Revista Brasileira de Computação Aplicada, Abril, 2024

DOI: 10.5335/rbca.v16i1.14955
Vol. 16, Nº 1, pp. 50–63
Homepage: seer.upf.br/index.php/rbca/index

ARTIGO ORIGINAL

H-sim: uma função de similaridade híbrida para identificação de correspondência de produtos

H-sim: a hybrid similarity function for product matching

Higor Moreira [1], and Edimar Manica [1]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul - Campus Ibirubá

*higormoreira999@hotmail.com; edimar.manica@ibiruba.ifrs.edu.br

Recebido: 10/06/2023. **Revisado:** 12/04/2024. **Aceito:** 30/04/2024.

Resumo

Uma empresa ao realizar compras de produtos de seus fornecedores, precisa importar as notas fiscais eletrônicas destes produtos para sua base de dados relacional para administrar o estoque de produtos, tributos e revenda. Esta não é uma tarefa trivial, pois as descrições dos produtos das notas fiscais e da base de dados apresentam variações. Este trabalho propõe a função de similaridade H-sim. Esta função combina métricas semânticas com métodos baseados em token ou distância de edição, visando identificar produtos correspondentes em diferentes bases de dados. Foram realizados experimentos utilizando dados reais de produtos, onde a função H-sim obteve 87,7% de F1. Os resultados indicam que a abordagem proposta tem potencial para melhorar a integração de dados de produtos em ambientes empresariais, ao mitigar as variações nas descrições dos produtos.

Palavras-Chave: correspondência de produtos; funções de similaridade; nota fiscal eletrônica; similaridade semântica.

Abstract

When a company purchases products from its suppliers, it needs to import the electronic invoices of these products into its relational database to manage the inventory, taxes, and resale. This is not a trivial task, as the product descriptions on the invoices and in the database vary. This paper proposes the H-sim similarity function. This function combines semantic metrics with token-based or edit distance methods, aiming to identify corresponding products in different databases. Experiments were conducted using real product data, where the H-sim function achieved an F1 score of 87.7%. The results indicate that the proposed approach has the potential to improve the integration of product data in business environments by mitigating variations in product descriptions.

Keywords: electronic invoices; product matching; semantic similarity; similarity functions.

1 Introdução

Uma empresa ao realizar compras de produtos de seus fornecedores precisa importar esses produtos para sua base de dados relacional para administrar o estoque de produtos, tributos e revenda. Os fornecedores emitem e enviam as notas fiscais eletrônicas (NFe) dos produtos em formato XML (*eXtensible Markup Language*) para a empresa. A NFe substitui a emissão do documento fiscal em papel, possui validade jurídica garantida pela assinatura digital do remetente, simplifica as obrigações acessórias dos contribuintes e permite o acompanhamento em tempo real das operações comerciais pelo Fisco (Ministério da Fazenda, 2022).

A Fig. 1 apresenta um exemplo real das etapas necessárias para realizar a importação das notas fiscais eletrônicas para a base de dados relacional de uma empresa com processo totalmente manual. Esse fluxograma é composto

Figura 1: Etapas da importação manual dos produtos das notas fiscais eletrônicas para a base de dados relacional
Fonte: Autor.

Tabela 1: Exemplos de produtos correspondentes com variações de descrição

Base de Dados	Nota Fiscal Eletrônica
CERV BRAH CHOP 473ML	BRAHMA CHOPP LT 473ML SH C/12 NPAL
MARGARINA BECEL MANTEIGA C/S 500G	BECEL CR VEG SABOR MANTEIGA CS 12X500G
TV LED SMART 43 PHILCO PTV43E60SN	TV PTV43E60SN LED
ARNO ASPIRADOR DE PO CYXL 220V	ASP PO ARNO CYXL 220V
NESTLE IOG.CHANDELLE CHANTILLY CHOCO 200G	CHANDELLE CHANTILLY CHOCOLATE 200G

por 3 etapas: Emissão, Verificação Manual e Importação.

Na primeira etapa, os fornecedores emitem e enviam notas fiscais eletrônicas (NFe) em formato XML para a empresa que adquiriu os produtos. A empresa recebe as NFe e inicia o processo de verificação para importar os produtos para sua base dados relacional. Na segunda etapa, o processo de verificação é feito de maneira manual por funcionários da empresa, pois não há nenhum software para auxiliar na importação. Essa etapa é muito dependente do conhecimento dos funcionários sobre os dados das NFe e da base de dados relacional, pois as especificações dos produtos apresentam variações na representação do conteúdo para um mesmo produto do mundo real. Esse processo é dispendioso, demorado e pode gerar cadastros incorretos. Por fim, na terceira etapa, após todo o processo de verificação manual, os produtos da NFe são cadastrados na base de dados relacional da empresa.

Automatizar esse processo não é uma tarefa trivial, porque existem variações entre a especificação dos produtos no XML da NFe e a especificação dos produtos da base de dados relacional da empresa. Também há variação no código utilizado para identificar e categorizar os produtos.

A Tabela 1 apresenta exemplos reais de produtos correspondentes com variação na descrição entre a base de dados e as notas fiscais eletrônicas. Entre as variações, destacam-se abreviações, acrônimos, sinônimos, omissão de informações e ordem da representação do conteúdo.

O objetivo deste trabalho é propor a função de similaridade *H-sim*[1] que combina funções de similaridade baseadas em *token* ou distância de edição com funções de similaridade semânticas para identificar produtos correspondentes de diferentes bases de dados. Enquanto a função da similaridade baseada em *token* analisa o conjunto das palavras e quantifica sua semelhança, a função de similaridade semântica analisa o contexto de cada palavra. A função baseada em *token* pode contribuir com o problema da ordem de representação das descrições, e a função semântica com as variações de sinônimos e abreviações das descrições.

Para os experimentos foram utilizados dados reais de uma empresa da região do Alto Jacuí comparando funções de similidade baseadas em distância de edição, *token* (frequência de um termo) e semânticas. O melhor resultado obtido foi com a combinação de uma função de similaridade semântica e a função de similaridade baseada em *token* Overlap resultando em 88% de precisão, 87,4% de revocação e 87,7% de F1.

O restante deste artigo está organizado como segue. A Seção 2 descreve os principais conceitos e técnicas necessárias para a compreensão deste trabalho. A Seção 3 discute os principais trabalhos relacionados com a temática deste artigo. A Seção 4 propõe e define a função *H-sim*. A Seção 5 detalha a avaliação experimental das funções de similaridade. Por fim, a Seção 6 apresenta as considerações finais e os trabalhos futuros.

2 Fundamentação Teórica

Nesta seção são descritos os principais conceitos e técnicas utilizadas durante o desenvolvimento deste trabalho. A Seção 2.1 define funções de similaridade e explica os tipos utilizados neste trabalho, enquanto a Seção 2.2 apresenta a técnica de processamento de linguagem natural Word2vec.

2.1 Funções de Similaridade

As funções de similaridade são usadas para medir a "distância" entre dois vetores, números ou pares. É uma medida de quão semelhantes são os dois objetos que estão sendo medidos. Os dois objetos são considerados seme-

[1]H: Hybrid - sim: Similarity - Similaridade Híbrida

lhantes se a distância entre eles for pequena e vice-versa (Cohen et al., 2003). Neste trabalho, são utilizados três tipos de funções de similaridade: baseadas em *token*, distância de edição e semânticas.

Funções de similaridade baseadas em *token* examinam as strings dos textos a serem comparados e definem um conjunto de *tokens*. Então contam o número desses *tokens* dentro de cada string e usam essa contagem para calcular uma medida de distância ou similaridade (Blurock, 2021; Cohen et al., 2003). São exemplos de funções de similaridade dessa categoria utilizadas neste trabalho: Cosine, Jaccard e Overlap.

Funções de similaridade baseadas em distância de edição quantificam quão diferente duas palavras são uma da outra, que é medido pela contagem do número mínimo de operações necessárias para transformar uma palavra na outra (Navarro, 2001; Cohen et al., 2003). São exemplos de funções de similaridade dessa categoria utilizadas neste trabalho: Jaro, JaroWinkler e Hamming.

Funções de similaridade semânticas são calculadas sobre um conjunto de documentos ou palavras, a distância entre as palavras é obtida a partir da semelhança de seus contextos ou na interpretação entre esses dois textos. Neste trabalho, é utilizada uma função de similaridade que obtém o contexto das palavras por meio da técnica de aprendizado de máquina Word2vec (Harispe et al., 2015).

