



UNIVERSIDADE FEDERAL DE SANTA CATARINA  
CENTRO TECNOLÓGICO  
DEPARTAMENTO DE ENGENHARIA DE AUTOMAÇÃO E SISTEMAS  
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

João Pedro DellOsbel da Silva

**Método Computacional para Validação e Preenchimento do Código GTIN em  
Notas Fiscais Eletrônicas**

Florianópolis  
2026

João Pedro DellOsbel da Silva

**Método Computacional para Validação e Preenchimento do Código GTIN em  
Notas Fiscais Eletrônicas**

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.

Orientador: Jomi Fred Hubner

Supervisora: Simone Silmara Werner

Florianópolis

2026

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.  
Dados inseridos pelo próprio autor.

Silva, João Pedro DelloSbel da  
Método computacional para validação e preenchimento do  
código GTIN em notas fiscais eletrônicas / João Pedro  
DelloSbel da Silva ; orientador, Jomi Fred Hubner, 2026.  
52 p.

Trabalho de Conclusão de Curso (graduação) -  
Universidade Federal de Santa Catarina, Centro  
Tecnológico, Graduação em Engenharia de Controle e  
Automação, Florianópolis, 2026.

Inclui referências.

1. Engenharia de Controle e Automação. 2.  
Correspondência de Produtos. 3. Processamento de Linguagem  
Natural. 4. Apache Solr. 5. Locality Sensitive Hashing. I.  
Hubner, Jomi Fred. II. Universidade Federal de Santa  
Catarina. Graduação em Engenharia de Controle e Automação.  
III. Título.

João Pedro DellOsbel da Silva

**Método Computacional para Validação e Preenchimento do Código GTIN em  
Notas Fiscais Eletrônicas**

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis, 06 de julho de 2026.

Prof. Marcelo De Lellis Costa de Oliveira, Dr.  
Coordenador do Curso

**Banca Examinadora:**



Documento assinado digitalmente

**Jomi Fred Hubner**

Data: 10/07/2026 17:26:25-0300

CPF: \*\*\*.334.769-\*\*

Verifique as assinaturas em <https://v.ufsc.br>

Prof. Jomi Fred Hubner, Dr.

Orientador

UFSC/CTC/EAS



Documento assinado digitalmente

**SIMONE SILMARA WERNER**

Data: 10/07/2026 17:16:58-0300

CPF: \*\*\*.925.219-\*\*

Verifique as assinaturas em <https://v.ufsc.br>

Prof. Simone Silmara Werner, Dr.

Supervisora

UFSC/INE/CEOS

Rodrigo Lange

Avaliador

IFRS

Prof. Hector Bessa Silveira, Dr.

Presidente da Banca

UFSC/CTC/EAS

Este trabalho é dedicado a minha família e amigos.

## **AGRADECIMENTOS**

Primeiramente, gostaria de agradecer aos meus pais, que sempre me incentivaram a estudar e me deram todo o apoio e suporte para seguir esse caminho. Agradeço à minha mãe, Mariza Odete DellOsbel da Silva, e ao meu pai, Edilson Gomes da Silva. Sem vocês eu não teria chegado até aqui. Também sou grato à minha irmã gêmea, Ana Laura, pela parceria e apoio.

Expresso meus agradecimentos ao time do projeto Céos, em especial à minha supervisora Simone Werner e ao professor André Zibetti, com os quais trabalhei em proximidade no meu tempo como bolsista e durante o PFC.

Sou grato ao meu orientador acadêmico, Jomi Fred Hübner, pelo apoio e ajuda no desenvolvimento deste trabalho. Por fim, expresso minha gratidão à Universidade Federal de Santa Catarina por todas as oportunidades que tive durante o curso.

## DECLARAÇÃO DE PUBLICIDADE

Florianópolis, 10 de 07 de 2026.

Na condição de representante do Projeto Céos no qual o presente trabalho foi realizado, declaro não haver ressalvas quanto ao aspecto de sigilo ou propriedade intelectual sobre as informações contidas neste documento, que impeçam a sua publicação por parte da Universidade Federal de Santa Catarina (UFSC) para acesso pelo público em geral, incluindo a sua disponibilização *online* no Repositório Institucional da Biblioteca Universitária da UFSC. Além disso, declaro ciência de que o autor, na condição de estudante da UFSC, é obrigado a depositar este documento, por se tratar de um Trabalho de Conclusão de Curso, no referido Repositório Institucional, em atendimento à Resolução Normativa n° 126/2019/CUn.

Por estar de acordo com esses termos, subscrevo-me abaixo.



Documento assinado digitalmente

**SIMONE SILMARA WERNER**

Data: 10/07/2026 17:18:29-0300

CPF: \*\*\*.925.219-\*\*-\*\*

Verifique as assinaturas em <https://v.ufsc.br>

---

Simone Silmara Werner

Projeto Céos

## RESUMO

O controle e a transparência nos gastos públicos exigem mecanismos eficazes para a identificação de sobrepreços em licitações. Para viabilizar essa análise em larga escala, é necessário um meio de agrupar registros de compras. O código GTIN (Global Trade Item Number), emitido pela organização GS1 e presente nas Notas Fiscais Eletrônicas (NF-e) de licitações, cumpriria esse papel, no entanto, o campo está frequentemente ausente ou incorretamente preenchido. Este trabalho, desenvolvido no âmbito do Projeto Céos, iniciativa de pesquisa entre a Universidade Federal de Santa Catarina (UFSC) e o Ministério Público de Santa Catarina (MPSC), propõe um método computacional para a validação e alocação automática desse código, estruturado em três etapas. Na primeira etapa, avaliaram-se diferentes abordagens de Processamento de Linguagem Natural (PLN) para classificar se um par de descrições textuais com mesmo GTIN, uma proveniente da NF-e e outra da base oficial da GS1, refere-se ao mesmo produto. Na segunda etapa, implementou-se um *pipeline* de busca com o Apache Solr para o preenchimento do código usando a abordagem escolhida na primeira etapa. Por fim, na terceira etapa, desenvolveu-se um método de agrupamento de produtos equivalentes na base da GS1, expandindo a classificação de pares equivalentes para formar grupos de produtos, com o objetivo de encontrar GTINs equivalentes. Para isso, fez-se uso do algoritmo Locality-Sensitive Hashing (LSH) junto do MinHash para validação dos pares e a formação dos grupos finais se dá por componentes conexas dos pares equivalentes encontrados. Para a tarefa de validação o método baseado na similaridade de Jaccard sobre trigramas teve o melhor resultado, alcançando  $F_{0.5}$  score de 0.916. Já a solução de preenchimento foi avaliada sobre 1104 notas fiscais sem GTIN, tendo uma taxa de acerto de 95.97%. Os resultados demonstram a viabilidade do método proposto em automatizar a identificação e agrupamento de produtos em compras públicas.

**Palavras-chave:** Correspondência de Produtos, Processamento de Linguagem Natural, Classificação, Apache Solr, Locality-Sensitive Hashing

## ABSTRACT

Control and transparency in public spending require effective mechanisms to identify overpricing in procurement processes. To enable this analysis on a large scale, a means to group purchasing records is necessary. The GTIN (Global Trade Item Number) code, issued by the GS1 organization and present in electronic invoices (NF-e), would fulfill this role however, this field is frequently missing or incorrectly filled. This work, developed in the Céos Project, a collaborative research initiative between the Universidade Federal de Santa Catarina (UFSC) and the Ministério Público de Santa Catarina (MPSC) proposes a computational method for the automatic validation and allocation of this code, structured into three stages. In the first stage, different Natural Language Processing (NLP) approaches were evaluated to classify whether a pair of textual descriptions with the same GTIN, one from the NF-e and the other from the official GS1 database, refers to the same product. In the second stage, a search pipeline was implemented using Apache Solr for code allocation, utilizing the approach selected in the first stage. Finally, in the third stage, a method was developed to cluster equivalent products within the GS1 database, expanding the pairwise equivalence classification to form product groups with the goal of identifying equivalent GTINs. To achieve this, the Locality-Sensitive Hashing (LSH) algorithm was utilized alongside MinHash for pair validation, and the final groups were formed by the connected components of the identified equivalent pairs. For the validation task, the method based on Jaccard similarity over trigrams achieved the best result, reaching an  $F_{0.5}$  score of 0.916. On the other hand, the allocation solution was evaluated on 1.104 invoices without GTIN, achieving an accuracy rate of 95.97%. The results demonstrate the viability of the proposed method for automating the identification and grouping of products in public purchasing.

**Keywords:** Product Matching, Natural Language Processing, Classification, Apache Solr, Locality-Sensitive Hashing

## LISTA DE FIGURAS

|                                             |    |
|---------------------------------------------|----|
| Figura 1 – Fluxo da solução . . . . .       | 28 |
| Figura 2 – Fluxo da validação . . . . .     | 32 |
| Figura 3 – Estrutura do NCM . . . . .       | 33 |
| Figura 4 – Fluxo do preenchimento . . . . . | 42 |
| Figura 5 – Formação dos grupos . . . . .    | 45 |

## LISTA DE TABELAS

|                                                                                |    |
|--------------------------------------------------------------------------------|----|
| Tabela 1 – Tipos de pares por produto . . . . .                                | 34 |
| Tabela 2 – Campos do conjunto anotado . . . . .                                | 34 |
| Tabela 3 – Grade de hiperparâmetros para o XGBoost . . . . .                   | 35 |
| Tabela 4 – Resultados dos métodos no conjunto de teste . . . . .               | 38 |
| Tabela 5 – Limiares de classificação ajustados . . . . .                       | 38 |
| Tabela 6 – Hiperparâmetros selecionados do XGBoost . . . . .                   | 38 |
| Tabela 7 – Etapas de pré-processamento selecionadas via Forward Selection .    | 39 |
| Tabela 8 – Definição do Schema do Apache Solr para o Core de Produtos . . .    | 41 |
| Tabela 9 – Resultados do pipeline de busca de GTIN para diferentes limiares .  | 42 |
| Tabela 10 – Resultados do agrupamento para diferentes limiares de similaridade | 45 |

## SUMÁRIO

|              |                                                         |           |
|--------------|---------------------------------------------------------|-----------|
| <b>1</b>     | <b>INTRODUÇÃO</b>                                       | <b>13</b> |
| 1.1          | O LABORATÓRIO                                           | 14        |
| 1.2          | OBJETIVOS                                               | 14        |
| 1.3          | ORGANIZAÇÃO DO DOCUMENTO                                | 15        |
| <b>2</b>     | <b>FUNDAMENTAÇÃO TEÓRICA</b>                            | <b>16</b> |
| 2.1          | PROCESSAMENTO DE LINGUAGEM NATURAL                      | 16        |
| 2.2          | FUNÇÕES DE SIMILARIDADE TEXTUAL                         | 17        |
| <b>2.2.1</b> | <b>Similaridade de Jaccard</b>                          | <b>17</b> |
| <b>2.2.2</b> | <b>Coeficiente de Dice</b>                              | <b>17</b> |
| <b>2.2.3</b> | <b>Levenshtein e Damerau-Levenshtein</b>                | <b>18</b> |
| <b>2.2.4</b> | <b>Similaridade do cosseno</b>                          | <b>18</b> |
| 2.3          | EMBEDDINGS                                              | 18        |
| 2.4          | APRENDIZADO DE MÁQUINA                                  | 19        |
| <b>2.4.1</b> | <b>Métricas de classificação</b>                        | <b>19</b> |
| 2.5          | RESOLUÇÃO DE ENTIDADES                                  | 20        |
| <b>2.5.1</b> | <b>Locality-Sensitive Hashing e MinHash</b>             | <b>21</b> |
| <b>2.5.2</b> | <b>Métricas de avaliação de clusters</b>                | <b>22</b> |
| 2.6          | APACHE SOLR                                             | 24        |
| <b>3</b>     | <b>DESCRIÇÃO DO PROBLEMA E REQUISITOS TÉCNICOS</b>      | <b>25</b> |
| 3.1          | REQUISITOS TÉCNICOS                                     | 26        |
| <b>3.1.1</b> | <b>Requisitos Funcionais</b>                            | <b>26</b> |
| <b>3.1.2</b> | <b>Requisitos Não Funcionais</b>                        | <b>26</b> |
| <b>4</b>     | <b>SOLUÇÃO PROPOSTA E METODOLOGIA UTILIZADA</b>         | <b>28</b> |
| 4.1          | VALIDAÇÃO                                               | 28        |
| 4.2          | PREENCHIMENTO DO GTIN                                   | 30        |
| 4.3          | DE-DUPLICAÇÃO E AGRUPAMENTO                             | 30        |
| <b>5</b>     | <b>DESENVOLVIMENTO E ANÁLISE DOS RESULTADOS OBTIDOS</b> | <b>32</b> |
| 5.1          | VALIDAÇÃO DE GTIN                                       | 32        |
| <b>5.1.1</b> | <b>Construção do conjunto anotado</b>                   | <b>33</b> |
| <b>5.1.2</b> | <b>Implementação dos Métodos</b>                        | <b>34</b> |
| <b>5.1.3</b> | <b>Estratégias de Pré-processamento</b>                 | <b>36</b> |
| <b>5.1.4</b> | <b>Resultados e Análise: Validação GTIN</b>             | <b>38</b> |
| 5.2          | PREENCHIMENTO DO GTIN                                   | 39        |
| <b>5.2.1</b> | <b>Construção do Core e Definição do Schema</b>         | <b>40</b> |
| <b>5.2.2</b> | <b>Pipeline de busca</b>                                | <b>41</b> |
| <b>5.2.3</b> | <b>Resultados e Análise: Preenchimento do GTIN</b>      | <b>42</b> |
| 5.3          | AGRUPAMENTO DE PRODUTOS EQUIVALENTES                    | 43        |

|       |                                                       |           |
|-------|-------------------------------------------------------|-----------|
| 5.3.1 | Construção de conjunto de dados para testes . . . . . | 43        |
| 5.3.2 | Desenvolvimento da solução de agrupamento . . . . .   | 43        |
| 5.3.3 | Resultados e Análise: Agrupamento de GTIN . . . . .   | 45        |
| 5.4   | RESULTADOS GERAIS . . . . .                           | 46        |
| 6     | <b>CONCLUSÃO</b> . . . . .                            | <b>47</b> |
| 6.1   | SUGESTÕES PARA TRABALHOS FUTUROS . . . . .            | 47        |
|       | <b>REFERÊNCIAS</b> . . . . .                          | <b>49</b> |

