Instituto de MATEMÁTICA E ESTATÍSTICA UFRGS

DEPARTAMENTO DE ESTATÍSTICA
E-mail: dest@mat.ufrgs.br

Trabalho de Conclusão de Curso

Extração de tópicos em Notas Fiscais Eletrônicas (NF-e): uma análise utilizando BERTopic

Antônio Oss Boll

26 de fevereiro de 2024

Antônio Oss Boll

Extração de tópicos em Notas Fiscais Eletrônicas (NF-e): uma análise utilizando BERTopic

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.

Orientadora: Profa. Dra. Márcia Helena Barbian

Porto Alegre
Fevereiro de 2024

Antônio Oss Boll

Extração de tópicos em Notas Fiscais Eletrônicas (NF-e): uma análise utilizando BERTopic

Este Trabalho foi julgado adequado para obtenção dos créditos da disciplina Trabalho de Conclusão de Curso em Estatística e aprovado em sua forma final pela Orientadora e pela Banca Examinadora.

Orientadora:_____
Profa. Dra. Márcia Helena Barbian, UFMG
Doutor(a) pela Universidade Federal de Minas Gerais, Belo Horizonte, MG

Banca Examinadora:

Profa. Dra. Viviane Pereira Moreira, MDX
Doutora pela Middlesex University London, Reino Unido

Ms Israel Campos Fama, ITA
Mestre pelo Instituto Tecnológico da Aeronáutica São José dos Campos, SP

Porto Alegre
Fevereiro de 2024

"So it begins." (Edward Richtofen)

Agradecimentos

Agradeço à Letícia, que me acompanhou em toda a graduação e continuará a me acompanhar no nosso futuro.

Agradeço à Cíntia e ao José Luis por serem pais incríveis que sempre me apoiaram e me ajudaram.

Agradeço à Heloísa por ser, desde pequena, minha parceira.

Agradeço ao Kid, meu melhor amigo.

Aos meus avós Eroí e Leda.

Aos meus avós Lourdes e Orlando, e tios Alexandre e Márcio.

Ao Binho, meu amigo de épocas.

Aos meus amigos Galvão e Léo, que dividiram papos e manhãs comigo no colégio.

À Márcia, que me trouxe e me formou do curso de Estatística.

Aos professores da banca, Viviane e Israel.

Aos meus amigos da faculdade.

Aos professores e funcionários do Instituto de Matemática e Estatística que contribuíram na minha formação acadêmica e profissional.

Resumo

À medida que informações são geradas, a busca por suas interpretações cresce. No entanto, em muitos bancos de dados, a falta de rotulação dificulta sua interpretabilidade. Assim, os modelos de aprendizado profundo surgem para abordar essas tarefas complexas de Processamento de Linguagem Natural. Utilizando dados não rotulados obtidos de Notas Fiscais Eletrônicas da Secretaria da Fazenda do Rio Grande do Sul, este trabalho visa construir um modelo BERTopic para agrupar produtos semelhantes em tópicos. Durante essa modelagem, diversos hiperparâmetros foram variados, com o objetivo de encontrar o melhor resultado com base em métricas como a silhueta e considerando também o número de tópicos gerados.

Palavras-Chave: BERTopic, BERT, Aprendizado Profundo, Processamento Natural de Linguagem, Dados não rotulados, Modelagem de tópico.

Abstract

As information is generated, the search for its interpretations grows. However, in many databases, the lack of labeling complicates their interpretability. Thus, deep learning models emerge to address these complex tasks in Natural Language Processing. Using unlabeled data obtained from Electronic Invoices of the Department of Finance of Rio Grande do Sul, this work aims to build a BERTopic model to group similar products into topics. During this modeling process, various hyperparameters were adjusted, aiming to find the best result based on metrics such as silhouette and also considering the number of generated topics.

