Instituto de MATEMÁTICA E ESTATÍSTICA UFRGS

UFRGS UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Trabalho de Conclusão de Curso

Comparação de arquiteturas de Word2Vec na análise de textos curtos

Tainá Ferreira Cabalheiro

1 de fevereiro de 2023

Tainá Ferreira Cabalheiro

Comparação de arquiteturas de Word2Vec na análise de textos curtos

Trabalho de Conclusão apresentado à comissão de Graduação do Departamento de Estatística da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para obtenção do título de Bacharel em Estatística.

Orientador(a): Profa. Dra. Márcia Barbian

Porto Alegre
1 de fevereiro de 2023

Agradecimentos

Agradeço aos meus pais, Amélia e Everardo, pelo apoio e incentivo incondicional em toda essa jornada. E por serem os melhores pais que eu poderia ter. Sempre lembrarei da primeira vez que nos perdemos no Vale e em como eu pude sentir o orgulho que irradiava de vocês, foi muito importante para mim.

Aos meus irmãos, Eduarda, Ramon, Taiane e Taise, agradeço por estarem sempre presentes na minha vida, me dando todo o suporte que necessito, serão sempre os melhores irmãos do mundo. Eu amo vocês.

À minha família por inteiro, por serem tão especiais e pacientes comigo, somos uma família louca, mas sou muito grata por ela.

Ao grupão da estatística, por terem me acolhido e me ajudado no que precisei. Vocês sempre foram mais do que colegas, foram amigos, tornando esses anos um pouco menos difíceis, Franciele, Bruna, Renan, Gabriela, Rafaela e todos os demais.

À Giulia, por ter seguido o conselho da minha mãe e falado comigo no primeiro dia de aula. Será para sempre minha dupla na estatística.

Ao Gabriel, por ter me convidado para almoçar quando me viu sozinha na fila do RU e ter se tornado um dos meus melhores amigos do curso, será sempre meu melhor bixo.

Aos meus amigos, Luiz e Juliana, por serem meu suporte no Vale no último semestre. Nunca esquecerei nossos almoços e como vocês são especiais.

Ao Fred, por sempre lembrar de mim.

Ao Andrey, por sempre me tirar um sorriso.

Às amigas de escola, Aléxia, Joka, Gabrielle e Thainá, por terem compartilhado a vida comigo e me ajudado a chegar até aqui.

Ao Departamento de Estatística da UFRGS como um todo, por terem me permitido realizar essa graduação.

À professora Márcia Barbian, por ter me guiado até aqui neste trabalho e por todos os conselhos e conhecimentos oferecidos, obrigada pela confiança.

À professora Luciana, por ter me apresentado o curso de estatística, foi uma grande inspiração.

Por fim, agradeço ao professor Hugo e ao Mestre Israel, por terem aceitado participar da minha banca.

Resumo

Em função do avanço na produção e armazenamento de dados de texto, houve uma grande procura pela área de Processamento de Linguagem Natural (NLP), o que acarretou o desenvolvimento de métodos cada vez mais complexos para lidar com tarefas relativas a diversas finalidades. Entre esses métodos encontra-se o *Word2Vec*, um algoritmo que utiliza redes neurais para aprender representações de palavras. Ele possui duas arquiteturas de rede: o CBoW, que tem como objetivo prever a palavra central de uma sentença através das palavras ao redor, o chamado contexto, e o Skip-gram, que faz o contrário, busca prever o contexto com base na palavra central. O presente trabalho visa aplicar as duas arquiteturas associadas ao *Word2Vec* a fim de obter representações *word embeddings* de palavras contidas em descrições de produtos de notas fiscais eletrônicas. Este dado é não estruturado, com tamanho máximo de 120 caracteres, possuindo vários desafios associados à análise de textos curtos além do vocabulário bastante específico das descrições. Foram ajustados alguns modelos para bancos de dados vinculados a dois produtos: leite e carne. Foram comparados ajustes considerando a repetição ou não dos documentos, o mínimo de vezes que as palavras aparecem no corpus e diferentes tamanhos de janela de contexto.

Palavras-Chave: Processamento de Linguagem Natural, Redes Neurais, Word2Vec, Continuous Bag of Words, Skip-gram, Notas Fiscais, Descrições de Produtos.

Abstract

Due to the advances in the production and storage of text data, there was a great demand for the area of Natural Language Processing (NLP), which led to the development of increasingly complex methods to deal with tasks related to different purposes. Among these methods is Word2Vec, an algorithm that uses neural networks to learn word representations. It has two network architectures: CBoW, which aims to predict the central word of a sentence through the surrounding words, the so-called context, and Skip-gram, which does the opposite, and seeks to predict the context based on the central word. The present work aims to apply the two architectures associated with Word2Vec to obtain word embeddings representations of words contained in product descriptions of electronic invoices. This data is unstructured, with a maximum size of 120 characters, with several challenges associated with the analysis of short texts in addition to the very specific vocabulary of the descriptions. Some models were adjusted for databases linked to two products: milk and meat. Adjustments were compared considering the repetition or not of the documents, the minimum number of times the words appear in the corpus, and different sizes of the context window.

Keywords: Natural Language Processing, Neural Network, Word2Vec, Continuous Bag of Words, Skip-gram, Invoices, Product Descriptions.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Atualmente uma quantidade enorme de dados é coletada automaticamente de celulares, computadores, máquinas industriais, antenas, câmeras e outros aparelhos que carregam informações para servidores constantemente (Segaran e Hammerbacher, 2009). A previsão é que em 2025, cada pessoa conectada terá ao menos uma interação de dados a cada 18 segundos, o que gerará cerca de 90 ZB de dados (Rydning et al., 2018). Com aumento da produção, capacidade de processamento e armazenamento de dados, o desenvolvimento de modelos mais complexos torna-se necessário. Especificamente, a área de análise de textos, possui uma crescente demanda, em razão do grande volume de informações textuais produzidos, exemplos são as redes sociais como o Facebook e o Twitter, reportagens associadas à diversos meios de comunicação, a avaliação de produtos ou serviços em diferentes plataformas ou a geração de diferentes dados por empresas de tecnologia.

A área de Processamento de Linguagem Natural (NLP) tem desenvolvido técnicas que abordam diferentes objetivos como: o reconhecimento de *spams*, exemplificado no artigo Kandasamy e Koroth (2014), em que os autores utilizaram técnicas de *Machine Learning* (ML) associado a conceitos de NLP na classificação de *spams* em publicações no Twitter; a análise de sentimento, como no trabalho de Devika et al. (2016), que comparou diversas técnicas incluindo Support Vector Machine, Naive Bayes e N-gram; reconhecimento de autoria, como em Kešelj et al. (2003) onde é aplicado N-gram; e análise de dados de redes sociais, tal como Acosta et al. (2017) que faz uma análise de sentimentos usando Word2Vec em dados do Twitter, entre outros.

Entre as diversas áreas que demandam a análise automática de texto, podemos destacar aquela que busca investigar o processo envolvendo a dinâmica da venda e compra de diferentes produtos dado a sua descrição, cujos objetivos podem ser a busca, estimativa de preços, análise de *reviews* ou agrupamento de produtos e consumidores. Os textos originados desse tipo de informação são bastante específicos e possuem diversas características como a quantidade limitada de caracteres, referência à marcas, códigos de diferentes modelos, utilização de diversas abreviações e distintas representações textuais para fracionamento. Dado essas especificidades, não é aconselhado utilizar técnicas que agrupam documentos como reportagens em agrupamentos de produtos (Yan et al., 2013). As diferentes reportagens possivelmente são compostas por centenas ou milhares de palavras, enquanto os documentos que compõem as descrições de produtos são formados por dezenas de palavras. Por isso, é importante avaliar como alguns dos principais algoritmos de NLP performam nesses dois cenários, neste trabalho especificamente, o objetivo será utilizar

algoritmos de NLP na identificação de mercadorias semelhantes dado a sua descrição, logo, envolve a análise de textos curtos não estruturados (Hossain et al., 2022). Entre os interessados nesse tipo de análise estão empresas como *marketplaces* e de marketing que buscam melhorar estratégias de negócio ou a experiência do usuário e órgãos públicos que buscam evitar fraudes ou aumentar a eficiência na gestão da informação.

O objetivo acima pode ser atingido de diferentes formas, dependendo do tipo de característica que busca-se ressaltar na análise. Em NLP muitas vezes trabalhamos com dados não estruturados de tamanhos arbitrários, que podem representar tanto documentos de textos curtos (descrição de produtos) quanto documentos mais longos (capítulos de livros). Tais documentos precisam ser representados de forma computacional, o que possibilita análises automáticas de linguagem. Isto significa que independentemente da origem do dado, o ideal é ser capaz de modelar semelhanças em tais estruturas, em alguns casos, isso significa codificar os dados como um vetor de largura fixa, que posteriormente será aplicado em algum método. Entre as possíveis formas de representar palavras podemos citar: *one-hot-encoding*, *bag of words*, *n-gram* e *word embeddings*. Métodos de *Machine Leaning*, como Naive Bayes e SVM fazem uso de *bag of words*. Ainda, existem os chamados Modelos N-gram (Brown et al., 1992) cujo o nome é auto-explicativo sobre qual representação estão relacionados. Métodos como Word2Vec (Mikolov et al., 2013a) e GloVe (Pennington et al., 2014) são usados para criar *word embeddings*.

Neste trabalho duas arquiteturas de redes neurais do método Word2Vec serão comparadas: o CBoW e o Skip-gram. O banco de dados a ser analisado é composto pela descrição de produtos contidas nas notas fiscais eletrônicas (NFe) de compras realizadas por órgãos públicos do estado do Rio Grande do Sul. Dado a grande quantidade de produtos e para fins acadêmicos, foram aplicados modelos para bancos de dados vinculados a dois produtos: leite e carne, comparando ajustes considerando a repetição ou não dos documentos, o mínimo de vezes que as palavras aparecem no corpus e diferentes tamanhos de janela de contexto. Para avaliar os modelos por meio de gráficos, foi utilizado o t-SNE (Van der Maaten e Hinton, 2008) a fim de reduzir a dimensionalidade dos vetores das palavras.

Esta monografia está organizada da seguinte forma: o Capítulo 2 apresenta os principais métodos de representação de texto utilizados na área de Processamento de Linguagem Natural, buscando apresentar de forma breve cada um. Em seguida serão abordados métodos de *Machine Learning*, como SVM, Naive Bayes, Random Forest e Boosting. O capítulo seguinte é sobre redes neurais, em que o foco é Word2Vec, mas também são citados outros métodos como Redes Neurais Recorrentes, GloVe e BERT. No Capítulo 4 apresenta-se os resultados das aplicações dos métodos através de gráficos e tabelas. Por fim, no Capítulo 5 são apresentados as conclusões e discussões finais do trabalho.

2 Metodologia

2.1 Processamento de Linguagem Natural

O processamento de linguagem natural, do inglês *Natural Language Processing* (NLP), trata-se de um conjunto de técnicas computacionais cujo intuito é analisar e representar de forma automática a linguagem humana (Cambria e White, 2014). Para tal, utiliza-se de diversas áreas do conhecimento, como linguística computacional, ciência cognitiva e inteligência artificial (Deng e Liu, 2018). Entre os exemplos de técnicas de NLP pode-se citar análise sintática (Socher et al., 2013a), análise de sentimento (Socher et al., 2013b) e tradução automática de textos (Chiang, 2007).

Devido ao avanço computacional e aos volumes gigantescos de dados textuais as técnicas de NLP tiveram grande evolução nos últimos anos, visto que para o desenvolvimento desses métodos é necessária uma grande base de dados que consiga capturar a complexa linguagem humana.

Uma tarefa muito importante ao analisar dados de texto é o pré processamento dos dados, que busca "limpar" o banco de inconsistências que possam atrapalhar a análise. Para isso é importante retirar as palavras que não contribuem para o texto, consideradas irrelevantes, as chamadas *stopwords*, também é necessário a retirada de acentos e outros símbolos comuns em dados de texto. A etapa seguinte é transformar o texto de forma que as máquinas entendam. Entre os algoritmos utilizados para representar essa linguagem pode-se citar: *bag of words*, *n-grams*, TF-IDF e os *word embeddings*.

Para compreender melhor os conceitos citados acima e que serão expostos no restante do texto, há algumas definições importantes que devem ser mencionadas:

- o *corpus* Γ é o conjunto total de documentos organizados em formato de um banco de dados;
- o documento d é uma observação do *corpus* Γ, podendo ser uma frase ou um texto inteiro;
- os *tokens* são separações dos documentos em pedaços, podendo ser palavras sub-palavras e outros. Neste trabalho o *token* representará uma palavra;
- o vocabulário V é o conjunto total de palavras distintas no *corpus* Γ.

2.2 Métodos de Representação Textual

2.2.1 One-hot-encoding

No *one-hot-encoding* cada palavra (*word*) no vocabulário do *corpus* Γ recebe um índice único (*id*) $word_{id}$ que está entre 1 e $|V|$, em que $|V|$ representa o tamanho do vocabulário no *corpus*. As palavras são simbolizadas por um vetor binário do tamanho de V composto de 0s, com exceção da célula do índice que é preenchido por 1 (Vajjala et al., 2020). Um exemplo de como um *corpus*, com um único documento composto pela frase **Eu adoro o curso de estatística** é transformado na linguagem *one-hot-encoding* é retratado na Figura 2.1:

Documento: "Eu adoro o curso de estatística"

Índices:

Eu	adoro	o	curso	de	estatística
1	2	3	4	5	6

Matriz one-hot:

	1	**2**	**3**	**4**	**5**	**6**
Eu	1	0	0	0	0	0
adoro	0	1	0	0	0	0
o	0	0	1	0	0	0
curso	0	0	0	1	0	0
de	0	0	0	0	1	0
estatística	0	0	0	0	0	1

Figura 2.1: Exemplo de representação *one-hot-encoding* de um documento d, composto por um conjunto de 6 *tokens*.

Observe que no exemplo acima:

$$\Gamma = (\text{Eu adoro o curso de estatística}),$$

$$\begin{aligned} V &= (word_1, word_2, word_3, word_4, word_5, word_6) \\ &= (\text{Eu, adoro, o, curso, de, estatística}), \end{aligned}$$

$$|V| = 6, \quad \text{e} \quad d = 1.$$

A codificação *one-hot* é de fácil compreensão e de simples implementação, mas possui algumas limitações, como não conseguir capturar similaridades entre as palavras e, ainda, o tamanho do vetor ser da dimensão do vocabulário. Como a maioria dos bancos de dados é composto por um grande vocabulário, esse representação é ineficiente quanto ao armazenamento de informação (Vajjala et al., 2020).

2.2.2 Bag of words

Lançado em 1975 por (Salton et al., 1975), o Bag of words (BoW) é um modelo muito comum em NLP (Vajjala et al., 2020). Nele, cada palavra é representada por

um índice e cada documento é convertido em um vetor $|V|$-dimensional. Nas células dos índices das palavras *word* são indicadas a quantidade de ocorrências da palavra no documento d (Vajjala et al., 2020). Entre os pontos positivos desse modelo está a fácil compreensão e implementação (Vajjala et al., 2020). No entanto, ele possui algumas deficiências, como a utilização de matrizes esparsas e o tamanho do vetor crescer conforme o vocabulário aumenta. Como o nome do modelo indica, é um "saco de palavras", isto é, não é considerada a ordem das palavras no documento. Ainda, o modelo não captura a semelhança de palavras diferentes com o mesmo significado, desconsiderando a semântica (Schütze et al., 2008). Na Figura 2.2 podemos ver um exemplo de representação do BoW.

Documento 1: "Eu adoro o curso de estatística"
Documento 2: "Eu faço curso de inglês"
Documento 3: "Eu adoro o curso de inglês e o curso de estatística"

Eu	adoro	o	curso	de	estatística	faço	inglês	e
1	2	3	4	5	6	7	8	9

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Documento 1	1	1	1	1	1	1	0	0	0
Documento 2	1	0	0	1	1	0	1	1	0
Documento 3	1	1	2	2	2	1	0	1	1

Figura 2.2: Exemplo da representação *bag of words* para a um conjunto de 3 documentos.

2.2.3 TF-IDF

A sigla TF-IDF é uma abreviação de *Term Frequency Inverse Document Frequency*, nessa representação textual é calculada a frequência relativa das palavras *word* em cada documento d dado a proporção inversa dessas palavras em todos os documentos (*corpus*).

Nota-se que, quando as palavras ocorrem muitas vezes em poucos documentos o valor TF-IDF é maior, pois, palavras que ocorrem muitas vezes em poucos documentos indicam alguma característica daquele documento, enquanto as palavras que ocorrem muito em vários documentos são palavras comuns, como artigos e preposições (Ramos et al., 2003), consequentemente recebem pontuações mais baixas para o TF-IDF. Logo a ideia do método é determinar a relevância das palavras nos documentos. A equação é expressa por Salton e Buckley (1988) da seguinte forma:

$$word_d = f_{word,d} \times \log(|\Gamma| f_{word,\Gamma}),$$

em que Γ representa o corpus, $|\Gamma|$ o tamanho do corpus, *word* a palavra, d um documento pertencente a Γ, $f_{word,d}$ é o numero de vezes que *word* aparece em d e $f_{word,\Gamma}$ é o numero de vezes que *word* aparece em Γ.

Para exemplificar, serão realizados os cálculos para as palavras *curso* e *estatística* dos documentos da Figura 2.2. Para a palavra *curso* no documento 1 e 2, temos

$$\begin{aligned} curso_1 = curso_2 &= 1 \times \log(9 \times 4) \\ &= \log(36) \\ &= 3.5835. \end{aligned}$$

Agora, para o documento 3 obtemos o seguinte:

$$\begin{aligned} curso_3 &= 1 \times \log(9 \times 4) \\ &= \log(36) \\ &= 7.1670. \end{aligned}$$

Para a palavra *estatística* no documento 1 e 3:

$$\begin{aligned} estatistica_1 = estatistica_3 &= 1 \times \log(9 \times 2) \\ &= \log(18) \\ &= 2.8904. \end{aligned}$$

Ainda, o valor obtido para *estatística* no documento 2 é:

$$\begin{aligned} estatistica_2 &= 0 \times \log(9 \times 2) \\ &= 0. \end{aligned}$$

Portanto, o valor TF-IDF de *curso* no documento 1 é 3.5835, o mesmo valor de *curso* no documento 2, por terem aparecido o mesmo número de vezes. Para a palavra *estatistica* o valor TD-IDF se repete para o documento 1 e 3 que é 2.8904, enquanto no documento 2 é 0, por não aparecer na frase "eu faço curso de inglês".

2.2.4 N-grams

Segundo Broder et al. (1997) um *n-gram* é um pedaço composto por n *tokens* contíguos de um texto maior. Os *n-grams* são modelos de Markov e tem como objetivo prever a próxima palavra baseado em uma janela fixa n, em que as probabilidades são estimadas através da contagem de ocorrências em um *corpus* Γ (Jurafsky e Martin, 2021a). Embora ainda seja bastante utilizado nos dias atuais o modelo é antigo, na década de 50, Shannon (1948) já aplicava *n-gram* para computar sequências de palavras em inglês. Apesar de ser muito comum na análise de textos e possuir uma grande referência teórica, o modelo possui atributos a serem melhorados, em Bengio et al. (2003) são citados dois deles, primeiro: os contextos geralmente são pequenos, de uma ou duas palavras apenas e segundo: não é considerada a semelhança entre as palavras.

Para exemplificar a representação de um documento através de um *n-gram*, considere novamente que a frase **Eu adoro o curso de estatística** é uma observação de um documento d que compõe o corpus Γ.

Se considerarmos um *n-gram* de tamanho $n = 2$, teremos o seguinte:

Eu adoro, **adoro o**, **o curso**, **curso de** e **de estatística**.

2.2.5 Word embeddings

Os modelos anteriores fazem representações apenas baseadas em contagem, por essa razão não trazem informações semânticas na análise dos documentos que compõem o *corpus*. Em 2003, Bengio et al. (2003) propuseram a metodologia de representações

distribuídas, através de um modelo de linguagem que usa redes neurais, em que era associado a cada palavra do vocabulário um vetor, denominados *embeddings*. Espera-se que os *word embeddings* de palavras semelhantes sejam parecidos.

A ideia principal dos *word embeddings* é que palavras similares ocorrem em contextos semelhantes, por exemplo:

A casa é amarela.

A casa é azul.

Amarela e azul são cores, então espera-se que as palavras em torno desses *tokens* também sejam semelhantes, ou seja, elas ocorram em contextos similares.