## 1 INTRODUÇÃO

As compras públicas no Brasil, realizadas por meio de processos licitatórios são a principal forma de aquisição de bens e serviços pelo estado e apenas no ano de 2025, movimentaram cerca de 1 trilhão de reais (GOV.BR, 2026). Nesse cenário, o sobrepreço, prática caracterizada pela aquisição de produtos ou serviços por valores expressivamente superiores aos praticados no mercado (Brasil, 2021) é uma forma comum de irregularidade encontrada em licitações.

A detecção dessa prática é feita através da comparação de preços efetivamente pagos por itens equivalentes em diferentes processos de compra. Essa comparação, por sua vez, exige que as mercadorias adquiridas sejam corretamente identificadas, de forma a tornar possível agrupar esses registros (Silva et al., 2023). Dentre os documentos gerados em licitações, a Nota Fiscal Eletrônica (NF-e) possui informação sobre os produtos comprados e os preços pagos por eles.

Nas notas fiscais a forma mais direta de identificar e agrupar os produtos é o código GTIN (Global Trade Item Number, Número Global de Item Comercial), padrão internacional de identificação de produtos gerenciado pela organização GS1. No entanto, na prática este campo frequentemente não é preenchido pelo emissor da nota ou é preenchido incorretamente.

Portanto, para viabilizar análises de sobrepreço em escala, torna-se necessário desenvolver métodos para verificar se o código GTIN foi preenchido corretamente e, em caso de não preenchimento ou preenchimento incorreto, realizar o preenchimento de forma automatizada. Para isso, pode-se fazer uso da descrição textual do produto presente na Nf-e, informação além do GTIN a qual caracteriza o produto da nota. Em uma nota fiscal corretamente preenchida, a descrição presente nela deve se referir ao mesmo produto da descrição do GTIN na base da GS1, mesmo que não escrito da mesma forma.

Além disso, na base da GS1 podem existir múltiplos códigos referentes ao mesmo produto por conta de variações de formato ou apresentação. Outro ponto relevante são itens de marcas diferentes que são essencialmente o mesmo produto. Portanto, para formar grupos mais relevantes para a análise de sobrepreço é preciso agrupar registros equivalentes na base da GS1.

Este trabalho apresenta um método para verificação e preenchimento dos campos do GTIN e agrupamento de GTINs referentes a produtos semelhantes para construção de um banco de preços. Primeiramente, investiga-se diferentes métodos para comparar pares de descrições de produtos, sendo um da Nf-e e outro da GS1 e classificá-los como equivalentes ou não. Em seguida, propõe-se um sistema de busca baseado no Apache Solr para preencher campos de GTIN ausentes em notas fiscais. Por fim, estende-se o método de classificação dos pares de descrições para formar gru-

pos de GTIN equivalentes na base da GS1, usando dos algoritmos Locality-Sensitive Hashing e MinHash.

Os resultados obtidos considerando os de conjuntos de dados de teste indicam que a solução proposta é capaz de cumprir com seus objetivos com alta precisão, permitindo a validação e preenchimento do código GTIN das notas fiscais com baixa taxa de erros, vital para análises de sobrepreço.

## 1.1 O LABORATÓRIO

O Projeto Céos é uma iniciativa de pesquisa da Universidade Federal de Santa Catarina (UFSC) em parceria com o Ministério Público de Santa Catarina (MPSC), com o objetivo de desenvolver soluções tecnológicas para auxiliar promotores e procuradores do MPSC na identificação de irregularidades em processos licitatórios. Ele é composto por pesquisadores dos seguintes laboratórios da UFSC: Laboratório de Estatística Aplicada (LEA), Laboratório de Inteligência Artificial e Algoritmos (LIAA), Laboratório de Integração de Sistemas e Aplicações Avançadas (LISA) e Laboratório de Pesquisa em Sistemas Distribuídos (LAPeSD).

O presente trabalho está inserido na linha de pesquisa e desenvolvimento dedicada à análise de sobrepreço em licitações. Seu principal foco é a construção de um banco de preços, ferramenta que permite a comparação de preços efetivamente pagos por produtos similares em diferentes licitações.

## 1.2 OBJETIVOS

O objetivo geral deste trabalho foi desenvolver um método computacional para a validação e alocação automática do código GTIN em notas fiscais eletrônicas de licitações públicas, comparando as descrições textuais das NF-e com a base oficial da GS1 Brasil, com o intuito de aprimorar e expandir o banco de preços do Céos. Para atingir esse objetivo geral, foram definidos os seguintes objetivos específicos:

- Desenvolver e avaliar métodos para classificar se um par de descrições de produtos refere-se a mesma entidade.
- Implementar um sistema de busca que, a partir dos dados de uma NF-e sem GTIN preenchido, recupere candidatos na base oficial da GS1 e selecione o código mais adequado utilizando o melhor método identificado no objetivo anterior.
- Propor um método de agrupamento de produtos equivalentes na base da GS1, permitindo encontrar GTINs equivalentes para formar grupos finais mais relevantes.

### 1.3 ORGANIZAÇÃO DO DOCUMENTO

Este trabalho está organizado da seguinte forma. O Capítulo 2 apresenta os fundamentos teóricos necessários para a compreensão do trabalho. O Capítulo 3 descreve o contexto, problema abordado e os requisitos técnicos definidos. O Capítulo 4 apresenta a solução proposta e o método adotado. O Capítulo 5 detalha a implementação realizada e analisa os resultados obtidos. Por fim, o Capítulo 6 sintetiza as contribuições do trabalho, discute as limitações encontradas e aponta direções para trabalhos futuros.

## 2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os fundamentos teóricos que serviram de base para o desenvolvimento deste trabalho, com o objetivo de fornecer ao leitor o conhecimento necessário para compreender os conceitos e métodos apresentados.

A Seção 2.1 introduz o conceito de processamento de linguagem natural (PLN). A Seção 2.2 apresenta funções matemáticas para medir similaridade textual. A Seção 2.3 descreve as representações vetoriais de textos, conhecidas como *embeddings*. A Seção 2.6 apresenta a ferramenta de busca de informações Apache Solr. A Seção 2.4 traz conceitos fundamentais de aprendizado de máquina para tarefas de classificação. Por fim, a Seção 2.5 apresenta o problema de resolução de entidades e métodos específicos utilizados neste trabalho.

### 2.1 PROCESSAMENTO DE LINGUAGEM NATURAL

O Processamento de Linguagem Natural (PLN) é a subárea da Inteligência Artificial (IA) dedicada ao desenvolvimento de métodos computacionais para permitir que computadores reconheçam, compreendam e gerem texto. O escopo do PLN é amplo e abrange tarefas como classificação de textos, análise de sentimento, extração de informação, resumo de conteúdo, geração de texto, entre outras (Stryker; Holdsworth, 2026).

Os métodos de Processamento de Linguagem Natural (PLN) podem ser organizados em três abordagens principais: a baseada em regras (ou simbólica), a estatística e a de deep learning. A abordagem simbólica funciona a partir de árvores de decisão com regras programadas previamente, sua capacidade se restringe a responder a comandos específicos e predefinidos. Com a introdução de representações vetoriais de textos, deu-se origem à abordagem estatística, que faz uso de métodos estatísticos e de aprendizado de máquina para ensinar modelos a realizar tarefas específicas. Por fim, o estado da arte de PLN atualmente é uma evolução da abordagem estatística usando deep learning, que consiste no treinamento de redes neurais com grandes volumes de dados (Stryker; Holdsworth, 2026).

Independentemente da abordagem adotada, a maioria das aplicações de PLN começa com uma etapa de processamento textual. As técnicas aplicadas visam reduzir o ruído presente nos textos e padronizá-los. A escolha entre elas depende do problema enfrentado e da solução adotada. As técnicas mais comuns são:

- Tokenização: segmentação do texto em unidades elementares (tokens).
- Normalização ortográfica: conversão para maiúsculas ou minúsculas, remoção de pontuação e acentos.

- Remoção de stopwords: eliminação de palavras muito frequentes que agregam pouco valor semântico à tarefa (como artigos, conjunções e preposições).
- Redução morfológica: redução das palavras à sua forma canônica ou radical, processo geralmente realizado por meio de radicalização (stemming) ou lematização.

## 2.2 FUNÇÕES DE SIMILARIDADE TEXTUAL

A similaridade textual é a tarefa de quantificar o quão próximos são dois textos e pode ser dividida em duas vertentes principais:

**Similaridade léxica:** mede o grau de semelhança comparando as palavras e o conjunto de letras que compõe os textos. Essa abordagem divide-se em duas categorias principais. A primeira engloba os métodos baseados em caracteres, que comparam a sequência direta das letras para identificar semelhanças. A segunda categoria abrange os métodos baseados em conjuntos de tokens, que decompõem o texto e o tratam como uma coleção de palavras (ou conjunto de  $n$  caracteres) isoladas (Tiwari, 2025).

**Similaridade semântica:** vai além da comparação de palavras e caracteres e busca capturar o significado e a intenção por trás das frases, identificando correspondências como sinônimos e paráfrases mesmo quando não há compartilhamento de vocabulário. Funciona principalmente por meio de representações vetoriais densas *embeddings*, que projetam os textos em um espaço numérico no qual a proximidade matemática entre vetores indica similaridade de sentido. A seguir são apresentadas as funções de similaridade utilizadas neste trabalho (Tiwari, 2025).

### 2.2.1 Similaridade de Jaccard

Dado dois textos, sejam  $A$  e  $B$  os conjuntos de *tokens* que formam os textos, a similaridade de Jaccard calcula o grau de semelhança entre eles a partir da razão entre a quantidade de *tokens* comuns (a interseção,  $A \cap B$ ) e o número total de *tokens* únicos presentes nos dois conjuntos (a união,  $A \cup B$ ):

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \quad (1)$$

A medida assume valores no intervalo  $[0, 1]$ , sendo igual a 0 quando os conjuntos não compartilham nenhum token e a 1 quando são idênticos (Noble, 2026).

### 2.2.2 Coeficiente de Dice

O coeficiente de Dice é uma medida calculada sobre tokens como a de Jaccard no entanto, ela dá o dobro do peso à interseção e divide pela soma dos tamanhos dos conjuntos (Harisha, 2023).

$$\text{Dice}(A, B) = \frac{2 |A \cap B|}{|A| + |B|}.$$

### 2.2.3 Levenshtein e Damerau-Levenshtein

A distância de Levenshtein cai na categoria de distâncias de edição, ela compara diretamente as sequências de caracteres entre dois textos  $A$  e  $B$  e é definida como o número mínimo de operações elementares (inserção, remoção e substituição) de um único caractere necessárias para transformar  $A$  em  $B$ . Damerau-Levenshtein é uma variação dessa medida que estende essa definição admitindo, além das três operações anteriores, a transposição de dois caracteres adjacentes como operação de custo unitário. Para obter uma medida de similaridade no intervalo  $[0, 1]$ , é comum normalizar a distância (Hossain et al., 2023):

$$\text{sim}(A, B) = 1 - \frac{d(A, B)}{\max(|A|, |B|)}.$$

### 2.2.4 Similaridade do cosseno

Quando os textos são representados como vetores, a medida de proximidade mais comumente utilizada é a similaridade do cosseno, que corresponde ao cosseno do ângulo formado pelos vetores:

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|}. \quad (2)$$

A medida assume valores no intervalo  $[-1, 1]$ , sendo igual a 1 para vetores colineares de mesmo sentido e a 0 para vetores ortogonais (Tom Krantz, 2026). Essa função está diretamente associada à similaridade semântica calculada por meio de representações vetoriais densas, apresentadas na seção seguinte.

