à escala territorial e à variabilidade dos dados.

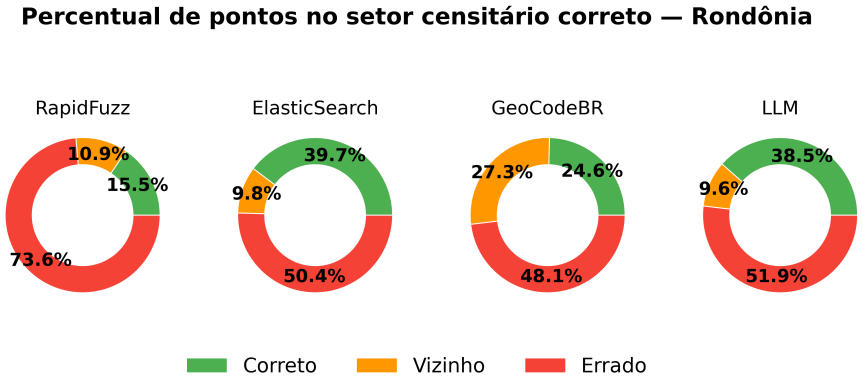

Figura 3.4: *Proporção de pontos localizados no setor correto, em setor vizinho ou setor incorreto — Gráfico de rosca: Rondônia.*

O gráfico de rosca para Rondônia confirma que esse conjunto apresenta um nível de dificuldade maior do que o observado em Diadema. Em todos os métodos, há uma proporção elevada de pontos classificados como incorretos, indicando que muitos endereços foram posicionados fora do setor censitário esperado.

O RapidFuzz apresenta o pior desempenho nesse cenário, com predominância de erros e baixa taxa de acertos diretos, o que sugere limitações da abordagem puramente lexical quando aplicada a bases mais extensas e heterogêneas.

Elasticsearch e LLM exibem resultados semelhantes, com cerca de 40% de acertos no setor correto e uma pequena parte de casos classificados como vizinhos. Ainda assim, ambos mantêm uma quantidade significativa de erros, evidenciando dificuldades diante da variabilidade espacial e textual dos dados.

O GeoCodeBR se destaca pela maior proporção de registros classificados como vizinhos, indicando que frequentemente posiciona os pontos próximos ao setor correto, em-

bora sem identificá-lo com precisão. Esse comportamento é compatível com o observado no gráfico de densidade.

Em conjunto, os resultados mostram que Rondônia constitui um cenário mais exigente do que o observado em Diadema e as diferenças entre os métodos se tornam mais evidentes.

3.3 Resultados para São Paulo

Os resultados para o município de São Paulo utilizam uma base de endereços da área da educação, também empregada pelo projeto CulturaEduca. Diferentemente de Diadema e de Rondônia, esse conjunto é composto por milhões de registros, o que traz uma complexidade muito maior tanto do ponto de vista do volume de dados quanto da diversidade na forma como os endereços aparecem descritos.

Essa característica faz com que São Paulo represente o cenário mais exigente considerado neste trabalho. O grande número de registros e a heterogeneidade dos endereços aproximam a avaliação de uma situação real de uso em larga escala, na qual erros textuais, variações de grafia e inconsistências espaciais tendem a impactar de forma mais significativa o desempenho dos métodos.

Nesse contexto, os endereços da base educacional do município foram utilizados como entrada dos algoritmos, enquanto o CNEFE de São Paulo serviu como base de referência para a correspondência espacial e para a verificação do setor censitário associado a cada ponto retornado.

Para esse município, os resultados do método semântico baseado em LLM não puderam ser incluídos na análise. A geração dos embeddings e a realização das consultas vetoriais exigiriam um volume de processamento e memória muito superior ao disponível nas máquinas utilizadas. Portanto, no contexto deste trabalho, a execução do método tornou-se inviável do ponto de vista computacional.

Assim, as análises apresentadas para São Paulo consideram apenas as abordagens RapidFuzz, Elasticsearch e GeoCodeBR. Ainda assim, o município oferece um cenário fundamental para observar como métodos baseados em similaridade textual e indexação se comportam quando aplicados a um ambiente urbano extenso e com grande variabilidade nos dados.