Mikolov et al. (2013a) apresentaram dois novos modelos para estimar as representações vetoriais em conjuntos de dados grandes, o CBoW e o Skip-gram. Essas arquiteturas obtiveram resultados satisfatórios com um baixo custo computacional, o que ajudou a popularizar o método proposto inicialmente por Bengio et al. (2003). A ideia é retratar as palavras como pontos em um espaço semântico multidimensional oriundo das distribuições das palavras ao redor (Jurafsky e Martin, 2021c), que produzam vetores em um espaço contínuo. Tais vetores capturam a semântica das palavras e permitem a realização de operações matemáticas. Exemplo famoso é o caso do modelo de Mikolov et al. (2013a) em que é feita a seguinte operação com os vetores das palavras:

$$\text{Rei} - \text{Homem} + \text{Mulher} \approx \text{Rainha}.$$

A Figura 2.3 apresenta a operação com os *word embeddings* indicados acima, em que diminui-se o vetor de *homem* do vetor de *rei* e soma-se com o vetor referente a *mulher*, dessa forma obtendo um vetor muito parecido com o da palavra *rainha*. Nesse exemplo didático o vetor das palavras é representado em duas dimensões.

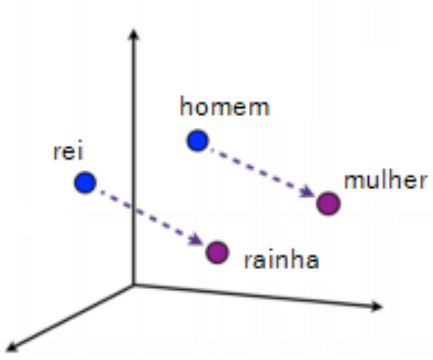

Figura 2.3: Exemplo 3D de uma representação de Word Embeddings
Fonte: Word Embedding: fazendo o computador entender o significado das palavras

2.3 Machine learning

Em NLP os dados geralmente são não estruturados, essa característica marcante torna comum que análise de dados de texto estejam associadas à técnicas de Aprendizado de Máquina, em inglês *Machine learning* (ML), uma área da Inteligência Artificial (IA) que desenvolve algoritmos capazes de aprender com os dados. O propósito de ML é que o algoritmo aprenda sozinho, utilizando uma parte dos dados

para treinar o modelo para reconhecer padrões. Uma definição mais formal é dada por Mitchell e Mitchell (1997): "Diz-se que um programa de computador aprende pela experiência E, com respeito a algum tipo de tarefa T e performance P, se sua performance P nas tarefas em T, na forma medida por P, melhoram com a experiência E". As técnicas de *Machine Learning* contribuem profundamente para classificação de texto, o que engloba diversas tarefas de NLP, como detecção de *spam* em e-mails, análise de sentimentos, identificação de autoria, identificação de idioma, categorização de tópicos de texto, entre outros.

2.3.1 Naive Bayes

Modelo muito famoso em NLP (Manning e Schutze, 1999) que utiliza o *bag of words* como método de representação textual. Este método é utilizado em problemas de classificação de texto, como em detecção de SPAM ou em análise de sentimento. Em detecção de *spam* o documento geralmente é classificado em dois grupos (*spam* ou não *spam*), em análise de sentimento o número de grupos ao qual o documento pode ser classificado pode ser maior, como identificar se determinado comentário é positivo, negativo ou neutro.

A técnica de *Naive Bayes* faz a classificação com base em probabilidades, desse modo, ao classificar um documento d, considerando todos os *grupos* possíveis, ele retorna o grupo com a maior probabilidade *a posteriori* dado as informações contidas no documento d (Jurafsky e Martin, 2021b):

$$\widehat{grupo} = \underset{grupo \in G}{\operatorname{argmax}} P(grupo|d) = \underset{grupo \in G}{\operatorname{argmax}} \frac{P(d|grupo)P(grupo)}{P(d)}.$$

Em que $P(grupo|d)$ representa a probabilidade *a posteriori*, $P(d|grupo)$ depende da função de verossimilhança, indicando a probabilidade de observar determinado documento d dado que ele pertence à classe *grupo*. Por fim, $P(grupo)$ é a probabilidade *a priori* de qualquer documento pertencer ao tópico *grupo* e $P(d)$ é a constante de normalização.

A suposição central assumida por esse algoritmo é que as variáveis possuem independência condicional, em análise de texto, essa característica indica que o método avalia a frequência que as palavras ocorrem no documento, a ordem em que elas são mencionadas não é considerada. Logo, se um documento d^* é composto por 3 palavras a distribuição conjunta $P(d|grupo)$ pode ser indicada como:

$$\begin{aligned} P(d^*|grupo) &= P(word_1 \cap word_2 \cap word_3|grupo) \\ &= P(word_1|grupo)P(word_2|grupo)P(word_3|grupo). \end{aligned}$$

Essa premissa é difícil de ser satisfeita, mas o modelo é frequentemente usado como um algoritmo inicial, justamente por esse suposição torná-lo simples e por ele ser fácil de treinar.

2.3.2 Support vector machines

Outro modelo bastante famoso de Machine Learning que utiliza *bag of words* como método de representação textual. Esse algoritmo, assim como o anterior, é utilizado, em geral, em classificações de texto, como em Devika et al. (2016). Ainda, ele pode ser usado para regressão ou agrupamento de documentos (Lorena e De Carvalho,

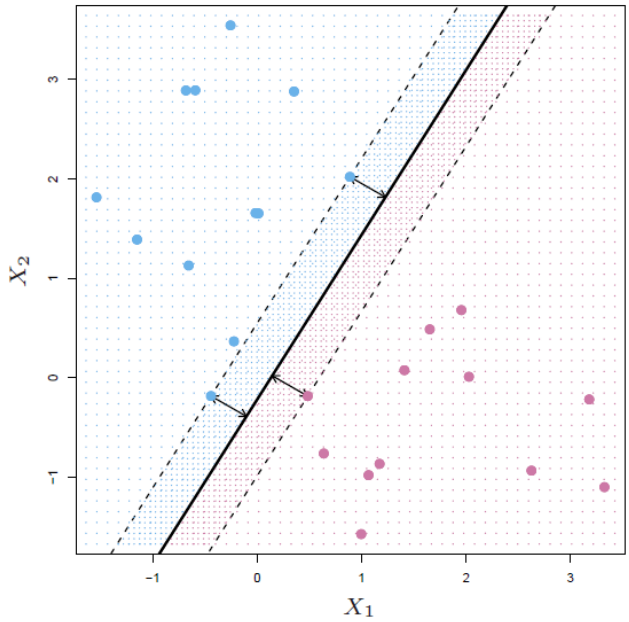

Figura 2.4: Representação SVM
Fonte: (James et al., 2013)

2007). Para a classificação ele busca aprender um limite de decisão que visa separar os dados em grupos de forma que a distância entre os pontos de cada grupo seja a maior possível (Vajjala et al., 2020). Um exemplo bidimensional pode ser visto na Figura 2.4, onde os pontos azuis representam uma classe e os pontos lilás representam outra.

2.3.3 Árvore de Decisão

Uma árvore de decisão é um algoritmo que utiliza a estrutura de árvore para dar suporte a uma decisão, pode ser usado tanto para regressão como para classificação. Em Manning e Schutze (1999), por exemplo, o foco é a categorização de tópicos de um texto. A classificação é realizada testando a pergunta do nó superior e ramificando para o nó apropriado. Em seguida, é repetido esse processo até a chegada a um nó folha (uma das classes) (Manning e Schutze, 1999). Na Figura 2.5 é possível ver a estrutura de uma árvore de decisão classificando um e-mail, primeiro é checado se foi enviado por “myEmployer.com”, se sim será classificado como “e-mail para ler quando estiver entediado”. Se não, é verificado se contém a palavra “hóquei” em seu corpo, se contiver é classificado como “e-mail de amigos", caso contrário, será identificado como “Spam: não leia” (Harrington, 2012). As árvores são simples e de fácil interpretação, no entanto não são competitivas em termos de decisão (James et al., 2013). Em seguida veremos métodos que utilizam de várias árvores para obterem melhores resultados.

2.3.4 Random Forest

O algoritmo *Random Forest* (Breiman, 2001) utiliza o *bootstrap* para selecionar de forma aleatória amostras dos dados de treino. Para cada amostra selecionada será construída uma árvore de decisão. A definição do primeiro nó de cada árvore é

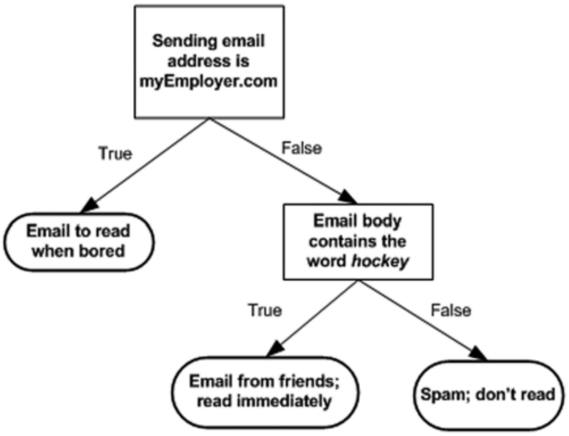

Figura 2.5: Exemplo de árvore de decisão
Fonte: (Harrington, 2012)

feita da seguinte forma: o algoritmo seleciona duas ou mais variáveis de forma aleatória com base na amostra em questão e então são realizados cálculos utilizando o algoritmo de entropia ou o índice de Gini para definir qual das variáveis será utilizada no primeiro nó. A seleção dos próximos nós é feita de forma semelhante, mas excluindo as variáveis utilizadas nos nós anteriores. Essa não é a melhor forma de construir a árvore, visto que o algoritmo pode selecionar variáveis ruins logo no primeiro nó, mas dado que são feitas muitas árvores, sendo elas diferentes umas das outras, o algoritmo se torna poderoso.

2.3.5 Boosting

Boosting é um método para melhorar previsões que pode ser aplicado em árvores ou outras técnicas. Nele são cultivadas diversas árvores também, mas ao invés de utilizar *bootstrap*, as árvores são cultivadas em sequência, utilizando as informações das outras árvores já crescidas. O procedimento é o seguinte: dado o modelo atual ajusta-se uma árvore de decisão aos seus resíduos, então adiciona-se essa árvore à função já ajustada para atualizar os resíduos. Ao realizar esse processo melhora-se lentamente a função estimada nas áreas que ela não funciona bem (James et al., 2013). Um exemplo de uso do Boosting em dados de texto é visto em Athanasiou e Maragoudakis (2017) onde é realizada uma análise de sentimentos.

3 Redes Neurais

McCulloch e Pitts (1943) deram início ao campo da computação neural com um artigo que formalizava a teoria de redes neurais (Raaijmakers, 2022). As redes neurais artificiais (RNAs) são baseadas nos neurônios do cérebro humano e em suas interações (Vajjala et al., 2020). Elas são formadas por camadas, a primeira representa as variáveis de entrada (*inputs*), em seguida há uma ou mais camadas ocultas e por fim a camada de saída (*outputs*). Cada camada é composta por neurônios artificiais que se conectam com as demais, transmitindo valores, como as sinapses em um cérebro humano. As conexões são chamadas de arestas e possuem pesos que se ajustam conforme a rede aprende, esses pesos podem aumentar ou diminuir o sinal das conexões entre os neurônios. O neurônio multiplica cada entrada por seu peso, essas multiplicações são somadas, então, aplica-se uma função (função de ativação) ao resultado dessa soma e o passa para sua saída.

Se os pesos forem definidos corretamente, uma rede neural com neurônios suficientes e com uma função de ativação não linear pode aproximar uma ampla gama de funções matemáticas, o que torna as redes neurais um método poderoso na representação de dados com diferentes características e estruturas de dependência.

Um exemplo de uma rede neural Feed-forward, isto é, uma rede em que a informação se move apenas em uma direção, pode ser visto na Figura 3.1.

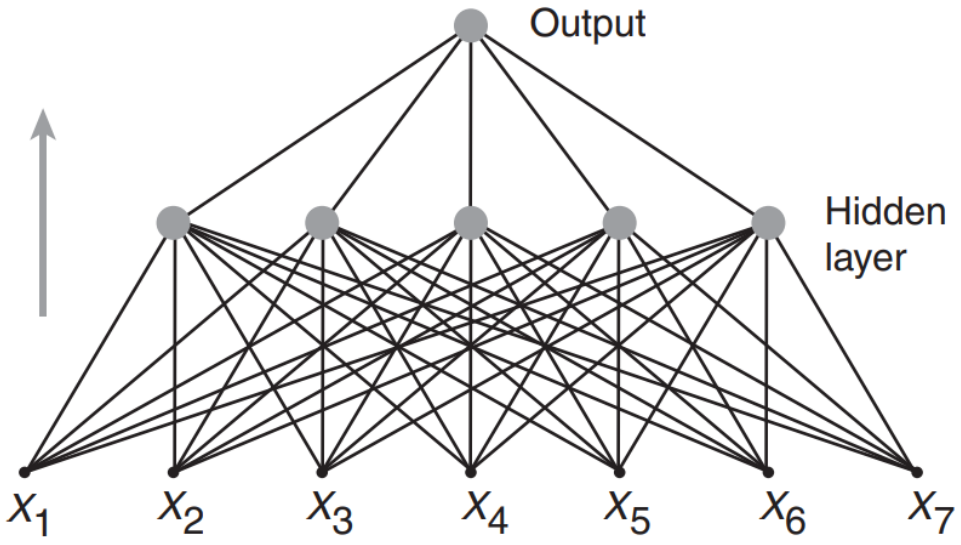

Figura 3.1: Exemplo Rede Neural Feed-forward
Fonte: (Krogh, 2008)

Cada círculo é um neurônio, com as setas de entrada representando as entradas do neurônio e as setas de saída as saídas do neurônio. Cada flecha carrega um peso, refletindo sua importância. Neurônios são organizados em camadas, refletindo o fluxo de informações. A camada inferior $X_1, X_2, \ldots, X_n$ não possui setas de entrada, essa camada representa as medidas das covariáveis que serão utilizadas na modelagem, os (*inputs*). A camada superior não possui setas de saída e representa o valor predito $\hat{y}$ da variável resposta y, o *output*. Se o objetivo da rede for predizer uma variável quantitativa o *output* será indicado por apenas um neurônio. Se o objetivo da rede for classificar determinada observação em um conjunto π de classes o *output* será um vetor de mesmo tamanho.

Na Figura 3.1, a rede possui apenas uma camada "oculta" e cada neurônio está conectado a todos os neurônios da próxima camada, nessas situações, denomina-se essa camada como totalmente conectada. É comum que dentro dos neurônios os pesos sejam combinados através de uma função não linear $g(x)$, a denominada função de ativação, ela é aplicada ao valor do neurônio antes de passá-lo para a saída). Entre as funções de ativação mais conhecidas na literatura podemos citar a sigmóide ($g(x) = \frac{1}{1+e^{-x}}$), a tangente hiporbólica e a ReLU (Goodfellow et al., 2016).

É possível representar a rede através de uma notação matemática, em que os valores de cada linha de neurônios na rede podem ser considerados como um vetor. A rede neural mais simples é denominada perceptron (Goodfellow et al., 2016) e é definida como uma função linear dos seus *inputs*

$$NN_{Perceptron}(\mathbf{x}) = \mathbf{x}\mathbf{W} + \mathbf{b},$$

$\mathbf{W}$ é a matriz de pesos e $\mathbf{b}$ é um vetor que representa o vício. Cada um dos termos da equação acima possuem diferentes dimensões, dependendo do tamanho dos vetores de *input* e *output*: $\mathbf{x} \in \mathbb{R}^{d_{input}}$, $\mathbf{W} \in \mathbb{R}^{d_{input} \times d_{output}}$ e $\mathbf{b} \in \mathbb{R}^{d_{output}}$.

Se a rede possuir uma camada oculta e uma função de ativação g não linear, a rede neural é chamada de *1-layer Multi Layer Perceptron*, exemplo da Figura 3.1. Nesse caso, a equação da rede neural pode ser definida como

$$NN_{MLP1}(\mathbf{x}) = g(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2,$$

$\mathbf{W}_2$ e $\mathbf{b}_2$ são a matriz de pesos e vetor do vício, respectivamente. A dimensão dos termos da equação acima não mudará para o vetor de *inputs* $\mathbf{x}$, os demais termos dependerão do número de neurônios da camada oculta e conexões entre a camada oculta e o *output*. Como ilustração $\mathbf{W_1} \in \mathbb{R}^{7\times5}$, $\mathbf{b_1} \in \mathbb{R}^5$, $\mathbf{W_2} \in \mathbb{R}^{5\times1}$ e $\mathbf{b_2} \in \mathbb{R}^1$ são as dimensões da NN_{MLP1} na Figura 3.1. A camada de entrada é um vetor de 7 dimensões $\mathbf{x}$, e a camada acima dela é um vetor de 5 dimensões $\mathbf{A}_1$. A camada totalmente conectada pode ser pensada como uma transformação linear de 7 para 5 dimensões que será transformado em um *output* de dimensão 1.

Uma camada totalmente conectada é uma matriz $\mathbf{A}_1 = \mathbf{x}\mathbf{W}$ onde o peso da conexão do i-ésimo neurônio na linha de entrada para o j-ésimo neurônio na linha de saída é W_{ij}. Os valores de $\mathbf{A}_1$ são então transformados por uma função não linear $g(\mathbf{A})$ que é aplicada a cada valor antes de ser passado para a próxima entrada. Todo o cálculo da entrada à saída pode ser escrito como: $(g(\mathbf{x}\mathbf{W}_1))$ onde $\mathbf{W}_1$ são os pesos da primeira camada.

Quando mais uma camada for adicionada a rede neural é chamada *2-layer Multi Layer Perceptron*, com equação

$$NN_{MLP2}(\mathbf{x}) = g_2(g_1(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2)\mathbf{W}_3 + \mathbf{b}_3. \quad (3.1)$$

É comum representar as redes da Equação 3.1 através de uma fórmula recursiva,

$$\begin{aligned} NN_{MLP2}(\mathbf{x}) &= \mathbf{A}_2\mathbf{W}_3 \\ \mathbf{A}_2 &= g_2(\mathbf{A}_1\mathbf{W}_2 + \mathbf{b}_2) \\ \mathbf{A}_1 &= g_1(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1). \end{aligned}$$

Generalizando a fórmula acima, o vetor resultante de cada transformação linear é chamado de camada. A camada mais externa da transformação linear resulta na camada de saída e as outras transformações lineares resultam em camadas ocultas ($\mathbf{A}_i$), em que cada camada oculta é seguida por uma ativação não linear. Redes com mais de uma camada oculta são chamadas de redes profundas, daí o nome aprendizagem profunda (*deep learning*).

3.1 Deep Learning

Deep Learning (DL) é um ramo de *Machine Learning* que se trata de um conjunto de técnicas de redes neurais artificiais com várias camadas, que capturam associações complexas entre as variáveis de entrada *inputs* (LeCun et al., 2015). Com a evolução da tecnologia nos últimos anos, o desenvolvimento de novas técnicas de DL tem se popularizado dentro de NLP, sendo usadas por exemplo, para melhorar o desempenho de reconhecimento de fala e tradução automática, (Vajjala et al., 2020).

Uma das arquiteturas apresentadas em DL são as Redes Neurais Recorrentes (RNNs), como a linguagem acontece em sequencia, flui em um sentido, e muitas vezes possui dependências de longo alcance (Raaijmakers, 2022) é interessante que um modelo seja capaz de ler um texto de forma sequencial e guardar memórias para usar em etapas seguintes das análises. Essa é a ideia das RNNs, apresentadas em Rumelhart et al. (1986) para modelar séries temporais. A estrutura é parecida com as de redes padrões como a apresentada na Figura 3.1, com a diferença de permitir conexões entre os neurônios das camadas ocultas referentes ao fluxo de informação no tempo. É através dessas conexões que o modelo guarda informações passadas, permitindo encontrar correlações temporais entre eventos distantes (Pascanu et al., 2013). As RNNs simples possuem uma quantidade limitada de memória. A cada espaço de tempo a rede implanta o estado da memória oculta do tempo anterior para produzir a saída do tempo atual. Assim, o estado de memória é composto por três fatores: o estado da memória no espaço de tempo anterior, uma matriz de pesos que pondera o estado da memória anterior e uma matriz de peso que pondera a entrada atual (Raaijmakers, 2022). Considerando que o tempo passa de forma discreta, a Figura 3.2 exemplifica uma rede neural recorrente simples, onde X são as entradas, S representa o estado da memória, W representa os pesos e y é a saída, é notável como os pesos são compartilhados por todas as atualizações (Raaijmakers, 2022).

