Análise de Similaridade: Uma Solução Estratégica para o Mapeamento na Gestão Pública

Kauã Rodrigo de Lima Barbosa
Secretaria de Estado do Planejamento, Gestão e Patrimônio, Brasil
E-mail: kauanrodrigoo25@gmail.com
Augusto Braga Rodrigues
Secretaria de Estado do Planejamento, Gestão e Patrimônio, Brasil
E-mail: augusto.rodrigues@seplag.al.gov.br

Resumo

O aumento exponencial de dados no setor público vem trazendo novos desafios, que demandam a adoção de ações mais eficazes, voltadas à automatização dos processos de tratamento e análise de dados. Um dos principais obstáculos está na realização de relacionamentos entre tabelas por meio de dados categóricos, que frequentemente apresentam variações ortográficas, ausência de acentuação ou erros de digitação. Essas inconsistências comprometem a precisão na correspondência entre registros e dificultam a consolidação das informações. Este artigo tem como objetivo demonstrar como técnicas de análise de similaridade textual podem ser aplicadas como solução estratégica para esse tipo de problema na gestão pública. Nesse contexto, a aplicação da distância de Levenshtein — também conhecida como distância de edição ou casamento aproximado — apresenta-se como uma alternativa eficaz. Essa técnica mede o grau de semelhança entre duas cadeias de caracteres e é amplamente utilizada em áreas como correção automática de textos, detecção de plágio e bioinformática (para comparação de sequências de DNA). Ao aplicar essa abordagem para relacionar dados categóricos em um conjunto real, adotando um limiar de similaridade de 90%, observou-se um ganho expressivo em produtividade. A técnica reduziu significativamente a necessidade de intervenção manual e aumentou a precisão no mapeamento das informações — contribuindo para uma gestão pública mais eficiente, moderna e orientada por dados. Resultados significativos foram observados em projetos do Governo do Estado de Alagoas nos quais mapeamentos que, se realizados manualmente, demandariam meses de trabalho, foram concluídos em poucos dias. Essa redução no tempo de execução não apenas otimizou recursos humanos e operacionais, como também acelerou a disponibilidade de informações qualificadas para subsidiar a tomada de decisão. Dessa forma, a aplicação da distância de Levenshtein demonstrou-se uma solução viável e estratégica para enfrentar os desafios relacionados à qualidade dos dados, agregando valor aos processos de integração e análise no setor público.

Palavras-chave: distância de Levenshtein; dados categóricos; similaridade textual.

1. Introdução

O aumento exponencial na geração de dados no setor público tem imposto desafios cada vez mais complexos à administração pública, exigindo ações mais eficazes e inovadoras voltadas à automação dos processos de tratamento, análise e integração dessas informações. Atualmente, estima-se que cerca de 328,77 milhões de terabytes de dados são gerados globalmente por dia, dos quais aproximadamente 20% são estruturados, enquanto 80% são não estruturados. Os dados estruturados compreendem registros organizados em tabelas e bancos de dados, ao passo que os não estruturados incluem textos livres, imagens, vídeos, áudios e outras mídias que não seguem uma estrutura predefinida.

Esse crescimento na produção de dados, especialmente em contextos governamentais, resulta da ampla adoção de sistemas digitais, como plataformas de serviços públicos on-line, redes sociais institucionais, sensores urbanos inteligentes, câmeras de vigilância e demais tecnologias voltadas ao monitoramento e ao registro contínuo de informações. Tal cenário representa uma oportunidade valiosa para a administração pública aprimorar sua eficiência operacional, transparência institucional e qualidade no atendimento ao cidadão. No entanto, também impõe desafios significativos relacionados à gestão, à padronização e à integração dessas bases de dados.

Entre esses desafios, destaca-se a necessidade recorrente de integrar bases heterogêneas, sobretudo em projetos interinstitucionais que envolvem múltiplos órgãos e secretarias governamentais. Um dos principais entraves nesse processo consiste na realização de relacionamentos entre tabelas que utilizam dados categóricos com diferentes formas de preenchimento, como variações ortográficas, ausência de acentuação, erros de digitação e formatos inconsistentes. Essas divergências comprometem a correspondência precisa entre registros, dificultando a consolidação das informações e impactando negativamente a qualidade da tomada de decisão baseada em dados.

Tradicionalmente, a abordagem mais comum para mitigar essas inconsistências envolve a revisão manual e a padronização por equipes especializadas, um processo que, além de ser moroso e oneroso, está sujeito a falhas humanas causadas pela repetitividade e monotonia da tarefa. Nesse contexto, torna-se essencial a adoção de métodos automatizados que proporcionem maior precisão, agilidade e confiabilidade, especialmente em ambientes públicos orientados por dados.

Diante desse cenário, o presente artigo tem como objetivo geral demonstrar como técnicas de análise de similaridade textual podem ser aplicadas de forma estratégica para solucionar problemas de inconsistência em dados categóricos no âmbito da gestão pública. Especificamente, propõe-se o uso da distância de Levenshtein como ferramenta para identificar, de maneira automatizada, similaridades entre registros distintos, contribuindo para a melhoria da precisão e da eficiência nos processos de integração e consolidação de bases governamentais.

Além desta introdução, o artigo está estruturado em quatro seções. A Seção 2, dedicada à metodologia, apresenta todas as etapas do processo de ETL (Extract, Transform, Load), desde o tratamento inicial dos dados até a disponibilização final, incluindo também os critérios utilizados para a aplicação da métrica de similaridade textual. A Seção 3 traz os resultados e discussões, com destaque para os impactos observados na qualidade dos dados e nos ganhos de desempenho no processo de integração. Por fim, a Seção 4 apresenta as considerações finais, destacando as contribuições do estudo, suas limitações e sugestões para pesquisas futuras.

2. Metodologia

A metodologia adotada neste estudo caracteriza-se como uma pesquisa aplicada, de natureza quantitativa e com abordagem exploratória, voltada à solução prática de um problema recorrente na administração pública: as inconsistências ortográficas e divergências na formatação de dados categóricos utilizados na integração de bases administrativas. Tal configuração metodológica mostrou-se adequada à proposta de avaliar, de forma mensurável, a eficácia de técnicas automatizadas para tratamento e consolidação de dados em grande escala.