## 2.3 EMBEDDINGS

*Embeddings* são representações numéricas de dados com alta dimensionalidade, como textos, em vetores densos de baixa dimensionalidade, o que permite que sejam analisados por métodos computacionais (Harsoor, 2024).

A literatura distingue duas grandes famílias de modelos. Os *embeddings* estáticos, como Word2Vec (Mikolov et al., 2013), atribuem um vetor fixo a cada palavra independentemente do contexto em que ela aparece. Já os *embeddings* contextuais, como o BERT (Devlin et al., 2019) são baseados na arquitetura Transformer e geram representações que variam conforme o contexto da sentença.

Para realizar medição de similaridade entre textos com base em *embeddings*, aplica-se a equação 2 do cosseno do ângulo entre os vetores. Ela se baseia no conceito

de que vetores que apontam para a mesma direção possuem significados parecidos (Tom Krantz, 2026).

## 2.4 APRENDIZADO DE MÁQUINA

O aprendizado de máquina (AM) é o campo da computação que estuda algoritmos capazes de aprender padrões a partir de dados e estender esse aprendizado para novos registros. Os métodos de AM dividem-se principalmente em supervisionados, nos quais os exemplos de treinamento são rotulados, e não supervisionados, nos quais o algoritmo busca padrões em dados sem anotação prévia.

A classificação é uma das tarefas centrais do aprendizado supervisionado, consistindo na atribuição de dados a classes pré-definidas com base em suas características (features) de entrada. Para isso, os algoritmos ajustam modelos matemáticos a partir de conjuntos de treinamento rotulados, com o objetivo de generalizar o aprendizado e associar padrões desconhecidos a categorias já conhecidas (MAmer F.A.H. ALNUAIMI, 2024).

### 2.4.1 Métricas de classificação

A avaliação de um classificador é fundamentada na matriz de confusão, que organiza as predições em quatro categorias: verdadeiros positivos (VP), falsos positivos (FP), verdadeiros negativos (VN) e falsos negativos (FN), comparando as predições do modelo com o rótulo. A partir desses valores, definem-se métricas específicas, como acurácia, precisão, revocação e F score (Kulkarni, 2023).

**Acurácia:** mede a proporção global de predições corretas:

$$Acc = \frac{VP + VN}{VP + FP + VN + FN}$$

**Precisão:** quantifica a fração dos exemplos classificados como positivos que são de fato positivos:

$$P = \frac{VP}{VP + FP}$$

**Revocação (recall):** indica a fração dos exemplos positivos que foram corretamente identificados:

$$R = \frac{VP}{VP + FN}$$

$F_\beta$ : combina precisão e revocação em uma única métrica, ponderando a importância relativa de cada uma por meio do parâmetro  $\beta$ :

$$F_\beta = (1 + \beta^2) \frac{P \cdot R}{\beta^2 P + R}$$

Para  $\beta = 1$  obtém-se o  $F_1$ , que trata precisão e revocação com pesos iguais. Para  $\beta = 0,5$ , a métrica  $F_{0,5}$  atribui maior importância à precisão em relação à revocação, sendo adequada a cenários nos quais o custo de um falso positivo supera o de um falso negativo. Portanto, a escolha do parâmetro  $\beta$  deve levar em conta o contexto e objetivo do problema em questão.

## 2.5 RESOLUÇÃO DE ENTIDADES

A resolução de entidades (ER) é o processo de agregar informações que se referem à mesma entidade do mundo real. De forma mais direta, o problema de ER consiste em avaliar pares de registros e classificá-los como correferentes (*links*) ou não-correferentes (*non-links*) (Olivier Binette, 2022). Esse problema abrange dois cenários principais: a vinculação de registros (*record linkage*), que trata de encontrar correspondências entre bases distintas, e a deduplicação, que atua sobre uma única base de dados com o propósito de identificar registros equivalentes internamente. O pipeline típico de ER compreende quatro etapas fundamentais:

(i) Alinhamento de atributos: registros são analisados para a identificação de um conjunto de características em comum;

(ii) Blocagem (*blocking*): restringe-se o espaço de comparação ao agrupar registros similares em blocos, assumindo automaticamente como não-correferentes os pares que não dividem o mesmo bloco com o intuito de reduzir a quantidade de comparações realizadas;

(iii) Resolução de entidades: Aplica-se solução para identificar os pares correferentes;

(iv) Canonicalização ou fusão: Responsável por mesclar as entidades resolvidas em um único registro representativo.

A principal dificuldade em ER reside na escalabilidade do processo. Para um total combinado de  $N$  registros, uma abordagem de comparação exaustiva exige a avaliação de  $N(N - 1)/2$  pares possíveis. O custo quadrático desse processo o torna inviável para grandes conjuntos de dados, destacando o importante papel da etapa de *blocking*.

Quando o problema de deduplicação necessita da formação dos grupos de entidades, avaliar os pares independentemente não basta, pois não assegura o fechamento transitivo (isto é, a garantia de que se um registro  $a$  está ligado a  $b$ , e  $b$  a  $c$ , então  $a$  também deve estar ligado a  $c$ ). Para solucionar este problema é possível adotar uma etapa de pós-processamento para a formação de grupos (clusters), formando um grafo com os pares correspondentes. Nesse cenário, constrói-se um grafo não direcionado no qual os nós representam os registros individuais e as arestas indicam os vínculos identificados. Os clusters são então definidos por meio da detecção das componentes conexas desse grafo (Olivier Binette, 2022).

### 2.5.1 Locality-Sensitive Hashing e MinHash

Conforme apresentado anteriormente, a comparação exaustiva de todos os pares de registros possui custo quadrático  $O(N^2)$ , tornando-se inviável para grandes bases de dados. O MinHash combinado com o *Locality-Sensitive Hashing* (LSH) constitui uma solução eficiente e amplamente adotada para esse problema, permitindo identificar pares de registros similares através de um cálculo aproximado da similaridade de Jaccard e diminuindo a quantidade de comparações necessárias, resultando em complexidade aproximadamente linear  $O(N)$  na prática (Brenndoerfer, 2026).

O objetivo do MinHash é representar cada documento por meio de uma assinatura compacta (vetor numérico) de tamanho  $k$ , sendo  $k$  o número de funções de hash utilizadas para transformar os *tokens* do documento em números inteiros. Para cada uma das  $k$  funções de hash  $h_1, h_2, \dots, h_k$ , aplica-se a função a todos os elementos do conjunto de *tokens* do documento  $D$  e retém-se apenas o menor valor obtido. Desse modo, a assinatura do documento é dada pelo vetor:

$$\text{sig}(D) = [\text{minhash}_1(D), \text{minhash}_2(D), \dots, \text{minhash}_k(D)]$$

A propriedade fundamental na qual o MinHash se baseia estabelece que a probabilidade de duas assinaturas concordarem em uma determinada posição  $i$  é exatamente igual à similaridade de Jaccard entre os conjuntos originais:

$$P(\text{minhash}_i(A) = \text{minhash}_i(B)) = J(A, B)$$

Como consequência direta dessa propriedade, o cálculo da similaridade aproximada de Jaccard entre dois documentos baseia-se exclusivamente na comparação de suas assinaturas. Basta calcular a razão do número de posições em que os elementos dos vetores  $\text{sig}(A)$  e  $\text{sig}(B)$  são idênticos sobre o tamanho  $k$  das assinaturas. O erro padrão dessa aproximação decresce em  $1/\sqrt{k}$ . Quanto maior o valor de  $k$ , mais precisa a estimativa, porém maior o custo de armazenamento e computação das assinaturas.

Mesmo com assinaturas compactas de tamanho  $k$ , comparar todos os pares de assinaturas ainda resulta em custo quadrático  $O(N^2)$ . O LSH resolve esse problema por meio da técnica de bandas (banding). A assinatura de tamanho  $k$  do MinHash é particionada em  $b$  bandas de  $r$  linhas cada, sendo  $k = b \times r$ . Para cada banda, aplica-se uma função de hash sobre o sub-vetor de tamanho  $r$  de forma que aqueles que possuem a mesma sequência exata de números em um determinado sub-vetor produzirão, obrigatoriamente, o mesmo valor de hash resultante. Esse valor serve como um identificador de recipiente (bucket). Ao final do processo, apenas os registros que compartilham pelo menos um bucket em qualquer uma das bandas são classificados como pares candidatos, eliminando a necessidade de comparar exaustivamente todos os pares de registros.

A probabilidade de um par com similaridade de Jaccard  $s$  tornar-se um par candidato é dada por:

$$P(\text{candidato} \mid s) = 1 - (1 - s^r)^b.$$

Essa função exibe o comportamento de uma curva em S, pares com similaridade acima de um limiar  $t$  têm alta probabilidade de serem capturados como candidatos, enquanto pares abaixo desse limiar são descartados com alta probabilidade, sem serem explicitamente comparados. O limiar pode ser aproximado por:

$$t \approx \left(\frac{1}{b}\right)^{1/r}.$$

### 2.5.2 Métricas de avaliação de clusters

O problema de deduplicação pode ser adaptado para um problema de clusterização quando é do interesse manter os registros duplicados e formar grupos de entidades equivalentes com eles. Para avaliar a qualidade desses agrupamentos quando se dispõe de um conjunto de dados anotados (com rótulos de grupo verdadeiros), recorre-se às métricas externas de avaliação de *clustering*. Essas métricas comparam os *clusters* gerados automaticamente com os grupos anotados previamente, quantificando o quão próximos eles são. Nas definições a seguir,  $K$  denota as classes verdadeiras (rótulos de referência) e  $C$  denota os *clusters* gerados pelo sistema. As métricas finais são calculadas com base nesses conceitos (Artus, 2024).

Para a formulação matemática, as seguintes variáveis são definidas:

- $N$ : representa o número total de registros no conjunto de dados;
- $n_k$ : é o número total de registros que pertencem à classe verdadeira  $k$ ;
- $n_c$ : é o número total de registros atribuídos pelo sistema ao *cluster*  $c$ ;
- $n_{ck}$ : denota o número de registros que pertencem à classe verdadeira  $k$  e foram atribuídos ao *cluster*  $c$ ;

Definidos esses parâmetros, podemos calcular os valores de entropia do conjunto. A entropia marginal das classes verdadeiras,  $H(K)$ , mede o quão misturados os dados estão inicialmente:

$$H(K) = - \sum_k \frac{n_k}{N} \log \left( \frac{n_k}{N} \right)$$

A entropia condicional das classes dado o agrupamento,  $H(K \mid C)$ , quantifica a incerteza que ainda resta sobre as classes reais *após* conhecermos os *clusters*

atribuídos pelo sistema:

$$H(K | C) = - \sum_c \sum_k \frac{n_{ck}}{N} \log \left( \frac{n_{ck}}{n_c} \right)$$

A entropia dos clusters,  $H(C)$ , mede a dispersão ou desordem dos dados entre os grupos gerados pelo sistema:

$$H(C) = - \sum_c \frac{n_c}{N} \log \left( \frac{n_c}{N} \right)$$

A entropia condicional dos clusters dadas as classes,  $H(C | K)$ , mede o quanto os membros de uma mesma classe real se fragmentaram e se espalharam por múltiplos agrupamentos:

$$H(C | K) = - \sum_k \sum_c \frac{n_{ck}}{N} \log \left( \frac{n_{ck}}{n_k} \right)$$

Tendo definido  $H(K)$ ,  $H(C)$ ,  $H(C | K)$  e  $H(K | C)$ , podemos calcular as métricas realmente usadas para avaliar os grupos formados, sendo elas: Homogeneidade, Completude e *V-measure*.

A Homogeneidade ( $h$ ) mede se cada *cluster* gerado contém apenas registros pertencentes a uma única classe verdadeira:

$$h = 1 - \frac{H(K | C)}{H(K)}$$

O valor  $h = 1$  indica que cada *cluster* é composto exclusivamente por membros de uma única classe.

A Completude ( $c$ ) mede se todos os registros de uma mesma classe verdadeira foram atribuídos ao mesmo *cluster*:

$$c = 1 - \frac{H(C | K)}{H(C)}$$

O valor  $c = 1$  indica que nenhuma classe verdadeira foi fragmentada em múltiplos *clusters*.

A *V-measure* ( $v$ ) pondera o *trade-off* entre os dois critérios anteriores, pois criar agrupamentos excessivamente pequenos e específicos aumenta a homogeneidade, mas diminui a completude. Para unificar ambas as métricas em um valor único, usa-se a média harmônica ponderada:

$$v = \frac{(1 + \beta^2) \cdot h \cdot c}{\beta^2 \cdot h + c}$$

Para  $\beta = 1$ , a métrica trata os dois critérios com pesos iguais. As três métricas ( $h$ ,  $c$  e  $v$ ) variam no intervalo  $[0, 1]$ , sendo 1 o valor ótimo.