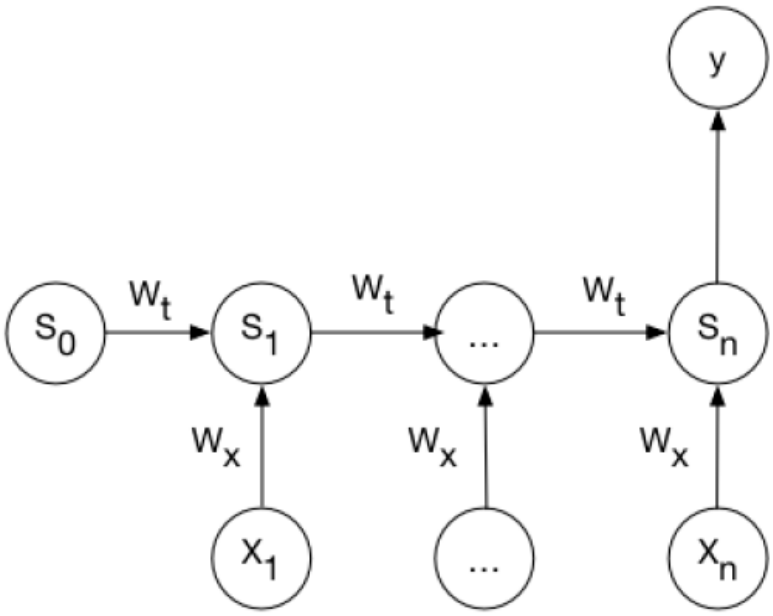

Figura 3.2: Representação de RNN simples desenrolado
Fonte: (Raaijmakers, 2022)

Apesar de suas habilidades, as RNNs simples sofrem com o fato de não conseguirem lidar com contextos mais longos, que geralmente é o caso de entradas de texto (Vajjala et al., 2020). O modelo de *Long Short Term Memory* (LSTM), introduzido por Hochreiter e Schmidhuber (1997), é uma variação das RNNs que atenua esse problema. A ideia dos LSTMs é lembrar apenas as partes do contexto necessárias para resolver a tarefa atual, deixando de lado o contexto irrelevante, assim amenizando a carga de guardar um contexto muito longo em uma representação vetorial (Vajjala et al., 2020). O LSTM possui três portas para administrar os dados de entrada: *forget gate*, *input gate* e *output gate*. As portas utilizam como entrada o estado oculto anterior e os dados de entrada da interação. O *input gate* indica quantos dados novos serão incorporados à nova memória. O *forget gate* tem como função calcular quantos dados da célula de memória anterior serão esquecidos. Ainda, cada unidade em LSTM possui uma célula de memória candidata, que usa a mesma entrada das portas e calcula a memória candidata. O modelo combina a saída do *input gate*, do **forget gate** e da célula de memória candidata para decidir o que irá lembrar. Por fim, utiliza-se do *input gate* para produzir o novo estado oculto da unidade. Assim é emitido pelo neurônio a nova memória e o novo estado oculto (Lin et al., 2019).

3.2 Word2Vec

A metodologia desenvolvida por Mikolov et al. (2013a) é a mais famosa para computar representações vetoriais contínuas de palavras. Bengio et al. (2003) foi o primeiro a utilizar redes neurais em dados de texto, desenvolvendo a representação distribuída *embeddings*. Mas foi através do algoritmo elaborado por Mikolov a popularização do método, devido ao fato de ter obtido resultados significativos tendo um custo computacional mais baixo. O propósito geral dele é, a partir do grande *corpus* Γ estabelecer *embeddings* para as palavras presentes no vocabulário V através das

palavras próximas, ou seja, os contextos em que elas se encontram (Vajjala et al., 2020).

Mikolov et al. (2013a) propõe duas arquiteturas para a estimação dos *word embeddings* (Figura 3.3), na esquerda o *Continuous bag of words* (CBoW) que tem como entrada o contexto e como saída palavra central $word_c$. Na direita está a arquitetura do *Skip-gram* que tem como objetivo predizer o contexto (seu output) com base na palavra central $word_c$. Ambas técnicas utilizam de redes neurais com uma camada oculta, possuem uma janela pré-definida e fixa k, que no caso da Figura 3.3 é 2, a palavra central $word_c$ e o chamado contexto $\{word_{c-k}, \ldots, word_{c-1}, word_{c+1}, \ldots, word_{c+k}\}$, que são as palavras ao redor da palavra central. O *input* de ambos os métodos é feito por meio de vetores *one-hot* das palavras. Ainda, o *output* são os vetores de pesos de saída que, por sua vez, são continuamente comparados com os vetores *one-hot* das palavras que o algoritmo busca prever a fim de treinar a rede.

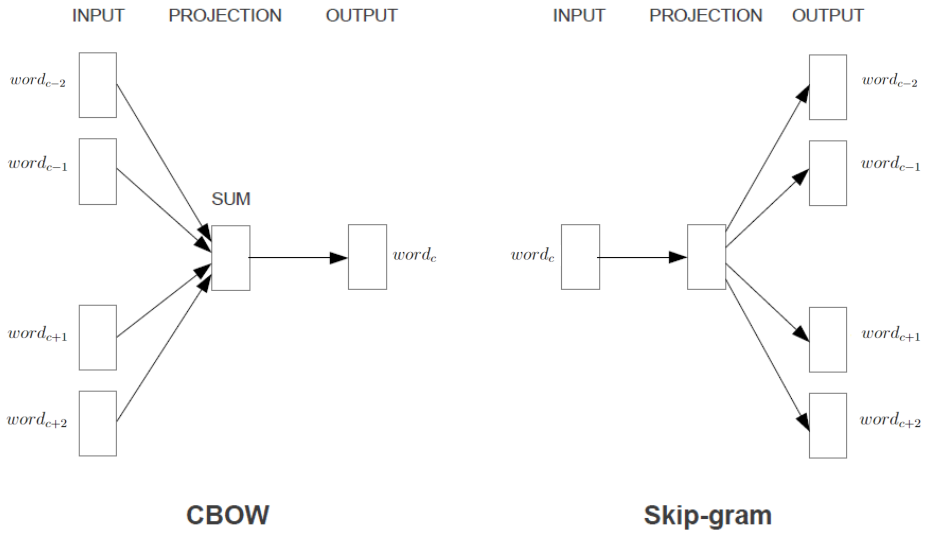

Figura 3.3: Arquiteturas do Word2Vec CBOW (esquerda) e Skip-gram (direita).
Fonte: (Mikolov et al., 2013a)

3.2.1 Arquitetura CBoW

Para demonstrar o funcionamento do CBoW a frase **"Eu adoro estudar estatística"** será usada como exemplo, além disso, será considerada como janela um valor de $k = 1$ e uma camada oculta de tamanho $N = 5$. O objetivo do algoritmo é prever a palavra central $word_c$ **"estudar"** através do contexto $(word_{c-1}, word_{c+1}) =$ ('adoro', 'estatística'). A arquitetura do exemplo pode ser ser vista na Figura 3.4

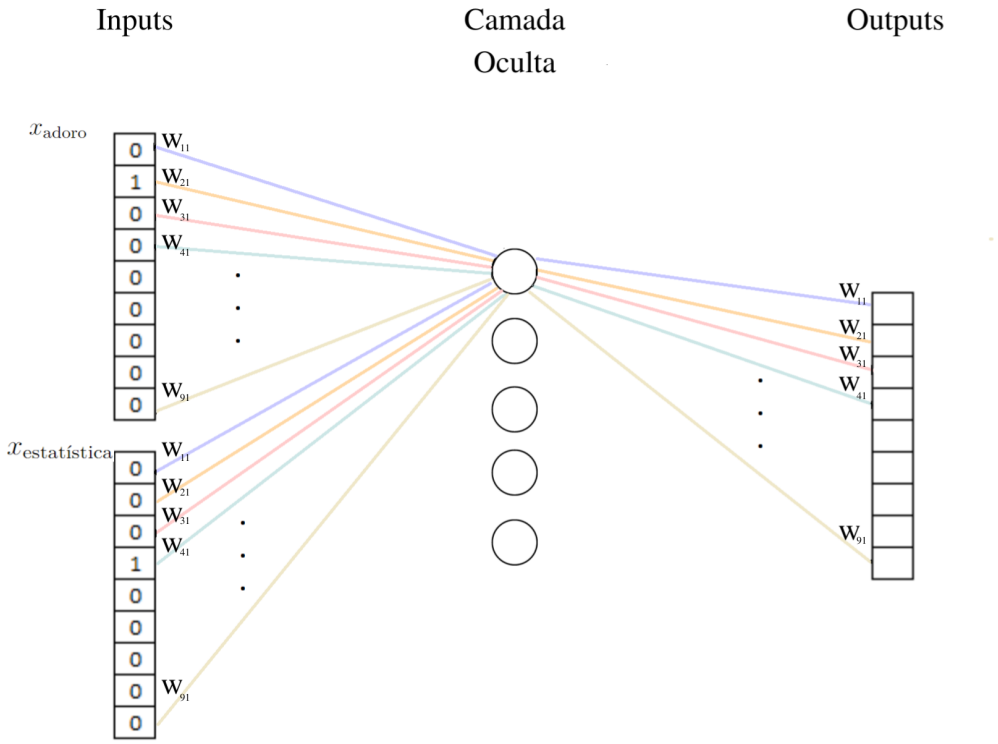

Figura 3.4: Exemplo de rede com a arquitetura CBoW.

O primeiro passo é entrar com os vetores *one-hot-encoder* das palavras de contexto na rede neural, como pode ser visto na Figura 3.4. Considerando que os índices de estudar e estatística no vocabulário sejam 2 e 4, respectivamente, e que $|V| = 9$ é o tamanho do vocabulário. Os vetores serão os seguintes:

$$x_{\text{adoro}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

e

$$x_{\text{estatística}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

A seguir, esses vetores multiplicam a matriz de pesos de entrada $W_{|V|\times N}$, em que N representa a quantidade de neurônios da camada oculta, portanto $W_{9\times 5}$, formada inicialmente por valores aleatórios. Na Figura 3.4 as ligações de cada item do vetor a cada neurônio é exemplificada com uma cor, pois cada valor de cada vetor *one-hot* de contexto irá multiplicar cada peso da matriz de entrada (não foram representadas as ligações de todos os neurônios para não poluir a imagem).

$$\begin{bmatrix} v_{\text{adoro}} \\ v_{\text{estatística}} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \\ w_{41} & w_{42} & w_{43} & w_{44} & w_{45} \\ w_{51} & w_{52} & w_{53} & w_{54} & w_{55} \\ w_{61} & w_{62} & w_{63} & w_{64} & w_{65} \\ w_{71} & w_{72} & w_{73} & w_{74} & w_{75} \\ w_{81} & w_{82} & w_{83} & w_{84} & w_{85} \\ w_{91} & w_{92} & w_{93} & w_{94} & w_{95} \end{bmatrix}.$$

Obtendo os vetores v_{adoro} e $v_{\text{estatística}}$ de dimensão= 5, o mesmo tamanho da quantidade de neurônios da camada oculta:

$$\begin{bmatrix} v_{\text{adoro}} \\ v_{\text{estatística}} \end{bmatrix} = \begin{bmatrix} w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{41} & w_{42} & w_{43} & w_{44} & w_{45} \end{bmatrix}.$$

Então, é feita a média desses vetores, visto que nesse método a saída é apenas um vetor, pois buscamos prever apenas a palavra central $word_c$ = 'estudar'.

$$\hat{v} = \frac{\begin{bmatrix} w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \end{bmatrix} + \begin{bmatrix} w_{41} & w_{42} & w_{43} & w_{44} & w_{45} \end{bmatrix}}{2} = \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \end{bmatrix}.$$

Após, é realizada a multiplicação de $\hat{v}$ pela matriz de pesos de saída $W'_{5\times9}$, que é a matriz de pesos de entrada $W_{9\times5}$ transposta:

$$z = \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \end{bmatrix} \times \begin{bmatrix} w_{11} & w_{21} & w_{31} & w_{41} & w_{51} & w_{61} & w_{71} & w_{81} & w_{91} \\ w_{12} & w_{22} & w_{32} & w_{42} & w_{52} & w_{62} & w_{72} & w_{82} & w_{92} \\ w_{13} & w_{23} & w_{33} & w_{43} & w_{53} & w_{63} & w_{73} & w_{83} & w_{93} \\ w_{14} & w_{24} & w_{34} & w_{44} & w_{54} & w_{64} & w_{74} & w_{84} & w_{94} \\ w_{15} & w_{25} & w_{35} & w_{45} & w_{55} & w_{65} & w_{75} & w_{85} & w_{95} \end{bmatrix}.$$

O resultado é um vetor de tamanho 9, a mesma dimensão do vocabulário V,

$$z = \begin{bmatrix} z_1 & z_2 & z_3 & z_4 & z_5 & z_6 & z_7 & z_8 & z_9 \end{bmatrix}.$$

O próximo passo é transformar o vetor z em probabilidades utilizando a regressão logística multinomial (McCullagh e Nelder, 2019), conhecida como função de ativação softmax, dada por

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{l=1}^{|V|} e^{z_l}}. \tag{3.2}$$

Para treinar o algoritmo e ele aprender as matrizes de pesos de entrada e de saída utiliza-se a função de otimização dada por:

$$\begin{aligned} \text{minimize } J &= -\log P(word_c | word_{c-1}, word_{c+1}) \\ &= -\log P(z_c | \hat{v}) \\ &= \log \frac{exp(z_c^T)}{\sum_{j=1}^{|V|} exp(z_j^T \hat{v})} \\ &= -z_c^T \hat{v} + \log \sum_{j=1}^{|V|} exp(z_j^T \hat{v}). \end{aligned} \tag{3.3}$$

Então utiliza-se gradiente descendente estocástico para atualizar todos os vetores z_c e v_j.

3.2.2 Arquitetura Skip-gram

Para exemplificar o Skip-gram será utilizada a mesma frase "Eu adoro estudar estatística". Ainda, serão utilizados os mesmos parâmetros, mas nesse caso se prevê o contexto $(word_{c-1}, word_{c+1}) = (\text{'adoro'}, \text{'estatística'})$ com base na palavra central $word_c = \text{'estudar'}$, como é possível analisar na Figura 3.5.

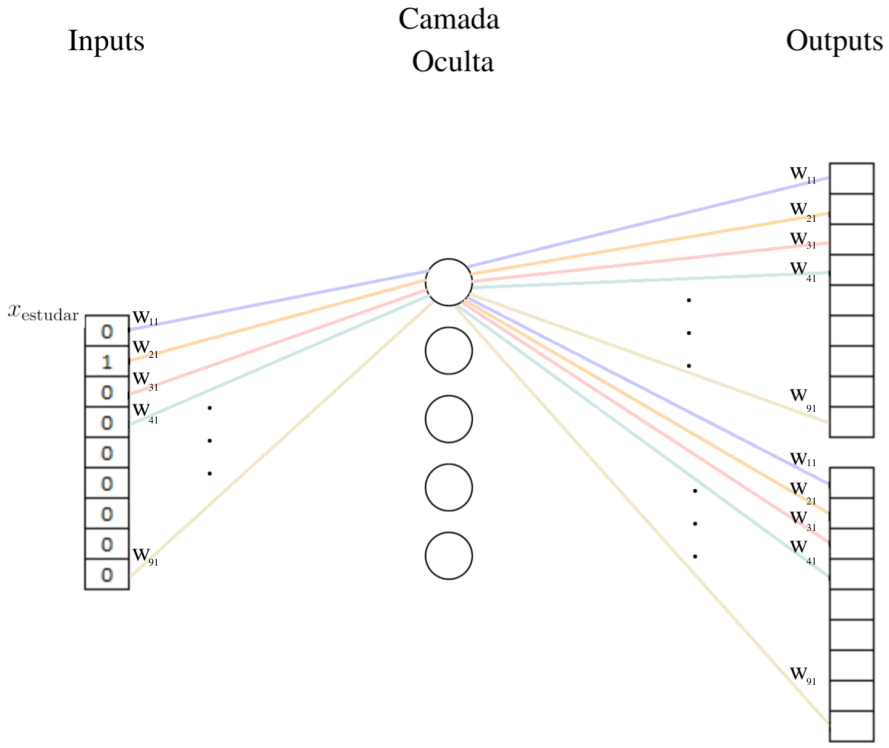

Figura 3.5: Exemplo Skip-gram

O *input* da rede neural nessa arquitetura é o vetor *one-hot* da palavra central 'estudar' cujo índice no vocabulário é 3,

$$x_{\text{estudar}} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

que multiplicará a matriz de peso de entrada $W_{9\times5}$:

$$[v_{\text{estudar}}] = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \\ w_{41} & w_{42} & w_{43} & w_{44} & w_{45} \\ w_{51} & w_{52} & w_{53} & w_{54} & w_{55} \\ w_{61} & w_{62} & w_{63} & w_{64} & w_{65} \\ w_{71} & w_{72} & w_{73} & w_{74} & w_{75} \\ w_{81} & w_{82} & w_{83} & w_{84} & w_{85} \\ w_{91} & w_{92} & w_{93} & w_{94} & w_{95} \end{bmatrix}.$$

Em seguida multiplicamos o vetor de pesos de entrada $v_{\text{estudar}} = \begin{bmatrix} w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \end{bmatrix}$,

que possui a mesma dimensão da quantidade de neurônios da camada oculta, pela matriz de pesos de saída $W_{5\times9}$:

$$z = \begin{bmatrix} w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \end{bmatrix} \times \begin{bmatrix} w_{11} & w_{21} & w_{31} & w_{41} & w_{51} & w_{61} & w_{71} & w_{81} & w_{91} \\ w_{12} & w_{22} & w_{32} & w_{42} & w_{52} & w_{62} & w_{72} & w_{82} & w_{92} \\ w_{13} & w_{23} & w_{33} & w_{43} & w_{53} & w_{63} & w_{73} & w_{83} & w_{93} \\ w_{14} & w_{24} & w_{34} & w_{44} & w_{54} & w_{64} & w_{74} & w_{84} & w_{94} \\ w_{15} & w_{25} & w_{35} & w_{45} & w_{55} & w_{65} & w_{75} & w_{85} & w_{95} \end{bmatrix}.$$

Obtendo um vetor com o mesmo tamanho do vocabulário V, ou seja, de dimensão $= 9$,

$$z = \begin{bmatrix} z_1 & z_2 & z_3 & z_4 & z_5 & z_6 & z_7 & z_8 & z_9 \end{bmatrix}.$$

Esse vetor de saída z é transformado em um vetor de probabilidades y através da Equação 3.2. Assim como no CBoW, será utilizada uma função de otimização para treinar a rede neural e obter as matrizes de peso. Nesse caso, recorre-se a uma suposição de Naive Bayes para quebrar as probabilidades, assim, dada a palavra central, todas as palavras de contexto são independentes. Dessa forma, a equação é dada por (Manning et al., 2019):

$$\begin{aligned} \text{minimize } J &= -\log P(word_{c-1}, word_{c+1}|word_c) \\ &= -\log \prod_{j=0, j\neq k}^{2k} P(word_{c-k+j|word_c}) \\ &= -\log \prod_{j=0, j\neq k}^{2k} P(z_{c-k+j|v_c}) \\ &= -\log \prod_{j=0, j\neq k}^{2k} \frac{exp(z_{c-k+j}^T v_c)}{\sum_{q=1}^{|V|} exp(z_q^T v_c)} \\ &= -\sum_{j=0, j\neq k}^{2k} z_{c-k+j}^T v_c + 2k \log \sum_{q=1}^{|V|} exp(z_q^T v_c). \end{aligned} \tag{3.4}$$

A partir dessa função objetivo é possível calcular os gradientes em relação aos parâmetros desconhecidos e atualizá-los através do Gradiente Descendente Estocástico.

3.2.3 *Hierarchical Softmax*

O softmax hieráquico foi inserido no contexto de redes neurais por Morin e Bengio (2005) e foi apresentado em Mikolov et al. (2013b) como uma alternativa ao softmax no caso do *Word2Vec*. Esse método utiliza uma representação em árvore binária da camada de saída.Cada folha da árvore é uma palavra, havendo um caminho único da raiz à cada folha (Mikolov et al., 2013b).

A probabilidade de uma palavra *word* dado um vetor $word_i$, $P(word|word_i)$, é igual a probabilidade de um passeio aleatório que começa na raiz da árvore e acaba no nó folha que representa a palavra *word*. Essa probabilidade é dada por Mikolov et al. (2013b):

$$P(word|word_i) = \prod_{j=1}^{L(word)-1} \sigma\Big([n(word, j+1) = ch(n(word, j))].\upsilon'_{n(word,j)}\upsilon_{word_i}\Big), \tag{3.5}$$

sendo $n(word, j)$ o j-ésimo nó no caminho da raiz ao nó que representa a palavra $word$ e $L(word)$ o comprimento desse caminho, então $n(word, 1) = raiz$ e $n(word, L(word)) = word$. Ainda, para qualquer nó interno n, $ch(n)$ é um filho fixo arbitrário de n e σ representa a função sigmóide dada por:

$$\sigma(x) = \frac{1}{(1 + \exp(-x))}. \tag{3.6}$$

Em Mikolov et al. (2013b) é citado que a estrutura de árvore utilizada influencia no desempenho obtido, os autores sugerem aplicar o algoritmo da árvore binária de Huffman, que demanda menos tempo de treinamento.