2.2 Word2vec

Word2vec é uma técnica para processamento de linguagem natural publicada em 2013 por uma equipe de pesquisadores do Google liderada por Tomas Mikolov em dois artigos (Mikolov, Chen, Corrado e Dean, 2013; Mikolov, Sutskever, Chen, Corrado e Dean, 2013). O algoritmo Word2vec usa um modelo de rede neural para aprender associações de palavras de um grande *corpus* de texto. Uma vez treinado, esse modelo pode detectar palavras sinônimas ou sugerir palavras adicionais para uma frase parcial. Como o nome indica, Word2vec representa cada palavra distinta com uma lista particular de números chamada de vetor. Os vetores são escolhidos cuidadosamente de forma que uma função matemática simples (a similaridade do cosseno entre os vetores) indique o nível de similaridade semântica entre as palavras representadas por esses vetores.

O Word2vec representa uma palavra com um vetor de tamanho fixo. Este vetor é capaz de capturar efetivamente semelhanças semânticas e sintáticas de uma palavra. Por exemplo, considerando que v(‘palavra’) representa o vetor de contexto de uma palavra, então v(‘rainha’) - v(‘mulher’) + v(‘homem’) = v(‘rei’) ou v(‘Brasil’) + v(‘Capital’) = v(‘Brasília’). O modelo Word2vec inclui duas arquiteturas para representar as palavras: *continuous bag-of-words* (CBOW) ou *continuous skip-gram* (*skip-gram*). Na arquitetura CBOW, o modelo prevê a palavra atual a partir de uma janela de palavras de contexto ao redor. A ordem das palavras de contexto não influencia a previsão (*bag-of-words assumption*). Na arquitetura *skip-gram*, o modelo usa a palavra atual para prever a janela circundante de palavras de contexto. A arquitetura *skip-gram* atribui um peso maior para as palavras de contexto próximas do que para as palavras de contexto mais distantes. De acordo com a nota dos autores, o CBOW é mais rápido enquanto o *skip-gram* faz um trabalho melhor para palavras pouco frequentes.

3 Trabalhos Relacionados

Esta seção discute os trabalhos relacionados ao contexto no qual este artigo está inserido. Esses trabalhos foram definidos a partir de uma pesquisa pelo termo *"product matching"* e outra pelo termo *"similarity function"* na DBLP[2] (*Digital Bibliography & Library Project* - Repositório Bibliográfico de Ciência da Computação) e no Google Acadêmico[3]. A partir dos resultados retornados, foram selecionados os quatro trabalhos mais recentes e relacionados com a proposta, que continham as seguintes palavras-chave como critério: correspondência de produtos (*product matching*) ou funções de similaridade (*similarity function*).

A Seção 3.1 apresenta um trabalho que utiliza BERT (*Bidirectional Encoder Representations for Transformers*) para identificar produtos correspondentes. A Seção 3.2 analisa outro trabalho relacionado com BERT, porém com foco na afinação do modelo. A Seção 3.3 discute o uso de Bi-LSTM (*Bidirectional Long Short-Term Memory Networks*) para identificar produtos correspondentes. A Seção 3.4 apresenta o uso de Word2vec para desambiguar o significado de palavras dependendo do contexto. Por fim, a Seção 3.5 apresenta uma análise comparativa entre os trabalhos relacionados e o trabalho proposto.

3.1 Tracz et al. (2020)

No trabalho de Tracz et al. (2020), é proposta a utilização do BERT com algumas modificações para fazer com que um sistema de deduplicação seja capaz de identificar produtos correspondentes e lidar com novas instâncias de produtos ainda não apresentadas para o modelo. Para essa tarefa, os autores dividiram os conjuntos de dados em um conjunto de Ofertas e um conjunto de Produtos (subdividido em conjuntos de produtos vistos e não vistos pelo modelo BERT). Os itens dos conjuntos são representados por um título, um conjunto de atributos e uma categoria. A representação de cada oferta e produto é concatenada, passada para minúsculo e por um codificador de par de bytes. Para cada exemplo de treinamento de oferta, é necessário fornecer um exemplo de duplicata e de não duplicata. A função de similaridade utilizada foi *Cosine Similarity*.

Para o treinamento, foram utilizadas bases de dados de produtos com as seguintes categorias: eletrônicos, beleza e cultura com respectivamente 800 mil, 400 mil e 200 mil produtos. Para a avaliação, os dados foram divididos em 80% para treino e 20% para teste.

3.2 Peeters et al. (2020)

O trabalho de Peeters et al. (2020) também propõem o uso de BERT para identificar produtos de diferentes lojas eletrônicas que se referem a mesma entidade do mundo

[2]Disponível em:<https://dblp.org>. Último Acesso em: 29/05/23.
[3]Disponível em:<https://scholar.google.com>. Último Acesso em: 29/05/23.

real. No entanto, esse trabalho é mais focado na importância do pré-treinamento do BERT, introduzindo uma etapa intermediária de treinamento antes do *fine-tuning* (refinamento) final do modelo para produtos.

Nos experimentos para treinamento, validação e testes foram utilizados dados disponibilizados pela *WDC Product Corpus for Large-Scale Product Matching* que oferece mais de 26 milhões de ofertas de produtos de mais de 79 mil lojas para treinamento. Para avaliar o modelo, há cerca de 2 mil pares de ofertas dessa base de dados verificados manualmente como duplicatas ou não. Para os experimentos, foram utilizados dois conjuntos de testes, um contendo apenas os dados da categoria Computadores e outro contendo 4 categorias: Computadores, câmeras, relógios e sapatos. Para testar a eficiência, os dados foram divididos em conjuntos de testes de tamanhos variados: pequeno, médio, grande e muito grande.

O conjunto de treinamento é formado por *Hard Positives* e *Hard Negatives*. Os *Hard Positives* são exemplos positivos de produtos correspondentes, mas com similaridade baixa. Metade desse conjunto é selecionado utilizando *Cosine Similarity* entre o título do produto e as 5 primeiras palavras da descrição da oferta e selecionando os pares de oferta com escore de similaridade mais baixos. Enquanto a outra metade é selecionada aleatoriamente. Por outro lado, os *Hard Negatives* são exemplos de produtos não correspondentes, mas com similaridade alta. Metade desse conjunto é selecionado utilizando o maior escore da função *Cosine Similarity* e a outra metade selecionada aleatoriamente.

3.3 Li et al. (2019)

Em Li et al. (2019), foi proposto um novo modelo de redes neurais para resolver o problema de deduplicação utilizando dois tipos de descrições (o título do produto e atributos). Essas informações foram coletadas de plataformas de comércio eletrônico utilizando técnicas de *web crawling*. Foram selecionados produtos de 20 categorias distintas de duas lojas populares na China, uma com 680 mil produtos e outra com 380 mil produtos.

O modelo proposto pelos autores, chamado de *Product Matching Model* (PMM) utiliza dois sub-módulos: *Product Title Matching Module* (TMM); e *Product Attributes Matching module* (AMM). O módulo TMM é designado para calcular a similaridade entre o título de dois produtos e o módulo AMM trabalha medindo a similaridade entre os atributos de dois produtos. Após obter esses dois dados de similaridade, eles são combinados para gerar o escore de similaridade final. Foram utilizadas as funções de similaridade *Cosine, Bilinear, Euclidean* e *Dot Product*.

O módulo TMM transforma dois títulos de produtos em dois grupos de vetores. A similaridade entre as palavras dos dois títulos é medida utilizando uma matriz com *Cosine Similarity*. Os padrões de similaridade são extraídos e a saída é concatenada gerando um escore de similaridade. O módulo AMM utiliza uma matriz interativa para agrupar os atributos dos produtos. Se um valor de um atributo *a* e um atributo *b* de diferentes produtos obtêm o escore igual a 1, ele é definido como correspondente, caso contrário é definido como não correspondente.

Tabela 2: Tabela comparativa entre os trabalhos relacionados e o trabalho proposto.

Características	Tracz et al. (2020)	Peeters, Bizer e Glavas (2020)	Li et al. (2019)	Orkphol e Yang (2019)	Trabalho Proposto
Função de Similaridade	Cosine	Cosine	Cosine, Bilinear Euclidean e Dot product	Cosine	Baseadas em edição, token e semânticas
Filtragem por categoria	Sim	Sim	Sim	Não especificado	Sim
Técnica de aprendizado de máquina	BERT	BERT	Bi-LSTM	Word2Vec	Word2Vec
Uso de base de dados livre	Não	Sim	Sim	Sim	Não
Disponibilização do código-fonte	Não	Sim	Sim	Sim	Sim

3.4 Orkphol e Yang (2019)

No trabalho de Orkphol e Yang (2019) é proposta a utilização de Word2Vec para desambiguar o significado de palavras dependendo do contexto utilizado. Para essa tarefa foram construídos vetores de sentença de contexto e vetores de definição de sentido. Para obter as palavras, os autores utilizaram a WordNet que contém substantivos, verbos, adjetivos e advérbios agrupados em conjuntos de sinônimos cognitivos interligados por meio de semântica conceitual e relações léxicas do idioma Inglês.