## 2.6 APACHE SOLR

O Apache Solr consiste em uma plataforma de busca e recuperação de informações de código aberto. Ele tem funcionamento distribuído e permite a criação de índices eficientes para otimizar buscas textuais em grandes volumes de dados (Sharma, 2023).

Um índice é o equivalente a uma tabela no Solr e ele é composto por documentos, que são os registros individuais armazenados no índice. As configurações que definem quais campos existem, seus tipos e quais formas de busca podem ser realizadas neles são definidas na estrutura chamada de schema (Apache Solr, s.d.).

### 3 DESCRIÇÃO DO PROBLEMA E REQUISITOS TÉCNICOS

Este capítulo apresenta o contexto, a motivação e os requisitos técnicos do problema abordado neste trabalho.

O Projeto Céos é uma iniciativa de pesquisa tecnológica da Universidade Federal de Santa Catarina (UFSC) em parceria com o Ministério Público de Santa Catarina (MPSC), com o objetivo de desenvolver soluções baseadas em dados para apoiar promotores e procuradores na identificação de irregularidades em processos licitatórios.

O projeto abrange diversas linhas de pesquisa, cujos resultados são concentrados no Painel Integrado Inteligente (PII). O painel centraliza e integra dados de diversas fontes, superando a necessidade de inspeções manuais em grandes volumes de dados dispersos. Além disso, faz uso das soluções desenvolvidas no projeto para automatizar a vigilância de processos licitatórios, gerando alertas automáticos sobre processos suspeitos.

Com isso, a ferramenta consolida todos os métodos desenvolvidos no Céos para automatizar a vigilância de contratações públicas, apoiando diretamente o Ministério Público de Santa Catarina (MPSC) na prevenção de fraudes e na geração de alertas automáticos sobre processos licitatórios suspeitos

Dentre as linhas de pesquisa do projeto, existe a de sobrepreço, que é definido como a prática fraudulenta pela qual o valor orçado ou cobrado em uma licitação é expressivamente superior aos preços referenciais de mercado (Brasil, 2021). Sua detecção depende da comparação sistemática dos preços de um mesmo item. Para que essa comparação seja realizada em larga escala, produtos adquiridos em licitações distintas precisam ser identificados como equivalentes, permitindo a construção de um banco de preços.

A principal fonte de dados de compras públicas é a Nota Fiscal Eletrônica (NF-e). Ela é emitida para toda compra feita em licitações, e nela estão informações dos produtos adquiridos, quantidade e preços pagos. O projeto Céos tem acesso, por meio do MPSC, à base completa de notas fiscais dos processos licitatórios do estado de Santa Catarina. Das informações presentes na nota fiscal, existe um código de identificação do produto chamado GTIN, que pode ser utilizado para agrupar as notas fiscais por produtos comprados, permitindo a realização de análises de preço.

O código GTIN (*Global Trade Item Number*), emitido pela organização GS1, é um padrão de identificação global de produtos, de forma que cada produto específico tenha um código único que o representa (GS1 Brasil, s.d.). A GS1 Brasil mantém uma base com milhões de produtos cadastrados pelos próprios fabricantes, contendo o código GTIN, a descrição do produto e a marca, e o Céos tem acesso a esses dados através do MPSC. Quando corretamente preenchido na NF-e, esse código permite o agrupamento das notas fiscais de forma direta. Porém, na prática, o campo GTIN está

frequentemente ausente e, mesmo quando preenchido, pode estar incorreto por conta de erro humano.

O desafio central deste trabalho é garantir o preenchimento correto do código GTIN nas NF-e ligadas a licitações do estado, de modo a viabilizar o agrupamento confiável de produtos para fins de análise de preços. Além disso, o GTIN é um identificador bastante específico de um produto e podem existir múltiplos códigos para variações do mesmo. Ademais, para a análise de sobrepreço é importante não separar os produtos por marca pois, em licitações, não é permitido indicar marcas específicas para serem compradas. Para isso, visando atender esta necessidade, foram estipuladas três etapas:

Etapas 1: para os casos nos quais o GTIN está preenchido na nota fiscal, devemos verificar se está correto.

Etapas 2: no caso de não ter sido preenchido ou estar errado, é necessário atribuir um valor para este campo.

Etapas 3: identificar produtos equivalentes na base de GTIN da GS1.

### 3.1 REQUISITOS TÉCNICOS

Com base no problema apresentado anteriormente, foram definidos os seguintes requisitos funcionais e não funcionais para o desenvolvimento da solução.

#### 3.1.1 Requisitos Funcionais

Os requisitos funcionais descrevem o que o sistema deve fazer de forma prática para resolver o problema:

**RF01 - Validação de GTIN existente:** Capacidade de verificar se o código GTIN que já veio preenchido na NF-e está correto em relação à base de dados da GS1.

**RF02 - Atribuição de GTIN ausente ou incorreto:** Preencher o código GTIN correto para as notas fiscais que estiverem com esse campo em branco ou que não passaram na validação.

**RF03 - Agrupamento dos GTIN:** O sistema deve ser capaz de identificar produtos equivalentes na base da GS1 e agrupá-los desconsiderando a marca.

#### 3.1.2 Requisitos Não Funcionais

Os requisitos não funcionais definem os critérios de qualidade e restrições com as quais o sistema deve operar:

**RNF01 - Escalabilidade:** Capacidade de processar o grande volume de dados de notas fiscais e produtos cadastrados na base da GS1.

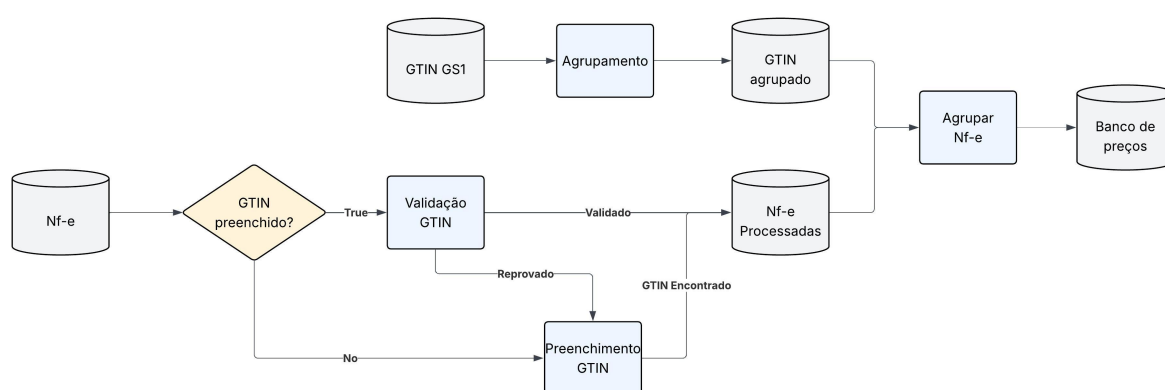
**RNF02 - Precisão dos métodos:** Os métodos devem ter um alto índice de acerto (precisão) para evitar agrupamentos errados que prejudiquem a análise de sobrepreço.

## 4 SOLUÇÃO PROPOSTA E METODOLOGIA UTILIZADA

Este capítulo apresenta a solução proposta para o problema descrito no Capítulo 3. A Seção 4.1 descreve a metodologia adotada para a validação de GTIN. Em seguida, a Seção 4.2 apresenta o *pipeline* para encontrar o GTIN de uma nota fiscal, buscando na base oficial. Por fim, a Seção 4.3 descreve o método de de-duplicação e agrupamento de produtos equivalentes na base oficial da GS1.

O problema apresentado se resume em desenvolver uma solução que garanta o preenchimento correto do código GTIN em notas fiscais. Como comentado anteriormente, o Céos tem acesso à base oficial do código GTIN da GS1 Brasil, portanto, ela pode ser usada como referência para compará-la às descrições dos produtos presentes na NF-e. Desta forma, buscam-se métodos capazes de identificar se textos diferentes referem-se ao mesmo produto. A solução proposta é organizada em três etapas complementares, de forma que cada uma resolva um dos problemas que compõe o projeto: validação, preenchimento e agrupamento. A Figura 1 apresenta o fluxograma geral do sistema.

Figura 1 – Fluxo da solução



### 4.1 VALIDAÇÃO

A primeira etapa consiste na validação do GTIN existente. Dada uma nota fiscal com o GTIN presente, busca-se esse código na base da GS1, formando um par composto pela descrição textual da NF-e e pela descrição oficial, o sistema deve então determinar se ambas as descrições referem-se ao mesmo item. Trata-se, portanto, de uma tarefa de classificação binária em que a classe positiva indica que o par descreve o mesmo produto e a classe negativa indica que são produtos distintos. Não existem soluções definitivas na literatura para este problema, portanto, para este trabalho, foram testados diferentes métodos.

A primeira abordagem investigada consiste no treinamento de um classificador para a tarefa, usando os dados de descrições dos produtos, com base no trabalho de (Arriaga et al., 2025). Nesta pesquisa, os autores testaram múltiplos modelos de aprendizado de máquina combinando informações textuais e imagens para treinar um modelo capaz de identificar se dois registros de produtos de e-commerce se referem ao mesmo produto. Levando em conta que nos dados de notas fiscais não há acesso a imagens dos produtos, adota-se a configuração com melhor desempenho sem dados visuais. Ela consiste no treinamento de um classificador XGBoost usando dados dos títulos dos produtos vetorizados por meio do Word2Vec, com vetores de tamanho 75. A informação de entrada do modelo é um vetor concatenado com 150 dimensões das duas descrições sendo avaliadas. O classificador XGBoost é também submetido a uma otimização de hiperparâmetros, na qual se testa o desempenho do modelo para diferentes combinações de parâmetros com o intuito de escolher as melhores.

A segunda abordagem é baseada em embeddings contextuais obtidos por meio do modelo BERT multilíngue (mBERT), conforme proposto no trabalho "Measuring Brazilian Portuguese Product Titles Similarity using Embeddings"(Silva Romualdo; Real; Medeiros Caseli) em um trabalho que avaliou o uso de diferentes embeddings para medir similaridade entre títulos de produtos em português brasileiro. A medida de similaridade entre os vetores de um par de descrições é calculada por meio da similaridade de cosseno, conforme definida na Equação 2. A classificação do par como *match* ou *non-match* é realizada com base em um limiar de similaridade.

A terceira abordagem consiste no uso de funções de similaridade léxica para a tarefa de classificação do par de descrições. Cada função é avaliada individualmente como classificador, usando um limiar de similaridade como no método anterior. As funções avaliadas são as seguintes:

- Jaccard
- Jaccard( $q=3$ )
- Levenshtein
- Damerau-Levenshtein
- Coeficiente de Dice
- Coeficiente de Dice( $q=3$ )

O q-grama com  $q = 3$  é sugerido por (Bezu et al., 2015) para lidar com descrições que contêm abreviações e erros ortográficos, ambos muito comuns em NF-e portanto, é adotado neste experimento para os métodos baseados em tokens. A avaliação dos métodos é realizada com base nas métricas de classificação apresentadas

na Seção 2.4.1: acurácia, precisão, revocação,  $F_1$  e  $F_{0,5}$ . A métrica adotada para a seleção do melhor método e ajuste dos limiares de classificação nas abordagens 2 e 3 é o  $F_{0,5}$ . Essa escolha se dá pela natureza do problema: um falso positivo, ou seja, classificar como *match* um par de produtos distintos, resulta no agrupamento indevido de itens diferentes, afetando as comparações de preços. Um falso negativo, por outro lado, representa apenas na perda de um registro para o banco de preços, sem introduzir dados incorretos na análise.

## 4.2 PREENCHIMENTO DO GTIN

A segunda etapa da solução proposta tem por objetivo atribuir o GTIN correto às notas fiscais que não possuem esse campo preenchido ou cujo código foi invalidado na etapa anterior. Para isso, é preciso comparar a descrição da NF-e com toda a base da GS1, que possui milhões de registros, tornando o processo inviável se feito de forma exaustiva. A solução proposta combina um sistema de recuperação de informação usando o Apache Solr para a seleção eficiente de candidatos com uma etapa de escolha final, por meio do melhor método definido na Etapa 1.

O sistema de recuperação de informação do Solr permite buscas eficientes em grandes bases de dados textuais, fazendo uso do algoritmo BM25 para busca de textos similares. Desta forma, é possível indexar a base da GS1 no Solr para realizar buscas com a descrição presente na NF-e, trazendo registros com algum nível de similaridade e reduzindo assim o número de comparações necessárias com o método da Etapa 1.

Além disso, antes de realizar a busca textual, aplica-se uma etapa de blocking para restringir o conjunto de candidatos da base da GS1 usando informações presentes tanto na NF-e quanto nesta base. Primeiramente, utiliza-se o código NCM (Nomenclatura Comum do Mercosul), que classifica os produtos em categorias hierárquicas com base em suas características gerais. É possível também extrair as unidades de medida e seus valores da descrição do produto. Descrições com unidades de medida ou valores diferentes já indicam que não se referem ao mesmo produto, sendo possível descartá-las sem necessidade de comparação.