3.2.4 *Negative Sampling*

Quanto maior o tamanho do vocabulário $|V|$ e da camada oculta N maior é a matriz de pesos W, consequentemente, mais alto é o custo computacional envolvido na minimização das funções objetivos 3.3 e 3.4. Para solucionar esse problema, Mikolov et al. (2013b) propôs que as funções fossem aproximadas, para isso em cada etapa do treinamento, no lugar de percorrer todo o vocabulário são selecionadas as palavras que farão parte da estimação. A nova função objetivo tem o propósito de maximizar a probabilidade de uma palavra e contexto fazerem parte do *corpus*.

Para o $CBoW$ Manning et al. (2019) definiu a nova função objetivo como

$$-\log \sigma(z_c^T \hat{v}) - \sum_{q=1}^{Q} \log \sigma(-\tilde{z}_q^T \hat{v}). \tag{3.7}$$

Já para o Skip-gram a função objetivo com *negative sampling* é dada

$$-\log \sigma(z_{c-k+j}^T \upsilon_c) - \sum_{q=1}^{Q} \log \sigma(-\tilde{z}_q^T \upsilon_c). \tag{3.8}$$

A função σ é a mesma apresentada na equação 3.6.

3.2.5 GloVe

Modelo criado em 2013 por pesquisadores de *Stanford* para representações distribuídas de palavras. O nome significa *Global Vectors*, pois as estatísticas globais do *corpus* são registradas diretamente pelo modelo (Pennington et al., 2014). Essa técnica tem como objetivo aproveitar as informações estatísticas do *corpus*, através da matriz de coocorrência global, e também possuir bom desempenho na tarefa de analogia entre palavras e análise semântica, através dos vetores (Pennington et al., 2014). Essa combinação permitiu que o modelo obtivesse melhor performance semântica e sintática.

3.2.6 BERT

O *Bidirectional Encoder Representations from Transformers*, o BERT (Devlin et al., 2018), desenvolvido por pesquisadores do Google, utiliza o algoritmo *Transformers*, desenvolvido por Vaswani et al. (2017).O *Transformers* é formado por codificadores e decodificadores, que permitem que sequências de dados sejam processados sem ordem fixa.

A principal característica do modelo BERT é a bidirecionalidade, pois até então, os algoritmos liam as palavras de forma sequencial. A sua aplicação é divida em duas partes: o pré-treinamento e o ajuste fino. O BERT utiliza duas estratégias de treinamento:o *Masked Language Model* (MLM) que mascara aleatoriamente alguns dos *tokens* da entrada, buscando prever o ID do vocabulário original da palavra mascarada dado o contexto, o que permite a bidirecionalidade do modelo e o *Next Sentence Prediction* que, como o nome já diz, tem como objetivo prever a próxima frase, assim, pré-treina conjuntamente representações de pares de texto (Devlin et al., 2018). Em seguida aplica-se o *fine tuning*, para o algoritmo realizar a tarefa desejada. Esse algoritmo utiliza técnicas mais complexas que os demais, assim como também necessita de mais armazenamento e processamento, o que traz uma dificuldade maior de uso.

4 Resultados

A primeira seção desse capítulo aborda as especificidades do banco de dados a ser analisado, além das técnicas utilizadas no pré-processamento dos dados. Na segunda parte serão apresentados os resultados, com gráficos ilustrando a similaridade entre palavras dado determinado tópico pré selecionado. A análise foi desenvolvida utilizando duas linguagens de programação, o **R** versão 4.0.4 junto do *RStudio* como ambiente de desenvolvimento e a linguagem **Python** em conjunto com o Jupyter.

4.1 Banco de dados

O banco de dados é composto por notas fiscais eletrônicas (NF-e) de identificação de transações (vendas) de diferentes produtos com destino à estabelecimentos (CNPJs) localizados no estado do Rio Grande do Sul, a base é gerada pelo Sistema da Nota Fiscal Eletrônica e não contém transações feitas à pessoas físicas. As NF-e‘s contêm informações sigilosas, disponibilizadas pelo Tesouro, uma das três áreas da Secretaria da Fazenda do Estado do Rio Grande do Sul. Apesar do banco possuir informações confidenciais, nenhum dos resultados ou dados apresentados possuem informação sensível.

As NF-e‘s são compostas por dados estruturados como: preço, quantidade, região de origem do produto, região de destino do produto, CNPJ codificado, entre outros. O dado não estruturado é a descrição do produto, um campo aberto preenchido pela empresa que vende a mercadoria, a única limitação dessa variável é o número de espaços que permite no máximo 120 caracteres. A Figura 4.1 exemplifica algumas das descrições encontradas nos documentos.

ID	Descrição das notas
1	CARNE TEMPERADA RESFRIADA DE SUINO COM OSSO - COSTELA
2	CARNE TEMPERADA COZIDA RESFRIADA DE SUINO COM OSSO COM MOLHO - COSTELA
3	CARNE RESFRIADA DE BOVINO COM OSSO COSTELA DO TRASEIRO (JANELA) 39
4	CARNE RESF. BOV. C/OSSO - COSTELA DO TRASEIRO
5	CARNE RESF. BOV. S/OSSO - FILE DE COSTELA (ENTRECORTE)
6	Carne Resfriada de Bovino Com Osso - COSTELA
7	Carne Resfriada de Suino Com Osso (Costela)
8	CARNE TEMP. CONG. DE SUINO S/ OSSO - COSTELA C/ PELE - DALIA
9	CARNE TEMPERADA RESFRIADA DE SUINO SEM OSSO - FILE DE COSTELA

Figura 4.1: Exemplo de 9 diferentes documentos indicando as descrições de produtos contidos nas notas fiscais.

A análise das descrições das notas fiscais possuem diversos desafios, presentes na análise de dados de linguagem natural, que são comuns devido à falta de padronização no preenchimento desse campo aberto. Especificamente, quando lidamos com descrição de produtos, é comum que algumas palavras apresentem várias versões diferentes de abreviações, por exemplo, o *token* gramas pode ser escrito como: **gramas, gr, gra, g, etc**. Outra dificuldade é a identificação de fracionamento do produto e identificação de marcas, que indicam palavras que não fazem parte do vocabulário. Além dessas questões, o limite de caracteres torna os textos mais curtos, fazendo com o que os documentos d que compõem o *corpus* Γ sejam formados por uma pequena quantidade de *tokens*, demandando uma análise específica que leve em consideração esse tipo de característica.

Como o banco de dados das notas fiscais é formado por uma gama gigantesca de mercadorias, foram selecionadas bases referentes a apenas dois tipos de produtos, que possuem diferentes características de preenchimento e diferentes tamanhos de *corpus*.

O banco de dados relativo ao produto **leite** é constituído de uma amostra de 184.119 NF-e‘s que contenham conjuntamente as palavras leite, integral e pó. As 184 mil notas representam o *corpus* Γ e cada nota retrata um documento d. A outra base de dados é relativa à produtos de **carne bovina**, cada uma das notas desse tópico possuem simultaneamente as palavras carne, osso e costela. Esse *corpus* é formado por 41.595 documentos e representa um cenário diferente do encontrado para o tópico **leite**.

A única informação utilizada na análise é a descrição dos produtos. Na subseção seguinte será apresentado o pré-processamento dos dados.

4.1.1 Pré-processamento

O pré-processamento é uma fase importante ao se trabalhar com dados de texto, neste trabalho foram efetuados os seguintes procedimentos através dos pacotes *dplyr*, *tm* e *stringr* da linguagem *R*:

- Transformação dos caracteres em minúsculos;
- Remoção de acentos, pontuações e símbolos;
- Remoção de stopwords em português.

Após a limpeza dos dados é feita a *tokenização* do texto, ou seja, a transformação das palavras em *tokens*. Nessa etapa foi utilizada a função *word_tokenize* do pacote *nltk.tokenize*, na linguagem *Python*. O *corpus*, portanto, é formado por uma frase em cada linha, onde as palavras são *tokenizadas*. Um exemplo pode ser visto a seguir:

- No documento:

  LEITE EM PO DANBY INTEGRAL INST 1 KG,

  LEITE EM PO INTEGRAL SANTA CLARA 1 KG SC.

- No corpus:

  [’leite’, ’po’, ’danby’, ’integral’, ’inst’, ’1’, ’kg’],,

['leite', 'po', 'integral', 'santa', 'clara', '1', 'kg', 'sc'].

Como a base é formada por notas fiscais é comum observar as mesmas descrições várias vezes, logo alguns dos 184.119 documentos do tópico **leite** e 41.595 do tópico **carne** serão iguais. Como na literatura há referencias de que a retirada de documentos duplicados no *corpus* pode melhorar a performance dos *word embeddings*, optou-se por avaliar dois cenários. Em um cenário todas as notas fiscais são consideradas, em outro as notas repetidas são retiradas da base, acarretando a diminuição das descrições do produto **leite** para 2293 documentos e do tópico **carne** para 1394 documentos.

4.2 Continuous bag of words (CBoW)

Com o pré-processamento e a *tokenização* prontos, foram ajustados dois modelos CBoW através da função Word2Vec do pacote *gensim* do *Python*.

Na aplicação dos algoritmos é necessário especificar *a priori* os hiperparâmetros do modelo, a janela de contexto e o tamanho do dicionário, que indica o número mínimo de vezes que a palavra aparece no *corpus*, palavras pouco frequentes, inferiores à quantidade escolhida são desconsideradas. O valor padrão utilizado no pacote *gensim* para k é 5, mas no geral utiliza-se um valor fixo arbitrário (Caselles-Dupré et al., 2018) Testou-se modelos com duas dimensões de dicionário 10 e 30, além de três tamanhos de janela de contexto $k = (2, 3, 5)$, os mesmos hiperparâmetros foram implementados para o Skip-gram.

Os resultados obtidos com os diferentes hiperparâmetros foram similares, optou-se por apresentar palavras que apareciam mais de 30 vezes no *corpus* e com tamanho do hiperparâmetro janela de contexto $k = 3$. Os demais resultados estão disponíveis nos anexos. A seguir os resultados para os produtos referentes ao tópico **leite**.

4.2.1 Leite

Devido ao fato dos vetores *embeddings* possuírem grande dimensão, utilizou-se a técnica de redução de dimensionalidade t-SNE(Van der Maaten e Hinton, 2008) com perplexicidade = 15, para visualizar os vetores por meio de gráficos 2D. O mesmo foi utilizado nas exposições gráficas dos *embeddings* do Skip-gram.

As palavras mais similares a italac (roxo), leite (azul), inst (azul claro), po (verde), 400g (laranja) e integral (vermelho) estão representadas por cores nos gráficos da Figura 4.2 para descrições únicas e duplicadas. No segundo gráfico nota-se que as palavras similares a italac são nomes de outras marcas da indústria de laticínios. Também é possível perceber como as diversas abreviações e erros de escrita da palavra instantâneo ficaram próximos na representação. Ainda, é necessário ressaltar que as palavras mais similares à 400g são relacionadas ao fracionamento do produto, como 400gr, 200g e kg. Embora alguns termos mais próximos a 400g no gráfico das descrições únicas sejam relativos

ao fracionamento, a representação das demais expressões não foram tão bem ajustadas.

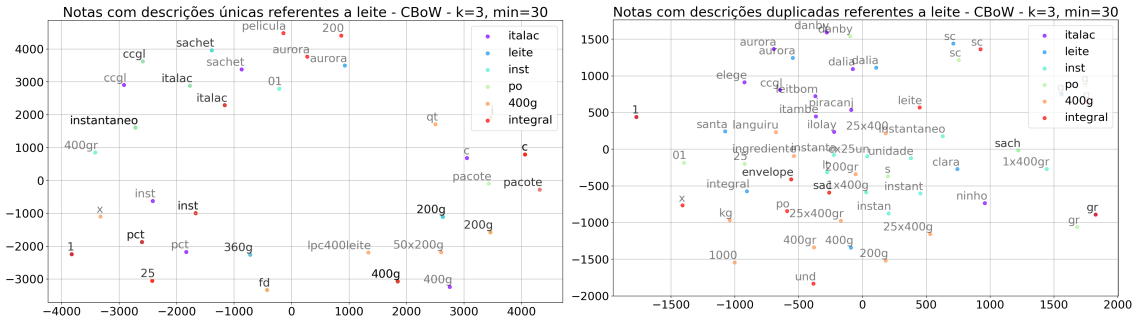

Figura 4.2: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a italac, leite, inst, po, 400g e integral, obtidas através do banco com descrições únicas utilizando o modelo CBoW. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

A semelhança entre as palavras também será avaliada através de tabelas, visto que a representação gráfica indica a similaridade através de duas dimensões, quando na verdade o tamanho dos vetores de cada palavra é muito maior. A Tabela 4.1 mostra as dez palavras mais similares à italac e 400g comparando o banco com descrições únicas e repetidas. O modelo com descrições repetidas conseguiu capturar as semelhanças entre as marcas e os fracionamentos das mercadorias. O ajuste com as descrições únicas não obteve resultados tão satisfatórios, mas ainda conseguiu registrar a proximidade de italac com as marcas tirol e ccgl. Ainda, com relação a 400g o modelo com descrições únicas obteve mais palavras associadas ao fracionamento, como 200g, 400gr e pct que se refere a pacote. O fato do banco de dados com descrições repetidas ter obtido resultados melhores talvez se deva ao tamanho do banco e pelas especificações dele, em outro contexto e com um número maior de documentos eventualmente os resultados seriam diferentes.

Tabela 4.1: Dez palavras mais similares à italac e 400g considerando descrições únicas e repetidas utilizando o modelo CBoW.

italac				**400g**			
Descrições únicas		**Descrições repetidas**		**Descrições únicas**		**Descrições repetidas**	
Palavra	**Similaridade**	**Palavra**	**Similaridade**	**Palavra**	**Similaridade**	**Palavra**	**Similaridade**
i	0.9988	danby	0.7199	i	0.9990	400gr	0.5660
400g	0.9987	aurora	0.7182	inst	0.9989	200g	0.5163
pct	0.9987	itambe	0.6896	200g	0.9989	200gr	0.3890
ccgl	0.9986	leitbom	0.6740	400gr	0.9989	25x400gr	0.3811
c	0.9986	ccgl	0.5878	tirol	0.9988	350g	0.3259
fd	0.9986	ninho	0.5833	pct	0.9988	25x400	0.3194
pacote	0.9986	elege	0.5556	c	0.9988	400	0.3057
qt	0.9986	dalia	0.5439	25x400g	0.9987	cxa	0.3042
inst	0.9985	piracanj	0.5227	qt	0.9987	kg	0.3036
tirol	0.9985	ilolay	0.4668	ccgl	0.9987	25x400g	0.3026

4.2.2 Carne

A Figura 4.3 indica a representação dos embeddings das palavras mais similares para o tópico carne para os *corpus* de descrições únicas e duplicadas utilizando o modelo CBoW. Era esperado que as palavras mais próximas de costela fossem outros tipos de cortes de carne, o que acontece com janela e com entrecote no gráfico à esquerda, mas não se repete no gráfico referente às notas duplicadas. Embora tenham conseguido capturar alguma analogia as representações não obtiveram o resultado esperado, o que pode ser devido ao número de documentos do banco de dado do tópico **carne**.

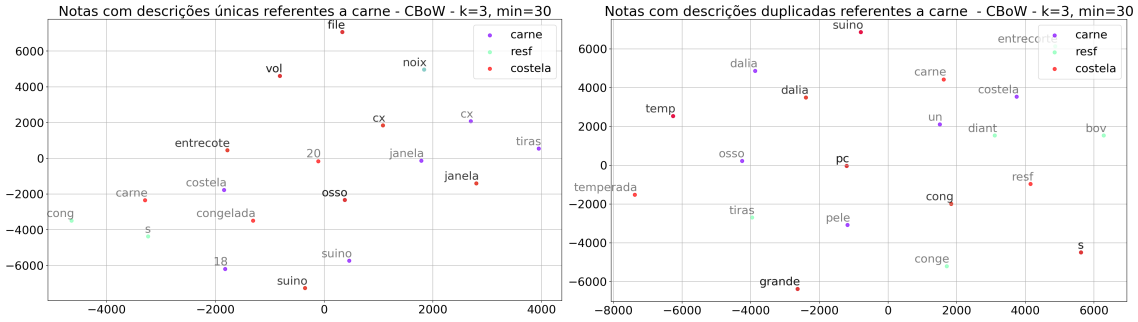

Figura 4.3: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a carne, resf e costela, obtidas através do banco com descrições únicas utilizando o modelo CBoW. Gráfico à direita retrata a mesma representação considerando descrições repetidas.

A Tabela 4.2 apresenta as dez palavras mais similares à costela e à abreviação de congelado, cong, para notas com escritas únicas ou duplicadas. Como já relatado anteriormente, era esperado que costela estivesse associada aos demais cortes de carne, mas as palavras mais semelhantes mesmo que relacionadas ao tópico em questão não exprimiam essa similaridade, independentemente do tipo de descrição utilizada. Em relação a cong, palavras como conge e res foram assinaladas como similares para o modelo com descrições repetidas e resf foi indicada como semelhante em ambas as descrições, como era previsto. No entanto, a maioria das palavras, como noix e pele, foram apontadas como similares pelos modelos sem possuir afinidade com cong.

Tabela 4.2: Dez palavras mais similares à costela e cong, considerando descrições únicas e repetidas utilizando o modelo CBoW.

costela				cong			
Descrições únicas		Descrições repetidas		Descrições únicas		Descrições repetidas	
Palavra	Similaridade	Palavra	Similaridade	Palavra	Similaridade	Palavra	Similaridade
carne	0.9983	carne	0.5085	cx	0.9985	resf	0.6863
cx	0.9978	un	0.2842	resf	0.9985	pc	0.5691
congelada	0.9975	resf	0.2507	c	0.9983	grande	0.5375
resf	0.9974	resfriada	0.2138	file	0.9982	dalia	0.5318
file	0.9974	care	0.1828	noix	0.9981	pele	0.4826
traseiro	0.9973	kg	0.1803	s	0.9981	res	0.4809
suino	0.9971	s	0.1791	qtd	0.9981	salgada	0.4498
bovino	0.9970	cong	0.1680	bov	0.9979	conge	0.4230
osso	0.9970	temperada	0.1673	suino	0.9979	c	0.4140
tiras	0.9969	osso	0.1662	osso	0.9978	cm	0.4080

4.3 Skip-gram

Neste espaço serão expostas as análises feitas com a arquitetura de Skip-gram. Foram utilizadas janelas de tamanho k igual a 3 e consideradas apenas palavras que apareciam em pelo menos 30 documentos.

4.3.1 Leite

A Figura 4.4 retrata as palavras mais semelhantes à italac, leite, inst, po, 400g e integral considerando os dois tipos de descrições já descritos e o modelo Skip-gram, que utiliza da palavra central para prever as palavras ao redor. O gráfico referente ao banco com descrições únicas conseguiu associar algumas marcas à italac como dalia e piracanjuba, por exemplo. O gráfico à direita também conseguiu ligar italac as demais marcas. Ainda, é notável que o gráfico com descrições repetidas associou com êxito os fracionamentos.

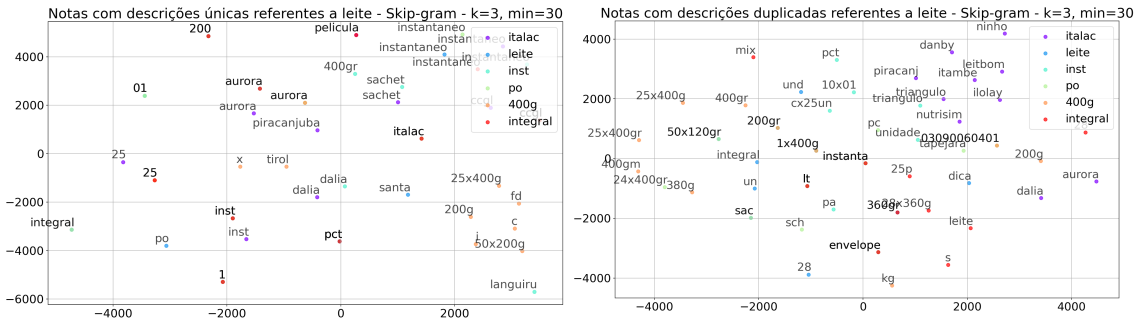

Figura 4.4: Gráfico à esquerda representa em duas dimensões as palavras mais similares a italac, leite, inst, po, 400g e integral, obtidas através do banco com descrições únicas utilizando o modelo Skip-gram. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

As palavras mais similares à italac e 400g estão representadas na Tabela 4.3.