Keywords: BERTopic, BERT, Deep Learning, Natural Language Processing, Non labeled data, Topic modeling.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Estamos inseridos em uma nova revolução, a dos computadores, da internet e, principalmente, dos dados, que carrega o nome de "Quarta Revolução Industrial" segundo Schwab (2016). Essa crescente quantidade de dados tem impulsionado a necessidade de modelos mais complexos para a sua análise. A *International Data Corporation* (IDC) (Reinsel et al. (2018)) prevê que a esfera de dados global crescerá de 33 zettabytes em 2018 para 175 zettabytes até 2025. À medida que a disponibilidade de informações aumenta exponencialmente, também crescem os desafios para extrair *insights* valiosos desses conjuntos massivos de dados. Modelos tradicionais muitas vezes não são adequados, visto que eles podem ser limitados em sua capacidade de capturar relacionamentos sutis, padrões não lineares e interações entre múltiplas variáveis.

Nesse contexto, modelos mais complexos, como os baseados em aprendizado de máquina, inteligência artificial e análise estatística avançada, tornam-se cruciais. Esses modelos têm a capacidade de lidar com grandes volumes de dados, capturar estruturas dos dados e fornecer previsões e classificações mais precisas.

Um artigo seminal da inteligência artificial e suas aplicações foi o de Rosenblatt (1958), que introduziu o perceptron, uma arquitetura base das redes neurais. A ideia do perceptron se remete à medicina, com aquilo que é conhecido como neurônio. Na rede neural, o neurônio recebe dados, os *inputs*, que são processados em camadas ocultas e apresentam valores finais, os *outputs*. Partindo disso, em 1967 o grupo GMDH (Ivakhnenko et al. (1967)) publicou o que foi considerada a primeira arquitetura de rede neural profunda. A grande diferença àquela de Rosenblatt (1958) é o processamento dos dados nas camadas ocultas, que agora recebe funções de ativação, aumentando a qualidade de predição das redes. Pensando em melhorar esse processo, em 1968 surge a retropropagação (Hinton et al. (1986)), que faz o caminho inverso na rede, isto é, após a saída final, os autores propõem um processamento reverso, buscando alterar pesos e diminuir erros para um melhor resultado.

Entre as principais aplicações de redes neurais pode-se citar a análise de imagens, textos, áudios, vídeos e medidas sequenciais. Dentre as principais metodologias criadas estão: as Redes Neurais Convolucionais (CNN), introduzidas por LeCun (1989) em 1989 e as Redes Neurais Recorrentes (RNN), que têm Werbos (1988) como seu pioneiro. Com base na ideia das RNNs, Hochreiter e Schmidhuber (1997) apresentaram a Memória Longa de Curto Prazo (LSTM), uma técnica que consegue superar as limitações das Redes Recorrentes e aumentar o espaço de dependência entre as observações, resultando em melhores estimativas.

Outra técnica importante, chamada Autoencoder, consiste em dividir a arqui-

tetura da rede em duas partes distintas: o Encoder, que mapeia e comprime a parte mais importante do *input*, e o Decoder, que reconstrói os dados comprimidos, buscando aproximar-se ao máximo dos dados originais. Dessa forma, essa poderosa ferramenta, desenvolvida por Hinton et al. (1986), LeCun et al. (1998) e Bengio et al. (2016) pode, por exemplo, remover ruídos insignificantes em uma imagem, aprender padrões em textos e preencher espaços em branco em frases, entre diversas outras tarefas. A versatilidade do Autoencoder o torna um método valioso e amplamente aplicável em diversas áreas da análise de dados e do processamento de informações.

O pensamento sobre a linguagem é algo quase intrínseco ao ser humano. Chomsky, em 1957, apresenta a ideia da palavra como duas estruturas. Chomsky (1957) define a estrutura profunda, que se refere à representação subjacente de uma frase que captura seu significado e estrutura sintática abstrata, independentemente de sua forma. A outra estrutura, denominada de superficial, representa a disposição real de palavras e frases em uma sentença, incluindo a ordem das palavras, inflexões e outras características de nível raso. Desde então, os modelos envolvendo Processamento de Linguagem Natural (PLN) ((Cambria e White, 2014)) evoluíram consideravelmente. No trabalho de Bengio et al. (2003) é proposta uma metodologia para lidar com dados de *string* utilizando redes neurais, em 2013 Mikolov et al. (2013b) apresenta o Word2Vec e o CBoW, que popularizaram o conceito de Word Embeddings proposto anteriormente por Bengio. Uma nova evolução desse tipo de análise surgiu no artigo Vaswani et al. (2017), em que é apresentado o "*transformer*". Neste trabalho, os autores propõem a utilização de uma estrutura de *encoder* e *decoder*, em que os vetores *embeddings* criados por Word2Vec, por exemplo, são *inputs* do modelo.