O gráfico de densidade para São Paulo evidencia diferenças mais claras entre as abordagens, o que é esperado diante da escala e da complexidade do município. O Elasticsearch se destaca por apresentar uma curva bastante concentrada, próxima de erro zero, indicando que, nos casos em que consegue localizar corretamente o endereço, o ponto retornado tende a estar muito próximo da localização verdadeira.

O RapidFuzz também concentra uma parcela relevante dos resultados em erros baixos, mas exibe uma cauda mais extensa. Esse comportamento revela a presença de desvios espaciais maiores em parte dos registros e sugere maior sensibilidade a variações textuais e ambiguidades.

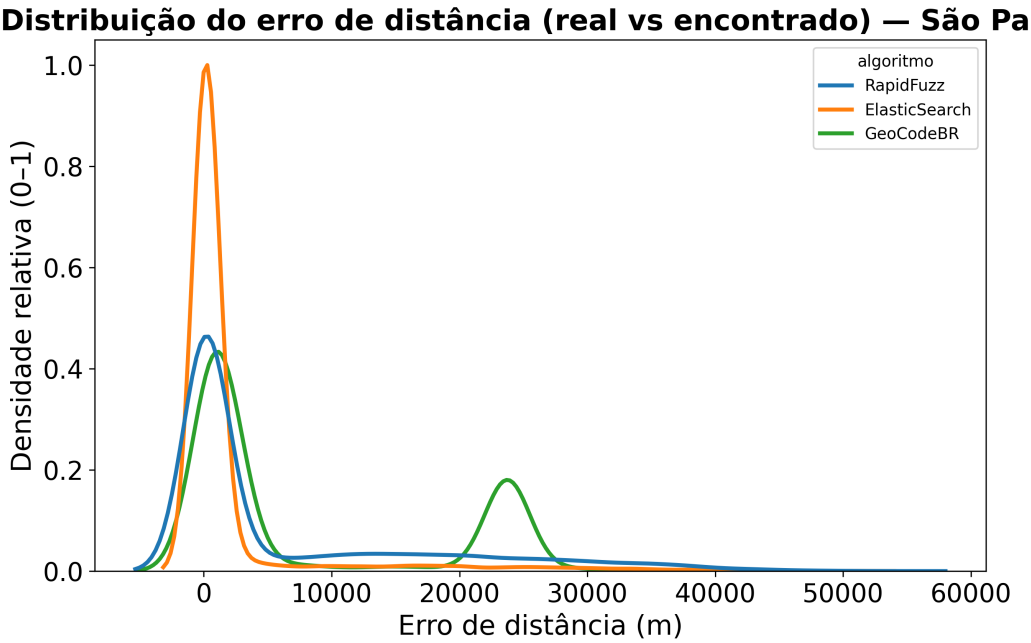

Figura 3.5: *Distribuição do erro de distância entre o ponto verdadeiro e o ponto estimado pelos quatro métodos — Gráfico de densidade: Cidade de São Paulo.*

O GeoCodeBR, por sua vez, apresenta uma distribuição mais dispersa, com a presença de um segundo pico em erros mais elevados. Esse padrão indica que, embora o método acerte em muitos casos, há situações recorrentes em que o ponto retornado se afasta consideravelmente da localização esperada, possivelmente em função de ambiguidades de logradouro ou da dificuldade em diferenciar endereços próximos.

De modo geral, os resultados para São Paulo mostram que o aumento da escala e da complexidade urbana torna mais evidentes as diferenças entre os métodos.

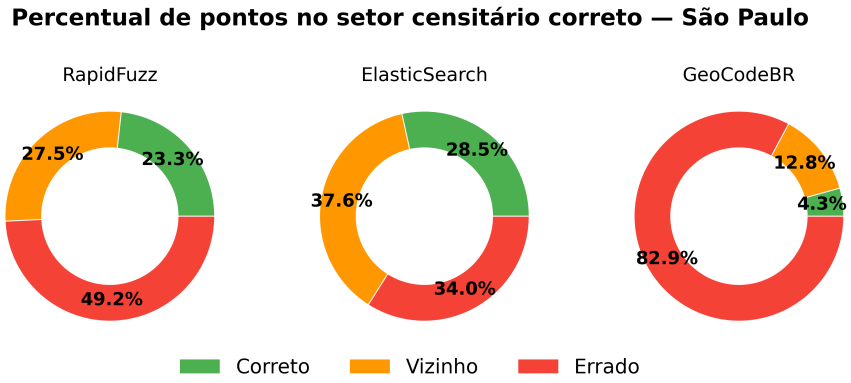

Figura 3.6: *Proporção de pontos localizados no setor correto, em setor vizinho ou setor incorreto — Gráfico de rosca: Cidade de São Paulo.*