O ajuste com descrições repetidas possui uma associação melhor de palavras, mas é notável que o outro modelo possui quatro nomes de marcas entre as dez palavras mais semelhantes à italac, são elas: ccgl, dalia, aurora, piracanjuba e quatro termos de fracionamento relacionados à 400g: 25x400g, 200g, 25 e 500x200g, o que indica que o ajuste com descrições únicas também conseguiu capturar semelhanças entre as palavras.

Tabela 4.3: Dez palavras mais similares à italac e 400g considerando descrições únicas e repetidas utilizando o modelo Skip-gram.

italac				**400g**			
Descrições únicas		**Descrições repetidas**		**Descrições únicas**		**Descrições repetidas**	
Palavra	**Similaridade**	**Palavra**	**Similaridade**	**Palavra**	**Similaridade**	**Palavra**	**Similaridade**
inst	0.9968	itambe	0.6827	25x400g	0.9967	200g	0.5274
ccgl	0.9967	leitbom	0.6758	200g	0.9965	200gr	0.5218
sachet	0.9967	triangulo	0.6242	tirol	0.9963	400gr	0.5039
dalia	0.9965	danby	0.5937	25	0.9963	25x400g	0.4668
aurora	0.9964	piracanj	0.5927	c	0.9962	kg	0.4667
piracanjuba	0.9964	ilolay	0.5756	x	0.9961	25x400gr	0.4619
25	0.9964	dalia	0.5714	i	0.9961	380g	0.4479
pelicula	0.9962	ninho	0.5601	50x200g	0.9961	400gm	0.4378
360g	0.9961	aurora	0.5565	fd	0.9959	pcte	0.4204
pct	0.9961	cabra	0.5357	aurora	0.9957	la	0.4200

4.3.2 Carne

A Figura 4.5 indica a representação dos *embeddings* em duas dimensões das palavras mais similares à carne, costela e resf para o tópico carne, considerando os corpus de descrições únicas e duplicadas utilizando o modelo CBoW. No primeiro gráfico destaca-se o registro da semelhança entre cortes de carne através da associação de entrecote, janela e file com costela. Já no gráfico à esquerda, referente às descrições duplicadas, salienta-se a observação de similaridade entre cong, conge e resf.

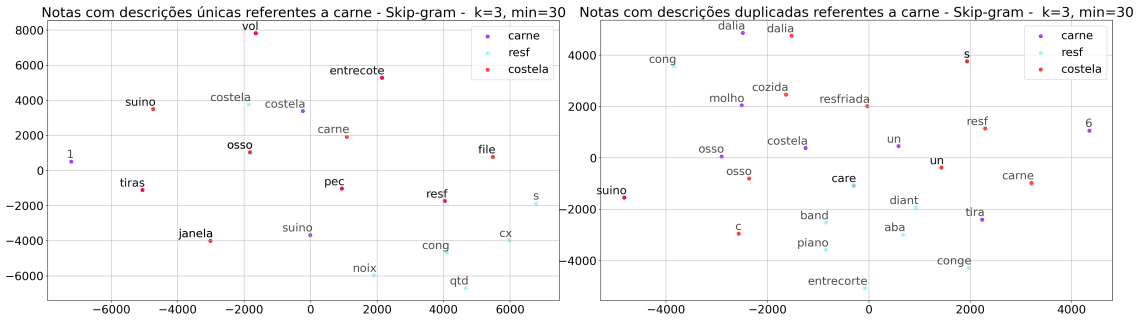

Figura 4.5: Gráfico à esquerda representa em duas dimensões as palavras mais similares à carne, resf e costela, obtidas através do banco com descrições únicas utilizando o modelo Skip-gram. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

A tabela 4.4 reforça o que já foi indicado pela Figura 4.5, ainda que haja relação entre algumas palavras, a maioria delas não possuem a compatibilidade

esperada. A maior parte dos termos são relacionados ao tópico carne, mas não necessariamente a palavra em questão.

Tabela 4.4: Dez palavras mais similares à costela e cong considerando descrições únicas e repetidas utilizando o modelo Skip-gram.

costela				**cong**			
Descrições únicas		**Descrições repetidas**		**Descrições únicas**		**Descrições repetidas**	
Palavra	**Similaridade**	**Palavra**	**Similaridade**	**Palavra**	**Similaridade**	**Palavra**	**Similaridade**
osso	0.9931	carne	0.4888	cx	0.9963	grande	0.6094
suino	0.9925	osso	0.4542	qtd	0.9948	pc	0.5852
carne	0.9921	carre	0.2928	c	0.9935	dalia	0.5769
vol	0.9877	c	0.2912	s	0.9902	pele	0.5401
pec	0.9846	resf	0.2870	resf	0.9879	conge	0.4949
tiras	0.9830	un	0.2824	noix	0.9873	salgada	0.4926
resf	0.9828	cozida	0.2631	bov	0.9841	temp	0.4791
entrecote	0.9825	diant	0.2631	file	0.9835	cm	0.4679
janela	0.9803	6	0.2604	janela	0.9774	resf	0.4571
file	0.9797	s	0.2593	pec	0.9725	5	0.4399

5 Conclusão

O presente trabalho apresentou representações vetoriais de palavras presentes em descrições de produtos contidos em notas fiscais do estado do Rio Grande do Sul referentes a dois tópicos: carne e leite. É importante ressaltar que os dados trazem consigo diversos desafios vinculados à textos curtos, frequentes quando os documentos possuem quantidade limitada de caracteres. Esse conjunto de características bem específicas, inerentes à dados de descrição de produtos, impossibilitando o uso de *word embeddings* pré-treinados disponíveis na literatura, já que estes são baseados em textos longos como reportagens e artigos do *Wikipédia*.

Em razão das características singulares do texto das notas fiscais, aplicou-se a técnica *Word2Vec* a uma base com descrições repetidas e outra em que estas foram retiradas. No banco referente ao tópico leite, a representação dos *word embeddings* utilizando as notas com descrições duplicadas pareceram obter resultados melhores, conseguindo agrupar marcas e capturar o fracionamento dos produtos, o que não aconteceu ao usar descrições sem repetição. Já na análise realizada na base de dados relativa ao tópico de carne não foi observada diferença tão consistente entre as notas com descrições únicas e repetidas. Ainda, as representações dos termos do tópico de carne não pareceram registrar tão bem as similaridades entre eles, o que pode ser devido à quantidade de documentos utilizados nos ajustes. Foram ajustados modelos com diferentes valores nos hiperparâmetros, mas eles obtiveram resultados muito próximos.

Concluí-se que a representação por *word embeddings* treinados diretamente em dados relativos a descrições de produtos é promissora para retratar as palavras das notas fiscais, pois conseguiu obter bons resultados para os exemplos abordados nesse trabalho. Ainda, a metodologia pode ser empregada em *marketplaces* e lojas de vendas online em que a descrição de mercadorias é recorrente, assim conseguindo filtrar produtos mais similares, fazendo recomendações úteis e específicas para cada usuário através do uso dos *embeddings* das palavras utilizadas na busca, por exemplo.

Considerando trabalhos futuros seria importante utilizar mais notas fiscais pertinentes a cada produto, assim como expandir a quantidade de produtos utilizados. Ainda, com maiores recursos computacionais, utilizar dados de notas fiscais no treinamento de métodos mais sofisticados, como o *GloVe* que usa a matriz de co-ocorrência e o *BERT*, que considera a ordem do texto, conhecido por ser um algoritmo "bidirecional".

Referências Bibliográficas

Acosta, J., Lamaute, N., Luo, M., Finkelstein, E., e Andreea, C. (2017). Sentiment analysis of twitter messages using word2vec. *Proceedings of student-faculty research day, CSIS, Pace University*, 7:1–7.

Athanasiou, V. e Maragoudakis, M. (2017). A novel, gradient boosting framework for sentiment analysis in languages where nlp resources are not plentiful: a case study for modern greek. *Algorithms*, 10(1):34.

Bengio, Y., Ducharme, R., Vincent, P., e Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Broder, A. Z., Glassman, S. C., Manasse, M. S., e Zweig, G. (1997). Syntactic clustering of the web. *Computer networks and ISDN systems*, 29(8-13):1157–1166.

Brown, P. F., Della Pietra, V. J., Desouza, P. V., Lai, J. C., e Mercer, R. L. (1992). Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–480.

Cambria, E. e White, B. (2014). Jumping nlp curves: A review of natural language processing research [review article]. *IEEE Computational Intelligence Magazine*, 9(2):48–57.

Caselles-Dupré, H., Lesaint, F., e Royo-Letelier, J. (2018). Word2vec applied to recommendation: Hyperparameters matter. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 352–356, New York, NY, USA. Association for Computing Machinery.

Chiang, D. (2007). Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

Deng, L. e Liu, Y. (2018). *Deep learning in natural language processing*. Springer.

Devika, M., Sunitha, C., e Ganesh, A. (2016). Sentiment analysis: A comparative study on different approaches. *Procedia Computer Science*, 87:44–49. Fourth International Conference on Recent Trends in Computer Science Engineering (ICRTCSE 2016).

Devlin, J., Chang, M.-W., Lee, K., e Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Goodfellow, I., Bengio, Y., e Courville, A. (2016). *Deep learning*. MIT press.

Harrington, P. (2012). *Machine learning in action*. Simon and Schuster.

Hochreiter, S. e Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hossain, M. S., Nayla, N., e Rassel, A. A. (2022). Product market demand analysis using nlp in banglish text with sentiment analysis and named entity recognition. In *2022 56th Annual Conference on Information Sciences and Systems (CISS)*, pages 166–171. IEEE.

James, G., Witten, D., Hastie, T., e Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer.

Jurafsky, D. e Martin, J. H. (2021a). N-gram language models. urlhttps://web.stanford.edu/ jurafsky/slp3/3.pdf.

Jurafsky, D. e Martin, J. H. (2021b). Naive bayes and sentiment classification. urlhttps://web.stanford.edu/ jurafsky/slp3/4.pdf.

Jurafsky, D. e Martin, J. H. (2021c). Vector semantics and embeddings. urlhttps://web.stanford.edu/ jurafsky/slp3/6.pdf.

Kandasamy, K. e Koroth, P. (2014). An integrated approach to spam classification on twitter using url analysis, natural language processing and machine learning techniques. In *2014 IEEE Students' Conference on Electrical, Electronics and Computer Science*, pages 1–5.

Kešelj, V., Peng, F., Cercone, N., e Thomas, C. (2003). N-gram-based author profiles for authorship attribution. In *Proceedings of the conference pacific association for computational linguistics, PACLING*, volume 3, pages 255–264.

Krogh, A. (2008). What are artificial neural networks? *Nature biotechnology*, 26(2):195–197.

LeCun, Y., Bengio, Y., e Hinton, G. (2015). Deep learning. *Nature*, 521:436–444.

Lin, H., Shi, X., Lausen, L., Zhang, A., He, H., Zha, S., e Smola, A. (2019). Dive into deep learning for natural language processing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): Tutorial Abstracts*, Hong Kong, China. Association for Computational Linguistics.

Lorena, A. C. e De Carvalho, A. C. (2007). Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, 14(2):43–67.

Manning, C. e Schutze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

Manning, R. S., Fang, G. G., Mundra, R., e Socher, R. (2019). Cs224n: Natural language processing with deep.

McCullagh, P. e Nelder, J. A. (2019). *Generalized linear models*. Routledge.

McCulloch, W. S. e Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Mikolov, T., Chen, K., Corrado, G., e Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., e Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Mitchell, T. M. e Mitchell, T. M. (1997). *Machine learning*, volume 1. McGraw-hill New York.

Morin, F. e Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *International workshop on artificial intelligence and statistics*, pages 246–252. PMLR.

Pascanu, R., Mikolov, T., e Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In Dasgupta, S. e McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA. PMLR.

Pennington, J., Socher, R., e Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Raaijmakers, S. (2022). *Deep Learning for Natural Language Processing*. Manning.

Ramos, J. et al. (2003). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. New Jersey, USA.

Rumelhart, D. E., Hinton, G. E., e Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Rydning, D. R.-J. G.-J., Reinsel, J., e Gantz, J. (2018). The digitization of the world from edge to core. *Framingham: International Data Corporation*, 16.

Salton, G. e Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Salton, G., Wong, A., e Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Schütze, H., Manning, C. D., e Raghavan, P. (2008). *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

Segaran, T. e Hammerbacher, J. (2009). *Beautiful Data: The Stories Behind Elegant Data Solutions*. Theory in practice. O'Reilly Media.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

Socher, R., Bauer, J., Manning, C. D., e Ng, A. Y. (2013a). Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., e Potts, C. (2013b). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Vajjala, S., Majumder, B., Gupta, A., e Surana, H. (2020). *Practical Natural Language Processing: A Comprehensive Guide to Building Real-world NLP Systems*. O'Reilly Media.

Van der Maaten, L. e Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., e Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Yan, X., Guo, J., Lan, Y., e Cheng, X. (2013). A biterm topic model for short texts. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1445–1456.

Anexos

CBoW

Leite

k=2, mínimo de palavras=10

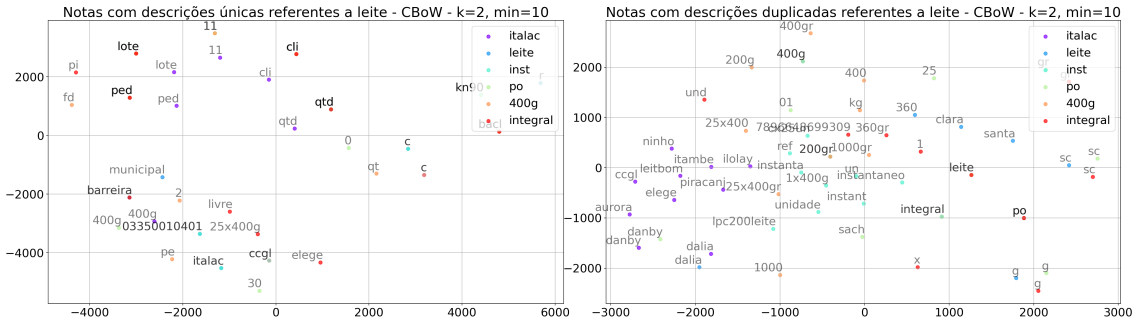

Figura 5.1: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a italac, leite, inst, po, 400g e integral, obtidas através do banco com descrições únicas utilizando o modelo CBoW, com k=2 e mínimo de palavras=10. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model.wv.most_similar('italac')
```

```
[('qtd', 0.9985781311988831),
 ('ped', 0.9984325170516968),
 ('bacl', 0.9984234571456909),
 ('lote', 0.9981926083564758),
 ('11', 0.9981444478034973),
 ('cli', 0.9981166124343872),
 ('r', 0.9981056451797485),
 ('400g', 0.9980681538581848),
 ('ccgl', 0.9980058670043945),
 ('c', 0.9979637861251831)]
```

```
model.wv.most_similar('italac')
```

```
[('danby', 0.7433639168739319),
 ('aurora', 0.7263697981834412),
 ('leitbom', 0.6969749331474304),
 ('itambe', 0.6919553875923157),
 ('ccgl', 0.5743505358695984),
 ('ninho', 0.5658228993415833),
 ('dalia', 0.5569595098495483),
 ('elege', 0.5520501732826233),
 ('ilolay', 0.5260247588157654),
 ('piracanj', 0.5111780166625977)]
```

```
model.wv.most_similar('400g')
```

```
[('qt', 0.9983846545219421),
 ('2', 0.9983721971511841),
 ('ped', 0.998338520526886),
 ('cli', 0.9983304738998413),
 ('c', 0.9983054995536804),
 ('lote', 0.9982946515083313),
 ('11', 0.9982900619506836),
 ('bacl', 0.9982681274414062),
 ('pe', 0.9982468485832214),
 ('fd', 0.9981966018676758)]
```

```
model.wv.most_similar('400g')
```

```
[('400gr', 0.5801095962524414),
 ('200g', 0.4998951554298401),
 ('25x400gr', 0.3998783528804779),
 ('1000gr', 0.3724437355995178),
 ('1000', 0.3307255506515503),
 ('200gr', 0.3212977349758148),
 ('400', 0.3125859797000885),
 ('po', 0.3111909329891205),
 ('25x400', 0.3052813410758972),
 ('kg', 0.2984739542074463)]
```

Figura 5.2: Saídas das 10 palavras mais similares a italac e 400g ajustadas através do modelo CBoW com k=2 e mínimo de palavras=10, considerando descrições únicas e duplicadas, respectivamente.

k=2, mínimo de palavras=30

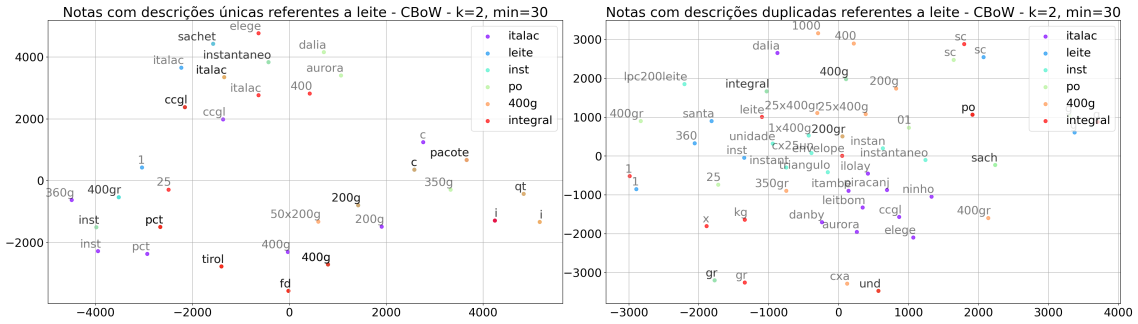

Figura 5.3: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a italac, leite, inst, po, 400g e integral, obtidas através do banco com descrições únicas utilizando o modelo CBoW, com k=2 e mínimo de palavras=30. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model.wv.most_similar('italac')
[('ccgl', 0.9976970553398132),
 ('400g', 0.9976260662078857),
 ('inst', 0.997592031955719),
 ('200g', 0.9975281357765198),
 ('pct', 0.9974644780158997),
 ('c', 0.9974374771118164),
 ('i', 0.9974169731140137),
 ('360g', 0.9974072575569153),
 ('sachet', 0.997294545173645),
 ('pacote', 0.9971904158592224)]
```

```
model.wv.most_similar('italac')
[('danby', 0.7263399958610535),
 ('aurora', 0.7154722809791565),
 ('itambe', 0.652317225933075),
 ('leitbom', 0.6389623880386353),
 ('ninho', 0.5711262226104736),
 ('elege', 0.5626891255378723),
 ('ccgl', 0.555997908115387),
 ('dalia', 0.5480556488037109),
 ('ilolay', 0.5196954011917114),
 ('piracanj', 0.5041142702102661)]
```

```
model.wv.most_similar('400g')
[('c', 0.9983635544776917),
 ('i', 0.9982756972312927),
 ('200g', 0.9981863498687744),
 ('qt', 0.9981397986412048),
 ('fd', 0.9980397820472717),
 ('50x200g', 0.9979283213615417),
 ('pct', 0.9977667331695557),
 ('tirol', 0.9977154731750488),
 ('italac', 0.997626006603241),
 ('pacote', 0.9976148009300232)]
```

```
model.wv.most_similar('400g')
[('400gr', 0.5440502762794495),
 ('200g', 0.4792003929615021),
 ('25x400gr', 0.3941911458969116),
 ('200gr', 0.3074856996536255),
 ('1000', 0.3024522364139557),
 ('25x400g', 0.29685866832733154),
 ('400', 0.29356053471565247),
 ('cxa', 0.2908402681350708),
 ('350gr', 0.27837231755256653),
 ('po', 0.27813097834587097)]
```

Figura 5.4: Saídas das 10 palavras mais similares a italac e 400g ajustadas através do modelo CBoW com k=2 e mínimo de palavras=30, considerando descrições únicas e duplicadas, respectivamente.