## 4.3 DE-DUPLICAÇÃO E AGRUPAMENTO

A terceira etapa da solução proposta aborda o problema de agrupamento de GTINs equivalentes na base oficial da GS1. O objetivo é identificar, entre os milhões de produtos cadastrados, aqueles que representam essencialmente o mesmo item, desconsiderando variações de marca ou especificações que não sejam relevantes para a análise de sobrepreço em licitações. Essa etapa configura um problema de de-duplicação e resolução de entidades, como apresentado na Seção 2.5.

A abordagem adotada consiste nos seguintes elementos: uma estratégia de blocking para reduzir o espaço de comparação, o MinHash com LSH para geração eficiente de pares candidatos e o melhor método da Etapa 1 para validação dos pares, seguido da formação de grupos por componentes conexas.

De forma análoga ao pipeline de busca descrito na Seção 4.2, a etapa de blocking restringe as comparações a pares de produtos pertencentes à mesma categoria NCM e com unidades de medida compatíveis. Mesmo após o blocking, o número de pares candidatos pode ser muito elevado para uma avaliação exaustiva. Para contornar essa limitação, utiliza-se o MinHash combinado com o Locality-Sensitive Hashing (LSH), conforme apresentado na Seção 2.5.1. As assinaturas do MinHash são calculadas sobre os trigramas das descrições, permitindo maior flexibilidade para abreviações e erros ortográficos (Bezu et al., 2015). O LSH é então aplicado para identificar, de forma aproximada e eficiente, os pares de produtos com alta similaridade de Jaccard, gerando um conjunto de pares candidatos sem a necessidade de comparação exaustiva.

Os pares candidatos gerados pelo LSH são submetidos ao melhor método identificado na Etapa 1, para uma classificação definitiva sobre se são referentes ao mesmo produto. Com isso, obtêm-se diversos pares de descrições da base da GS1 considerados equivalentes. Para formar grupos efetivamente, é necessário construir um grafo não direcionado, no qual cada nó representa um registro e as arestas conectam os registros classificados como *match*. Os grupos de produtos equivalentes são então formados pela identificação das componentes conexas desse grafo.

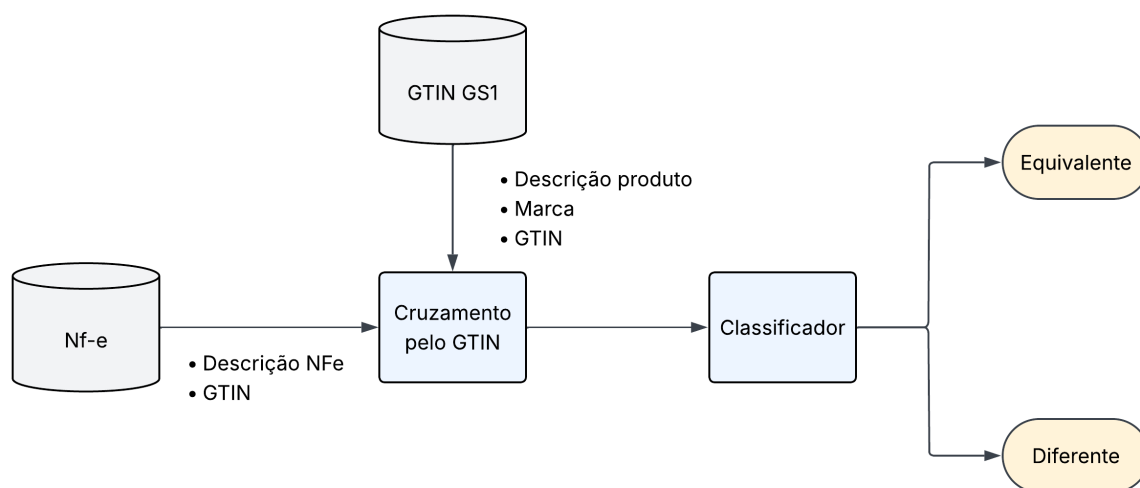
## 5 DESENVOLVIMENTO E ANÁLISE DOS RESULTADOS OBTIDOS

Este capítulo descreve o desenvolvimento e os resultados obtidos neste trabalho. A seção 5.1 apresenta os métodos testados para a tarefa de validação de GTIN e os resultados obtidos. A seção 4.2 apresenta a solução proposta para a etapa de preenchimento com base no Apache Solr e a avaliação de seu desempenho. A seção 4.3 descreve a abordagem utilizada para estender a classificação dos pares em um método de agrupamento usando dos algoritmos LSH com MinHash e Union Find. Por fim, a seção 5.4 apresenta os resultados gerais do trabalho relacionando com os requisitos definidos.

### 5.1 VALIDAÇÃO DE GTIN

A primeira etapa do projeto e a fundação dele se baseia em definir um método capaz de identificar se um par de descrições de produtos se refere a mesma entidade ou não. A figura 2 apresenta o fluxo dessa etapa. Dada uma nota fiscal com descrição textual do produto e código GTIN preenchido, é feita uma consulta a base da GS1, retornando a descrição do produto na GS1 e a marca. Em seguida, o par descrições é alimentado a um método que irá classificá-los como equivalentes ou não.

Figura 2 – Fluxo da validação

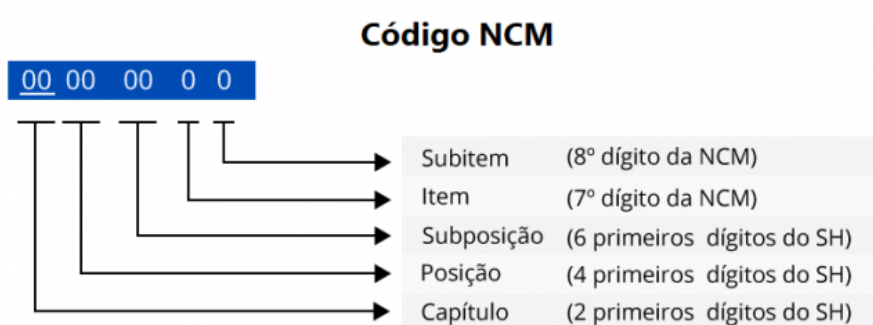


Com o objetivo de avaliar diferentes abordagens e desenvolver uma solução para o domínio específico de notas fiscais eletrônicas de compras públicas, o primeiro passo é construir um conjunto de dados anotado com os dados das bases de Nfe e da GS1.

### 5.1.1 Construção do conjunto anotado

Ambas as bases utilizadas neste trabalho estão disponíveis como tabelas em um banco de dados PostgreSQL do projeto Céos. Desta forma, é feito um cruzamento entre as tabelas usando como chave o código GTIN, presente em ambas. Para obter-se uma amostra representativa dos dados, faz-se uso do NCM presente nas Nf-e para agrupar os registros usando ambos os códigos e obter-se os GTIN que mais aparecem na base de Nf-e por grupo de produto. O NCM (Nomenclatura Comum do Mercosul) é um código hierárquico de oito dígitos usado para classificação de mercadorias, o qual pode ser quebrado em sub-valores que representam classes mais abrangentes, a figura 3 apresenta sua estrutura. Como esse dado também é preenchido por pessoas e está sujeito a erros, opta-se por usar o capítulo do NCM (2 primeiros dígitos) ao agrupar os registros para mitigar efeito de preenchimentos incorretos.

Figura 3 – Estrutura do NCM



Fonte: fazcomex (2026)

Neste ponto, tendo obtido uma lista de GTINs mais comuns nas notas fiscais dos mais diversos tipos de produtos, é feito o cruzamento entre as tabelas obtendo-se os pares de descrições. Para montar efetivamente o conjunto de dados, seguiu-se uma lógica similar a proposta no artigo "Measuring Brazilian Portuguese Product Titles Similarity using Embeddings" (Silva Romualdo; Real; Medeiros Caseli, 2021), que consiste em trazer três tipos de pares para cada produto: equivalentes (EQ), parecidos mas, não equivalentes (PNEQ) e combinações aleatórias, não equivalentes (NEQ). A proporção e quantidade absoluta de cada classe é apresentada na tabela 1.

Desta forma, foi feita uma etapa manual onde era escolhido um produto da lista gerada, selecionava 20 pares em que ambas as descrições se referiam a mesma entidade e buscava-se na base de Nf-e 10 registros de produtos parecidos para formar os pares PNEQ. Após completar essas etapas para todos os GTIN selecionados, realizou-se um cruzamento interno para gerar os pares aleatórios. O conjunto final é formado por 26 produtos, totalizando 1040 registros, com 520 pares equivalentes e

Tabela 1 – Tipos de pares por produto

| Classe | Proporção | Quantidade |
|--------|-----------|------------|
| EQ     | 50%       | 20         |
| PNEQ   | 25%       | 10         |
| NEQ    | 25%       | 10         |

520 não equivalentes. Os campos presentes neste arquivo são evidenciados na tabela 2.

Tabela 2 – Campos do conjunto anotado

| Campo         | Descrição                                                      |
|---------------|----------------------------------------------------------------|
| gtin_produto  | Código GTIN do par avaliado                                    |
| descricao_gs1 | Descrição textual oficial do produto na base da GS1            |
| descricao_nfe | Descrição textual do produto presente na NF-e                  |
| marca         | Nome da marca do produto cadastrada na GS1                     |
| label         | Rótulo booleano que indica se o par representa o mesmo produto |

### 5.1.2 Implementação dos Métodos

A primeira abordagem testada consiste no treinamento de um classificador XGBoost que recebe como parâmetros os pares de descrições transformados em vetores por meio do modelo Word2Vec(W2V) (Arriaga et al.). Para gerar os vetores baixou-se o modelo do W2V NILC skip-gram de 50 dimensões através da biblioteca de Python Huggingface. Este modelo foi treinado para gerar vetores especificamente para o português brasileiro, assim sendo uma boa escolha para geração dos *embeddings* das descrições de produtos. Quanto a escolha do tamanho dos vetores, no trabalho de referência o autor tem bons resultados com vetores pequenos (25,50 e 75 dimensões), portanto como no NILC W2V skip-gram desses três só tem a opção 50, foi escolhido usar essa versão.

O ajuste do modelo começa com o pré-processamento das descrições, que consiste na remoção de todos os caracteres não alfanuméricos e na conversão do texto para letras minúsculas. Em seguida, cada descrição é convertida em um vetor de tamanho 50 usando do modelo do W2V. O vetor que representa um par de descrições é formado pela concatenação dos dois vetores individuais, resultando em uma representação de 100 dimensões, que é então utilizada como entrada do classificador XGBoost.

Em seguida, o treinamento do modelo é feito junto da otimização dos hiperparâmetros, através de uma busca em grade combinada com validação cruzada de cinco

partições (*5-fold cross-validation*). Nesta etapa o conjunto de dados é dividido em cinco partes iguais e o modelo é treinado em cima de 4 partições e testado na 5ª, o processo se repete cinco vezes de forma que todas as partições tenham sido usadas para teste. Ao combinar com a otimização dos hiperparâmetros, para cada configuração os hiperparâmetros é aplicado o *5-fold cross-validation*, buscando encontrar a configuração com melhor desempenho. Os parâmetros e os valores investigados na busca estão apresentados na Tabela 3, e o critério utilizado para selecionar a melhor combinação foi o  $F_{0,5}$  médio obtido na validação cruzada.

Tabela 3 – Grade de hiperparâmetros para o XGBoost

| Hiperparâmetro                                | Valores avaliados    |
|-----------------------------------------------|----------------------|
| Número de estimadores ( <i>n_estimators</i> ) | 100, 500, 1000, 1200 |
| Profundidade máxima ( <i>max_depth</i> )      | 3, 5, 20, 50         |
| Regularização gama ( <i>gamma</i> )           | 0, 5, 10             |
| Taxa de aprendizado ( <i>learning_rate</i> )  | 0,01; 0,1; 0,3; 0,5  |

A segunda abordagem utiliza da similaridade cosseno sobre *embeddings* gerados através do modelo BERT multilíngue (mBERT) para cálculo de similaridade semântica, conforme proposto por (Silva Romualdo; Real; Medeiros Caseli). Primeiramente é feito o pré-processamento textual, igual ao aplicado à abordagem anterior, seguido pela geração do vetor através do modelo mBERT. A similaridade entre os *embeddings* de um par de descrições é calculada por meio da similaridade de cosseno, conforme definida na Equação 2, a qual foi traduzida em uma função em Python para aplicar sobre os vetores gerados. É feita uma pequena alteração a qual considera valores de similaridade negativos como 0, afinal um valor negativo indicaria significado oposto mas, na prática indica que não é o mesmo produto. A classificação do par como *match* ou *non-match* é determinada com base em um limiar de similaridade. Esse limiar é ajustado percorrendo valores de 0 a 1 em passos de 0.05, e o valor que maximiza o  $F_{0,5}$  é selecionado. Para o ajuste do limiar, o conjunto de dados é separado em dois, treinamento(dados usados para ajustar limiar de classificação) e teste(aplica-se o método com limiar ajustado para avaliar desempenho) na proporção 80/20.