Variações do *transformer* são o BERT (Devlin et al. (2018)), o GPT (Generative Pre-trained Transformer) e seus derivados. No BERT diversos *encoders* são empilhados, sendo utilizados em problemas que buscam avaliar o contexto do documento, como por exemplo análise de sentimento e previsão de palavras omitidas. No GPT (Radford et al. (2018)), a arquitetura da rede é composta por múltiplos *decoders*. É utilizado principalmente em traduções, *chatbots* e previsão de palavras.

Entre os diversos desafios envolvendo técnicas de NLP estão a rotulação e análise de dados não estruturados, chamada de modelagem de tópico. Análises que ocorreriam com uma melhor fluidez em dados rotulados passam a ser tarefas complexas, dificultando o trabalho e a manipulação do corpus. Problemas como a identificação de padrões, agrupamento de observações similares e análises descritivas do corpus se tornam tarefas muito desafiadoras. Além disso, há poucas técnicas capazes de realizar esse trabalho, sendo algoritmos de *"Latent Dirichlet Allocation"* (LDA) (Blei et al. (2003)) e *"Latent Semantic Analysis"* (LSA) (Deerwester et al. (1990)) os mais recorrentes, mesmo que não utilizem procedimentos mais atualizados para a extração dos tópicos, como mecanismos de atenção.

Além disso, a falta de análise em dados não estruturados e não rotulados representa um desafio, levando muitas corporações, por exemplo, a negligenciarem conjuntos de informações relevantes que poderiam contribuir para a melhoria de seus estabelecimentos. No contexto fiscal, os órgãos governamentais buscam identificar fraudes, como o superfaturamento de produtos. No entanto, devido à enorme quantidade de mercadorias, é necessário automatizar o processo de identificação, se tornando uma tarefa difícil devido à falta de padronização na descrição dessas mercadorias não rotuladas. Além disso, essas descrições podem conter erros ou inconsistências, comprometendo a eficiência e a confiabilidade das análises automáticas

realizadas.

No contexto da Secretaria da Fazenda do Rio Grande do Sul, milhares de produtos de diferentes empresas (CNPJs) são comercializados, gerando múltiplas descrições para o mesmo produto. Dessa forma, trabalhar com o agrupamento de dados se torna uma tarefa difícil, uma vez que esse volume de dados para uma mesma mercadoria com diversas rotulações é muito grande, ocorrendo problemas como abreviações, erros ortográficos, adição de números, entre diversas outras dificuldades.

Este trabalho busca agrupar tópicos de dados não rotulados para que seja possível obter características dos diversos conjuntos de mercadorias, como o preço médio, preço de produtos semelhantes, entre outros. Para isso, será utilizada a estrutura derivada do modelo BERT, o BERTopic, que será aplicado a dois bancos de dados não estruturados da Secretaria da Fazenda do Rio Grande do Sul. Eles são compostos por milhares de produtos que possuem apenas sua descrição.

O método BERTopic Grootendorst (2022) trabalha com diferentes técnicas para que os tópicos sejam formados de forma robusta, criando *embeddings*, *clusters* e representações de tópico que agrupam diferentes conjuntos de dados. Como resultado, o modelo se torna mais eficiente e eficaz em suas previsões, adaptando-se dinamicamente às complexidades dos dados que está processando.