Para calcular os vetores de sentença foram utilizadas três configurações diferentes. A primeira configuração soma todos os vetores de palavras. A segunda configuração soma todos os vetores de palavras e divide pela quantidade de palavras. Por fim, a terceira configuração utiliza a mesma técnica que a segunda configuração, porém cada palavra é multiplicada pelo peso do cálculo de frequência inversa do documento, que atribui pesos baseado na frequência de cada palavra. Para os experimentos foi utilizada a base de dados de notícias disponibilizada pela Google.

3.5 Análise Comparativa

A Tabela 2 apresenta uma análise comparativa entre os trabalhos relacionados e o trabalho proposto. As seguintes características são analisadas: função de similaridade, filtragem por categoria, técnica de aprendizado de máquina, uso de base de dados livre e disponibilização do código-fonte.

Em todos os trabalhos relacionados apresentados, a função de similaridade *Cosine* é utilizada para comparar especificações de produtos ou sentenças. Apenas o trabalho de Li et al. (2019) e o trabalho proposto utilizam mais de uma função de similaridade. O diferencial do trabalho proposto é utilizar funções de similaridade semânticas, baseadas em distância de edição e baseadas em *token* (inclui a *Cosine*) e avaliar a diferença de eficácia entre essas diferentes funções de similaridade individuais e combinadas.

A filtragem por categoria ou tipo de produtos é utilizada em todos os trabalhos, exceto no trabalho de Orkphol e Yang (2019). O trabalho proposto também utilizou a técnica de filtragem por categoria para diminuir o número de comparações entre os produtos. Para filtrar os produtos por categoria foi utilizado o NCM (Nomenclatura Comum do Mercosul) que é usado para categorizar mercadorias nos países do Mercosul.

Todos os trabalhos utilizam algum tipo de técnica de aprendizado de máquina combinado com funções de si-

milaridade, sendo BERT para os trabalhos de Tracz et al. (2020) e Peeters et al. (2020). Bi-LSTM para o trabalho de Li et al. (2019). Por fim, Word2Vec para o trabalho de Orkphol e Yang (2019). O trabalho proposto utilizou as mesmas técnicas de construção de vetor de sentença do trabalho de Orkphol e Yang (2019) para criar vetores de contexto para as palavras das descrições dos produtos, a fim de calcular a similaridade semântica entre os produtos.

Entre os trabalhos que utilizaram bases de dados livres ou disponíveis gratuitamente na Internet se destacam os trabalhos de Peeters et al. (2020), Li et al. (2019) e Orkphol e Yang (2019). O trabalho de Tracz et al. (2020) utilizou a própria base de dados da empresa, por isso ela não é disponibilizada no trabalho. O trabalho proposto também utilizou uma base de dados privada, disponibilizada por uma empresa da região do Alto Jacuí, por esse motivo a mesma não foi disponibilizada.

Por fim, todos os trabalhos disponibilizaram o código-fonte das principais etapas desenvolvidas, exceto o trabalho de Tracz et al. (2020). O trabalho proposto também disponibilizou a função *H-sim* e as etapas necessárias para utilizá-la no Github[4].

4 H-sim

Esta seção define a função de similaridade *H-sim*, proposta neste trabalho, que objetiva identificar correspondência de produtos. Essa seção está organizada da seguinte forma. A Seção 4.1 descreve o contexto de aplicação dessa função. A Seção 4.2 define o cálculo da *H-sim*. A Seção 4.3 apresenta o cálculo da similaridade semântica. Por fim, a Seção 4.4 descreve a construção do modelo Word2vec.

4.1 Visão geral da aplicação da H-sim

A Fig. 2 apresenta a aplicação da função *H-sim* para semi-automatizar a importação de notas fiscais eletrônicas para a base de dados relacional da empresa que adquiriu os produtos. A importação ocorre de forma automática quando a correspondência *H-sim* dos produtos é acima de um determinado limiar, enquanto que os produtos com correspondência menor que determinado limiar passam por uma conferencia manual, onde o esforço manual é reduzido, uma vez que é apresentada uma lista com os produtos mais similares.

A entrada do fluxo é a nota fiscal eletrônica emitida pelo fornecedor e a base dados da empresa que adquiriu os produtos. Na primeira etapa, é realizado o cálculo da *H-sim* entre cada produto da NFe e os produtos da base de dados que possuem o mesmo NCM, onde é gerada uma lista com os pares de produtos e seu respectivo escore de similaridade calculado por meio da função *H-sim*.

Para cada produto da nota fiscal eletrônica, se os pares de produtos possuem escore de similaridade maior ou igual ao limiar, o produto com o maior índice de similaridade é importado automaticamente para a base de dados da empresa. Se nenhum dos pares de produtos possui escore de similaridade maior ou igual ao limiar, é criada uma lista de sugestões de importação com os pares de produtos ordenados pelo índice de similaridade em ordem decrescente. Essa lista é enviada para um funcionário especialista responsável pela importação. O funcionário deve selecionar manualmente o produto equivalente. Após essa seleção, esse produto é importado.

[4]Disponível em: https://github.com/999higor/word2vec_git

4.2 Definição do Cálculo da H-sim

Esta seção descreve o cálculo da função de similaridade *H-sim*, que tem como objetivo encontrar os produtos da base de dados relacional da empresa que são mais similares aos produtos da NFe.

Algorithm 1: Algoritmo para encontrar os produtos mais similares

```
   Input  : P, B
   Output : prod_list
1  for P_i ∈ P do
2      for B_i ∈ B do
3          if P_i[ncm] = B_i[ncm] then
4              prod_list[P_i] ←
                 (B_i, H-sim(P_i[desc], B_i[desc]))
5          end
6      end
7      prod_list[P_i] ← OrdenaLista(prod_list[P_i])
8  end
9  return prod_list
```

O Algoritmo 1 descreve o algoritmo para encontrar os produtos da base de dados que são mais similares aos produtos da NFe. A entrada é composta por produtos de uma nota fiscal eletrônica P, e pelos produtos da base de dados relacional da empresa B. Todos os produtos de P e B possuem dois atributos, sendo eles descrição (*desc*) e NCM (*ncm*). A saída é composta por uma lista *prod_list* contendo a descrição dos produtos, NCM e o escore de similaridade entre os produtos da NFe e da base de dados, ordenados pelo escore de similaridade por meio da função *OrdenaLista*. A Linha 1 percorre todos os produtos da nota fiscal e seleciona um produto P_i para comparar com a base de dados. A Linha 2 percorre os produtos da base de dados da empresa e seleciona um produto B_i para comparar com o produto selecionado da NFe. A Linha 3 é uma condicional que compara se os produtos P_i e B_i possuem o mesmo NCM. A Linha 4 calcula a similaridade entre os produtos selecionados com a função *H-sim* utilizando sua descrição e armazenando os valores em uma lista *prod_list*. Uma lista é criada para cada produto da nota fiscal. A linha 7 ordena os produtos da lista *prod_list* com a função *OrdenaLista* pelo escore de similaridade em ordem decrescente, ou seja, os produtos mais similares ficam no topo da lista. Por fim, a Linha 9 retorna a lista *prod_list* contendo as descrições, NCM e escore de similaridade entre os produtos de uma NFe e os produtos da base de dados relacional da empresa.

A função *H-sim* calcula a média harmônica entre o escore de similaridade de uma função de similaridade semântica e uma função de similaridade baseada em token