A terceira abordagem consiste na aplicação de funções de similaridade léxica para a classificação dos pares de descrições. O processamento textual é similar ao usado no outros métodos, exceto que para as similaridades baseadas em tokens(Jaccard e Dice) são retirado espaços em branco e adiciona-se \_ no começo e fim das descrições (Lukasik et al., 2021). Cada função é avaliada individualmente como classificador, e o limiar de decisão é ajustado da mesma forma descrita para o mBERT. As funções implementadas em código Python são:

- Jaccard por palavras

- Jaccard com trigramas
- Distância de Levenshtein normalizada
- Distância de Damerau-Levenshtein normalizada
- Coeficiente de Dice por palavras
- Coeficiente de Dice com trigramas

A formulação matemática de cada uma dessas funções está apresentada no Capítulo 5.

### 5.1.3 Estratégias de Pré-processamento

Além de implementar cada método conforme proposto na literatura, foram investigadas estratégias adicionais de pré-processamento com o objetivo de melhorar o desempenho dos modelos. A metodologia adotada para essa investigação é a seleção sequencial progressiva (forward selection). Funciona da seguinte forma: partindo da configuração de pré-processamento base de cada método, uma nova etapa é adicionada e o desempenho é avaliado, caso haja ganho no  $F_{0,5}$ , ela é mantida, caso contrário, descartada. Esse processo é repetido até passar por todas as etapas sendo testadas. Os processos avaliados são apresentados a seguir.

1) Enriquecimento condicional com a marca (EM): considerando que na base da GS1 existe o campo com o nome da marca do produto, propõe-se que quando a marca está presente na descrição da NF-e mas, não consta na descrição da GS1, ela é concatenada ao final da descrição oficial, para aumentar a similaridade nesses casos. O campo da marca também é limpo, aplicando remoção de caracteres não alfanuméricos e conversão para minúsculo. Essa verificação é realizada após as demais etapas de pré-processamento da descrição, exceto aquelas que removem espaços em branco, para não prejudicar a identificação da marca.

2) Remoção de stopwords (RS): consiste na remoção de termos do português brasileiro que possuem alta frequência no idioma, mas com baixo valor semântico (ex: "e", "de", "o", "que"). Realizado através da biblioteca Python Spacy, com o modelo `pt_core_news_sm`.

3) Normalização de unidades (NU): nesta fase, termos que representam a mesma unidade de medida, mas que aparecem com grafias diferentes, são padronizados por meio de um dicionário de unidades. Além disso, as grandezas são padronizadas de forma que unidades de massa são convertidas para quilograma(kg), volume para litros(l) e comprimento para metros(m). Por exemplo, 100 g se torna 0.1kg e 20cm vira 0.2m. Abaixo o dicionário de termos de unidades de medida construído é apresentado.

```

MAPA_UNIDADES = {
    'kg':      ['kg', 'kgs', 'quilo', 'quilos', 'kilograma', 'kilogramas',
               'kilo', 'kilos', 'quilograma', 'kl', 'kls', 'kig', 'k'],
    'g':       ['g', 'gr', 'grs', 'gramas', 'grama', 'grm', 'grms',
               'gr.', 'g.', 'gram', 'gramm', 'gs'],
    'mg':      ['mg', 'mgs', 'miligramas', 'miligrama', 'mlg', 'mlgs', 'mg.'],
    'ton':     ['t', 'ton', 'tons', 'tonelada', 'toneladas', 'tonel',
               'tonel.', 't.', 'tn', 'tns'],
    'lt':      ['l', 'lt', 'lts', 'ltr', 'ltrs', 'litros', 'litro',
               'litrs', 'li', 'lts.', 'lt.', 'l.', 'litr', 'lit'],
    'ml':      ['ml', 'mls', 'mlitros', 'mili', 'mililitros',
               'mlitro', 'ml.', 'mls.', 'mlt'],
    'un':      ['un', 'und', 'unid', 'unidades', 'unidade', 'unit',
               'ud', 'unid.', 'u', 'un.', 'uni', 'unt'],
    'fardo':   ['fd', 'fardo', 'fardos', 'frd', 'fard'],
    'm':       ['m', 'mt', 'mts', 'metros', 'metro', 'm.', 'mt.', 'mtr', 'mtrs'],
    'cm':      ['cm', 'cms', 'centimetro', 'centimetros', 'cm.'],
    'mm':      ['mm', 'mms', 'milimetro', 'milimetros', 'mm.'],
    'pol':     ['pol', 'polg', 'polegadas', 'inch', 'in', '"', 'poleg'],
    'pcs':     ['pc', 'pca', 'peca', 'pecas', 'pcs'],
    'cpr':     ['cpr', 'comp', 'cp', 'cps', 'cmpr', 'cmp', 'comprimido',
               'compr', 'rev c', 'rev comp', 'cp rev'],
}

```

4) Validação por veto de unidades (VU): após padronizar as unidades de medida no passo anterior, elas são extraídas e usadas para comparar as descrições (Nfe e GS1) com o intuito de identificar se possuem unidades distintas. Independentemente de como foi classificado pelo método, um par é classificado automaticamente como non-match caso as unidades de medida extraídas das duas descrições pertençam à mesma grandeza física, porém com valores incompatíveis. Para exemplificar, considere Nfe com descrição 'Cimento 25kg CPII 32 Todas as Obras Votoran' e descrição do GTIN da GS1 'Cimento Todas As Obras CP IV-32 Cinza 50kg Votoran'. Elas compartilham quase todas as palavras que as compõe, sendo bastante similares. Com o veto, mesmo se o modelo classificasse o par como sendo o mesmo produto, ao comparar as unidades extraídas, 25kg e 50kg, seria definido como non-match.

5) Remoção de unidades (RU): as unidades de medida e seus respectivos valores numéricos são identificados e removidos das descrições antes de serem processados pelos modelos, sendo usados apenas para a etapa de veto.

### 5.1.4 Resultados e Análise: Validação GTIN

A Tabela 4 apresenta os resultados finais de todos os métodos avaliados no conjunto de teste, após a aplicação das estratégias de pré-processamento selecionadas via forward selection (tabela 7), utilizando os limiares de classificação otimizados (tabela 5) e para o XGBoost os hiperparâmetros selecionados (tabela 6).

Tabela 4 – Resultados dos métodos no conjunto de teste

| Método              | $F_{0,5}$ | Precisão | Revocação | Acurácia |
|---------------------|-----------|----------|-----------|----------|
| Jaccard trigrama    | 0,9159    | 0,9444   | 0,8173    | 0,8846   |
| Dice trigrama       | 0,9152    | 0,9535   | 0,7885    | 0,8750   |
| Dice palavras       | 0,9102    | 0,9740   | 0,7212    | 0,8510   |
| mBERT cosseno       | 0,8726    | 0,9250   | 0,7115    | 0,8269   |
| Jaccard palavras    | 0,8709    | 0,8854   | 0,8173    | 0,8558   |
| Levenshtein         | 0,8405    | 0,8667   | 0,7500    | 0,8173   |
| Damerau-Levenshtein | 0,8333    | 0,8571   | 0,7500    | 0,8125   |
| XGBoost + Word2Vec  | 0,7979    | 0,7826   | 0,8654    | 0,8125   |

Tabela 5 – Limiares de classificação ajustados

| Método              | Limiar ( $\tau$ ) |
|---------------------|-------------------|
| Jaccard trigrama    | 0,20              |
| Jaccard palavras    | 0,20              |
| Dice trigrama       | 0,35              |
| Dice palavras       | 0,40              |
| Levenshtein         | 0,35              |
| Damerau-Levenshtein | 0,35              |
| mBERT cosseno       | 0,70              |

Tabela 6 – Hiperparâmetros selecionados do XGBoost

| Hiperparâmetro | Grade de busca         | Valor selecionado |
|----------------|------------------------|-------------------|
| n_estimators   | {100, 500, 1000, 1200} | 1200              |
| max_depth      | {3, 5, 20, 50}         | 3                 |
| learning_rate  | {0,01, 0,1, 0,3, 0,5}  | 0,3               |
| gamma          | {0, 5, 10}             | 0                 |

O método com melhor desempenho global foi a similaridade de Jaccard com trigramas, atingindo  $F_{0,5} = 0.9159$ , precisão de 0.9444, revocação de 0.8173 e acurácia de 0.8846. Para alcançar esse resultado, o método utilizou um limiar de classificação de  $\tau = 0.20$  (Tabela 5) e o pipeline de pré-processamento que incluiu a remoção de stopwords, enriquecimento condicional da marca, normalização de unidades e o veto de unidades, conforme a Tabela 7.

Tabela 7 – Etapas de pré-processamento selecionadas via Forward Selection

| Método              | RS | EM | NU | RU | VU |
|---------------------|----|----|----|----|----|
| Jaccard trígama     | ✓  | ✓  | *  |    | ✓  |
| Dice trígama        | ✓  | ✓  | ✓  |    | ✓  |
| Jaccard palavras    | ✓  | ✓  | ✓  |    | ✓  |
| Dice palavras       |    | ✓  | *  |    | ✓  |
| Levenshtein         | ✓  | ✓  | *  | ✓  | ✓  |
| Damerau-Levenshtein | ✓  | ✓  | *  | ✓  | ✓  |
| mBERT               |    |    | *  |    | ✓  |
| XGBoost + Word2Vec  |    |    | *  | ✓  | ✓  |

\*NU é aplicada para todos os métodos para realizar as etapas de extração e veto.

A alta precisão desse método é especialmente relevante para o contexto do trabalho, pois a classificação incorreta de um par de produtos distintos como equivalentes introduziria dados equivocados no banco de preços, comprometendo diretamente a validade das análises de sobrepreço.

A superioridade dos métodos léxicos baseados em q-gramas de caracteres em relação às abordagens baseadas em palavras inteiras e de edição de caracteres é consistente com as características do domínio: as descrições de NF-e frequentemente contêm abreviações, erros tipográficos e termos truncados, que levam a palavras que não coincidem lexicalmente com as da base GS1, mas compartilham sequências de caracteres comuns quando decompostas em trigamas. O modelo mBERT cosseno, embora capture representações semânticas contextuais mais ricas, obteve  $F_{0,5} = 0.8726$ , desempenho inferior ao dos três melhores métodos léxicos. Esse resultado pode ser explicado pelo fato de que as descrições de produtos em NF-e são textos curtos, com presença comum de erros ortográficos e abreviações o que limita a capacidade do modelo de explorar o contexto semântico para o qual foi treinado. Além disso, o vocabulário altamente específico do domínio de produtos (com marcas, códigos técnicos e abreviações) pode não estar adequadamente representado no corpus em que o mBERT foi treinado.

O método XGBoost combinado com Word2Vec apresentou  $F_{0,5} = 0.7979$ , configurando o desempenho mais baixo entre todas as abordagens avaliadas. A principal limitação dessa abordagem resulta do tamanho pequeno do conjunto de dados utilizado para treinar o modelo. Além de enfrentar os mesmos desafios que o método do mBERT sofre em comparação a abordagens léxicas.

## 5.2 PREENCHIMENTO DO GTIN

Tendo definido uma metodologia para comparar pares de descrições e identificar se referem-se ao mesmo produto, o próximo passo é dado uma nota fiscal na qual o

GTIN não está preenchido ou foi invalidado pelo método da seção 5.1, buscar na base da GS1 o código referente a esse produto.

Se para todo registro de Nf-e for necessário comparar um a um a base inteira da GS1, se torna computacionalmente caro e demorado, tendo complexidade quadrática. Por conta disso, se faz necessário adotar estratégias para otimizar esse processo. Propõe-se então o uso do Apache Solr, um sistema de busca e indexação de documentos textuais de código aberto, amplamente utilizado em aplicações de recuperação de informação em larga escala. Ele permite armazenar dados textuais e criar índices eficientes para buscas de similaridade, assim pode ser utilizado para diminuir o espaço de comparação necessário para a busca.

### 5.2.1 Construção do Core e Definição do Schema

No Apache Solr, o core é uma instância de um banco de dados de busca que contém configurações próprias e armazena os dados seguindo essas diretrizes. Cada core possui um schema, que é a estrutura responsável por definir as regras que o regem. Ele funciona como um mapeamento que dita quais campos existem, quais são seus tipos de dados e de que forma o Solr deve tratá-los durante a indexação e a pesquisa.

A Tabela 8 apresenta o schema configurado para este projeto, detalhando as colunas existentes no banco, seus tipos de dados e quais foram indexadas para poderem ser usadas na busca. 'Tipo' indicado o formato de dado e 'Indexado' configura se a coluna pode ser utilizada como parâmetro de busca. Os campos do tipo string servem para busca exata, enquanto o tipo `text_general`, permite busca por similaridade, a qual usa do método BM25 baseado em frequência de termos.