O Trabalho de Conclusão de Curso está organizado da seguinte forma: o Capítulo 2 abordará temas como o pré-processamento e métodos de representações de texto, enquanto que o Capítulo 3 fornecerá uma visão inicial sobre redes neurais. O Capítulo 4 apresentará diversos modelos de redes neurais e suas aplicações, de RNNs até os modelos de Linguagem Natural de Grande Escala, além de algumas técnicas de aprendizado de máquina. No Capítulo 5, os materiais e os métodos utilizados serão exemplificados, exibindo das linguagens de programação às métricas do modelo. No Capítulo 6, serão apresentados os resultados da aplicação dos modelos BERTopic em dois tipos de corpus, compostos por descrições de produtos contidos em notas fiscais eletrônicas. Os bancos de dados foram disponibilizados pela SEFAZ-RS. Por fim, o Capítulo 7 abordará as conclusões do trabalho. Todos os códigos foram implementados utilizando as linguagens **Python** e **R**.

2 Tipos de texto

Para realizar tratamentos de texto, é essencial compreender a estrutura dos dados a serem utilizados. Diversos tipos de bancos de dados podem ser explorados, incluindo:

Dados Numéricos: Expressos em números, como por exemplo, a idade:
[1, 4, 6, 21, 42, 51, 96].

Dados Categóricos: Representam categorias, como o tipo sanguíneo:
[A, B, O, AB].

Dados de Séries Temporais: Conjuntos coletados ao longo do tempo, com dependência temporal, como a temperatura em diferentes datas:

- **23/12/2023:** 23 Graus Celsius.
- **24/12/2023:** 31 Graus Celsius.
- **25/12/2023:** 17 Graus Celsius.

Dados de Texto: Informações expressas por palavras ou frases, seguindo uma estrutura específica:

- **"Amanhã iremos ao parque."**
- **"Aquele feijão estava ótimo."**

Entre diversos outros. Uma complexidade adicional pode ser explorada considerando as seguintes categorias:

Dados Rotulados: Cada dado possui a sua categoria associada no próprio banco, por exemplo:

- **"água" = Bebida**.
- **"pastel" = Comida**.

Dados Não Rotulados: Os dados não possuem uma classe determinada no banco, como por exemplo:

- **Batata**.
- **Banana**.

Em suma, a compreensão das diversas categorias de dados, desde numéricos até rotulados e não rotulados, é essencial para conduzir tratamentos e análises avançadas.

Além disso, é preciso compreender alguns termos utilizados para dados de texto:

- **“Documento”**: Cada observação no conjunto de dados é chamada de “documento”;
- **“Corpus”**: Um conjunto de “Documentos” é chamado de “corpus”;
- **“Corpora”**: Um conjunto de “Corpus” é chamado de “corpora”.

2.1 Pré processamento

O pré-processamento é uma etapa fundamental na análise de dados de texto (Hvitfeldt e Silge (2021)). Esses métodos foram criados visando a melhor adaptação dos modelos aos corpus, buscando transformar e ajustar os dados brutos de uma forma que os torne mais compatíveis e otimizados para o treinamento e para a operação de métodos de redes neurais. Alguns deles serão citados nesse trabalho.

2.1.1 Tokenização

Tido como o primeiro passo do pré processamento, a tokenização consiste na divisão do texto em fragmentos, conhecidos como *tokens*. Diferentes tarefas requerem diferentes tipos de tokenização. Assim, diversos métodos de tokenização surgiram, sendo alguns citados abaixo:

- **Por palavra**, na qual o texto é dividido em palavra por palavra:
  - **Eu comi bolo hoje. = [“Eu” “comi” “bolo” “hoje” “.”]**.
- **Por sentença**, na qual o texto é dividido de frase em frase:
  - **Hoje o inter vai ganhar. Sou colorado. = [“Hoje o inter vai ganhar.” “Sou colorado.”]**.
- **Por letra**, na qual o texto é dividido de letra em letra:
  - **Boa noite. = [“B” “o” “a” “n” “o” “i” “t” “e” “.”]**.
- **Por *n-grams***, em que as palavras são divididas por tamanho:
  - **Unigram = [“Bala”, “Bolo”]**.
  - **Bigram = [“Couve-Flor”, “Casa velha”]**.
  - **trigram = [“Você e eu”, “Casa mal assombrada”]**.