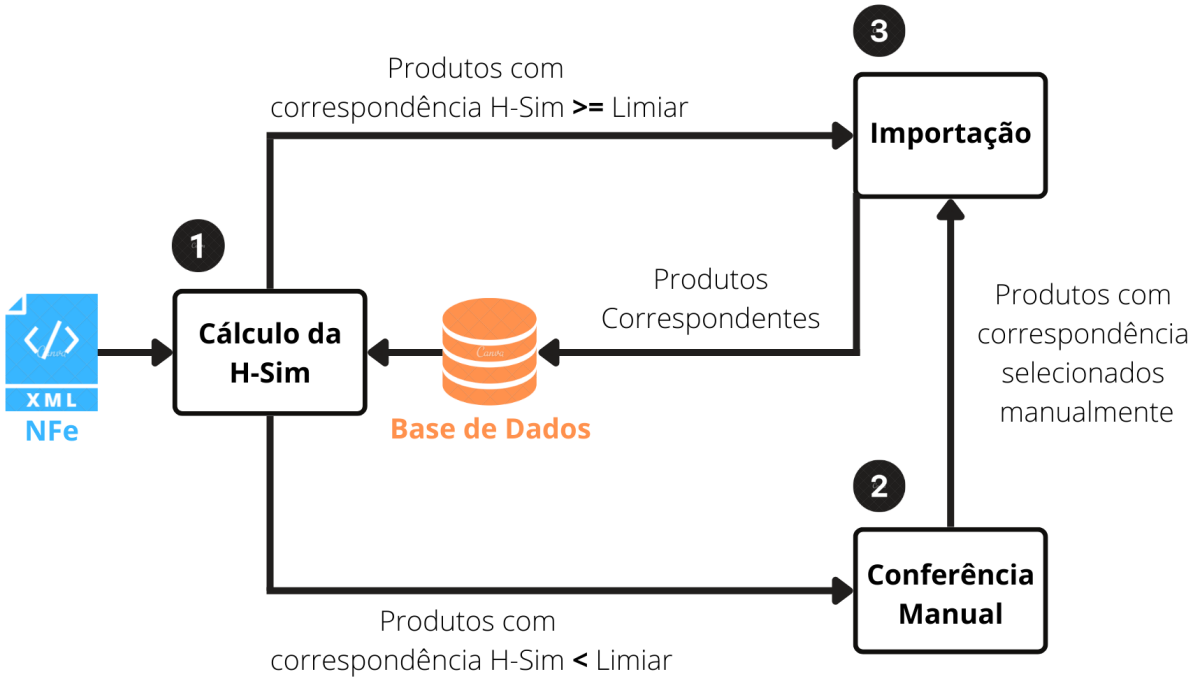

Figura 2: Etapas da importação semi-automática dos produtos da NFe para a base de dados relacional da empresa.
Fonte: Autor.

ou em distância de edição, conforme a Eq. (1), onde S é o escore de similaridade da função semântica e T é o escore de similaridade da função baseada em token ou distância de edição.

$$H\text{-}sim = 2 \cdot \frac{S \cdot T}{S + T} \tag{1}$$

4.3 Cálculo da Similaridade Semântica

A Fig. 3 apresenta as etapas para a obtenção da similaridade semântica, que são: pré-processamento, obtenção dos vetores das palavras, construção dos vetores de sentença e cálculo da similaridade. A entrada do fluxograma é um produto da nota fiscal eletrônica e um produto da base de dados relacional da empresa que possuem o mesmo NCM.

O objetivo da etapa de pré-processamento é limpar e padronizar as descrições dos produtos. Nessa etapa, são removidas pontuações, acentuações e *stopwords*. Essa etapa ainda acessa um modelo Word2vec para verificar a frequência mínima das palavras definida pelo modelo e remover as palavras abaixo do valor definido. O resultado é uma lista de palavras que restaram após a limpeza.

A etapa de obtenção dos vetores das palavras tem como objetivo obter um vetor de contexto para cada palavra da descrição do produto da NFe e para cada palavra da descrição do produto da base de dados relacional. Os vetores de contexto são obtidos acessando um modelo Word2vec.

A etapa de construção dos vetores de sentença visa obter um único vetor de contexto para cada produto a partir dos vetores de contexto das palavras que compõem sua descrição. Para construir o vetor de sentença foram testadas três técnicas: soma, média aritmética e média ponderada.

A primeira técnica é a soma (*Sum*) de todos os vetores das palavras de um produto gerando um vetor único. A soma do vetor é construída usando a Eq. (2), onde W é a lista de palavras de um determinado produto, $|W|$ é o tamanho da lista e $v(W[i])$ é o vetor de contexto da palavra $W[i]$.

$$Sum = \sum_{i=0}^{|W|} v(W[i]) \tag{2}$$

A segunda técnica é a média aritmética (*Average*) de todos os vetores das palavras de um produto gerando um vetor único. A média aritmética é calculada usando a Eq. (3), onde as variáveis têm a mesma nomenclatura utilizada na Eq. (2).

$$Average = \frac{1}{|W|} \sum_{i=0}^{|W|} v(W[i]) \tag{3}$$

A terceira técnica é a média ponderada (*Weight*) dos vetores das palavras de um produto gerando um vetor único. Para calcular a média ponderada é feito o cálculo da frequência inversa do documento (IDF) que valoriza palavras mais raras aumentando seu peso, e diminui o peso

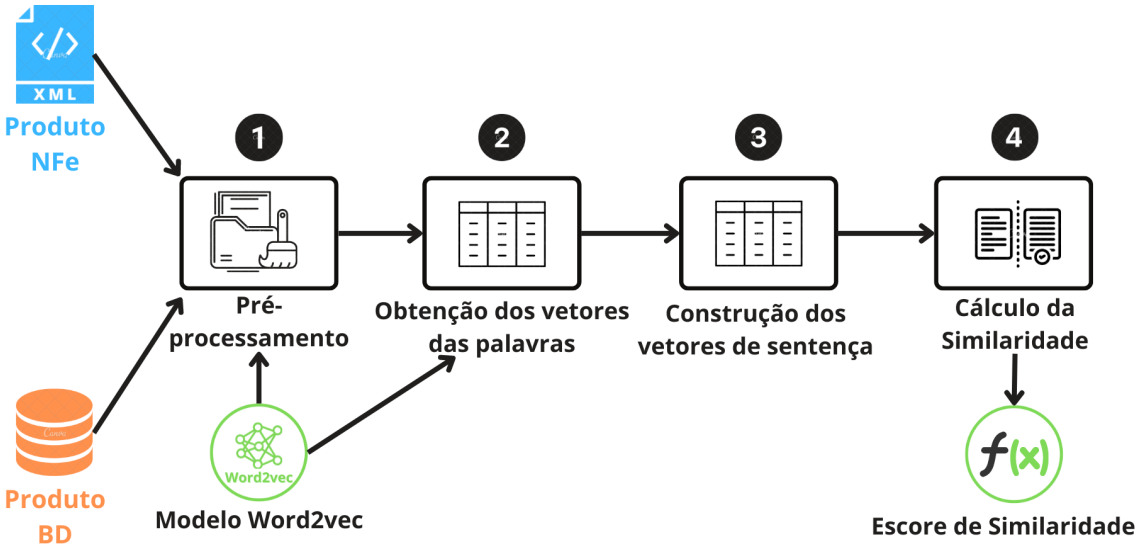

Figura 3: Etapas para obtenção da similaridade semântica.
Fonte: Autor.

de palavras que ocorrem em mais produtos. O cálculo do *IDF* é feito através da Eq. (4), onde t é uma palavra e D é o conjunto de todos os produtos das notas fiscais eletrônicas e da base de dados relacional, $|D|$ é o número total de produtos e $df(D,t)$ é o número de produtos que contém a palavra t.

$$idf(t,D) = \log_{10} \frac{|D|}{df(D,t)} \tag{4}$$

A terceira técnica é calculada conforme a Eq. (5), onde as variáveis também possuem a mesma nomenclatura utilizada na Eq. (2).

$$Weight = \frac{1}{|W|} \sum_{i=0}^{|W|} [v(W[i]) \cdot idf(W[i],D)] \tag{5}$$

Por fim, o objetivo da etapa de cálculo da similaridade é calcular a similaridade semântica entre um produto da NFe e um produto da base de dados relacional que possuem o mesmo NCM por meio de suas descrições. Esse cálculo é feito utilizando uma das três técnicas de construção dos vetores de sentença. A similaridade semântica entre os produtos é calculada a partir do cosseno de seus vetores de sentença. O cosseno de dois produtos é calculado pelo produto escalar de seus vetores dividido pelo produto de seus comprimentos, conforme a Eq. (6), onde A e B são os vetores de sentença dos produtos a e b, bem como A_i e B_i são os $i$ésimos componentes dos vetores A e B.

$$S = Cosine\ similarity = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}, \tag{6}$$

O resultado final é o escore de similaridade semântica calculado pela função de similaridade *Cosine*. Esse escore varia entre os valores 0 e 1. Quanto mais próximo de 1 for o resultado entre a comparação de duas descrições, mais similar ele é considerado. Quanto mais próximo de 0, menos similar ele é considerado.

Na Fig. 4, é apresentado um exemplo real do cálculo da similaridade semântica entre um produto da NFe e um produto da base de dados relacional da empresa.

Na primeira etapa, é realizada a limpeza das descrições dos produtos, onde as palavras são convertidas para minúsculo e são removidas acentuações, pontuações e as palavras com frequência mínima abaixo do valor definido pelo modelo Word2vec.

Na segunda etapa, são obtidos os vetores de contexto de cada palavra das descrições dos produtos. Para obter o vetor de contexto é utilizado o modelo Word2vec onde cada palavra recebe sua representação no plano do modelo.

Na terceira etapa, após obter o vetor de contexto das palavras, é calculo o vetor da sentença. Para calcular o vetor da sentença é utilizada umas das três técnicas apresentadas: *Sum*, *Average* e *Weight*.

Na quarta e última etapa, é utilizada a função de similaridade *cosine* para calcular a similaridade entre os vetores da sentença e obter o escore de similaridade semântica.