O gráfico de rosca para São Paulo evidencia o efeito da escala e da complexidade urbana sobre o desempenho dos métodos. O Elasticsearch apresenta o resultado mais equili-

brado, com a maior proporção de acertos diretos no setor censitário correto e uma parcela relevante de registros em setores vizinhos, o que indica que o método costuma posicionar os pontos na região correta, mesmo quando não acerta exatamente o setor.

O RapidFuzz apresenta um desempenho intermediário. Embora concentre parte dos registros nas categorias correto e vizinho, ainda mantém uma proporção significativa de erros, refletindo as limitações de abordagens textuais em um contexto com grande diversidade de logradouros.

O GeoCodeBR apresenta o pior desempenho nesse cenário, com predominância de registros classificados como incorretos e uma taxa muito reduzida de acertos. Esse resultado é consistente com o comportamento observado no gráfico de densidade, no qual o GeoCodeBR apresenta uma distribuição mais dispersa e um segundo pico em erros mais elevados. Isso indica que, embora o método acerte em parte dos casos, há ocorrências em que o ponto retornado se afasta consideravelmente da localização esperada, o que impacta diretamente a identificação correta do setor censitário.

De forma geral, os resultados indicam que, em São Paulo, a vinculação correta de endereços ao setor censitário se torna significativamente mais desafiadora, evidenciando as limitações das abordagens avaliadas quando aplicadas a bases extensas e heterogêneas.

3.4 Comparação geral entre as abordagens

A comparação dos resultados obtidos em Diadema, Rondônia e São Paulo evidencia como o comportamento das abordagens varia conforme aumentam a escala e a complexidade dos dados. Em bases menores, os métodos tendem a apresentar desempenhos semelhantes, mas as diferenças tornam-se mais claras à medida que o volume de endereços cresce e a heterogeneidade aumenta.

Em bases menores, como Diadema, os métodos apresentam resultados semelhantes, com erros baixos e taxas de acerto de setores censitários próximas, indicando que diferentes estratégias conseguem produzir correspondências consistentes em cenários mais controlados.

Em Rondônia, que representa um cenário intermediário, as diferenças tornam-se mais visíveis. A dispersão dos erros aumenta e cresce a proporção de registros fora do setor correto. Nesse contexto, o RapidFuzz é mais sensível à variabilidade textual, enquanto Elasticsearch e o método semântico mantêm desempenho relativamente superior. O GeoCodeBR tende a posicionar os pontos próximos da região correta, mas com menor precisão para acertar os setores censitários.

São Paulo representa o cenário mais desafiador para as abordagens, pois o grande volume de dados e a diversidade dos endereços ampliam as diferenças entre os métodos. O Elasticsearch se destaca por apresentar o melhor equilíbrio entre acertos diretos e setores vizinhos, enquanto o RapidFuzz e o GeoCodeBR apresentam um aumento significativo de erros.

A ausência do método semântico em São Paulo evidencia uma limitação prática da abordagem. Apesar do bom desempenho em bases menores, o custo computacional invi-

abilizou sua execução em larga escala com a estrutura computacional disponível.

De forma geral, os resultados indicam que não há uma abordagem única que se destaque em todos os cenários. O RapidFuzz tende a funcionar bem em bases menores, enquanto a abordagem baseada em indexação tem uma maior estabilidade à medida que o volume de dados cresce. Nesse sentido, o Elasticsearch se destaca especialmente em cenários intermediários e de grande escala, mantendo um bom equilíbrio entre precisão espacial e viabilidade de execução. A abordagem semântica apresenta potencial em contextos mais controlados, embora ainda enfrente limitações computacionais em bases muito extensas. Esses resultados mostram que a escolha da estratégia de geocodificação depende diretamente do contexto de uso, sobretudo quando se busca apoiar análises territoriais em larga escala, como no CulturaEduca.