Sobre os campos definidos para a base de busca, 'Desc\_clean' é a descrição do produto da GS1 processado usando a pipeline definida na seção 5.1 para o método escolhido, exceto a remoção de espaços em branco. Já 'unidades\_norm' é uma string das unidades e seus valores extraídos das descrições. Se houver mais de uma unidade identificada, elas serão concatenadas, ordenando primeiramente pela unidade de forma alfabética e depois pelo valor em ordem crescente. Desta forma, se diferentes descrições tiverem as mesmas unidades, formará o mesmo valor para `unidades_norm`, independente de como aparecem na descrição. No exemplo fictício 'Cabo flexível 2.5mm 100m rolo 5kg' a string resultante seria '5kg-0.0025m-100m'. Temos também os campos do código NCM e seus sub níveis, além do nome da marca e do código GTIN do produto.

Sobre os tipos de dados utilizados, suas definições são as seguintes:

- **`text_general`**: Este tipo é projetado para buscas por aproximação ou similaridade. Quando um dado entra em um campo `text_general`, o Solr aplica um processo

Tabela 8 – Definição do Schema do Apache Solr para o Core de Produtos

| Nome do Campo | Tipo         | Indexado |
|---------------|--------------|----------|
| marca         | string       | False    |
| GTIN          | string       | False    |
| Desc_clean    | text_general | True     |
| NCM           | string       | True     |
| NCM_2         | string       | True     |
| NCM_4         | string       | True     |
| NCM_6         | string       | True     |
| unidades_norm | string       | True     |

de análise textual que divide a frase em palavras individuais (tokens), remove caracteres especiais e padroniza o texto em letras minúsculas. Essa preparação permite que o algoritmo padrão BM25 calcule a relevância dos termos.

- **string:** Ao contrário do `text_general`, o tipo `string` não sofre nenhuma modificação ou divisão textual. O Solr armazena e lê o valor exatamente como ele foi enviado, de forma literal e em bloco único. Esse comportamento é ideal para buscas exatas e filtros rígidos.

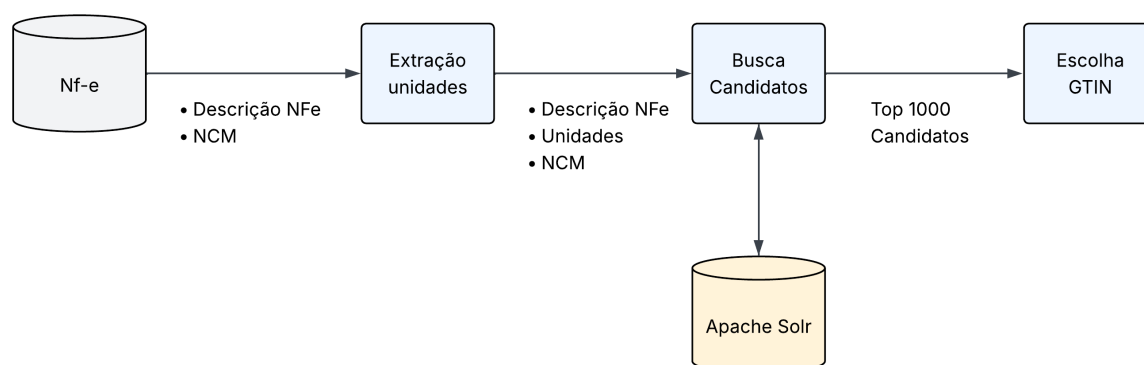
### 5.2.2 Pipeline de busca

O *pipeline* de busca é composto por duas etapas: a busca de candidatos pelo Solr e a seleção final do GTIN mais adequado por meio do método definido na seção 5.1. A estratégia de busca faz uso de busca exata sobre o NCM posição (quatro primeiros dígitos) e unidades extraídas quando presentes na Nf-e sendo processada. Mesmo se não tiver nenhuma dessas informações, é realizada busca por similaridade usando como entrada a descrição da nota fiscal processada pela pipeline de tratamento textual. O sistema retorna então, os mil candidatos com maior pontuação de similaridade da base. Para esta primeira etapa, escolheu-se usar do NCM posição como forma de mitigar erros humanos no preenchimento desse campo na Nf-e. A figura 4 demonstra o funcionamento da solução de preenchimento.

Já a decisão de trazer mil candidatos se dá pois essa quantidade é facilmente processável e traz alta probabilidade de que o produto correto estará presente entre os candidatos retornados pelo Solr, mesmo que não no topo da lista de candidatos, pois esses são os itens mais similares.

Sobre o conjunto de candidatos resultante, aplica-se a Similaridade de Jaccard com trigramas para calcular a pontuação de similaridade entre a descrição da NF-e e cada candidato. O candidato com maior pontuação é o GTIN selecionado. Contudo, para evitar alocações incorretas em casos nos quais nenhum candidato é suficientemente similar, o GTIN só é atribuído à NF-e caso a pontuação do melhor candidato supere um limiar mínimo.

Figura 4 – Fluxo do preenchimento



Para validar o sistema proposto, foram realizados experimentos sobre um conjunto de 1.104 notas fiscais com diferentes NCM posição as quais o campo do GTIN não estava preenchido. Testou-se nove valores distintos de limiar de aceitação que variam de 0.20 a 0.75. O experimento consistiu em aplicar o pipeline de busca para todas essas Nf-e fazer uma anotação manual do resultado, classificando se o GTIN atribuído é de fato referente ao mesmo produto da nota fiscal, comparando as descrições de ambos.

### 5.2.3 Resultados e Análise: Preenchimento do GTIN

A Tabela 9 apresenta os resultados do pipeline de busca para cada valor de limiar testado, indicando o número de registros corretamente alocados (acertos), o número de registros não alocados (abaixo do limiar), o número de erros (GTIN incorreto atribuído), a proporção de registros acima do limiar(% alocados), bem como a taxa de acerto entre os registros efetivamente alocados.

Tabela 9 – Resultados do pipeline de busca de GTIN para diferentes limiares

| Limiar | Acertos | Não Alocados | Erros | % Alocados | % Acerto |
|--------|---------|--------------|-------|------------|----------|
| 0.20   | 431     | 343          | 330   | 68,93%     | 56,64%   |
| 0.30   | 424     | 533          | 147   | 51,72%     | 74,26%   |
| 0.40   | 415     | 584          | 105   | 47,10%     | 79,81%   |
| 0.50   | 334     | 717          | 53    | 35,05%     | 86,31%   |
| 0.55   | 294     | 785          | 25    | 28,89%     | 92,16%   |
| 0.60   | 262     | 831          | 11    | 24,73%     | 95,97%   |
| 0.65   | 223     | 874          | 7     | 20,83%     | 96,96%   |
| 0.70   | 194     | 907          | 3     | 17,84%     | 98,48%   |
| 0.75   | 172     | 930          | 2     | 15,76%     | 98,85%   |

Os resultados revelam um claro trade-off entre a precisão dos preenchimentos

realizadas e a cobertura do sistema, à medida que o limiar aumenta, a taxa de acerto entre os registros alocados cresce consideravelmente, porém ao custo de um maior número de registros que permanecem sem GTIN atribuído. Com o limiar mais baixo testado de 0.2, 761 dos 1.104 registros recebem um GTIN, mas apenas 56,64% dessas alocações estão corretas, um volume inaceitável para fins de análise de sobrepreço. No extremo oposto, com um limiar de 0.75, a taxa de acerto sobe para 98.85% mas, somente 15.76% dos registros tem o campo de GTIN preenchido. Considerando a importância da precisão deste método para seu caso de uso, o limiar a ser utilizado na aplicação real deve ser algo entre 0.6 a 0.7.

### 5.3 AGRUPAMENTO DE PRODUTOS EQUIVALENTES

A terceira e última etapa do projeto tem por objetivo identificar e agrupar, na base da GS1, produtos que representam essencialmente o mesmo item. Isso tem como objetivo formar grupos maiores para análise de sobrepreço, o que melhora a qualidade da análise.

Nas etapas anteriores foi validado uma pipeline que usa da similaridade de Jaccard sobre trigrams de caracteres para identificar pares de descrições equivalentes. Com base nisso, pode-se extrapolar essa abordagem para identificar par a par de todas as combinações de registros na base da GS1, quais seriam referentes ao mesmo item. Para realizar esta tarefa, sem enfrentar complexidade quadrática de comparar todas as opções, faz-se uso do algoritmo Locality-Sensitive Hashing (LSH) junto do MinHash, apresentados na seção 2.5.1, permitindo solucionar esse problema em ordem aproximadamente linear  $O(N)$ .

#### 5.3.1 Construção de conjunto de dados para testes

Para o desenvolvimento e a avaliação do método de agrupamento, foi construído um conjunto de dados de referência composto por 77 grupos, cada qual formado por até 10 registros, totalizando em 636 itens da base GS1 com diferentes descrições textuais do mesmo produto por grupo. Esses grupos foram definidos manualmente, representando o agrupamento ideal que o método deve ser capaz de reproduzir.

#### 5.3.2 Desenvolvimento da solução de agrupamento

A solução de agrupamento foi desenvolvida em Python, fazendo uso da biblioteca datasketch, que oferece implementações dos algoritmos MinHash e LSH. A primeira etapa da solução é o pré-processamento das descrições. Aplicou-se o mesmo conjunto de etapas definido nas seções anteriores, com ajustes para o novo problema. A marca não é mais adicionada na descrição e sim removida pois, busca-se agrupar produtos iguais de diferentes marcas. Além disso, as unidades extraídas nesta etapa

são usadas como estratégia de *blocking*, isso é aplica-se o algoritmo apenas para registros com unidades iguais, diminuindo o espaço de comparação e impedindo agrupamento de registros com unidades divergentes. A extração é feita de forma análoga a secção 4.2, formando a string `unidades_norm`. Abaixo apresenta-se os processos de tratamento textual realizados.

- Remoção de caracteres não alfanuméricos
- Conversão para letras minúsculas
- Remoção de stopwords
- Normalização de unidades
- Remoção de marca
- Extração de unidades
- Remoção de espaços em branco

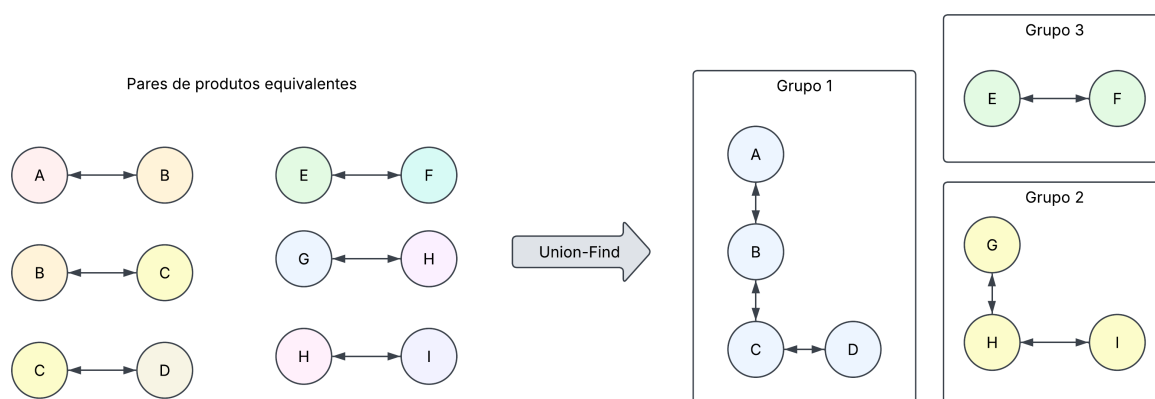
Tendo processado as descrições, faz-se uso da biblioteca `datasketch` para criar assinaturas do MinHash e indexá-los em uma estrutura do LSH a qual faz uso das assinaturas para identificar candidatos a equivalentes, com base em uma pontuação aproximada da similaridade de Jaccard e um limiar pré-definido.

Os pares candidatos gerados pelo LSH são validados pelo cálculo da similaridade de Jaccard com trigramas exato, agora só tendo que ser aplicado a esses candidatos, em vez de todos os pares possíveis da base. O limiar de aceitação é o mesmo usado para o LSH.

Por fim, a formação dos grupos de produtos equivalentes é modelada como um problema de identificação de componentes conexas em um grafo, onde cada produto é um nó e cada par candidato validado representa uma aresta. Para agrupar esses nós de forma eficiente, implementou-se a estrutura de dados Union-Find.

Essa estrutura opera fundamentalmente através de duas operações: a Find, que determina a qual grupo um elemento pertence, e a Union, que funde dois grupos distintos. A propriedade de fechamento transitivo é uma consequência natural desse modelo: se o produto *A* é equivalente a *B*, e *B* é equivalente a *C*, a operação Union conecta suas estruturas, reunindo automaticamente *A*, *B* e *C* no mesmo *cluster*. Desta forma, os pares validados que inicialmente estão isolados, são agrupados com outros pares que compartilham algum registro em comum. Esta dinâmica é apresentada na figura 5.