A abordagem empírico-quantitativa permitiu mensurar os impactos da técnica aplicada, especialmente no que se refere à precisão das correspondências entre registros e à eficiência computacional alcançada. O caráter exploratório da pesquisa justifica-se pela busca por métodos alternativos e inovadores de enfrentamento de um desafio técnico ainda pouco sistematizado no setor público: o tratamento automatizado de dados inconsistentes e heterogêneos em ambientes descentralizados.

O estudo foi conduzido com base em dados reais, disponibilizados pela Secretaria de Estado de Assistência e Desenvolvimento Social (Seades), referentes ao Cadastro Único (CadÚnico) e coletados em abril de 2024. Ao todo, foram analisados 59.075 registros, referentes a famílias em situação de vulnerabilidade social. Esses dados, essenciais para a formulação e monitoramento de políticas públicas, demandam elevado grau de precisão quando utilizados em processos de integração com outras bases governamentais.

Para atingir os objetivos propostos, foi adotada a técnica de análise de similaridade textual, por meio da distância de Levenshtein, uma métrica reconhecida por seu desempenho na detecção de correspondências entre cadeias de texto com pequenas variações. Essa técnica já se mostrou eficaz em diversos domínios, como correção automática de textos, verificação de plágio, bioinformática e integração de bases heterogêneas, o que reforça sua aplicabilidade ao contexto da gestão pública.

A metodologia foi organizada em etapas sequenciais e sistemáticas, conforme descrito a seguir:

1. Coleta e preparação dos dados: os dados foram inicialmente importados e submetidos a um processo de pré-processamento, que incluiu remoção de acentuação, conversão para letras minúsculas, eliminação de espaços duplicados e exclusão de caracteres especiais. Essas ações visaram padronizar os campos textuais e reduzir a interferência de ruídos na análise de similaridade.

2. Construção de chave composta: os campos de logradouro, bairro e município foram concatenados em uma coluna-chave padronizada, facilitando a comparação direta entre registros potencialmente semelhantes.

3. Aplicação da técnica de similaridade textual: a distância de Levenshtein foi aplicada utilizando um limiar de 90% de similaridade, valor definido com base em testes exploratórios iniciais. Esse parâmetro buscou equilibrar sensibilidade e especificidade na identificação de correspondências relevantes.

4. Otimização computacional: em função do grande volume de dados, foram empregadas estratégias de filtragem prévia por município e paralelismo computacional, visando otimizar o tempo de execução e viabilizar a escalabilidade do método.

5. Validação dos resultados: uma validação manual amostral foi realizada com o auxílio de ferramentas geográficas (como o Google Earth), a fim de verificar a precisão das correspondências geradas automaticamente e garantir a confiabilidade dos resultados.

6. Entrega da base consolidada: ao final do processo, os registros identificados como correspondentes foram integrados em uma base de dados unificada, apta para ser utilizada em sistemas de gestão, análise e visualização de dados.

Ao longo da execução, foram documentados os resultados intermediários, os desafios enfrentados e os ajustes metodológicos necessários, assegurando a transparência e a reprodutibilidade da pesquisa. A metodologia demonstrou que a aplicação da distância de Levenshtein constitui uma solução viável, eficaz e replicável para o tratamento de inconsistências em dados categóricos. Essa solução foi implementada por meio de um processo de ETL (Extract, Transform, Load) bem estruturado, que abrangeu desde a extração dos dados brutos, transformação criteriosa — com padronizações, limpezas, concatenações e aplicação da métrica de similaridade — até a carga final em uma base consolidada, apta para uso em sistemas analíticos e de apoio à decisão. A adoção desse fluxo organizado de tratamento de dados não apenas garantiu o desempenho técnico da solução proposta, como também reforçou seu potencial de reuso em outras iniciativas de integração e qualificação de dados no setor público, contribuindo de forma concreta para a modernização da gestão pública orientada por dados.

2.1 Dataset utilizado

O conjunto de dados utilizado neste estudo baseia-se em três fontes principais com diferentes origens e formatos, o que exigiu um processo rigoroso de padronização e integração. A primeira dessas fontes refere-se aos registros de endereços de famílias cadastradas no Cadastro Único (CadÚnico) em situação de vulnerabilidade social, disponibilizados pela Secretaria de Estado de Assistência e Desenvolvimento Social (Seades). O dataset original era composto por 59.075 registros brutos, cada um correspondente a uma pessoa cadastrada. Os dados foram coletados no mês de abril de 2024 e abrangem os 102 municípios do Estado de Alagoas.

As demais bases utilizadas foram: (i) uma base georreferenciada construída pela Superintendência de Informações e Cenários (SINC), previamente estruturada com coordenadas geográficas vinculadas a localidades de referência em todo o estado; e (ii) uma base complementar fornecida pela empresa Equatorial, contendo endereços e coordenadas extraídas dos sistemas da concessionária de energia. Enquanto a base do CadÚnico apresentava dados preenchidos de forma livre — sujeitos a erros de grafia, abreviações e inconsistências —, as bases da SINC e da Equatorial possuíam estruturação padronizada e confiável, o que as qualificou como fontes de validação.

As três bases envolvidas no estudo estavam em formatos distintos: duas em .xlsx (Excel) e uma em .csv. Para garantir eficiência computacional e compatibilidade com os scripts desenvolvidos em Python, todas foram convertidas para o formato .csv, considerado mais leve, robusto e adequado para processamento em larga escala.

Durante a análise inicial, foram observadas diversas variações ortográficas, duplicidades e inconsistências de grafia em nomes de logradouros, bairros e municípios — fenômeno comum em cadastros manuais e descentralizados. Essas inconsistências dificultavam significativamente a realização de junções entre as bases, impactando negativamente a acurácia na correspondência entre registros e, por consequência, comprometendo a qualidade

das análises que dependem da georreferenciação precisa dos dados.