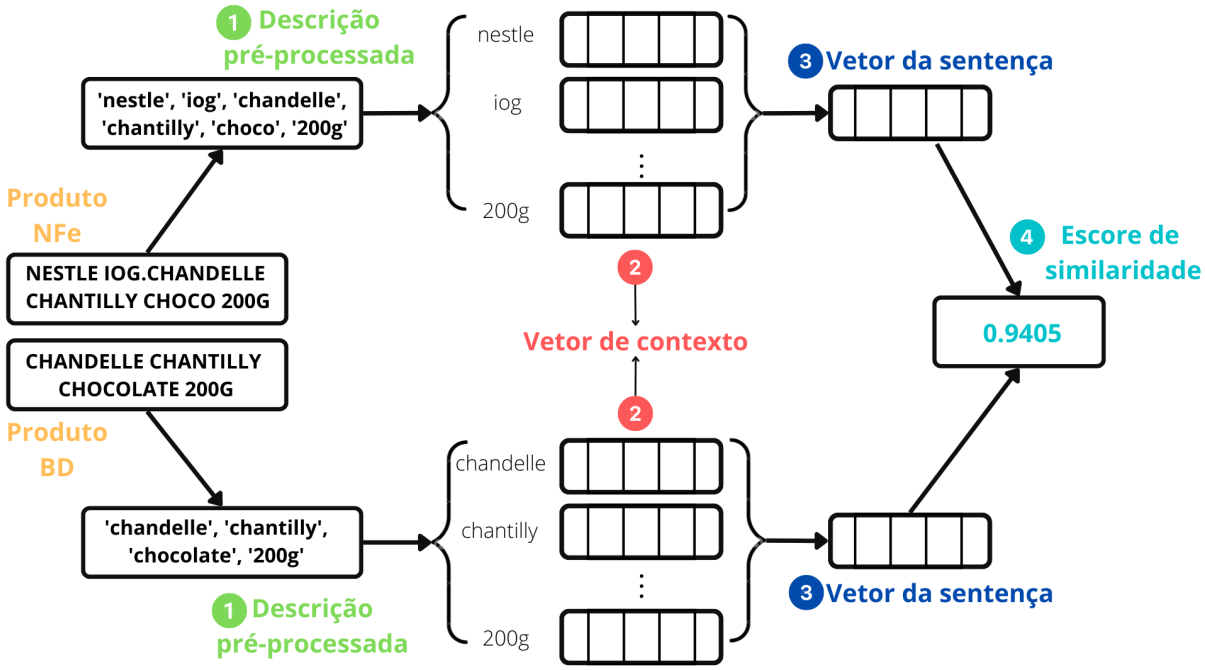

Figura 4: Exemplo das etapas de obtenção da similaridade semântica.
Fonte: Autor.

4.4 Construção do Modelo Word2vec

O objetivo da construção de um modelo Word2vec neste trabalho é obter o contexto das palavras que compõem a descrição de dois produtos a fim de calcular a sua similaridade semântica. A Fig. 5 apresenta as etapas da criação do modelo Word2vec utilizando uma amostra de notas fiscais eletrônicas e a base dados relacional da empresa. Essa criação é composta por 3 fases, sendo elas: agrupamento, pré-processamento e treinamento.

O objetivo da etapa de agrupamento é integrar os dados das notas fiscais eletrônicas e da base de dados relacional da empresa para a criação do modelo Word2Vec. Nessa etapa, são extraídos das NFe em formato XML as descrições e o NCM de cada produto e armazenados em um arquivo CSV. O mesmo processo é realizado para a base de dados relacional da empresa, são extraídos as descrições e o NCM de cada produto e agrupados em um arquivo CSV. No fim desse processo, o arquivo CSV das NFe e da base de dados relacional são integrados em um único arquivo.

A etapa de pré-processamento visa remover inconsistências nos dados. Nessa etapa, as descrições do produtos são convertidas para minúsculo, são removidas pontuações, acentuações e *stopwords* do idioma Português. Além disso, o NCM é padronizado como número inteiro, e os produtos que não possuem NCM ou que possuem caracteres alfabéticos recebem o valor 0 para evitar erros no código ao tentar comparar diferentes tipos de variáveis.

Na etapa de treinamento, é treinado um modelo Word2vec para identificar o contexto das palavras que compõem a descrição dos produtos nas diferentes bases de dados. Para essa etapa, é necessário definir parâmetros padrões para o treinamento do modelo Word2Vec. Os principais parâmetros que precisam ser definidos são: frequência miníma das palavras, taxa de aprendizado, algoritmo de treinamento, número de processadores, tamanho da janela de palavras, tamanho da dimensão dos vetores das palavras e épocas.

Após a etapa de treinamento, o modelo Word2vec é criado e salvo em um arquivo de formato MODEL. Esse arquivo pode ser utilizado para carregar o modelo Word2vec, sem a necessidade de treiná-lo toda vez que for necessário utilizá-lo.

5 Avaliação Experimental

Esta Seção descreve os experimentos realizados para avaliar a função de similaridade *H-sim*, bem como os experimentos que determinaram a configuração de seus parâmetros. A Seção 5.1 descreve as bases de dados utilizadas. A Seção 5.2 define as métricas adotadas. A Seção 5.3 descreve o ambiente de configuração utilizado. A Seção 5.4 apresenta a configuração utilizada para calcular a similaridade semântica da *H-sim*. Por fim, a Seção 5.5 apresenta a configuração da *H-sim*.

5.1 Base de Dados

Para o treinamento do modelo Word2Vec e para a avaliação da função de similaridade *H-sim* foi utilizada uma base de dados composta por: a base de dados relacional de uma empresa real contendo todos os seus produtos, uma amostra das notas fiscais eletrônicas dessa empresa e um gabarito para calcular a eficácia da proposta.

A base de dados relacional da empresa original foi fornecida apenas com as descrições e o NCM dos produtos em

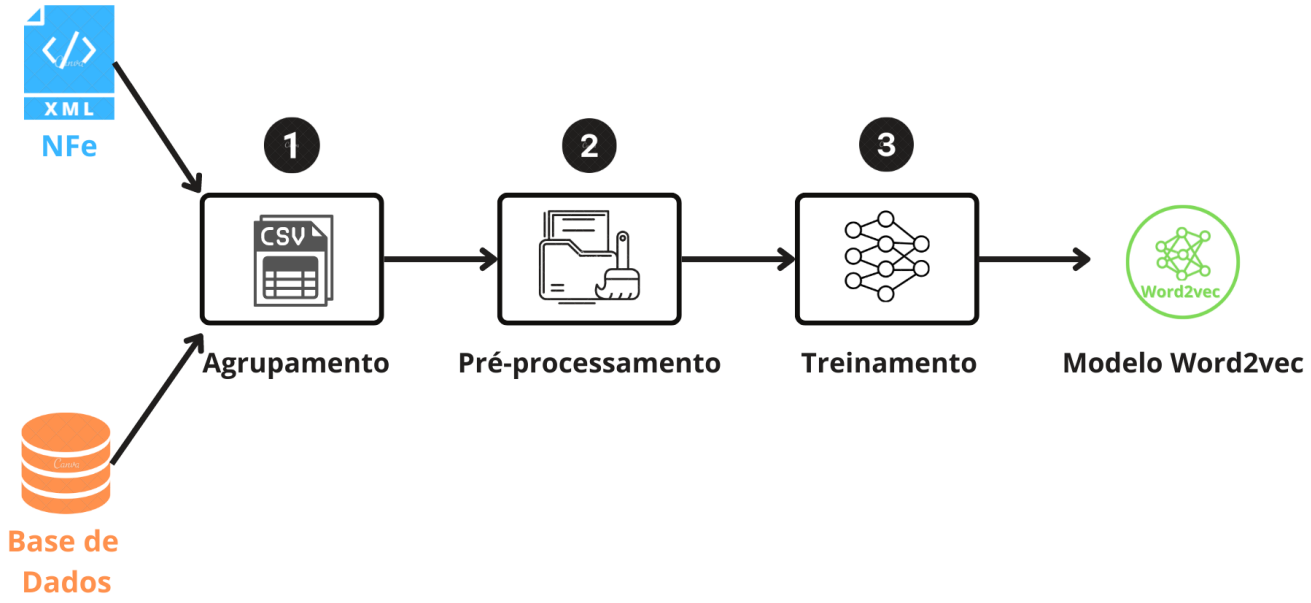

Figura 5: Etapas da criação do modelo Word2vec.
Fonte: Autor.

um arquivo CSV, com cerca de 138 mil produtos. Após um pré-processamento na base de dados foi verificado que cerca de 30 mil produtos não possuíam NCM ou tinham um NCM inválido. Esses produtos foram removidos uma vez que o NCM foi utilizado para filtrar os produtos por categoria[5]. A base final utilizada para os experimentos ficou com cerca de 108 mil produtos.