Dependendo do objetivo, diversos outros métodos de tokenização podem ser utilizados.

2.1.2 Stop Words

Esse passo consiste na retirada de palavras que não agregam significado ao contexto. Assim, as *stop words* são descartadas sem comprometer o entendimento da frase.

Normalmente, elas são palavras de ligação, como:

- **[“De”, “Eu”, “Esse”]**.

Com a remoção das *stop words*, é possível utilizar apenas as palavras mais relevantes que agregam mais ao contexto da frase e reduzir a dimensionalidade do banco de dados, tornando os processos subsequentes mais rápidos e dinâmicos.

Uma das maneiras de obter *stop words* é através de listas previamente criadas utilizando diferentes métodos, como os conjuntos de *stop words* disponíveis no NLTK (Bird et al. (2023)) e no spaCy (spaCy Contributors (2023)).

Além disso, alguns métodos foram desenvolvidos para identificar a presença de termos nos textos e podem ser utilizados para encontrar palavras que são *stop words*, uma vez que geram resultados de frequência de palavras. Entre essas técnicas, destacam-se a *TF* (Frequência do Termo), a *IDF* (Frequência Inversa do Documento) e a *TF-IDF* (Frequência do Termo-Inversa da Frequência do Documento), que serão discutidas posteriormente neste trabalho.

2.1.3 Stemming

Os métodos de *stemming* visam extrair a raiz das palavras, ou seja, identificar a parte central do seu significado. Essa técnica é utilizada para eliminar variações como conjugações, plurais e outras flexões, garantindo que diferentes formas de uma palavra mantenham o mesmo sentido. Por exemplo:

- **“Corro” , “Correndo”, “Correu”, “Corrida” = “Corr”**.

Essa abordagem facilita a identificação de palavras similares nos textos, tratando-as como iguais em diferentes contextos. Isso reduz significativamente o tempo de busca e processamento algorítmico.

Dentro dos métodos diversos métodos de stemming, o Removedor de Sufixos da Língua Portuguesa (RSLP) proposto por Moreira et al. (2003) se destaca como uma abordagem eficaz.

2.2 Representações do texto

2.2.1 TF-IDF

O método de **TF-IDF** possui dois componentes fundamentais: o **TF** (Frequência do Termo) e o **IDF** (Frequência Inversa do Documento). A medida **TF** corresponde ao número de ocorrências de uma palavra em um documento. Palavras altamente frequentes recebem valores elevados de **TF**, enquanto palavras menos utilizadas no texto são associadas a valores mais baixos. Sua Equação é apresentada a seguir:

$$\boldsymbol{TF}(\textbf{term}) = \left(\frac{n_{\text{aparição do termo no documento}}}{n_{\text{total de termos no documento}}} \right). \tag{2.1}$$

Utilizando a Equação (2.1), se uma palavra como “Balde” aparecer em 4 de 100 documentos, seu **tf** será 0,04. Em contrapartida, se uma palavra como “Soube” aparecer em 99 de 100 documentos, seu **tf** será 0,99. Com esses resultados é possível listar as palavras mais frequentes no documento.

O **IDF**, por sua vez, atribui valores baixos para palavras comuns e valores altos para palavras que não são frequentemente utilizadas. Sua Equação é:

$$\boldsymbol{IDF}(\textbf{term}) = \ln\left(\frac{n_{\text{documentos}}}{n_{\text{documentos contendo termo}}}\right) \tag{2.2}$$

Com a Equação (2.2), se uma palavra como "Balde" aparecer em 4 de 100 documentos, sua **idf** será 3,22. Por outro lado, se uma palavra como "Soube" aparecer em 99 de 100 documentos, seu **idf** será 0,01. Dessa forma o método destaca palavras que são específicas no texto, além de reduzir a dimensionalidade do banco.