k=3, mínimo de palavras=10

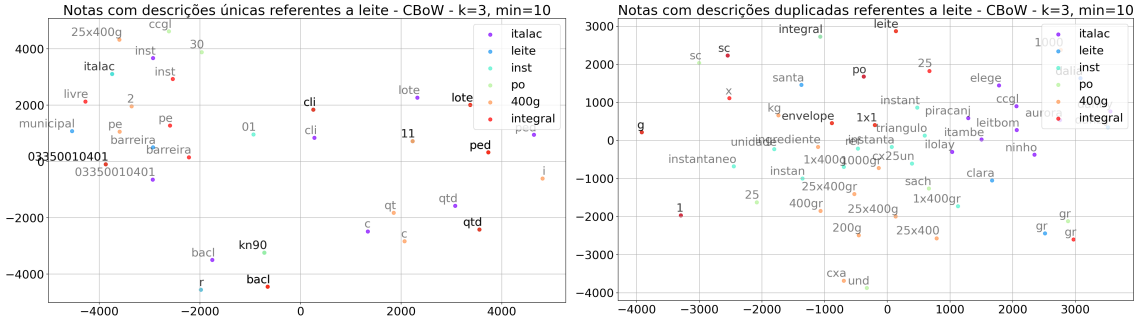

Figura 5.5: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a italac, leite, inst, po, 400g e integral, obtidas através do banco com descrições únicas utilizando o modelo CBoW, com k=3 e mínimo de palavras=10. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model.wv.most_similar('italac')
[('qtd', 0.9987131953239441),
 ('ped', 0.9986443519592285),
 ('bacl', 0.9986147880554199),
 ('lote', 0.9984583854675293),
 ('11', 0.998301088809967),
 ('cli', 0.9981916546821594),
 ('inst', 0.9981592297554016),
 ('03350010401', 0.9981139302253723),
 ('c', 0.998109877195276),
 ('r', 0.9981095194816589)]
```

```
model.wv.most_similar('italac')
[('danby', 0.7654082179069519),
 ('aurora', 0.7433123588562012),
 ('leitbom', 0.7085146307945251),
 ('itambe', 0.6967071890830994),
 ('dalia', 0.5979601740837097),
 ('ninho', 0.5797885060310364),
 ('ccgl', 0.5657677054405212),
 ('elege', 0.5375797748565674),
 ('piracanj', 0.5337328314781189),
 ('ilolay', 0.484075665473938)]
```

```
model.wv.most_similar('400g')
[('2', 0.9983033537864685),
 ('pe', 0.9982673525810242),
 ('ped', 0.9982092976570129),
 ('c', 0.9982085227966309),
 ('lote', 0.9981549978256226),
 ('qt', 0.9981313943862915),
 ('i', 0.9981300830841064),
 ('25x400g', 0.9981117844581604),
 ('11', 0.9980888962745667),
 ('bacl', 0.9980828166007996)]
```

```
model.wv.most_similar('400g')
[('400gr', 0.5607271790504456),
 ('200g', 0.4911058843135834),
 ('25x400gr', 0.4289669394493103),
 ('25x400g', 0.35446614027023315),
 ('kg', 0.3003881871700287),
 ('25x400', 0.2996869385242462),
 ('1000', 0.2952816486356426),
 ('ingrediente', 0.29230231046676636),
 ('cxa', 0.26772674918174744),
 ('1000gr', 0.2644481062889099)]
```

Figura 5.6: Saídas das 10 palavras mais similares a italac e 400g ajustadas através do modelo CBoW com k=3 e mínimo de palavras=10, considerando descrições únicas e duplicadas, respectivamente.

k=5, mínimo de palavras=10

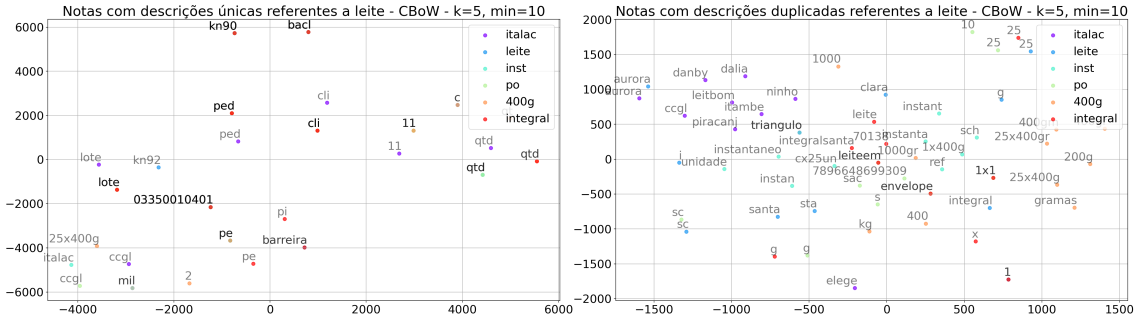

Figura 5.7: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a italac, leite, inst, po, 400g e integral, obtidas através do banco com descrições únicas utilizando o modelo CBoW, com k=5 e mínimo de palavras=10. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model1.wv.most_similar('italac')
[('qtd', 0.9969996809959412),
 ('ped', 0.9968470335006714),
 ('lote', 0.996762216091156),
 ('11', 0.9966918230056763),
 ('mil', 0.9965558052062988),
 ('03350010401', 0.9963632225990295),
 ('cli', 0.996350347957581),
 ('mgz', 0.9960650205612183),
 ('rot', 0.9959735870361328),
 ('120g', 0.9953961968421936)]
```

```
model.wv.most_similar('italac')
[('danby', 0.7506375908851624),
 ('aurora', 0.7421299815177917),
 ('itambe', 0.7395166158676147),
 ('leitbom', 0.7189064025878906),
 ('ninho', 0.6124374270439148),
 ('dalia', 0.5963461995124817),
 ('ccgl', 0.5565580129623413),
 ('piracanj', 0.5543262362480164),
 ('elege', 0.5526439547538757),
 ('triangulo', 0.4653870761394501)]
```

```
model1.wv.most_similar('400g')
[('25x400g', 0.9929735064506531),
 ('8', 0.9928779602050781),
 ('x', 0.9927301406860352),
 ('lpi400leite', 0.9924392700195312),
 ('50', 0.9924055337905884),
 ('20', 0.9923669099807739),
 ('15', 0.9923402070999146),
 ('leitbom', 0.9918145537376404),
 ('30', 0.9917315244674683),
 ('4', 0.9916440844535828)]
```

```
model.wv.most_similar('400g')
[('400gr', 0.571864128112793),
 ('200g', 0.4859218597412109),
 ('25x400gr', 0.4127757251262665),
 ('kg', 0.3724862337112427),
 ('25x400g', 0.3190397918224334),
 ('1000', 0.3059081733226776),
 ('400gm', 0.28440675139427185),
 ('gramas', 0.2799767255783081),
 ('400', 0.2727184295654297),
 ('1000gr', 0.2674514949321747)]
```

Figura 5.8: Saídas das 10 palavras mais similares a italac e 400g ajustadas através do modelo CBoW com k=5 e mínimo de palavras=10, considerando descrições únicas e duplicadas, respectivamente.

k=5, mínimo de palavras=30

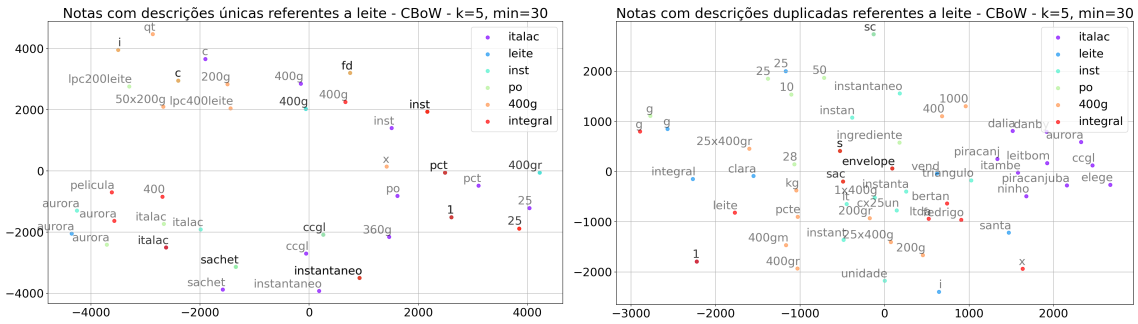

Figura 5.9: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a italac, leite, inst, po, 400g e integral, obtidas através do banco com descrições únicas utilizando o modelo CBoW, com k=5 e mínimo de palavras=30. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model.wv.most_similar('italac')
[('inst', 0.9976500868797302),
 ('ccgl', 0.9975937604904175),
 ('sachet', 0.9972784519195557),
 ('pct', 0.9972085952758789),
 ('400g', 0.9972079396247864),
 ('360g', 0.9972063899040222),
 ('c', 0.9972009658813477),
 ('instantaneo', 0.9971624612808228),
 ('po', 0.9970316290855408),
 ('25', 0.9970086216926575)]
```

```
model.wv.most_similar('italac')
[('danby', 0.778860867023468),
 ('aurora', 0.7371287941932678),
 ('itambe', 0.7315088510513306),
 ('leitbom', 0.6776179671287537),
 ('ninho', 0.5821608304977417),
 ('dalia', 0.5738474726676941),
 ('elege', 0.5653489232063293),
 ('ccgl', 0.5628876686096191),
 ('piracanj', 0.5609527826309204),
 ('piracanjuba', 0.4767930209636688)]
```

```
model.wv.most_similar('400g')
[('c', 0.9981478452682495),
 ('i', 0.9980194568634033),
 ('fd', 0.9979709386825562),
 ('lpc400leite', 0.9978612661361694),
 ('qt', 0.9978231191635132),
 ('50x200g', 0.9977579712867737),
 ('200g', 0.997702419757843),
 ('x', 0.9976641535758972),
 ('25', 0.9976573586463928),
 ('inst', 0.9974718689918518)]
```

```
model.wv.most_similar('400g')
[('400gr', 0.5984495282173157),
 ('200g', 0.5259324908256531),
 ('25x400gr', 0.4200347363948822),
 ('kg', 0.37995749711990356),
 ('200gr', 0.36943319439888),
 ('25x400g', 0.3448587656021118),
 ('400', 0.3100040555000305),
 ('400gm', 0.3009301424026489),
 ('1000', 0.27376997470855713),
 ('25x400', 0.26114562153816223)]
```

Figura 5.10: Saídas das 10 palavras mais similares a italac e 400g ajustadas através do modelo CBoW com k=5 e mínimo de palavras=30, considerando descrições únicas e duplicadas, respectivamente.

Carne

k=2, mínimo de palavras=10

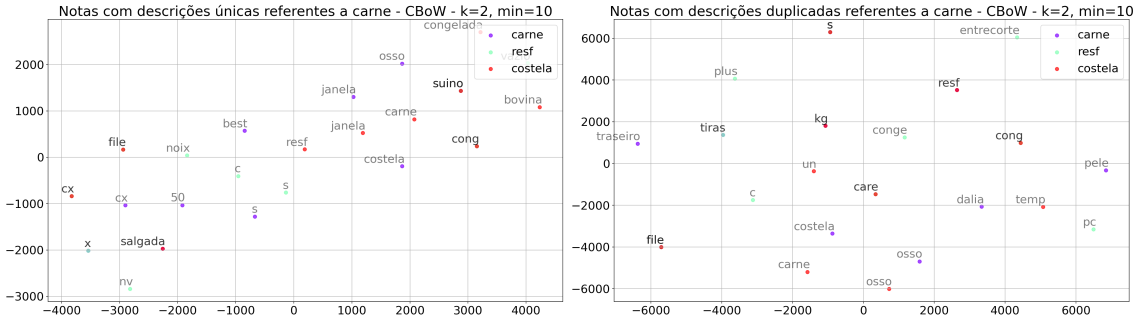

Figura 5.11: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a carne, resf e costela, obtidas através do banco com descrições únicas utilizando o modelo CBoW, com k=2 e mínimo de palavras=10. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model.wv.most_similar('costela')
[('carne', 0.9966051578521729),
 ('cx', 0.9962249398231506),
 ('file', 0.9957903623580933),
 ('suino', 0.9956934452056885),
 ('janela', 0.9955876469612122),
 ('salgada', 0.9955293536186218),
 ('congelada', 0.9954741597175598),
 ('bovina', 0.9953872561454773),
 ('cong', 0.9953694939613342),
 ('resf', 0.9952256083488464)]
```

```
model.wv.most_similar('costela')
[('carne', 0.5004284977912903),
 ('un', 0.26241534948349),
 ('resf', 0.2502754032611847),
 ('osso', 0.22116689383983612),
 ('cong', 0.21475663781166077),
 ('kg', 0.20624710619449615),
 ('care', 0.1991443783044815),
 ('temp', 0.19445101916790009),
 ('s', 0.179080069065094),
 ('file', 0.16725701093673706)]
```

```
model.wv.most_similar('cong')
[('cx', 0.9968040585517883),
 ('bov', 0.9966610670089722),
 ('resf', 0.9966250658035278),
 ('c', 0.996433436870575),
 ('vazio', 0.9963963031768799),
 ('noix', 0.9960867762565613),
 ('s', 0.9960496425628662),
 ('file', 0.9957361221313477),
 ('qtd', 0.9954788684844971),
 ('costela', 0.9953694343566895)]
```

```
model.wv.most_similar('cong')
[('resf', 0.6691081523895264),
 ('pc', 0.6344509720802307),
 ('pele', 0.5780466198921204),
 ('grande', 0.5463154911994934),
 ('dalia', 0.5193973183631897),
 ('temp', 0.45528972148895264),
 ('res', 0.44736817479133606),
 ('salgada', 0.41531816124916077),
 ('c', 0.4017359912395477 3),
 ('cm', 0.392093300819397)]
```

Figura 5.12: Saídas das 10 palavras mais similares a costela e cong ajustadas através do modelo CBoW com k=2 e mínimo de palavras=10, considerando descrições únicas e duplicadas, respectivamente.

k=2, mínimo de palavras=30

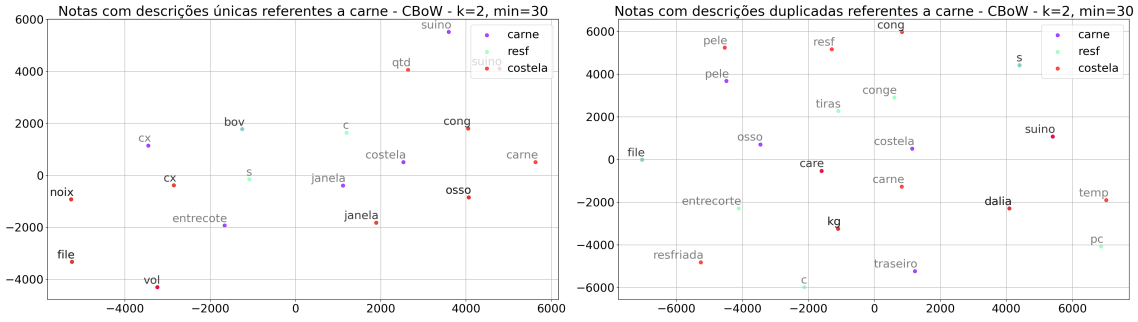

Figura 5.13: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a carne, resf e costela, obtidas através do banco com descrições únicas utilizando o modelo CBoW, com k=2 e mínimo de palavras=30. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model.wv.most_similar('costela')
[('carne', 0.997033417224884),
 ('cx', 0.996315062046051),
 ('suino', 0.9958614110946655),
 ('janela', 0.9956712126731873),
 ('cong', 0.9955790042877197),
 ('file', 0.9954695105552673),
 ('noix', 0.9954307675361633),
 ('osso', 0.9953967928886414),
 ('qtd', 0.9953119158744812),
 ('vol', 0.9951756000518799)]
```

```
model.wv.most_similar('costela')
[('carne', 0.5059338212013245),
 ('care', 0.2371264100074768),
 ('resf', 0.22271119058132172),
 ('suino', 0.1683918535709381),
 ('temp', 0.16153140366077423),
 ('kg', 0.16117921471595764),
 ('dalia', 0.15209361910820007),
 ('cong', 0.152058407664299),
 ('pele', 0.15129630267620087),
 ('resfriada', 0.14948144555091858)]
```

```
model.wv.most_similar('cong')
[('cx', 0.997184693813324),
 ('resf', 0.9968161582946777),
 ('noix', 0.9966411590576172),
 ('bov', 0.9964728355407715),
 ('file', 0.9963889122009277),
 ('s', 0.9961468577384949),
 ('c', 0.9960992932319641),
 ('qtd', 0.9958515167236328),
 ('suino', 0.9956249594688416),
 ('costela', 0.995578944683075)]
```

```
model.wv.most_similar('cong')
[('resf', 0.6712180972099304),
 ('pc', 0.6175458431243896),
 ('grande', 0.5483700037002563),
 ('dalia', 0.5180974006652832),
 ('pele', 0.4870462417602539),
 ('res', 0.43839001655578613),
 ('temp', 0.42977243661880493),
 ('salgada', 0.41897276043891907),
 ('conge', 0.40222135186195374),
 ('cm', 0.3817197680473327 6)]
```

Figura 5.14: Saídas das 10 palavras mais similares a costela e cong ajustadas através do modelo CBoW com k=2 e mínimo de palavras=30, considerando descrições únicas e duplicadas, respectivamente.

k=3, mínimo de palavras=10

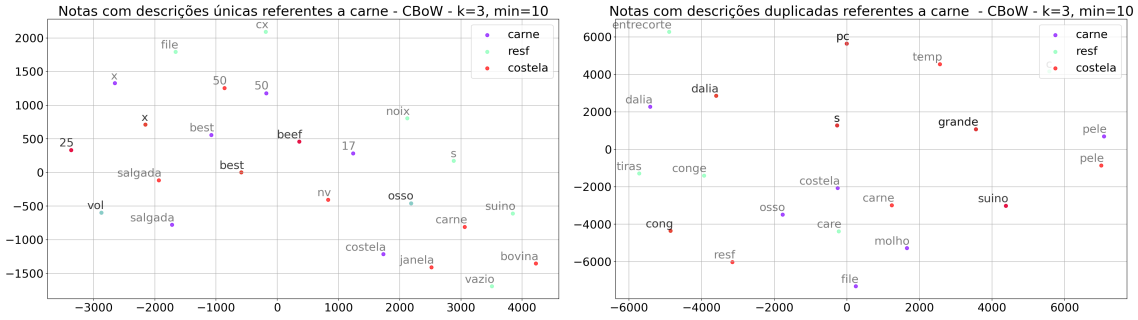

Figura 5.15: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a carne, resf e costela, obtidas através do banco com descrições únicas utilizando o modelo CBoW, com k=3 e mínimo de palavras=10. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model.wv.most_similar('costela')
[('salgada', 0.9976506233215332),
 ('50', 0.9975670576095581),
 ('bovina', 0.9975345134735107),
 ('x', 0.9974790215492249),
 ('carne', 0.9974650740623474),
 ('best', 0.9972710609436035),
 ('beef', 0.9972118735313416),
 ('nv', 0.9971515536308289),
 ('25', 0.9969820976257324),
 ('janela', 0.9969691634178162)]
```

```
model.wv.most_similar('costela')
[('carne', 0.4194093942642212),
 ('pc', 0.18777696788311005),
 ('eti', 0.16303144439535141),
 ('s', 0.16032496094703674),
 ('grande', 0.15400074422359467),
 ('osso', 0.15351708233356476),
 ('temp', 0.14855743944644928),
 ('plast', 0.14469438791275024),
 ('resf', 0.13608984649181366),
 ('nv', 0.1354183554649353)]
```

```
model.wv.most_similar('cong')
[('bov', 0.997229278087616),
 ('cx', 0.9971966743469238),
 ('qtd', 0.9965389966964722),
 ('vazio', 0.9964970946311951),
 ('c', 0.9960126876831055),
 ('noix', 0.9958891272544861),
 ('resf', 0.9958323836326599),
 ('s', 0.9954601526260376),
 ('suino', 0.9951326847076416),
 ('file', 0.9949309825897217)]
```

```
model.wv.most_similar('cong')
[('resf', 0.6452097296714783),
 ('grande', 0.5726568698883057),
 ('pc', 0.5477542877197266),
 ('dalia', 0.5123492479324341),
 ('pele', 0.4831806123256833),
 ('salgada', 0.43721067905426025),
 ('c', 0.4201160967350006),
 ('temp', 0.4088694751262665),
 ('conge', 0.39570674300193787),
 ('cm', 0.3808104693889618)]
```

Figura 5.16: Saídas das 10 palavras mais similares a costela e cong ajustadas através do modelo CBoW com k=3 e mínimo de palavras=10, considerando descrições únicas e duplicadas, respectivamente.