Para obter a amostra de produtos das notas fiscais eletrônicas da empresa, foi desenvolvida uma aplicação em Java que foi enviada para a empresa. Essa aplicação recebe como entrada o caminho de uma pasta com arquivos XML e o usuário pode selecionar quais dados deseja extrair das NFe. A saída são cópias dos arquivos XML contendo apenas os dados selecionados. Nessa etapa, foram extraídos a descrição e o NCM de 876 NFes dos arquivos XML. A amostra de produtos final possui cerca de 10 mil produtos, sendo 5.700 produtos com descrições únicas.

Os produtos com descrições únicas das NFe foram reunidos em um arquivo CSV para desenvolver um gabarito para validar as técnicas utilizadas durante o andamento deste trabalho. Para cada produto da NFe, foi identificado seu par equivalente na base de dados da empresa. Para esse processo, foi utilizada a função de similaridade *Cosine Similarity*, onde os pares com resultado de similaridade igual à 1, foram considerados equivalentes automaticamente. Os pares com resultado menor que 1 foram avaliados manualmente para identificar o par equivalente utilizando o auxílio de um especialista da empresa.

[5]Foi verificado em um trabalho anterior que utilizar o NCM como filtro possui maior custo-benefício (Moreira et al., 2020).

5.2 Métricas

Para avaliar a eficácia da função de similaridade *H-sim* para identificar produtos correspondentes, foram adotadas as métricas de reconhecimento de padrões com classificação binária: Precisão, revocação e F1 (*F-Measure*). A Precisão é calculada conforme a Eq. (7) e a Revocação conforme a Eq. (8).

$$P = \frac{m}{Ti} \tag{7}$$

$$R = \frac{m}{T} \tag{8}$$

onde P significa Precisão, R significa Revocação, m é o número de produtos da NFe cujo produto equivalente na base de dados relacional da empresa foi identificado corretamente, Ti é o número total de produtos da NFe que tiveram um par identificado e T é o número total de produtos da NFe. $F1$ é a média harmônica entre precisão e revocação. Quanto maior o índice de $F1$, maior a eficácia da identificação de correspondência de produtos, sendo calculada conforme a Eq. (9). A função de similaridade com o maior índice de $F1$ é considerada a mais eficaz.

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{9}$$

5.3 Ambiente de Configuração

Esta seção descreve a configuração do ambiente utilizado durante os experimentos deste trabalho. Para criar a *envi-*

roment (ambiente isolado), administrar bibliotecas, versões e pacotes foi utilizada a ferramenta Anaconda na versão 2.1.4. A *enviroment* utilizada foi o Jupyter Notebook na versão 6.4.8. A linguagem de programação utilizada foi Python na versão 3.9.12. A ferramenta de edição de código-fonte utilizada foi o Visual Studio Code na versão 1.70.2.

Os experimentos foram executados em um notebook Acer Predator Helios 300 G3-572-75L9, com um processador Intel(R) Core(TM) i7-7700HQ de 2.80GHz (8 CPUs), 32GB de memória RAM, placa de vídeo GTX 1060 de 6GB, um SSD de 1TB e o Sistema Operacional (SO) Windows 10 Single Language.

5.4 Configuração do Cálculo da Similaridade Semântica da H-sim

O objetivo desta seção é avaliar os melhores parâmetros do modelo Word2vec para calcular a similaridade semântica utilizando as técnicas de construção de vetor de sentença. O ambiente de configuração para o modelo Word2vec utilizou alguns parâmetros fixos que se repetiram em todos os experimentos, como taxa de aprendizado inicial (*alpha*), decaimento da taxa de aprendizado (*min_alpha*), número de *threads* (*workers*) e algoritmo de treinamento (sg) com os respectivos valores: 0,03; 0,0007; 7 e 1 (skip-gram).

O restante das seções está organizado da seguinte forma. A Seção 5.4.1 apresenta os experimentos de definição da técnica de construção do vetor de sentença. A Seção 5.4.2 descreve os experimentos de definição do tamanho do vetor. A Seção 5.4.3 apresenta os experimentos da frequência mínima das palavras. A Seção 5.4.4 discute os experimentos do tamanho da janela de contexto. Por fim, a Seção 5.4.5 apresenta os experimentos do número de épocas.

5.4.1 Definição da Técnica de Construção do Vetor de Sentença

Este experimento tem como objetivo identificar a técnica de construção de vetor de sentença mais eficaz. Para esse experimento foram testadas as três técnicas propostas neste trabalho *Sum*, *Average* e *Weight*. Foram utilizados os parâmetros padronizados para o modelo Word2vec, como tamanho do vetor, tamanho do vocabulário, tamanho de janela, e épocas com os respectivos valores: 300, 2, 5 e 500.

A Tabela 3 apresenta os resultados para as três técnicas *Sum*, *Average* e *Weight*. Os seguintes resultados são apresentados: precisão, revocação e F1. Observa-se que as técnicas *Sum* e *Average* obtiveram um escore de F1 de 0,8635, enquanto a função *Weight* obteve um escore de F1 de 0,8603. Devido a proximidade dos resultados, as três técnicas foram mantidas para os próximos experimentos.

Tabela 3: Comparação das técnicas de construção de vetor de sentença

Técnica	Precisão	Revocação	F1
Sum	**0,8666**	**0,8605**	**0,8635**
Average	**0,8666**	**0,8605**	**0,8635**
Weight	0,8634	0,8573	0,8603

5.4.2 Definição do Tamanho do Vetor

O objetivo desse experimento é verificar qual é o tamanho de vetor de palavras com maior eficácia em identificar produtos correspondentes. Tamanho do vetor ou *vector size* é um valor definido para o tamanho do vetor que é utilizado para representar o contexto de uma palavra no modelo Word2vec. A qualidade da representação dos dados aumenta gradativamente quanto maior a dimensionalidade do tamanho do vetor. No entanto, após atingir algum limite ou valor estipulado, o ganho marginal diminuirá e chegará a um valor insignificante para a eficácia do modelo Word2vec, apenas aumentando seu tempo de treinamento. Neste experimento, foram testados valores entre 200 e 400. As três técnicas de construção de vetor de sentença *Sum*, *Average* e *Weight* foram testadas com o valores fixos de tamanho de vocabulário, janela e épocas: 2, 5 e 500, respectivamente.

Tabela 4: Comparação entre diferentes tamanhos de vetor de palavras

Técnica	Tamanho do Vetor	Precisão	Revocação	F1
Sum	200	0,8644	0,8583	0,8614
	300	**0,8666**	**0,8605**	**0,8635**
	400	0,8657	0,8595	0,8626
Average	200	0,8644	0,8583	0,8614
	300	**0,8666**	**0,8605**	**0,8635**
	400	0,8657	0,8595	0,8626
Weight	200	0,8621	0,8559	0,859
	300	**0,8634**	**0,8573**	**0,8603**
	400	0,862	0,855	0,858

A Tabela 4 apresenta os resultados para as técnicas de construção de vetor de sentença *Sum*, *Average* e *Weight*. Os seguintes resultados são apresentados: precisão, revocação e F1 para os tamanhos de vetor testados. Para as três técnicas, o tamanho do vetor com a maior F1 foi 300.

Durante os experimentos as técnicas *Sum* e *Average* obtiveram o mesmo valor de precisão, revocação e F1. Por esse motivo, para os experimentos seguintes são utilizadas apenas as técnicas *Average* e *Weight* e o tamanho do vetor com maior eficácia em identificar produtos correspondentes definido neste experimento.

5.4.3 Definição da Frequência Mínima das Palavras

O objetivo desse experimento é verificar qual é a frequência mínima das palavras (*min_count*) necessária para criar o vocabulário de palavras do modelo Word2vec. Palavras abaixo da frequência mínima são ignoradas e retiradas do vocabulário do modelo Word2vec. Neste experimento, foram testados valores entre 1 e 3. Para os experimentos, as técnicas de construção de vetor de sentença *Average* e *Weight* foram testadas com parâmetros fixos de tamanho de janela e épocas: 5 e 300, respectivamente.

A Tabela 5 apresenta os resultados para as técnicas de construção de vetor de sentença *Average* e *Weight*. São apresentados os seguintes resultados: precisão, revocação e F1 para a frequência mínima das palavras. Para as duas técnicas, a frequência mínima das palavras com maior F1 foi 2. Para os experimentos seguintes é utilizado somente a frequência mínima das palavras com maior F1.