Os maiores desafios de padronização ocorreram nas zonas rurais, onde a informalidade na designação dos endereços era mais evidente. Nessas regiões, observou-se uma ampla variedade de formas de escrita para um mesmo local, além do uso frequente de apelidos ou nomes populares para comunidades, sítios e povoados, muitas vezes diferentes da nomenclatura oficial. Essa multiplicidade de grafias e denominações aumentou substancialmente a complexidade da etapa de correspondência textual, exigindo maior sensibilidade dos algoritmos para detecção de similaridade.

O desafio de integração envolveu três fontes principais:

1. Base CadÚnico/Seades: contendo endereços declarados no momento do cadastramento, muitas vezes preenchidos de forma livre, o que potencializa erros de grafia, ausência de acentuação e abreviações inconsistentes.

2. Base georreferenciada construída pela Superintendência de Informações e Cenários (SINC): desenvolvida previamente com dados limpos e estruturados, essa base continha coordenadas geográficas (latitude e longitude) vinculadas a localidades de referência em todo o estado.

3. Base complementar da empresa Equatorial: fornecida em parceria com a SINC, essa base também continha o endereço e coordenadas geográficas extraídas de sistemas próprios da concessionária de energia, cobrindo áreas urbanas e rurais.

A base da SINC foi primeiramente limpa e reestruturada, e posteriormente concatenada com a base da Equatorial. Esse processo resultou em uma planilha-chave contendo 35.086 registros únicos com endereços georreferenciados padronizados, a qual foi utilizada como base de referência para o processo de comparação e correspondência com os registros do CadÚnico. Essa planilha-chave foi essencial para a validação cruzada dos dados e possibilitou identificar, por meio de técnicas de similaridade textual, quais endereços do CadÚnico possuíam correspondência plausível com registros georreferenciados confiáveis.

Esse processo de construção e padronização das bases de dados — desde a extração, passando pela transformação (limpeza, concatenação e padronização), até a carga final da planilha-chave — representa a fase operacional do ETL (Extract, Transform, Load) mencionado na seção de metodologia. A etapa foi determinante para garantir a confiabilidade do algoritmo de correspondência e evidencia a importância da organização de dados como etapa prévia e imprescindível à aplicação de técnicas avançadas de análise e integração de informações públicas.

2.2 Ferramentas e técnicas aplicadas

Para o tratamento e análise dos dados, foi utilizada a linguagem de programação Python, amplamente adotada em projetos de ciência de dados e análise automatizada de informações, devido à sua versatilidade, sintaxe acessível e vasto ecossistema de bibliotecas especializadas. Dentre essas, destacam-se as bibliotecas Pandas e RapidFuzz, empregadas em diferentes fases do processamento.

A biblioteca Pandas foi responsável pela manipulação estruturada dos dados, oferecendo recursos poderosos para leitura, filtragem, transformação e análise de grandes volumes de informação. Ao contrário de ferramentas como o Microsoft Excel, que possuem limites rígidos (1.048.576 linhas por 16.384 colunas, segundo a documentação oficial da Microsoft), o Pandas permite trabalhar com datasets significativamente maiores, com mais

flexibilidade e controle programático.

Já a biblioteca RapidFuzz foi utilizada para aplicação da distância de Levenshtein, sendo escolhida por sua eficiência de desempenho. Essa biblioteca é implementada em C++ com bindings em Python, o que resulta em maior velocidade de execução em comparação a bibliotecas puramente escritas em Python, como o FuzzyWuzzy. Segundo sua documentação oficial, a RapidFuzz é otimizada para grandes volumes de dados e oferece operações vetorizadas, o que reduz drasticamente o tempo de comparação entre strings em cenários com milhares de registros.

O processo metodológico envolveu inicialmente uma etapa de pré-processamento dos dados textuais, com as seguintes operações:

- Remoção de acentuação com o uso da biblioteca unicodedata, que converte caracteres acentuados para sua forma base;
- Conversão para caixa baixa, garantindo padronização independente da capitalização original;
- Eliminação de espaços extras e outros caracteres inconsistentes.

Em seguida, foi criada uma coluna-chave padronizada por meio da concatenação dos campos logradouro, bairro e município, originando a variável “Concatenado com Bairro – Formato Google”. Essa padronização foi essencial para viabilizar a comparação direta entre registros aparentemente distintos, mas semanticamente similares.

Após o preparo dos dados, foi aplicada a distância de Levenshtein com um limiar de similaridade de 90%, valor definido com base em testes exploratórios preliminares e validação manual por amostragem. Esse parâmetro buscou equilibrar a detecção de correspondências relevantes sem gerar falsos positivos em excesso.

A distância de Levenshtein é uma métrica clássica de similaridade textual que mensura o número mínimo de operações necessárias para transformar uma cadeia de caracteres em outra. Essas operações incluem inserção, deleção ou substituição de caracteres. A fórmula clássica pode ser visualizada na Figura 1, sendo fundamental para o entendimento do funcionamento interno do algoritmo.

Figura 1 - Fórmula da distância de Levenshtein

$$
\text{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{se } \min(i,j) = 0 \\ \min \begin{cases} \text{lev}_{a,b}(i-1,j) + 1 \\ \text{lev}_{a,b}(i,j-1) + 1 \\ \text{lev}_{a,b}(i-1,j-1) + C \end{cases} & \text{nos outros casos} \end{cases}
$$

em que $C_i = 0$, se $a_i = b_i$, ou $C_i = 1$ se $a_i \mathrel{!}= b_i$

Fonte: Elaboração própria com base em Levenshtein (1966)

A implementação da métrica baseia-se em programação dinâmica, uma técnica de construção de algoritmos que resolve problemas complexos ao dividi-los em subproblemas menores e interdependentes. O principal benefício da programação dinâmica reside na memorização de resultados intermediários, evitando cálculos redundantes e otimizando o desempenho computacional. Essa abordagem é especialmente útil em algoritmos de comparação de sequências, como é o caso da distância de Levenshtein, onde o número de possibilidades de transformação cresce exponencialmente conforme o tamanho das strings.
Para o desenvolvimento da solução computacional, foi utilizada como ambiente de desenvolvimento integrado (IDE) o **PyCharm**, da **JetBrains**, ferramenta amplamente reconhecida por seu suporte dedicado à linguagem Python. O PyCharm oferece recursos como **destaque de sintaxe, auto-complete inteligente, integração com sistemas de controle de versão (Git), ambientes virtuais e execução interativa de scripts**, o que facilita significativamente o ciclo de desenvolvimento, teste e depuração de projetos em ciência de dados.