k=5, mínimo de palavras=10

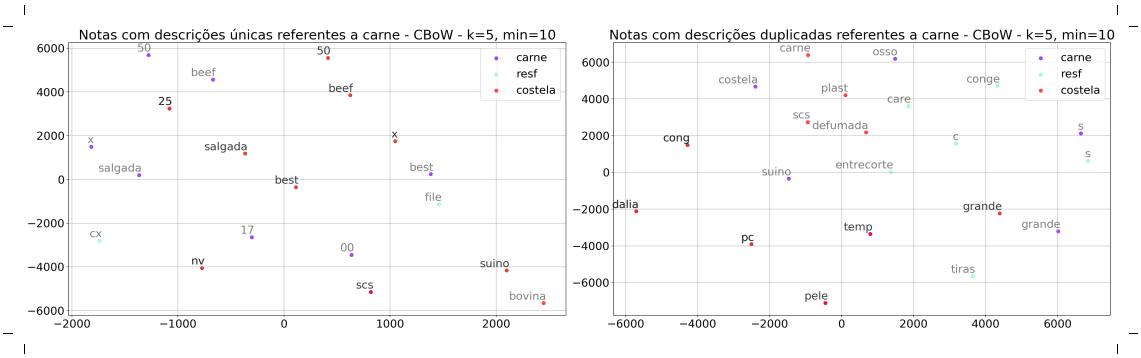

Figura 5.17: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a carne, resf e costela, obtidas através do banco com descrições únicas utilizando o modelo CBoW, com k=5 e mínimo de palavras=10. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model.wv.most_similar('costela')
```

```
[('50', 0.9982132911682129),
 ('salgada', 0.9981255531311035),
 ('beef', 0.9980220794677734),
 ('x', 0.9978853464126587),
 ('nv', 0.9978196620941162),
 ('scs', 0.9977953433990479),
 ('best', 0.9977739453315735),
 ('bovina', 0.9977673292160034),
 ('suino', 0.9976909756660461),
 ('25', 0.9976162314414978)]
```

```
model.wv.most_similar('costela')
```

```
[('50', 0.9982132911682129),
 ('salgada', 0.9981255531311035),
 ('beef', 0.9980220794677734),
 ('x', 0.9978853464126587),
 ('nv', 0.9978196620941162),
 ('scs', 0.9977953433990479),
 ('best', 0.9977739453315735),
 ('bovina', 0.9977673292160034),
 ('suino', 0.9976909756660461),
 ('25', 0.9976162314414978)]
```

```
model.wv.most_similar('cong')
```

```
[('cx', 0.997762143611908),
 ('bov', 0.9974201917648315),
 ('c', 0.9970346689224243),
 ('vazio', 0.996910572052002),
 ('noix', 0.9965801239013672),
 ('qtd', 0.9962615966796875),
 ('resf', 0.9956342577934265),
 ('s', 0.9955843091011047),
 ('file', 0.9950324892997742),
 ('vol', 0.9946025609970093)]
```

```
model.wv.most_similar('cong')
```

```
[('resf', 0.629227876663208),
 ('pc', 0.5740641355514526),
 ('grande', 0.5299052596092224),
 ('pele', 0.5041018724441528),
 ('dalia', 0.498626708984375),
 ('temp', 0.43402156233787537),
 ('c', 0.4081015884876251),
 ('suino', 0.37285006046295166),
 ('salgada', 0.3694275617594873),
 ('cm', 0.360083669424057)]
```

Figura 5.18: Saídas das 10 palavras mais similares a costela e cong ajustadas através do modelo CBoW com k=5 e mínimo de palavras=10, considerando descrições únicas e duplicadas, respectivamente.

k=5, mínimo de palavras=30

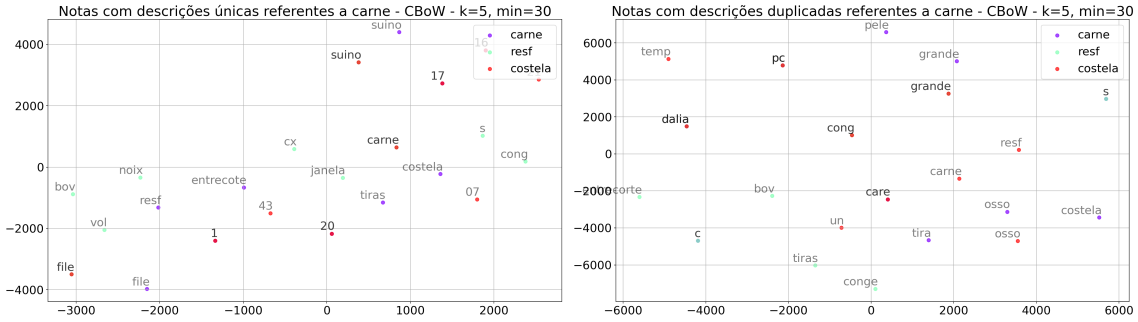

Figura 5.19: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a carne, resf e costela, obtidas através do banco com descrições únicas utilizando o modelo CBoW, com k=5 e mínimo de palavras=30. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model.wv.most_similar('costela')
```

```
[('carne', 0.9969007968902588),
 ('1', 0.996778666973114),
 ('16', 0.9965826272964478),
 ('20', 0.9965519309043884),
 ('07', 0.9962959885597229),
 ('file', 0.996290922164917),
 ('43', 0.9962381720542908),
 ('17', 0.9961970448493958),
 ('suino', 0.9961892366409302),
 ('21', 0.9961551427841187)]
```

```
model.wv.most_similar('costela')
```

```
[('carne', 0.271891325712204),
 ('osso', 0.23733694851398468),
 ('grande', 0.19498053193092346),
 ('temp', 0.17859573662281036),
 ('cong', 0.16970007121562958),
 ('un', 0.13747768104076385),
 ('resf', 0.13389602303504944),
 ('care', 0.13074007630348206),
 ('pc', 0.129682719707489),
 ('dalia', 0.11303083598613739)]
```

```
model1.wv.most_similar('cong')
```

```
[('cx', 0.9959734082221985),
 ('c', 0.9944297075271606),
 ('bov', 0.9940274953842163),
 ('qtd', 0.9889444708824158),
 ('noix', 0.9839360117912292),
 ('s', 0.9761021733283997),
 ('resf', 0.9728521704673767),
 ('file', 0.9683929681777954),
 ('pec', 0.9670810103416443),
 ('vol', 0.9574176073074341)]
```

```
model.wv.most_similar('cong')
```

```
[('resf', 0.6245023012161255),
 ('grande', 0.5420145392417908),
 ('pc', 0.5235846638679504),
 ('dalia', 0.4750169813632965),
 ('pele', 0.44637125730514526),
 ('c', 0.39063066244125366),
 ('temp', 0.37896108627319336),
 ('suino', 0.3749130070209503),
 ('salgada', 0.35683658719062805),
 ('conge', 0.31759631633758545)]
```

Figura 5.20: Saídas das 10 palavras mais similares a costela e cong ajustadas através do modelo CBoW com k=5 e mínimo de palavras=30, considerando descrições únicas e duplicadas, respectivamente.

Skip-gram

Leite

k=2, mínimo de palavras=10

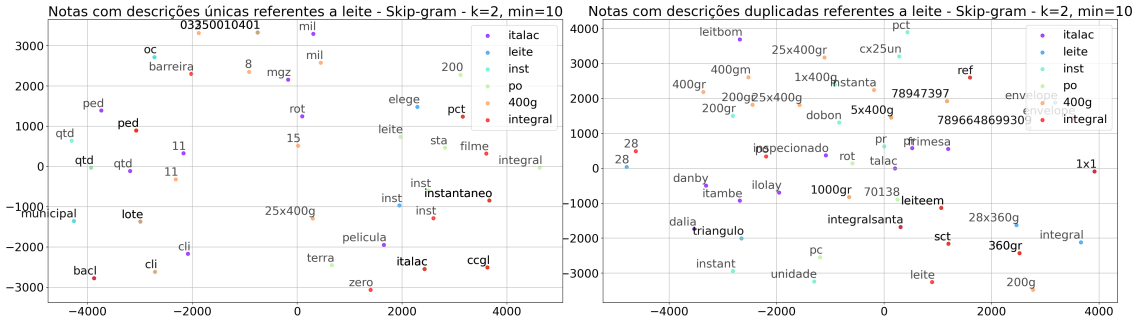

Figura 5.21: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a italac, leite, inst, po, 400g e integral, obtidas através do banco com descrições únicas utilizando o modelo Skip-gram, com k=2 e mínimo de palavras=10. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model1.wv.most_similar('italac')
[('qtd', 0.9973050951957703),
 ('ped', 0.9971298575401306),
 ('cli', 0.9966568946838379),
 ('lote', 0.9966012835502625),
 ('11', 0.996592104434967),
 ('pelicula', 0.9964343309402466),
 ('03350010401', 0.9964070320129395),
 ('mil', 0.9963859915733337),
 ('rot', 0.9962877035140991),
 ('mgz', 0.9962742328643799)]
```

```
model1.wv.most_similar('italac')
[('itambe', 0.6815569996833801),
 ('leitbom', 0.6779002547264099),
 ('frimesa', 0.6354796886444092),
 ('danby', 0.6313402056694031),
 ('triangulo', 0.6110437512397766),
 ('talac', 0.6041343212127686),
 ('ilolay', 0.5924270749092102),
 ('dalia', 0.5910006165504456),
 ('ninho', 0.5844177603721619),
 ('piracanj', 0.5802014470100403)]
```

```
model1.wv.most_similar('400g')
[('8', 0.9961234927177429),
 ('mil', 0.9959920644760132),
 ('cli', 0.9959891438484192),
 ('2', 0.9959313869476318),
 ('lote', 0.9959283471107483),
 ('ped', 0.9959236979484558),
 ('03350010401', 0.9958850145339966),
 ('11', 0.995841811665344),
 ('25x400g', 0.9958023428916931),
 ('15', 0.9957455992698669)]
```

```
model1.wv.most_similar('400g')
[('5x400g', 0.5786008238792419),
 ('200g', 0.5442792773246765),
 ('1000gr', 0.5432426333427429),
 ('200gr', 0.5192037224769592),
 ('400gr', 0.5107606053352356),
 ('25x400g', 0.5014740824699402),
 ('78947397', 0.47202181816101074),
 ('25x400gr', 0.46885955333709717),
 ('400gm', 0.46297967433929443),
 ('cx=25x400gr', 0.4497072696685791)]
```

Figura 5.22: Saídas das 10 palavras mais similares a italac e 400g ajustadas através do modelo Skip-gram com k=2 e mínimo de palavras=10, considerando descrições únicas e duplicadas, respectivamente.

k=2, mínimo de palavras=30

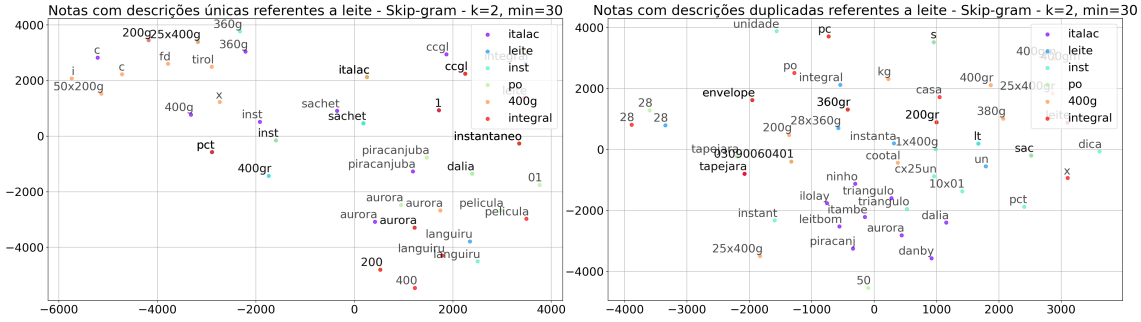

Figura 5.23: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a italac, leite, inst, po, 400g e integral, obtidas através do banco com descrições únicas utilizando o modelo Skip-gram, com k=2 e mínimo de palavras=30. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model1.wv.most_similar('italac')
[('sachet', 0.9968138337135315),
 ('ccgl', 0.9967314600944519),
 ('360g', 0.9966109395027161),
 ('c', 0.9965085983276367),
 ('piracanjuba', 0.9964998960494995),
 ('aurora', 0.9964846968650818),
 ('inst', 0.99644935131073),
 ('400g', 0.99644935131073),
 ('pct', 0.9964187741279602),
 ('200g', 0.9963269829750061)]
```

```
model1.wv.most_similar('italac')
[('itambe', 0.6873412728309631),
 ('leitbom', 0.6698965430259705),
 ('triangulo', 0.6167023777961731),
 ('danby', 0.6024251580238342),
 ('dalia', 0.5908340811729431),
 ('ilolay', 0.5832481384277344),
 ('ninho', 0.5574222803115845),
 ('aurora', 0.5549507737159729),
 ('piracanj', 0.5526666045188904),
 ('tapejara', 0.5220687985420227)]
```

```
model1.wv.most_similar('400g')
[('c', 0.9968553185462952),
 ('25x400g', 0.9967989921569824),
 ('fd', 0.9967110753059387),
 ('200g', 0.9966920018196106),
 ('50x200g', 0.9966744184494019),
 ('tirol', 0.9965183734893799),
 ('italac', 0.99644935131073),
 ('i', 0.9962940812110901),
 ('x', 0.9962404370307922),
 ('aurora', 0.9960182309150696)]
```

```
model1.wv.most_similar('400g')
[('200g', 0.5637043714523315),
 ('25x400g', 0.5164771676063538),
 ('200gr', 0.5078656077384949),
 ('400gr', 0.4932786226272583),
 ('25x400gr', 0.4609701931476593),
 ('kg', 0.450455904006958),
 ('cootal', 0.4502250850200653),
 ('380g', 0.4495340287685394 3),
 ('400gm', 0.43500062823295593),
 ('03090060401', 0.423556923866272)]
```

Figura 5.24: Saídas das 10 palavras mais similares a italac e 400g ajustadas através do modelo Skip-gram com k=2 e mínimo de palavras=30, considerando descrições únicas e duplicadas, respectivamente.

k=3, mínimo de palavras=10

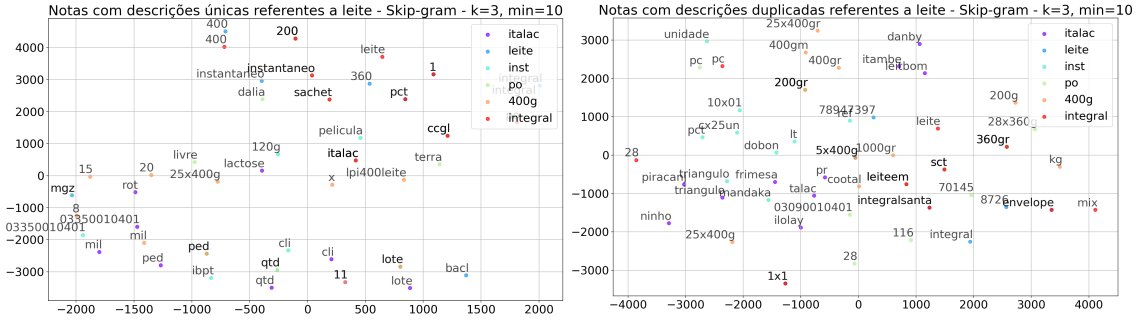

Figura 5.25: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a italac, leite, inst, po, 400g e integral, obtidas através do banco com descrições únicas utilizando o modelo Skip-gram, com k=3 e mínimo de palavras=10. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model1.wv.most_similar('italac')
[('ped', 0.9971354603767395),
 ('qtd', 0.9970654249191284),
 ('lote', 0.996741533279419),
 ('03350010401', 0.9964642524719238),
 ('mil', 0.9964489340782166),
 ('11', 0.9964068531990051),
 ('rot', 0.9963377714157104),
 ('cli', 0.9963144063949585),
 ('mgz', 0.9962159395217896),
 ('lactose', 0.9959915280342102)]
```

```
model1.wv.most_similar('italac')
[('itambe', 0.6737256050109863),
 ('leitbom', 0.6558588743209839),
 ('talac', 0.6242527961730957),
 ('triangulo', 0.6145452260971069),
 ('danby', 0.5988436937332153),
 ('frimesa', 0.5941998362541199),
 ('dalia', 0.5879644751548767),
 ('ninho', 0.5723675489425659),
 ('ilolay', 0.5658648014068604),
 ('piracanj', 0.5479641556739807)]
```

```
model1.wv.most_similar('400g')
[('25x400g', 0.9946848750114441),
 ('8', 0.9934117197990417),
 ('20', 0.9930076599121094),
 ('15', 0.9928886294364929),
 ('lote', 0.9928362369537354),
 ('11', 0.9928333759307861),
 ('ped', 0.9928291440010071),
 ('leitbom', 0.9928010702133179),
 ('x', 0.9927646517753601),
 ('mil', 0.9927636981010437)]
```

```
model1.wv.most_similar('400g')
[('5x400g', 0.5876803398132324),
 ('400gr', 0.538892924785614),
 ('200g', 0.536467969417572),
 ('200gr', 0.5350725054740906),
 ('1000gr', 0.5150796175003052),
 ('25x400gr', 0.5107477903366089),
 ('25x400g', 0.510165810585022),
 ('400gm', 0.452287495136261),
 ('kg', 0.44957807660102844),
 ('cx=25x400gr', 0.4309195876121521)]
```

Figura 5.26: Saídas das 10 palavras mais similares a italac e 400g ajustadas através do modelo Skip-gram com k=3 e mínimo de palavras=10, considerando descrições únicas e duplicadas, respectivamente.

k=5, mínimo de palavras=10

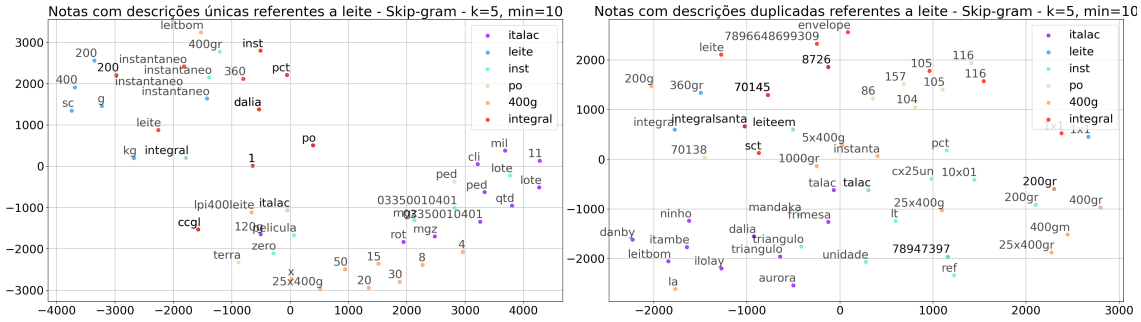

Figura 5.27: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a italac, leite, inst, po, 400g e integral, obtidas através do banco com descrições únicas utilizando o modelo Skip-gram, com k=5 e mínimo de palavras=10. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model1.wv.most_similar('italac')
[('qtd', 0.9969996809959412),
 ('ped', 0.9968470335006714),
 ('lote', 0.996762216091156),
 ('11', 0.9966918230056763),
 ('mil', 0.9965558052062988),
 ('03350010401', 0.9963632225990295),
 ('cli', 0.9963503479957581),
 ('mgz', 0.9960650205612183),
 ('rot', 0.9959735870361328),
 ('120g', 0.9953961968421936)]
```

```
model1.wv.most_similar('italac')
[('itambe', 0.6658581495285034),
 ('leitbom', 0.6541362404823303),
 ('triangulo', 0.6079100370407104),
 ('danby', 0.6032209396362305),
 ('talac', 0.5944196581840515),
 ('dalia', 0.5940057635307312),
 ('ninho', 0.5768882632255554),
 ('frimesa', 0.5531383752822876),
 ('ilolay', 0.5494088530540466),
 ('aurora', 0.5398407578468323)]
```

```
model1.wv.most_similar('400g')
[('25x400g', 0.9929735064506531),
 ('8', 0.9928779602050781),
 ('x', 0.9927301406860352),
 ('lpi400leite', 0.9924392700195312),
 ('50', 0.9924055337905884),
 ('20', 0.9923669099807739),
 ('15', 0.992340207099146),
 ('leitbom', 0.9918145537376404),
 ('30', 0.9917315244674683),
 ('4', 0.9916440844535828)]
```

```
model1.wv.most_similar('400g')
[('25x400g', 0.5118014216423035),
 ('400gr', 0.4995588958263397),
 ('5x400g', 0.4949145019054413),
 ('1000gr', 0.4927072823047638),
 ('200g', 0.491255551576144),
 ('25x400gr', 0.4841216206550598),
 ('400gm', 0.4544804990291595),
 ('200gr', 0.44971370697021484),
 ('1a', 0.4426812827571277),
 ('instanta', 0.4322047233581543)]
```

Figura 5.28: Saídas das 10 palavras mais similares a italac e 400g ajustadas através do modelo Skip-gram com k=5 e mínimo de palavras=10, considerando descrições únicas e duplicadas, respectivamente.