Tabela 5: Comparação entre diferentes frequências mínimas de palavras

Técnica	Frequência Mínima das Palavras	Precisão	Revocação	F1
Average	1	0,8626	0,8564	0,8606
	2	**0,8666**	**0,8605**	**0,8635**
	3	0,8637	0,8576	0,8606
Weight	1	0,8607	0,8546	0,8576
	2	**0,8634**	**0,8573**	**0,8603**
	3	0,86	0,8539	0,8569

5.4.4 *Definição do Tamanho da Janela*

O objetivo desse experimento é verificar qual é o tamanho da janela mais eficaz para identificar produtos correspondentes. O tamanho da janela (*window*) é a quantidade de palavras que fazem parte do contexto de uma palavra *X*. Por exemplo, se o tamanho da janela for 5, são consideradas duas palavras antes e duas depois da palavra de entrada, além da própria palavra de entrada. O tamanho da janela tem diferentes efeitos dependendo das características da base de dados utilizada para o treinamento do modelo Word2vec. Janelas maiores (15 a 50) são conhecidas por induzir incorporações que são mais 'tópicas' ou 'associativas', melhorando seu desempenho em conjuntos de testes de analogia, enquanto janelas menores (3 a 15) induzem modelos mais 'funcionais' e 'sinonímicos', levando a um melhor desempenho em conjuntos de testes de similaridade (Lison e Kutuzov, 2017). Neste experimento, as técnicas de construção de vetor de sentença *Average* e *Weight* foram testadas com parâmetro de épocas fixo em 500, variando o tamanho da janela entre 3 e 9.

Tabela 6: Comparação entre diferentes tamanhos de janela

Técnica	Tamanho da Janela	Precisão	Revocação	F1
Average	3	0,8643	0,8582	0,8612
	5	0,8666	0,8605	0,8635
	7	**0,867**	**0,8609**	**0,8639**
	9	0,8665	0,8604	0,8634
Weight	3	0,8625	0,8563	0,8594
	5	**0,8634**	**0,8573**	**0,8603**
	7	0,8624	0,8562	0,8593
	9	0,8606	0,8545	0,8575

A Tabela 6 apresenta os resultados das técnicas de construção de vetor de sentença *Average* e *Weight*. Os resultados de precisão, revocação e F1 para os tamanhos de janela testados são apresentados. O tamanho de janela que obteve a maior F1 para a técnica *Average* foi 7. Enquanto que para a técnica *Weight* foi 5. Devido à diferença entre o tamanho de janela mais eficaz para as técnicas *Average* e *Weight*, é utilizado o valor mais eficaz para cada técnica nos próximos experimentos.

5.4.5 *Definição do Número de Épocas*

O objetivo desse experimento é verificar qual é o número de épocas com mais eficácia para identificar produtos correspondentes. Em aprendizado de máquina, uma interação inteira dos dados de treinamento através de um algoritmo é conhecido como época ou *epoch*. O número de épocas é um parâmetro crítico para o algoritmo. Ele especifica o número de épocas ou de iterações que um conjunto de dados irá passar pelo treinamento ou aprendizado de um algoritmo. Os parâmetros internos de um modelo de conjunto de dados são atualizados a cada época (Goodfellow et al., 2016). Neste experimento, as técnicas *Average* e *Weight* foram testadas com os melhores parâmetros obtidos através dos experimentos anteriores. Os testes de número de épocas iniciaram em 500, e variaram entre 400 e 1000 épocas.

Tabela 7: Comparação entre diferentes número de épocas

Técnica	Épocas	Precisão	Revocação	F1
Average	400	0,8659	0,8597	0,8628
	500	0,867	0,8609	0,8639
	750	**0,8672**	**0,861**	**0,864**
	1000	0,8663	0,8601	0,8632
Weight	400	0,863	0,8568	0,8599
	500	**0,8634**	**0,8573**	**0,8603**
	750	0,8613	0,8552	0,8583

A Tabela 7 apresenta os resultados das técnicas de construção de vetor de sentença *Average* e *Weight*. São apresentados os resultados de precisão, revocação e F1 para os números de épocas testados. Para a técnica *Average*, o número de épocas que obteve a maior F1 foi 750, enquanto para a técnica *Weight* o número de épocas que obteve a maior F1 foi 500. Para a função *Weight* não foi realizado o teste de 1000 épocas, pois os testes iniciaram com 500 épocas e, no experimento com 750 épocas, a técnica já apresentava perda de F1.

5.5 Configuração da H-sim

O objetivo desta seção é avaliar a eficácia de funções de similaridade individuais e combinadas para identificar correspondência de produtos. São analisadas funções baseadas em *token*, baseadas em distância de edição e semânticas.

O restante das seções está organizado da seguinte forma. A Seção 5.5.1 apresenta os experimentos com funções de similaridade individuais, enquanto a Seção 5.5.2 descreve os experimentos combinando funções de similaridade baseadas em *token* ou distância de edição com funções de similaridade semânticas.

5.5.1 *Eficácia das Funções de Similaridade Individuais*

O objetivo desse experimento é comparar a eficácia para identificar produtos correspondentes entre diferentes funções de similaridade individuais. Foram comparadas funções de similaridade textuais baseadas em *token* e baseadas em distância de edição com a função de similaridade semântica proposta neste trabalho. Neste experimento, foram testadas as funções de similaridade baseadas em *token*: Cosine, Jaccard, Sorensen, Tversky e Overlap. As seguintes funções de similaridade baseadas em distância de edição foram testadas: Jaro, JaroWinkler, StrCmp95 e Hamming. Para as funções de similaridade baseadas em *token* e distância de edição foram selecionadas as funções que retornavam um escore normalizado entre 0 e 1

da biblioteca Python Textdistance[6], sendo 1 totalmente correspondente, e 0 totalmente diferente. Para calcular a similaridade semântica foram testadas as técnicas de construção de vetor de sentença *Average* e *Weight*.

Para calcular a similaridade das funções textuais baseadas em token e distância de edição, foi realizado um pré-processamento nas descrições dos produtos, onde foram removidas as pontuações, as acentuações e as palavras foram convertidas para minúsculo.

A Fig. 6 apresenta os resultados de F1 para as funções testadas. A função de similaridade baseada em *token* mais eficaz foi a Cosine e a menos eficaz foi a Overlap. A função de similaridade baseada em distância de edição mais eficaz foi Jaro e a menos eficaz foi Hamming.

A função semântica *Average* foi 1,30% mais eficaz que a função baseada em *token* Cosine, e 2,36% mais eficaz que a função baseada em distância de edição Jaro. Enquanto que, a função semântica *Weight* foi 0,88% mais eficaz que a função baseada em *token* Cosine, e 1,91% mais eficaz que a função baseada em distância de edição Jaro.

5.5.2 *Eficácia das Funções de Similaridade Combinadas*

Este experimento visa analisar a eficácia das funções de similaridade semânticas combinadas com funções de similaridade baseadas em token ou distância de edição. Para esse experimento foram selecionadas seis funções de similaridade baseadas em *token* e distância de edição. Onde três dessas funções são baseadas em *token*, e foram escolhidas as duas funções com o maior escore de F1 (Cosine e Jaccard) e a função com o menor escore de F1 (Overlap). As outras três funções são baseadas em distância de edição, onde foram escolhidas as duas funções com o maior escore de F1 (Jaro e Jaro-Winkler) e a função com o menor escore de F1 (Hamming). Para as funções de similaridade semântica, foram testadas as técnicas de construção de vetor de sentença *Average* e *Weight*. Foram selecionadas as piores funções baseadas em token e distância de edição para analisar a complementariedade que uma função semântica pode trazer.

Tabela 8: Comparação entre as combinações de funções de similaridade utilizando a técnica *Average*

Função	Precisão	Revocação	F1
Average	0,8672	0,861	0,864
Average + Cosine	0,8794	0,8731	0,8762
Average + Jaccard	0,8725	0,8663	0,8694
Average + Jaro	0,8764	0,8702	0,8733
Average + JaroWinkler	0,8724	0,8662	0,8693
Avarage + Hamming	0,7975	0,7918	0,7946
Average + Overlap	**0,8802**	**0,8739**	**0,877**

A Tabela 8 apresenta os resultados das combinações da função semântica com a técnica *Average* com as funções de similaridade baseadas em *Token* e distância de edição. São apresentados os resultados de precisão, revocação e F1. A combinação com maior F1 foi *Average + Overlap*.

A Tabela 9 apresenta os resultados das combinações da função semântica com a técnica *Weight* com as funções de similaridade baseada em *Token* e distância de edição. São apresentados os resultados de precisão, revocação e F1. A combinação com maior F1 foi *Weight + Cosine*.

Tabela 9: Comparação entre as combinações de funções de similaridade utilizando a técnica *Weight*

Função	Precisão	Revocação	F1
Weight	0,8634	0,8573	0,8603
Weight + Cosine	**0,8789**	**0,8727**	**0,8758**
Weight + Jaccard	0,8722	0,866	0,8691
Weight + Jaro	0,8746	0,8684	0,8715
Weight + JaroWinkler	0,8719	0,8657	0,8688
Weight + Hamming	0,7986	0,793	0,7958
Weight + Overlap	0,876	0,8698	0,8729

Destaca-se que a função de similaridade *H-sim* proposta neste trabalho, combina uma função de similaridade semântica com uma função de similaridade baseada em *token* ou distância de edição.