Capítulo 4

Conclusão

Este trabalho teve como objetivo investigar métodos de correspondência de endereços aplicados à geocodificação de dados públicos brasileiros, com foco na avaliação espacial dos resultados obtidos. A proposta central consistiu em analisar não apenas a similaridade textual entre endereços, mas principalmente a qualidade geométrica das soluções produzidas, considerando métricas como distância em metros e correspondência de setor censitário.

Foram implementadas e avaliadas três abordagens de correspondência: lexical, baseada em indexação e baseada em modelos de linguagem. Os resultados experimentais indicaram que a abordagem indexada, implementada com Elasticsearch, apresentou o melhor desempenho geral nos conjuntos de dados analisados. Esse método demonstrou maior eficiência e maior acurácia nos resultados, recuperando com mais frequência candidatos corretos e apresentando desvios geográficos menores em comparação às demais abordagens. A integração com um procedimento de desempate espacial no PostgreSQL, utilizando funcionalidades do PostGIS, mostrou-se fundamental para a escolha consistente do melhor candidato em cenários com múltiplos empates.

Além dos resultados experimentais, o trabalho se destaca pelo uso de ferramentas de código aberto. A solução foi desenvolvida como um software livre, com o objetivo de viabilizar uma API aberta de geocodificação, oferecendo uma alternativa às soluções pagas. Essa escolha reforça a viabilidade de construir sistemas de geocodificação precisos e reprodutíveis para aplicações públicas, acadêmicas e culturais, como o CulturaEduca.

Entre as limitações do estudo, destaca-se a abordagem baseada em modelos de linguagem, que apresentou maior custo computacional e desempenho inferior em relação aos demais métodos. Dificuldades iniciais no uso dos embeddings em memória impactaram o desempenho dessa abordagem, restringindo sua aplicação em cenários de maior volume de dados.

Como trabalhos futuros, propõe-se o aprimoramento da precisão da geocodificação, a automatização da integração com APIs de dados culturais e o desenvolvimento de dashboards geográficos integrados ao CulturaEduca, com foco na análise e visualização dos resultados.

Referências

[Bachmann 2024] Max Bachmann. *RapidFuzz Documentation — Version 3.14.3*. Acesso em: 09 jul. 2025. 2024. URL: https://rapidfuzz.github.io/RapidFuzz/ (acesso em 09/02/2025) (citado nas pgs. 6, 7).

[Brunner e Stockinger 2020] Ulf Brunner e Kurt Stockinger. "Entity matching with transformer architectures – a step forward in data integration". In: *Proceedings of the 23rd International Conference on Extending Database Technology (EDBT)*. Copenhagen, Denmark, 2020, pp. 463–473. URL: https://arxiv.org/abs/2004.00584 (citado na pg. 12).

[Gormley e Tong 2015] Clinton Gormley e Zachary Tong. *Elasticsearch: The Definitive Guide*. O'Reilly Media, 2015 (citado na pg. 9).

[Manning *et al.* 2008] Christopher D. Manning, Prabhakar Raghavan e Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. URL: https://nlp.stanford.edu/IR-book/ (citado na pg. 8).

[Navarro 2001] Gonzalo Navarro. "A guided tour to approximate string matching". *ACM Computing Surveys* 33.1 (2001), pp. 31–88. ISSN: 0360-0300. DOI: 10.1145/375360.375365 (citado nas pgs. 4–6, 9).

[Reimers e Gurevych 2019] Nils Reimers e Iryna Gurevych. "Sentence-BERT: sentence embeddings using siamese bert-networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2019. URL: https://arxiv.org/abs/1908.10084 (citado na pg. 11).

[Robertson e Zaragoza 2009] Stephen Robertson e Hugo Zaragoza. "The probabilistic relevance framework: bm25 and beyond". *Foundations and Trends in Information Retrieval* 3.4 (2009), pp. 333–389 (citado na pg. 8).

[Wang *et al.* 2021] Wenhui Wang *et al.* "MiniLMv2: multi-head self-attention relation distillation for compressing pretrained transformers". *arXiv preprint arXiv:2012.15828* (2021). URL: https://arxiv.org/abs/2012.15828 (citado na pg. 11).