Figura 5 – Formação dos grupos



### 5.3.3 Resultados e Análise: Agrupamento de GTIN

A Tabela 10 apresenta os resultados da pipeline de agrupamento no conjunto de dados de teste. As métricas analisadas são homogeneidade, que mede se todos os *clusters* contêm apenas membros de uma única classe de dados, completude que quantifica se todos os membros de uma determinada classe foram atribuídos ao mesmo *cluster* e a *V-measure*, que representa a média entre as duas métricas anteriores.

Tabela 10 – Resultados do agrupamento para diferentes limiares de similaridade

| Limiar | Homogeneidade | Compleitude | <i>V-measure</i> |
|--------|---------------|-------------|------------------|
| 0,20   | 0,929         | 0,919       | 0,924            |
| 0,30   | 0,965         | 0,872       | 0,916            |
| 0,42   | 0,988         | 0,804       | 0,886            |
| 0,50   | 0,992         | 0,751       | 0,855            |
| 0,55   | 0,993         | 0,732       | 0,843            |
| 0,60   | 0,995         | 0,715       | 0,832            |
| 0,65   | 0,996         | 0,705       | 0,825            |
| 0,71   | 0,997         | 0,692       | 0,817            |

Os resultados revelam um padrão semelhante ao observado na etapa de preenchimento do GTIN, o aumento do limiar de similaridade eleva a pureza dos agrupamentos formados, refletida na homogeneidade, às custas de uma menor completude, resultado de mais itens não sendo agrupados.

Por fim, observa-se que os valores de homogeneidade se mantêm elevados mesmo para os limiares mais baixos, divergindo dos resultados obtidos nas seções anteriores. Isso ocorre pois, o conjunto de dados de teste não apresenta uma complexidade suficientemente alta comparado com a base real e seus milhões de itens.

Portanto, essas métricas podem estar infladas e para a aplicação real sugere-se o uso de um limiar análogo ao da seção 4.2 de 0.6 a 0.7.

#### 5.4 RESULTADOS GERAIS

Retomando a lista dos requisitos deste trabalho apresentados na seção 3.1, primeiramente a solução de validação dos GTINs proposta alcançou um bom desempenho, com score  $F_{0.5}$  de 0,916 e alta precisão de 0,944, cumprindo os requisitos RF01 e RNF02.

A etapa seguinte, de atribuição do código GTIN (requisito RF02), também foi construída e alcançou uma alta taxa de acerto, próxima de 96%. No entanto, para o limiar de similaridade mínimo proposto, o método definiu o GTIN para apenas 24,73% das notas fiscais avaliadas, demonstrando uma limitação significativa no escopo em que é capaz de atuar resolvendo o problema com alta precisão.

Quanto ao requisito RF03, que definia a necessidade de agrupar produtos equivalentes na base da GS1, o objetivo foi alcançado estendendo o método de validação de pares para a formação de grupos.

Por fim, sobre a necessidade de os métodos serem escaláveis (RNF01) para lidarem com grandes volumes de dados, o Apache Solr utilizado no sistema de busca é uma ferramenta desenvolvida para buscas textuais eficientes em grandes bases. Já a solução de agrupamento com LSH e MinHash baseia-se em algoritmos projetados especificamente para mitigar o custo computacional da comparação quadrática em grandes volumes de dados. Por último, o método de validação baseia-se na similaridade de Jaccard, que possui cálculo simples e baixo custo computacional.

## 6 CONCLUSÃO

O presente trabalho teve como objetivo desenvolver um método para a validação e alocação automática do código GTIN em notas fiscais eletrônicas para viabilizar análises de sobrepreço em compras públicas. A solução foi estruturada em três etapas complementares: validação dos GTINs existentes, preenchimento dos campos ausentes ou incorretos e agrupamento de produtos equivalentes na base oficial da GS1.

Para a etapa de validação, foram investigadas e comparadas diferentes abordagens, incluindo funções de similaridade léxica, similaridade semântica com base em embeddings do modelo mBERT e um classificador XGBoost treinado com vetores do Word2Vec. Além disso, foram testados estratégias de pré-processamento textual, selecionadas via *forward selection* para todas as abordagens. O método de similaridade de Jaccard com trigramas de caracteres obteve o melhor desempenho para a tarefa de classificação dos pares de descrições, alcançando  $F_{0.5}$  score de 0,916. A superioridade das abordagens léxicas baseadas em trigramas evidencia a adequação dessa abordagem ao domínio das descrições de NF-e, marcado por textos curtos, com frequente presença de abreviações e erros ortográficos.

Para a etapa de preenchimento, foi implementado um pipeline de busca combinando o Apache Solr para recuperação de candidatos com o BM25, e o método de Jaccard com trigramas para a seleção final do GTIN entre os candidatos. Os experimentos realizados sobre 1104 notas fiscais demonstraram um claro *trade-off* entre cobertura e precisão, controlado pelo limiar de aceitação. Com um limiar de 0.60, o sistema atinge taxa de acerto de 95.97% sobre os registros alocados, abrangendo 24.73% das notas fiscais avaliadas.

Por fim, na etapa de agrupamento, foi desenvolvido uma solução utilizando os métodos MinHash e LSH para identificação eficiente de pares candidatos e o algoritmo de componentes conexas via Union-Find para a formação dos grupos finais com base nos pares equivalentes. Essa combinação permite aplicar a similaridade de Jaccard de trigramas de forma eficiente em grandes bases de dados e agrupar os itens com base nos pares identificados com equivalentes.

### 6.1 SUGESTÕES PARA TRABALHOS FUTUROS

O principal ponto a ser melhorado na metodologia proposta é aumentar a cobertura da solução sem prejudicar sua precisão. Para isso, na etapa de validação pode-se investigar a combinação de diferentes abordagens, por exemplo trazendo um fator de similaridade semântica junto da léxica. Além disso, sugere-se aumentar o conjunto de dados para permitir o treinamento de modelos de aprendizado de máquina para o problema, sendo uma boa forma de combinar as abordagens. Para o sistema de busca,

seria interessante testar outras configurações de busca no Apache Solr, testando novos algoritmos de similaridade além do BM25 e até fazendo mudanças nos parâmetros dele para otimizar a recuperação de candidatos.

## REFERÊNCIAS

APACHE SOLR. **Apache Solr Reference Guide :: Apache Solr Reference Guide — solr.apache.org**. [S.l.: s.n.].

<https://solr.apache.org/guide/solr/latest/index.html>. [Accessed 07-06-2026].

ARRIAGA, B.; GÓMEZ, A.; PALACIOS, A.; ALIAGA, W. A Peruvian Machine Learning Model for E-Commerce Product Matching in South America, Spain and Portugal. In:

ARTUS, Vladimir. **Clustering metrics: evaluate the complex, make it simple**.

[S.l.: s.n.], 2024. [Accessed 25-05-2026]. Disponível em:

<https://medium.com/@vladimir-artus/%D1%81lustering-metrics-evaluate-the-complex-make-it-simple-6ae70c0f164b>.

BEZU, Ronald van; BORST, Sjoerd; RIJKSE, Rick; VERHAGEN, Jim; VANDIC, Damir; FRASINCAR, Flavius. Multi-component Similarity Method for Web Product Duplicate Detection. In:

BRASIL. **Lei nº 14.133, de 1º de abril de 2021**. [Accessed 26-05-2026]. 2021.

Disponível em:

[https://www.planalto.gov.br/ccivil\\_03/\\_ato2019-2022/2021/lei/l14133.htm](https://www.planalto.gov.br/ccivil_03/_ato2019-2022/2021/lei/l14133.htm).

BRENDOERFER, Michael. **MinHash: Jaccard Similarity, LSH, and Near-Duplicate Detection - Interactive | Michael Brenndoerfer — mbrenndoerfer.com**. [S.l.: s.n.], 2026.

<https://mbrenndoerfer.com/writing/minhash-algorithm-jaccard-similarity-lsh-deduplication#the-minhash-algorithm>. [Accessed 26-05-2026].

DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In:

GOV.BR. **Ano de 2025 é marcado pela inovação no mercado de compras públicas brasileiro — gov.br**. [S.l.: s.n.], 2026.

<https://www.gov.br/gestao/pt-br/assuntos/noticias/2026/janeiro/ano-de-2025-e-marcado-pela-inovacao-no-mercado-de-compras-publicas-brasileiro>. [Accessed 14-06-2026].

GS1 BRASIL. **GTIN - Identificação de Produtos**. [Accessed 26-05-2026]. Disponível em: <https://www.gs1br.org/gtin-identificacao-de-produtos>.

HARISHA, Lathashree. **Dice Coefficient! What is it?** [S.l.: s.n.], 2023.

<https://lathashreeh.medium.com/dice-coefficient-what-is-it-ff090ec97bda>. [Accessed 15-06-2026].

HARSOOR, Sharan. **Embeddings: A Deep Dive from Basics to Advanced Concepts — sharanharsoor**. [S.l.: s.n.], 2024.

<https://medium.com/@sharanharsoor/embeddings-a-deep-dive-from-basics-to-advanced-concepts-f092765476fc>. [Accessed 23-05-2026].

HOSSAIN, Elias; RANA, Rajib; HIGGINS, Niall; SOAR, Jeffrey; BARUA, Prabal Datta; PISANI, Anthony R.; TURNER, Kathryn. Natural Language Processing in Electronic Health Records in relation to healthcare decision-making: A systematic review. **Computers in Biology and Medicine**, 2023. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0010482523001142>.

KULKARNI, Vyankatesh. **Classification Metrics**. [S.l.: s.n.], 2023. <https://medium.com/@vyankatesh.kulkarni20/classification-metrics-63c635852995>. [Accessed 24-05-2026].

LUKASIK, Szymon; MICHALOWSKI, Andrzej; KOWALSKI, Piotr A.; GANDOMI, Amir H. Text-Based Product Matching with Incomplete and Inconsistent Items Descriptions. In:

MAMER F.A.H. ALNUAIMI, Tasnim H.K. ALBALDAWI. An overview of machine learning classification techniques. In:

MIKOLOV, Tomas; SUTSKEVER, Ilya; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. Distributed Representations of Words and Phrases and Their Compositionality. In:

NCM o que é e onde consultar — fazcomex.com.br. [S.l.: s.n.]. <https://www.fazcomex.com.br/ncm/o-que-e-onde-consultar/>. [Accessed 06-06-2026].

NOBLE, Joshua. **What is Jaccard Similarity? | IBM — ibm.com**. [S.l.: s.n.], 2026. <https://www.ibm.com/think/topics/jaccard-similarity>. [Accessed 15-06-2026].

OLIVIER BINETTE, Rebecca C. Steorts. (Almost) all of entity resolution. In:

SAYED, Ebad. **Mastering ML Metrics: Definitions, Mathematics, and Implementation Part 1**. [S.l.: s.n.], 2024. <https://medium.com/@sayedebad.777/mastering-ml-metrics-definitions-mathematics-and-implementation-9a9bd9a46df8>. [Accessed 26-05-2026].

SECRETARIA DE ESTADO DA FAZENDA. **Nota Fiscal Eletrônica**. [Accessed 26-05-2026]. 2024. Disponível em: <https://www.nfe.fazenda.gov.br/portal/principal.aspx>.

SHARMA, Manish. **Apache Solr Fundamentals**. [S.l.: s.n.], 2023. <https://mansha99.medium.com/apache-solr-fundamentals-43e360962cc8>. [Accessed 21-06-2026].

SILVA, Mariana O.; COSTA, Lucas L.; BEZERRA, Guilherme; GOMIDE, Larissa D.; HOTT, Henrique R.; OLIVEIRA, Gabriel P.; BRANDAO, Michele A.; LACERDA, Anisio;

PAPPA, Gisele. *Análise de Sobrepreço em Itens de Licitações Públicas*, 2023. [Accessed 21-06-2026].

SILVA ROMUALDO, Alan da; REAL, Livy; MEDEIROS CASELI, Helena de. Measuring Brazilian Portuguese Product Titles Similarity using Embeddings. In:

STRYKER, Cole; HOLDSWORTH, Jim. **O que é PLN (processamento de linguagem natural)? | IBM — ibm.com**. [S.l.: s.n.], 2026.

<https://www.ibm.com/br-pt/think/topics/natural-language-processing>. [Accessed 23-05-2026].

TIWARI, Rahul. **Unlocking Text Similarity: Comprehensive Methods and Real-World Use Cases**. [S.l.: s.n.], 2025.

<https://medium.com/@rahultiwari065/ultimate-guide-to-text-similarity-from-basics-to-advanced-applications-1492f82c0269>. [Accessed 23-05-2026].

TOM KRANTZ, Alexandra Jonker. **O que é similaridade de cosseno? | IBM — ibm.com**. [S.l.: s.n.], 2026.

<https://www.ibm.com/br-pt/think/topics/cosine-similarity>. [Accessed 15-06-2026].