A versão utilizada do Python foi a **3.12.4**, a mais recente estável disponibilizada até o momento do desenvolvimento, o que garantiu compatibilidade com as bibliotecas modernas e melhor performance nas operações de texto e manipulação de arquivos.

Além das bibliotecas principais já mencionadas, outras ferramentas e módulos utilizados no projeto incluem:

- os e glob: para manipulação de arquivos e automação de leitura de múltiplos datasets;
- numpy: para operações vetoriais e manipulação eficiente de arrays em etapas intermediárias do processo;
- re (expressões regulares): para limpeza textual avançada;
- openpyxl: para leitura e escrita de arquivos nos formatos .xlsx;
- logging (Documentação): para controle e registro da execução do script, permitindo identificar pontos críticos do processamento, erros silenciosos e o andamento das rotinas automatizadas, conforme documentação oficial da linguagem Python (PYTHON SOFTWARE FOUNDATION, 2025);
- concurrent.futures.ThreadPoolExecutor: empregado para paralelização de tarefas críticas, como a aplicação da métrica de similaridade sobre grandes volumes de dados, permitindo aproveitar múltiplos núcleos de CPU e reduzir significativamente o tempo de execução em comparação ao processamento sequencial. Essa abordagem é particularmente eficaz quando as comparações entre strings podem ser feitas de forma independente.

Todas as bibliotecas mencionadas foram compatíveis com a versão Python 3.12.4 e instaladas por meio do gerenciador de pacotes pip. As versões utilizadas no momento do desenvolvimento foram:

- **pandas 2.2.2**
- **rapidfuzz 3.6.1**

- **numpy 1.26.4**
- **openpyxl 3.1.2**
- **logging** (módulo nativo da biblioteca padrão Python)
- **re** e **unicodedata** (também da biblioteca padrão)
- **concurrent.futures** (ThreadPoolExecutor, da lib padrão futures)

Todas foram testadas em ambiente local com sistema operacional Windows 11, arquitetura x64 e 16 GB de RAM. O código foi modularizado para facilitar eventuais reusos e possíveis atualizações futuras nas bibliotecas.

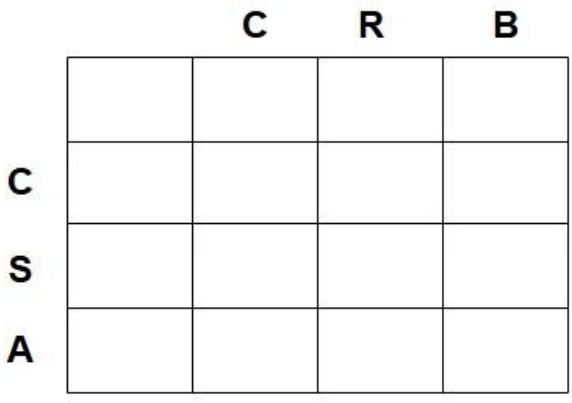
A construção do algoritmo foi dividida em **quatro etapas principais**, que serão descritas nas subseções seguintes (2.2.1 a 2.2.4), utilizando como base exemplos didáticos com **nomes de clubes de futebol tradicionais do Estado de Alagoas**, permitindo ilustrar desde a concepção lógica até a implementação prática em código Python.

2.2.1 Primeiro passo: construção da matriz

O primeiro passo para calcular a distância de Levenshtein consiste na criação de uma matriz bidimensional de tamanho (n+1)×(m+1), onde n e m representam, respectivamente, o número de caracteres das duas strings a serem comparadas. Essa matriz representa uma tabela de programação dinâmica, na qual cada célula armazena o custo mínimo de edição necessário para transformar um prefixo da primeira string em um prefixo da segunda string.

A adição de uma linha e uma coluna extras é necessária para incluir as comparações com a string vazia (caso base). Por exemplo, ao comparar as strings "CRB" (3 caracteres) e "CSA" (3 caracteres), a matriz terá dimensão 4×4, conforme ilustrado na Figura 2.

Figura 2 - Criação da matriz 4x4



Fonte: (Os autores, 2025)

Essa matriz é a espinha dorsal do algoritmo, e seu preenchimento obedece a regras matemáticas específicas baseadas em operações elementares: inserção, deleção e substituição.

2.2.2 Segundo passo: construção da matriz

Após a construção da matriz, realiza-se a inicialização da primeira linha e da primeira coluna com os valores de seus respectivos índices. Isso representa o custo de transformar qualquer string em uma string vazia apenas por meio de operações de deleção (linha) ou inserção (coluna).

Matematicamente, isso significa:

$$distância(i, 0) = i \quad e \; distância(0, j) = j$$

Essa etapa estabelece as condições iniciais para a aplicação da programação dinâmica, e permite que os demais valores da matriz sejam preenchidos recursivamente. A Figura 3 ilustra essa fase, com os valores representando o custo cumulativo de transformar uma string de tamanho i em outra de tamanho zero (e vice-versa).

Figura 3 - Inicialização da matriz 4x4

		C	R	B
	0	1	2	3
C	1			
S	2			
A	3			

Fonte: (Os autores, 2025)

2.2.3 Primeiro passo: construção da matriz

A partir da célula [1][1], inicia-se o preenchimento da matriz aplicando-se a recorrência da distância de Levenshtein, conforme mostrada na Figura 1 na secção 2.2.

Onde:

- distância(i − 1, j) : custo para deletar um caractere;
- distancia(i , j − 1): custo para inserir um caractere;
- distância(i − 1 , j − 1): custo para substituir um caractere.

Caso os caracteres si e tj sejam iguais, o custo é zero (sem operação). Caso contrário, adiciona-se uma unidade ao custo mínimo entre as três operações possíveis.