Considerando os resultados das Tabela 8 e Tabela 9, é possível concluir que houve uma melhoria de 0,15% na precisão, 0,14% na revocação e 0,14% de escore de F1 ao utilizar a função de similaridade semântica *Average* combinada com a função de similaridade baseada em *token* Overlap em comparação com a função de similaridade semântica *Weight* combinada com a função de similaridade baseada em *token* Cosine.

6 Conclusão

Este trabalho propõe a função de similaridade *H-sim* que combina funções de similaridade semânticas com funções de similaridade baseadas em *token* ou funções baseadas em distância de edição para identificar correspondência de produtos. Ao invés de utilizar apenas a verificação da escrita ou caracteres das palavras que compõem a descrição dos produtos, este trabalho analisa também o contexto dessas palavras por meio de uma função de similaridade semântica. Para calcular a similaridade semântica, foram testadas três técnicas de construção de vetor de sentença, que permitem definir o contexto dos produtos: *Sum*, *Average* e *Weight*.

Os experimentos utilizaram a base de dados relacional de uma empresa real e uma amostra de notas fiscais eletrônicas de fornecedores dessa empresa. Através dos experimentos foram definidos os melhores parâmetros para o modelo Word2vec e a melhor técnica de construção de vetor de sentença para identificar produtos correspondentes. A técnica *Sum* foi descartada durante os experimentos pois seus resultados de F1 eram idênticos à técnica *Average*. O uso de IDF para a técnica de construção de vetor *Weight* não resultou em ganhos de F1 em relação a técnica *Average*. Após definir os melhores parâmetros do modelo Word2vec e a melhor técnica de construção de vetor de sentença, foram combinadas as funções de similaridade semânticas com funções de similaridade baseadas em *token* ou distância de edição. A combinação da função de similaridade semântica com a técnica de construção de vetor de sentença *Average* com a função de similaridade baseada em *token* Overlap se mostrou a opção mais eficaz.

Para trabalhos futuros, foram identificadas algumas melhorias e experimentos que podem ser desenvolvidas:

[6] Disponível em: https://pypi.org/project/textdistance/

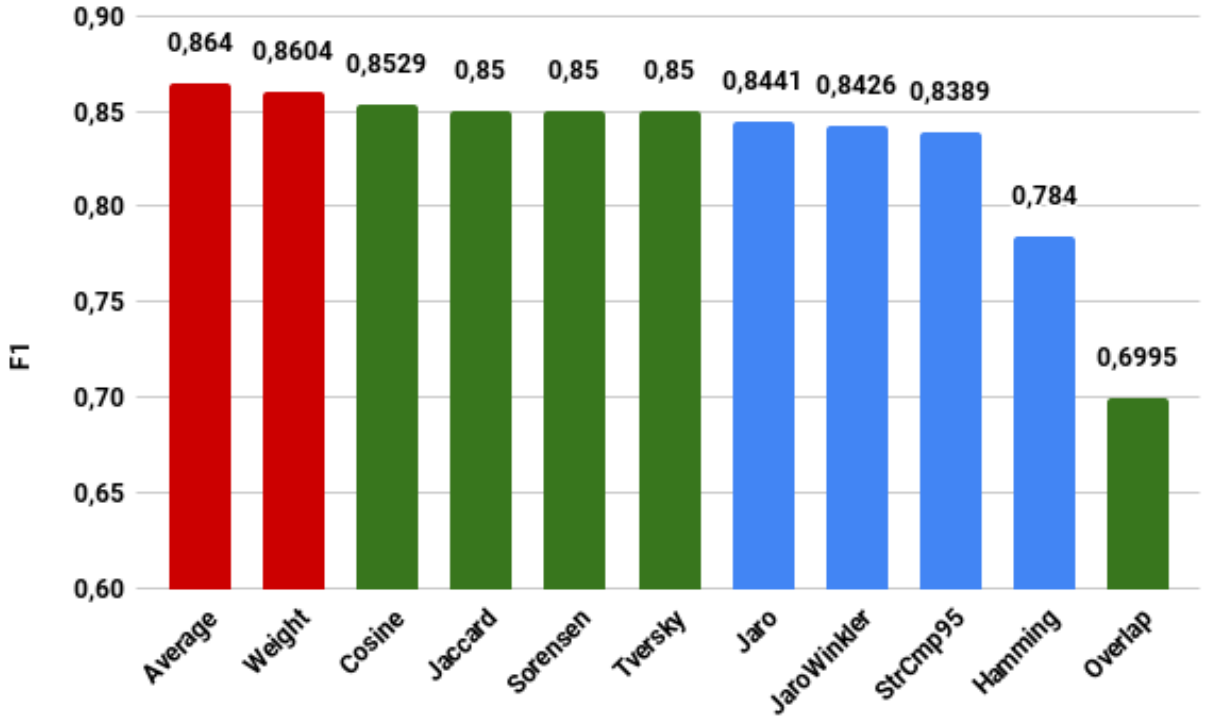

Figura 6: Comparação entre as funções de similaridade individuais. Vermelho: semânticas, Verde: baseadas em *token*, Azul: baseadas distância de edição.
Fonte: Autor.

(i) testar a função de similaridade *H-sim* em outras bases de dados de produtos; (ii) definir o limiar de similaridade que permite a importação automática sem a necessidade de verificação humana; (iii) comparar com soluções baseadas em BERT; e (iv) combinar diferentes funções de similaridade usando algoritmos de classificação.

Referências

Blurock, E. (2021). String similarity metrics: Token methods, Disponível em:https://www.baeldung.com/cs/string-similarity-token-methods. Último acesso em: 27/08/2022.

Cohen, W., Ravikumar, P. e Fienberg, S. (2003). A comparison of string distance metrics for name-matching tasks, *IIWeb* **2003**: 73–78.

Goodfellow, I. J., Bengio, Y. e Courville, A. (2016). *Deep Learning*, MIT Press, Cambridge, MA, USA. http://www.deeplearningbook.org.

Harispe, S., Ranwez, S., Janaqi, S. e Montmain, J. (2015). *Semantic Similarity from Natural Language and Ontology Analysis*, Springer International Publishing. https://doi.org/10.1007/978-3-031-02156-5.

Li, J., Dou, Z., Zhu, Y., Zuo, X. e Wen, J.-R. (2019). Deep cross-platform product matching in e-commerce, *Information Retrieval Journal* **23**: 136–158. https://doi.org/10.1007/s10791-019-09360-1.

Lison, P. e Kutuzov, A. (2017). Redefining context windows for word embedding models: An experimental study, *Proceedings of the 21st Nordic Conference on Computational Linguistics*, Association for Computational Linguistics, Gothenburg, Sweden, pp. 284–288. Disponível em https://aclanthology.org/W17-0239.pdf.

Mikolov, T., Chen, K., Corrado, G. e Dean, J. (2013). Efficient estimation of word representations in vector space. https://doi.org/10.48550/arXiv.1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. e Dean, J. (2013). Distributed representations of words and phrases and their compositionality. https://doi.org/10.48550/arXiv.1310.4546.

Ministério da Fazenda (2022). Sobre a nf-e, Disponível em: http://www.nfe.fazenda.gov.br/portal/sobreNFe.aspx?tipoConteudo=PEhYdxncZBE=. Último acesso em: 27/08/2022.

Moreira, H., Manica, E. e Mocelin, I. (2020). Avaliação da eficácia de funções de similaridade para importação de notas fiscais eletrônicas, *Anais do 5° SALÃO DE PESQUISA, EXTENSÃO E ENSINO DO IFRS*, SICT - SEMINÁRIO DE INICIAÇÃO CIENTÍFICA E TECNOLÓGICA, IFRS, Bento Gonçalves, Brasil.

Navarro, G. (2001). A guided tour to approximate string matching, *ACM Comput. Surv.* **33**(1): 31–88. https://doi.org/10.1145/375360.375365.

Orkphol, K. e Yang, W. (2019). Word sense disambiguation using cosine similarity collaborates with word2vec and wordnet, *Future Internet* **11**(5). https://doi.org/10.3390/fi11050114.

Peeters, R., Bizer, C. e Glavas, G. (2020). Intermediate training of bert for product matching, *DI2KG@VLDB*, Tokyo, Japan.

Tracz, J., Wójcik, P. I., Jasinska-Kobus, K., Belluzzo, R., Mroczkowski, R. e Gawlik, I. (2020). BERT-based similarity learning for product matching, *Proceedings of Workshop on Natural Language Processing in E-Commerce*, Association for Computational Linguistics, Barcelona, Spain, pp. 66–75. Disponível em https://aclanthology.org/2020.ecomnlp-1.7.pdf.