k=5, mínimo de palavras=30

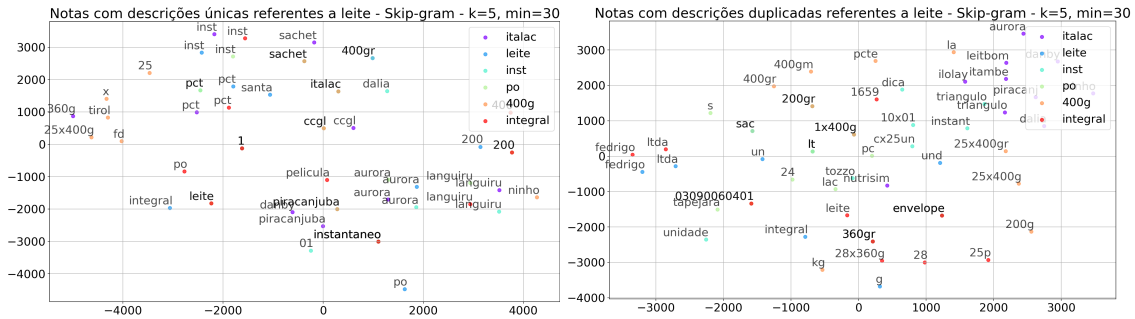

Figura 5.29: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a italac, leite, inst, po, 400g e integral, obtidas através do banco com descrições únicas utilizando o modelo Skip-gram, com k=5 e mínimo de palavras=30. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model1.wv.most_similar('italac')
```

```
[('sachet', 0.9969974160194397),
 ('inst', 0.9963617920875549),
 ('ccgl', 0.9963254332542419),
 ('aurora', 0.9963021874427795),
 ('piracanjuba', 0.996268630027771),
 ('360g', 0.9961210489273071),
 ('pct', 0.9960179328918457),
 ('languiru', 0.9959417581558228),
 ('danby', 0.9958115816116333),
 ('400gr', 0.9957740902900696)]
```

```
model1.wv.most_similar('italac')
```

```
[('itambe', 0.6819268465042114),
 ('leitbom', 0.6759076714515686),
 ('triangulo', 0.6104618906974792),
 ('danby', 0.610425142362976),
 ('dalia', 0.5827434062957764),
 ('ninho', 0.5570520162582397),
 ('ilolay', 0.556452751159668),
 ('aurora', 0.5552259683609009),
 ('piracanj', 0.5538938641548157),
 ('nutrisim', 0.5331317186355591)]
```

```
model1.wv.most_similar('400g')
```

```
[('x', 0.9960638880729675),
 ('25x400g', 0.9958857297897339),
 ('25', 0.9957056641578674),
 ('tirol', 0.9950570464134216),
 ('italac', 0.9949941635131836),
 ('ccgl', 0.9948644638061523),
 ('fd', 0.9946457147598267),
 ('piracanjuba', 0.99449223279953),
 ('sachet', 0.9943937659263611),
 ('ninho', 0.9942871928215027)]
```

```
model1.wv.most_similar('400g')
```

```
[('25x400gr', 0.5018100142478943),
 ('25x400g', 0.5008977651596069),
 ('200g', 0.4983437657356262),
 ('200gr', 0.4970008432865143),
 ('400gr', 0.4910828173160553),
 ('400gm', 0.4483771920241626),
 ('1a', 0.43201661109924316),
 ('1x400g', 0.42606157064437866),
 ('kg', 0.42093369364738464),
 ('pcte', 0.4193660020828247)]
```

Figura 5.30: Saídas das 10 palavras mais similares a italac e 400g ajustadas através do modelo Skip-gram com k=5 e mínimo de palavras=30, considerando descrições únicas e duplicadas, respectivamente.

Carne

k=2, mínimo de palavras=10

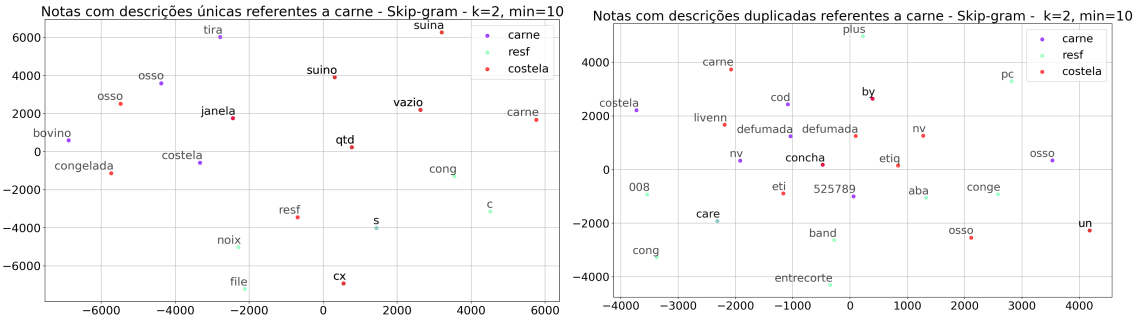

Figura 5.31: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a carne, resf e costela, obtidas através do banco com descrições únicas utilizando o modelo Skip-gram, com k=2 e mínimo de palavras=10. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model1.wv.most_similar('costela')
[('carne', 0.9912598133087158),
 ('cx', 0.9884175658226013),
 ('osso', 0.9882950782775879),
 ('suino', 0.9876320958137512),
 ('janela', 0.9867284297943115),
 ('congelada', 0.9862384796142578),
 ('vazio', 0.9849092364311218),
 ('suina', 0.9842367768287659),
 ('qtd', 0.9841836094856262),
 ('resf', 0.983933687210083)]
```

```
model1.wv.most_similar('costela')
[('carne', 0.5533263087272644),
 ('osso', 0.4504404664039612),
 ('un', 0.4269315302371979),
 ('by', 0.4116562604904175),
 ('etiq', 0.3920234739780426),
 ('livenn', 0.382631778717041),
 ('concha', 0.3810414671897888),
 ('defumada', 0.3795975148677826),
 ('nv', 0.37617674469947815),
 ('eti', 0.3709667921066284)]
```

```
model1.wv.most_similar('cong')
[('resf', 0.9912347197532654),
 ('c', 0.9908450841903687),
 ('bov', 0.9906103014945984),
 ('cx', 0.9901217818260193),
 ('s', 0.9897152781486511),
 ('qtd', 0.9892752170562744),
 ('file', 0.98759925365448),
 ('noix', 0.9873141646385193),
 ('vazio', 0.986405611038208),
 ('costela', 0.9819980263710022)]
```

```
model1.wv.most_similar('cong')
[('grande', 0.6661185622215271),
 ('concha', 0.5873591303825378),
 ('dalia', 0.5759792923927307),
 ('pc', 0.5649744868278503),
 ('pele', 0.5570927858352661),
 ('resf', 0.5521571040153503),
 ('cm', 0.5197209119796753),
 ('conge', 0.5078935027122498),
 ('salgada', 0.4712666869163513),
 ('temp', 0.47094210982322693)]
```

Figura 5.32: Saídas das 10 palavras mais similares a costela e cong ajustadas através do modelo Skip-gram com k=2 e mínimo de palavras=10, considerando descrições únicas e duplicadas, respectivamente.

k=2, mínimo de palavras=30

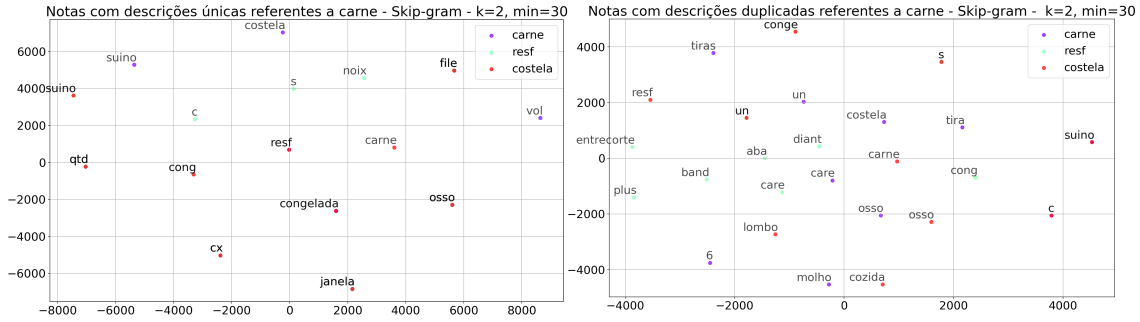

Figura 5.33: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a carne, resf e costela, obtidas através do banco com descrições únicas utilizando o modelo Skip-gram, com k=2 e mínimo de palavras=30. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model1.wv.most_similar('costela')
[('carne', 0.9940324425697327),
 ('suino', 0.9891255497932434),
 ('cx', 0.9879783987998962),
 ('qtd', 0.9874058961868286),
 ('osso', 0.9860990643501282),
 ('congelada', 0.9848609566688538),
 ('cong', 0.9832682013511658),
 ('janela', 0.9830074310302734),
 ('file', 0.9828535318374634),
 ('resf', 0.9822533130645752)]
```

```
model1.wv.most_similar('costela')
[('carne', 0.5326460599899292),
 ('osso', 0.44715723395347595),
 ('un', 0.3359328806400299),
 ('c', 0.33065253496170044),
 ('cozida', 0.30655187368392944),
 ('resf', 0.3046990633010864),
 ('suino', 0.3036859631538391),
 ('s', 0.3005034923553467),
 ('conge', 0.29363441467285156),
 ('lombo', 0.27216196060180664)]
```

```
model1.wv.most_similar('cong')
[('cx', 0.9939401149749756),
 ('resf', 0.9928678870201111),
 ('qtd', 0.9924399852752686),
 ('bov', 0.9904197454452515),
 ('s', 0.9898952841758728),
 ('c', 0.9896417260169983),
 ('noix', 0.9895050525665283),
 ('file', 0.9893287420272827),
 ('janela', 0.9862093925476074),
 ('osso', 0.9851577877998352)]
```

```
model1.wv.most_similar('cong')
[('grande', 0.646914541721344),
 ('pc', 0.5892188549041748),
 ('dalia', 0.5891667604446411),
 ('pele', 0.5599035024642944),
 ('conge', 0.5381131768226624),
 ('resf', 0.5166202783584595),
 ('cm', 0.4883727729320526),
 ('temp', 0.4783293306827545),
 ('salgada', 0.47055456042289734),
 ('c', 0.43213242292404175)]
```

Figura 5.34: Saídas das 10 palavras mais similares a costela e cong ajustadas através do modelo Skip-gram com k=2 e mínimo de palavras=30, considerando descrições únicas e duplicadas, respectivamente.

k=3, mínimo de palavras=10

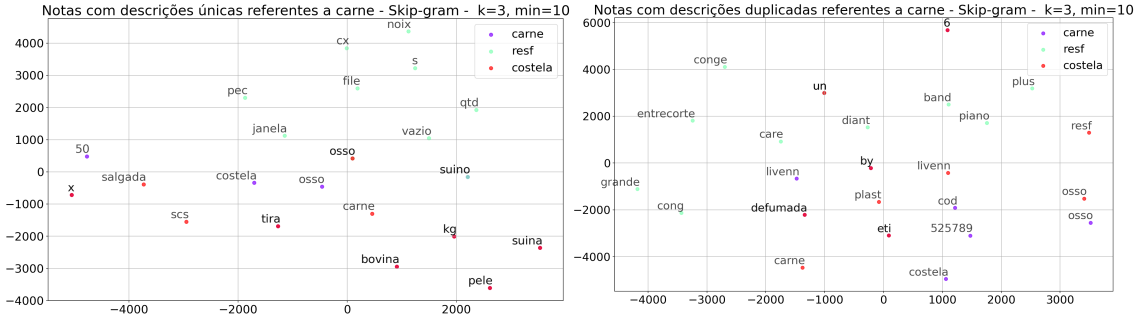

Figura 5.35: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a carne, resf e costela, obtidas através do banco com descrições únicas utilizando o modelo Skip-gram, com k=3 e mínimo de palavras=10. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model1.wv.most_similar('costela')
[('bovina', 0.992912769317627),
 ('kg', 0.9917370080947876),
 ('carne', 0.9912992715835571),
 ('suina', 0.9906684756278992),
 ('tira', 0.9905689358711243),
 ('pele', 0.9903030395507812),
 ('osso', 0.9887755513191223),
 ('x', 0.9867561459541321),
 ('salgada', 0.9867229461669922),
 ('scs', 0.986287534236908)]
```

```
model1.wv.most_similar('costela')
[('carne', 0.5236494541168213),
 ('osso', 0.46729639172554016),
 ('by', 0.4347184300422685),
 ('defumada', 0.3997511565685272),
 ('livenn', 0.3739941120147705),
 ('nv', 0.3609841465950012),
 ('eti', 0.3426154553890228),
 ('etiq', 0.32832878828048706),
 ('resf', 0.32157033681869507),
 ('scs', 0.3203642964363098)]
```

```
model1.wv.most_similar('cong')
[('bov', 0.9935435056686401),
 ('cx', 0.992958128452301),
 ('c', 0.9892528653144836),
 ('qtd', 0.9886031150817871),
 ('noix', 0.9839872121810913),
 ('s', 0.9839469790458679),
 ('file', 0.9729112386703491),
 ('resf', 0.9728201627731323),
 ('vazio', 0.9665607213973999),
 ('suino', 0.9563465714454651)]
```

```
model1.wv.most_similar('cong')
[('grande', 0.6317011117935181),
 ('pc', 0.5820369124412537),
 ('dalia', 0.556498110294342),
 ('concha', 0.5436303615570068),
 ('pele', 0.5430871248245239),
 ('conge', 0.5346552729606628),
 ('temp', 0.4877704679965973),
 ('cm', 0.48554500937461853),
 ('resf', 0.4817686080932617),
 ('salgada', 0.4772508144378662)]
```

Figura 5.36: Saídas das 10 palavras mais similares a costela e cong ajustadas através do modelo Skip-gram com k=3 e mínimo de palavras=10, considerando descrições únicas e duplicadas, respectivamente.

k=5, mínimo de palavras=10

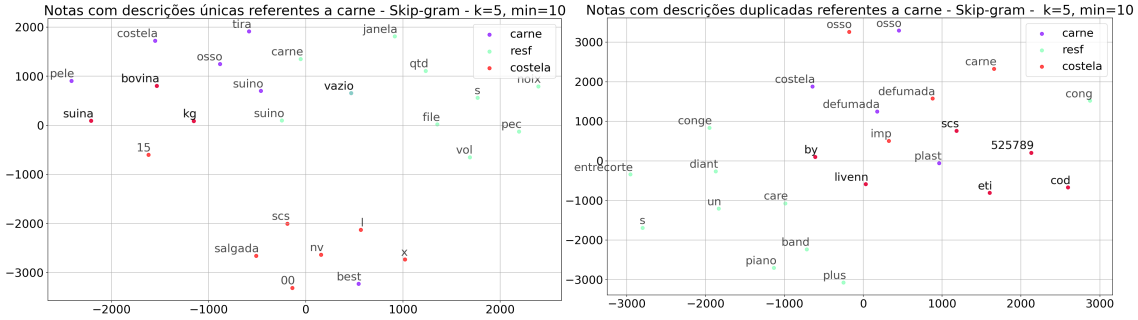

Figura 5.37: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a carne, resf e costela, obtidas através do banco com descrições únicas utilizando o modelo Skip-gram, com k=5 e mínimo de palavras=10. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model1.wv.most_similar('costela')
[('bovina', 0.9923550486564636),
 ('kg', 0.9918103218078613),
 ('suina', 0.9908814430236816),
 ('salgada', 0.9905691742897034),
 ('scs', 0.9903624057769775),
 ('15', 0.9900128245353699),
 ('00', 0.9897267818450928),
 ('nv', 0.9896392822265625),
 ('l', 0.9896288514137268),
 ('x', 0.9894399046897888)]
```

```
model1.wv.most_similar('costela')
[('carne', 0.42088842391967773),
 ('eti', 0.40762394666671753),
 ('defumada', 0.3948610723018646),
 ('osso', 0.38517463207244873),
 ('525789', 0.3788119852542877),
 ('by', 0.3398497402667999),
 ('livenn', 0.337126225233078),
 ('scs', 0.33165913820266724),
 ('cod', 0.3292901813983917),
 ('imp', 0.31713199615478516)]
```

```
model1.wv.most_similar('cong')
[('cx', 0.993830144405365),
 ('bov', 0.9934337139129639),
 ('c', 0.9932611584663391),
 ('noix', 0.9791951179504395),
 ('qtd', 0.9759823083877563),
 ('s', 0.969399452209472 7),
 ('resf', 0.9450236558914185),
 ('file', 0.9443848729133606),
 ('pec', 0.943852961063385),
 ('vol', 0.9387668371200562)]
```

```
model1.wv.most_similar('cong')
[('grande', 0.5932654738426208),
 ('pc', 0.5656672120094299),
 ('dalia', 0.5594320893287659),
 ('pele', 0.5344663262367249),
 ('temp', 0.4950645864009857),
 ('cm', 0.46992990374565125),
 ('conge', 0.4697481691837311),
 ('resf', 0.46199196577072144),
 ('vazio', 0.4559665024280548),
 ('2', 0.4543234705924988)]
```

Figura 5.38: Saídas das 10 palavras mais similares a costela e cong ajustadas através do modelo Skip-gram com k=5 e mínimo de palavras=10, considerando descrições únicas e duplicadas, respectivamente.

k=5, mínimo de palavras=30

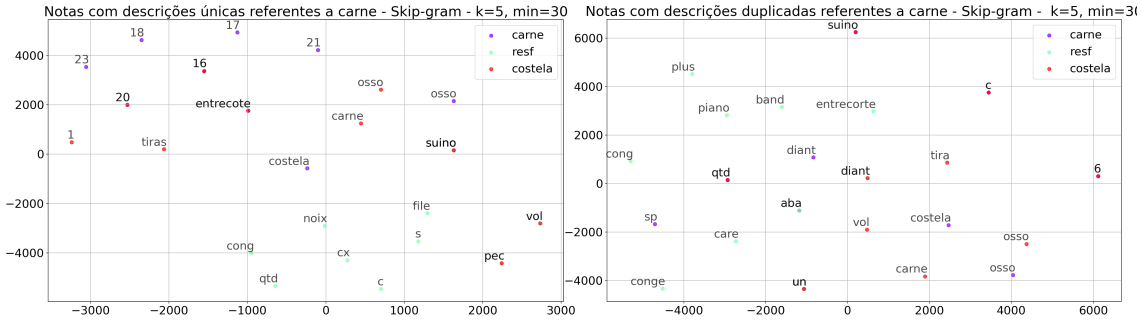

Figura 5.39: Gráfico à esquerda retrata a representação em duas dimensões das palavras mais similares a carne, resf e costela, obtidas através do banco com descrições únicas utilizando o modelo Skip-gram, com k=5 e mínimo de palavras=30. Gráfico à direita retrata a mesma representação apenas considerando descrições repetidas.

```
model1.wv.most_similar('costela')
[('suino', 0.9860814213752747),
 ('carne', 0.9858625531196594),
 ('osso', 0.9769167900085449),
 ('entrecote', 0.970565140247345),
 ('20', 0.9704501628875732),
 ('vol', 0.9703953862190247),
 ('16', 0.9676927924156189),
 ('tiras', 0.9662087559700012),
 ('1', 0.9599408507347107),
 ('pec', 0.9593482613563538)]
```

```
model1.wv.most_similar('costela')
[('carne', 0.41057464480400085),
 ('osso', 0.40733081102371216),
 ('un', 0.3177036941051483),
 ('qtd', 0.3097965121269226),
 ('6', 0.27705806493759155),
 ('c', 0.27424320578575134),
 ('diant', 0.2678067088127136),
 ('suino', 0.256149023771286),
 ('vol', 0.25386807322502136),
 ('tira', 0.24465347826480865)]
```

```
model1.wv.most_similar('cong')
[('cx', 0.9959734082221985),
 ('c', 0.9944297075271606),
 ('bov', 0.9940274953842163),
 ('qtd', 0.9889444708824158),
 ('noix', 0.9839360117912292),
 ('s', 0.9761021733283997),
 ('resf', 0.9728521704673767),
 ('file', 0.9683929681777954),
 ('pec', 0.9670810103416443),
 ('vol', 0.9574176073074341)]
```

```
model1.wv.most_similar('cong')
[('grande', 0.5932654738426208),
 ('pc', 0.5656672120094299),
 ('dalia', 0.5594320893287659),
 ('pele', 0.5344663262367249),
 ('temp', 0.4950645864009857),
 ('cm', 0.46992990374565125),
 ('conge', 0.4697481691837311),
 ('resf', 0.46199196577072144),
 ('vazio', 0.4559665024280548),
 ('2', 0.4543234705924988)]
```

Figura 5.40: Saídas das 10 palavras mais similares a costela e cong ajustadas através do modelo Skip-gram com k=5 e mínimo de palavras=30, considerando descrições únicas e duplicadas, respectivamente.