O valor final, presente na célula [n][m], representa a distância total de edição entre as

duas strings. Por exemplo, transformar "CRB" em "CSA" requer duas operações (substituir 'R' por 'S' e 'B' por 'A'), como demonstrado na Figura 4.

Figura 4 - Aplicação da fórmula de Levenshtein entre “CRB” e “CSA”

		C	R	B
	0	1	2	3
C	1	0	1	2
S	2	1	1	2
A	3	2	2	2

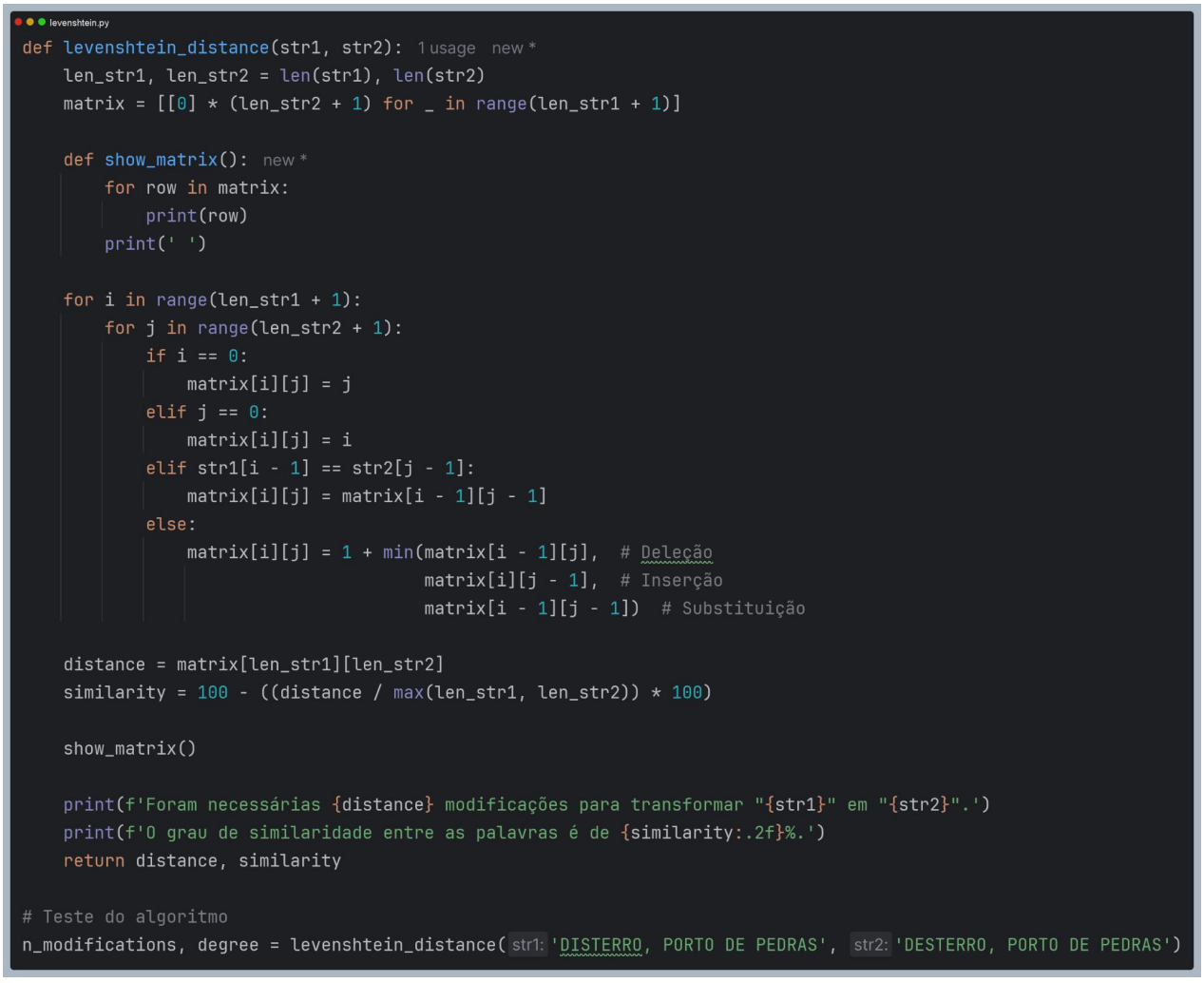
Fonte: (Os autores, 2025)

2.2.4 Implementação dos passos em python

Conforme descrito nas subseções anteriores (2.2.1 a 2.2.3), o cálculo da distância de Levenshtein baseia-se em programação dinâmica e envolve quatro etapas principais: construção da matriz, definição das condições iniciais, preenchimento da tabela com base na função de recorrência e obtenção do valor final de distância. A presente subseção apresenta a **implementação prática dessas etapas em Python**, utilizando um exemplo real extraído do contexto analisado.

O código da Figura 5 realiza o cálculo da distância de Levenshtein entre dois endereços que diferem por um pequeno erro ortográfico — uma ocorrência comum nas bases manuais analisadas neste estudo:

Figura 5 - Implementação da distância de Levenshtein em Python



```python
levenshtein.py
def levenshtein_distance(str1, str2):  1 usage  new *
    len_str1, len_str2 = len(str1), len(str2)
    matrix = [[0] * (len_str2 + 1) for _ in range(len_str1 + 1)]

    def show_matrix():  new *
        for row in matrix:
            print(row)
        print(' ')

    for i in range(len_str1 + 1):
        for j in range(len_str2 + 1):
            if i == 0:
                matrix[i][j] = j
            elif j == 0:
                matrix[i][j] = i
            elif str1[i - 1] == str2[j - 1]:
                matrix[i][j] = matrix[i - 1][j - 1]
            else:
                matrix[i][j] = 1 + min(matrix[i - 1][j],  # Deleção
                                       matrix[i][j - 1],  # Inserção
                                       matrix[i - 1][j - 1])  # Substituição

    distance = matrix[len_str1][len_str2]
    similarity = 100 - ((distance / max(len_str1, len_str2)) * 100)

    show_matrix()

    print(f'Foram necessárias {distance} modificações para transformar "{str1}" em "{str2}".')
    print(f'O grau de similaridade entre as palavras é de {similarity:.2f}%.')
    return distance, similarity

# Teste do algoritmo
n_modifications, degree = levenshtein_distance( str1: 'DISTERRO, PORTO DE PEDRAS',  str2: 'DESTERRO, PORTO DE PEDRAS')
```

Fonte: (Os autores, 2025)

Esse código traduz, de forma estruturada, os conceitos teóricos previamente abordados:

- A **criação da matriz** e sua **inicialização com índices** seguem exatamente o descrito nas seções 2.2.1 e 2.2.2;
- O **preenchimento da matriz com base na função de recorrência** e a obtenção da **distância final** correspondem ao apresentado na seção 2.2.3;
- O cálculo adicional da **similaridade percentual** permite interpretar os resultados com base em um limiar pré-definido (90%), como discutido anteriormente na seção 2.2.

Além de calcular o número mínimo de operações para conversão entre duas strings, a

função imprime a matriz de programação dinâmica utilizada no processo, o que facilita a visualização da lógica do algoritmo. Essa visualização é especialmente útil em contextos educacionais e de validação manual, como os realizados durante a fase de testes exploratórios deste estudo.

No exemplo apresentado, a distância calculada entre "DISTERRO, PORTO DE PEDRAS" e "DESTERRO, PORTO DE PEDRAS" foi de apenas **duas modificações**, o que resultou em um grau de similaridade de **96%** — valor acima do limiar mínimo adotado, indicando correspondência válida.

2.3 Validação dos resultados

Para realizar a análise de similaridade entre os registros, foi estruturado um algoritmo baseado em **dois laços de repetição aninhados**: o primeiro percorreu os **59.075 registros da base do Cadastro Único (CadÚnico)**, enquanto o segundo iterava sobre os **35.086 registros da base de referência georreferenciada**. Essa estrutura resultou em um total de **aproximadamente 2.072.740.536 comparações**, caracterizando uma **complexidade computacional de ordem quadrática** — O(n^2), segundo a notação Big O. Embora funcional, essa abordagem inicial revelou-se altamente custosa em termos de tempo de execução: a primeira execução integral do algoritmo levou **cerca de dois dias** para ser concluída, mesmo em ambiente de processamento local com recursos intermediários.

Diante desse gargalo de desempenho, o código foi **refatorado** com foco na eficiência. A estratégia reformulada passou a **restringir as comparações ao nível municipal**, utilizando filtros para agrupar os registros por município. Essa segmentação reduziu drasticamente o volume de comparações em cada iteração e, ao mesmo tempo, preservou a precisão da análise, uma vez que a grande maioria dos registros compartilha o mesmo contexto geográfico.

Além disso, foi implementado **paralelismo computacional** com o uso da biblioteca multiprocessing, permitindo a **execução simultânea de diferentes segmentos da tarefa** em múltiplos núcleos de CPU. O paralelismo divide o processamento em subtarefas independentes e distribuídas, explorando os recursos de hardware de forma mais eficiente e acelerando significativamente o tempo total de execução.

Para ilustrar a eficiência da nova abordagem, no caso do município de **Maceió**, que possuía **8.908 registros no CadÚnico** e **7.047 na base georreferenciada**, o número total de comparações caiu para aproximadamente **62.774.676** — ainda alto, mas **reduzido em mais de 30 vezes** em relação à abordagem completa. Com a nova lógica e o uso da biblioteca **RapidFuzz**, o tempo de execução global da tarefa foi reduzido para **cerca de cinco minutos**, um ganho expressivo em comparação com os dois dias anteriores.

A **validação dos dados correspondentes** foi conduzida manualmente por **três técnicos especializados**, com o suporte da ferramenta **Google Earth**, que possibilitou verificar a **coerência espacial das coordenadas atribuídas automaticamente**. A análise considerou tanto a correspondência textual dos registros quanto a **verossimilhança geográfica**, assegurando que os dados integrados fossem compatíveis com a localização real dos endereços.

Como resultado da validação, obteve-se um índice de **acerto de 97,35%**, o que equivale a **57.508 registros corretamente pareados** entre os 59.075 analisados. Esse desempenho evidencia a **robustez e a confiabilidade da técnica de similaridade textual** aplicada, demonstrando sua viabilidade como solução automatizada para tarefas que, se realizadas manualmente, exigiriam **meses de trabalho de uma equipe especializada**.

Esse resultado não apenas valida a técnica empregada, como também **reforça a importância da otimização computacional em projetos de análise de dados em larga escala**, especialmente no setor público, onde o volume de informações e a necessidade de

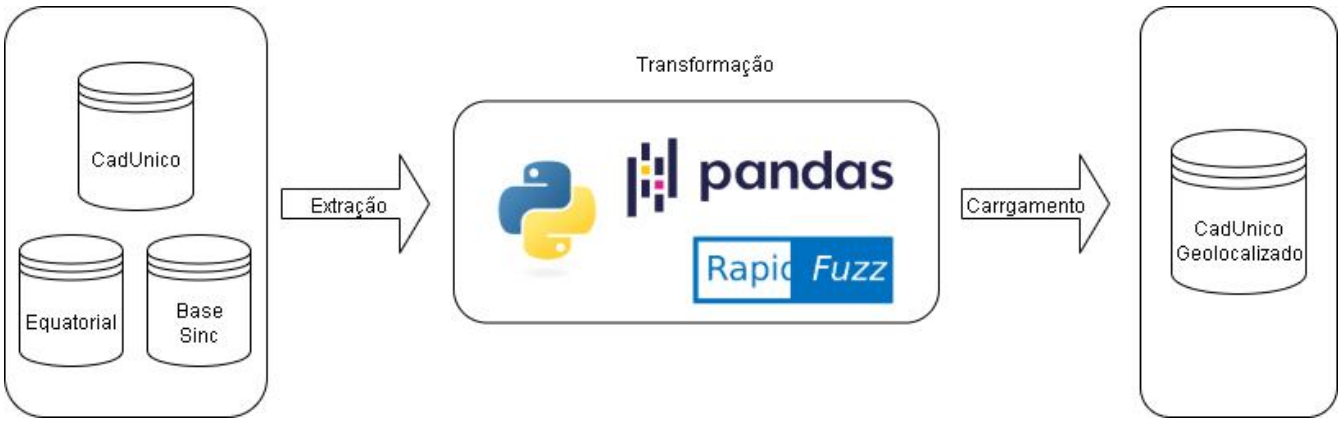
integração entre bases são crescentes.

3. Resultados e Discussão

A aplicação da distância de Levenshtein sobre o conjunto de dados categóricos demonstrou resultados expressivos em termos de eficiência e precisão na correspondência entre registros. Com a definição de um limiar de similaridade de 90%, foi possível identificar automaticamente correspondências entre diferentes grafias de um mesmo endereço, reduzindo significativamente a necessidade de intervenção manual.

Do total de 59.075 registros analisados, 57.508 apresentavam variações ortográficas ou inconsistências textuais passíveis de tratamento por meio da técnica de similaridade textual. Para avaliar a efetividade do algoritmo, foi selecionada uma amostra de 2.000 registros, que passou por validação manual. A aplicação da distância de Levenshtein permitiu consolidar essas variações em grupos únicos de correspondência, com uma taxa de acerto de 96% na amostra validada, conforme o fluxo demonstrado na Figura 6.

Figura 6 - Etapas de aplicação e validação da distância de Levenshtein sobre os dados



Fonte: (Os autores, 2025)

A comparação entre o tempo estimado para padronização manual e o tempo de execução automatizada evidencia um ganho operacional substancial. Enquanto a revisão manual completa demandaria cerca de três meses de trabalho contínuo por uma equipe dedicada, o processo automatizado foi concluído em menos de 48 horas — considerando as etapas de pré-processamento, execução dos algoritmos e validação amostral.

Esses resultados reforçam a eficácia da abordagem baseada em similaridade textual como alternativa estratégica para a gestão pública orientada por dados. Além da expressiva redução de tempo, a técnica contribuiu para o aumento da qualidade da base consolidada, impactando diretamente na confiabilidade das análises gerenciais subsequentes.

Contudo, algumas limitações foram observadas. Em casos com grafias extremamente distintas ou abreviações não convencionais, o algoritmo apresentou dificuldades na identificação de similaridade, resultando em falsos negativos. Um exemplo disso foi o caso do endereço “CON RES ERNESTO MARANHÃO, BARRA DE SANTO ANTÔNIO”, que retornou como correspondência “HERNESTO MARANHÃO, BARRA DE SANTO ANTÔNIO”, com um grau de similaridade de 89,88% — valor abaixo do limiar de 90% adotado. Esse tipo de variação, ainda que foneticamente próximo, não foi reconhecida como correspondência válida.

Adicionalmente, o uso exclusivo da distância de Levenshtein pode levar à formação de agrupamentos incorretos quando nomes semanticamente distintos apresentam alta similaridade textual. Para mitigar essas limitações, recomenda-se a combinação da métrica com outras abordagens complementares, como algoritmos fonéticos (por exemplo, Soundex ou Metaphone) ou a incorporação de regras semânticas auxiliares.
Além disso, o pipeline foi concebido de forma modular, o que permite sua adaptação e execução em ambientes de computação em nuvem, ampliando sua escalabilidade e viabilidade em contextos interinstitucionais.

4. Considerações Finais

Este estudo demonstrou que a aplicação da distância de Levenshtein constitui uma solução viável, eficiente e estratégica para o enfrentamento de inconsistências em dados categóricos, especialmente no contexto da gestão pública orientada por dados. Com base em um caso real envolvendo a integração de bases administrativas heterogêneas, foi possível evidenciar que a técnica adotada reduz significativamente o esforço manual necessário para a consolidação de registros, além de aumentar a precisão nas correspondências entre informações textuais com variações ortográficas.

A adoção de um limiar de similaridade de 90% permitiu a automatização robusta do processo de mapeamento entre endereços, alcançando uma taxa de acerto de 97,35% na amostragem validada manualmente. Tais resultados indicam um avanço expressivo na modernização dos processos de tratamento de dados no setor público, reforçando o potencial de reaplicação da metodologia em outros projetos com grandes volumes de dados e baixa padronização textual.

Contudo, a abordagem baseada unicamente na distância de Levenshtein apresenta limitações em cenários com grafias muito divergentes, registros curtos ou nomes ambíguos, podendo gerar falsos negativos ou associações imprecisas. Para mitigar esses efeitos, recomenda-se a combinação dessa métrica com técnicas complementares, tais como algoritmos fonéticos (Soundex, Metaphone), regras contextuais específicas e modelos de aprendizado supervisionado treinados com dados locais.

Além disso, vislumbra-se como caminho promissor a incorporação de métricas alternativas de similaridade, como o índice de Jaccard, que avalia a interseção de subconjuntos de caracteres ou palavras, podendo ser mais eficaz em determinadas estruturas de texto. Métodos baseados em n-gramas, vetores semânticos (como Word2Vec ou BERT) e abordagens híbridas combinando múltiplas heurísticas podem ampliar significativamente a capacidade discriminativa e contextual das análises, especialmente em ambientes multissistêmicos e com bases oriundas de diferentes fontes.

Com o avanço da inteligência artificial, surgem novas possibilidades para o tratamento de inconsistências em dados textuais, por meio de modelos treinados em grandes corpora linguísticos, capazes de capturar nuances semânticas e contextuais além da mera comparação de caracteres. Tecnologias como **spaCy**, **Hugging Face Transformers** (incluindo modelos como **BERT**, **RoBERTa** e **DistilBERT**) e **OpenAI Embeddings** oferecem elevado desempenho na detecção de similaridades complexas, mesmo em expressões textuais que compartilham pouca ou nenhuma semelhança lexical. Esses modelos operam por meio de **vetores semânticos** — representações matemáticas capazes de codificar o significado de palavras e frases — o que permite inferir relações entre termos mesmo em estruturas sintáticas distintas.

Além de seu uso isolado, tais modelos vêm sendo incorporados a pipelines de deduplicação, entity resolution e enriquecimento semântico de dados, com aplicações consolidadas em setores como saúde, educação, segurança e auditoria governamental. Em

contextos públicos, sua adoção pode representar um diferencial estratégico, sobretudo em projetos de interoperabilidade entre órgãos, onde a padronização textual não é garantida.

A integração dessas técnicas avançadas em soluções programadas em linguagens como **Python** favorece o desenvolvimento de rotinas de automação inteligente, que combinam eficiência computacional com capacidade analítica. Ao mesmo tempo em que aceleram tarefas operacionais, essas soluções liberam equipes técnicas para se dedicarem a atividades de maior valor agregado, como modelagem de políticas públicas, avaliação de impactos e produção de evidências.

Em síntese, a aplicação da análise de similaridade textual, especialmente quando aliada a um pipeline de ETL bem estruturado e a abordagens computacionalmente eficientes, representa um salto qualitativo na forma como o setor público pode organizar, integrar e utilizar seus dados. A consolidação de informações com maior acurácia e menor custo operacional contribui diretamente para a construção de políticas públicas mais eficazes, oportunas e ancoradas em evidências confiáveis. O aprofundamento e a diversificação dessas técnicas, portanto, constituem um caminho promissor para a inovação e a inteligência na administração pública.

O reuso dessa abordagem não se restringe a projetos de integração de endereços. Bases relacionadas a nomes de escolas, unidades de saúde, denominações de programas sociais ou estabelecimentos comerciais podem ser beneficiadas com o mesmo princípio técnico. Em um contexto de transformação digital e aumento das iniciativas de governança de dados, soluções como essa favorecem não apenas a eficiência, mas também a padronização e interoperabilidade entre sistemas, ampliando a capacidade analítica das instituições públicas.

Nesse contexto, torna-se essencial que os gestores públicos incorporem, em suas estratégias de transformação digital, diretrizes claras de governança de dados, interoperabilidade e uso ético da inteligência artificial. A adoção de tecnologias de NLP (Natural Language Processing) e *machine learning* deve ser acompanhada por políticas institucionais que assegurem a rastreabilidade dos algoritmos, a qualidade dos dados de entrada e a transparência nos critérios de correspondência textual utilizados em bases administrativas. Além disso, é fundamental que haja capacitação contínua das equipes técnicas envolvidas, promovendo o domínio dos recursos computacionais modernos e estimulando a cultura de dados nos órgãos públicos.

A institucionalização de práticas de qualidade de dados e de automação inteligente, aliada ao uso de pipelines auditáveis e reprodutíveis, permitirá que o setor público avance para um novo patamar de maturidade digital. Com isso, além de resolver problemas operacionais relacionados à inconsistência de dados, será possível produzir conhecimento estratégico, avaliar políticas com maior robustez e orientar decisões baseadas em evidências concretas. O uso inteligente da similaridade textual, portanto, deixa de ser apenas uma solução técnica para se tornar um pilar de apoio à inovação na gestão pública contemporânea.

Referências

BAEZA-YATES, Ricardo; RIBEIRO-NETO, Berthier. **Modern information retrieval:** the concepts and technology behind search. 2. ed. New York: Addison Wesley, 2011.

CUNHA, Maria Alexandra; MEIRELLES, Fernando. Governança digital: um modelo para governos locais. **Revista de Administração Pública**, Rio de Janeiro, v. 45, n. 1, p. 1–26, jan./fev. 2011.

GOOGLE. **Google Earth**. Disponível em: https://www.google.com/earth/. Acesso em: 30 jun. 2024.

HUGGING FACE. **Transformers**: State-of-the-art Machine Learning for PyTorch, TensorFlow and JAX. Disponível em: https://huggingface.co/transformers/. Acesso em: 1 jul. 2024.

LEVENSHTEIN, Vladimir I. Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady, v. 10, n. 8, p. 707–710, 1966.

MANNING, Christopher D.; RAGHAVAN, Prabhakar; SCHÜTZE, Hinrich. **Introduction to information retrieval.** Cambridge: Cambridge University Press, 2008.

MARTINS, Hélio. **Tratamento e qualidade de dados**: princípios e técnicas para melhoria da informação. São Paulo: Saraiva, 2017.

NAVARRO, Gonzalo. A guided tour to approximate string matching. ACM Computing Surveys, v. 33, n. 1, p. 31–88, 2001.

OLIVEIRA, Kelly; CRUZ, Sérgio. Qualidade de dados e interoperabilidade no setor público: um estudo sobre iniciativas brasileiras. **Revista Eletrônica de Administração** (REAd), Porto Alegre, v. 24, n. 2, p. 166–192, 2018.

PANDAS DEVELOPMENT TEAM. pandas: powerful Python data analysis toolkit. Disponível em: https://pandas.pydata.org/. Acesso em: 30 jun. 2024.

PINHEIRO, Fabrício. **Dados abertos e transparência pública**: caminhos para a inovação no setor público brasileiro. Brasília: Enap, 2020.

PYTHON SOFTWARE FOUNDATION. logging — Logging facility for Python. Disponível em: https://docs.python.org/3/library/logging.html. Acesso em: 30 jun. 2024.

RAPIDFUZZ. **Rapid fuzzy string matching.** Disponível em: https://github.com/maxbachmann/RapidFuzz. Acesso em: 30 jun. 2024.

SILVA, Luiz Carlos da; MACHADO, Roberto. Aplicação de técnicas de mineração de dados no serviço público: estudo de caso na análise de benefícios sociais. **Revista do Serviço Público**, Brasília, v. 69, n. 4, p. 625–644, out./dez. 2018.

SPACY. spaCy: Industrial-strength Natural Language Processing in Python. Disponível em: https://spacy.io/. Acesso em: 1 jul. 2024.

SUPERINTENDÊNCIA DE INFORMAÇÕES E CENÁRIOS – SINC. **Dados georreferenciados do Estado de Alagoas.** Maceió, 2024. Documento interno.