Combinando os dois métodos, o **TF-IDF** calcula um peso para cada termo, levando em consideração tanto a sua relevância em um documento específico (TF) quanto a sua importância em todo o corpus (IDF). O ***TF-IDF*** é definido da seguinte forma:

$$\boldsymbol{TF\text{-}IDF}(\textbf{term}) = \text{TF}(\text{term}) \times \text{IDF}(\text{term}). \tag{2.3}$$

A partir da Equação (4.9), se uma palavra como "Balde" obtiver um **TF** igual a 0,04 e um **IDF** igual a 3,22, seu **TF-IDF** será 0,1288. Em contrapartida, se uma palavra como "Soube" obtiver um **TF** de 0,99 e um **IDF** de 0,01, seu **TF-IDF** será 0,0099. Esses valores indicam a importância relativa de cada termo nos documentos, considerando tanto a frequência local quanto a raridade global.

Essa abordagem resulta em uma métrica que destaca termos significantes em um contexto local, ao mesmo tempo em que penaliza termos que são comuns em vários documentos.

Algumas aplicações do ***TF-IDF*** são: sumarização de texto (Panchal (2019)), clusterização (Salnikov (2018)), entre diversas outras que envolvem texto.

2.3 Word Embeddings

As *word embeddings* são representações de palavras por meio de vetores (Mikolov et al. (2013b)) de modo que palavras semanticamente relacionadas possuem representações vetoriais próximas umas das outras. Dessa forma, as *word embeddings* mapeiam palavras para um espaço vetorial onde a proximidade geométrica entre os vetores reflete a proximidade semântica entre as palavras. Por exemplo:

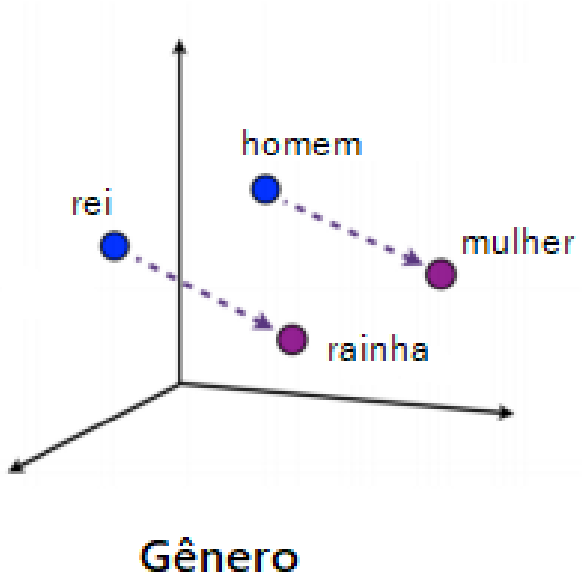
O doce é vermelho.
O doce é roxo.

Vermelho é semanticamente semelhante à palavra **roxo**, logo seus vetores terão valores parecidos. A Figura (2.1) apresenta os *word embeddings* de 4 palavras em um espaço tridimensional. Observe que a distância entre as palavras **rei** e **rainha** é muito similar à distância entre as palavras **homem** e **mulher**, tornando possível a seguinte operação:

Rei - Homem + Mulher = Rainha.

Bengio et al. (2003) introduziu o conceito fundamental de representação vetorial de palavras usando redes neurais, mas somente em 2013 Mikolov et al. (2013b) propôs as arquiteturas textitCBow e o *Word2Vec* que possibilitaram a construção dos *word embeddings*. Em 2014, Pennington et al. (2014) propuseram uma modificação das arquiteturas propostas por Mikolov, o *GloVe* (Global Vectors for Word Representation), utilizando informações sobre a frequência e a distribuição das palavras nos documentos. O *GloVe* incorporou a ideia de que as palavras que ocorrem

Figura 2.1: Exemplo vetorial de um Word Embedding.[1]



Fonte: Artigo da Medium

com frequência juntas e com menos frequência em relação a outras palavras compartilham significados semelhantes. Ao combinar informações de coocorrência global e local das palavras, o *GloVe* produz representações vetoriais de palavras que capturam de forma mais precisa as nuances semânticas e relacionamentos entre palavras em um corpus de texto.

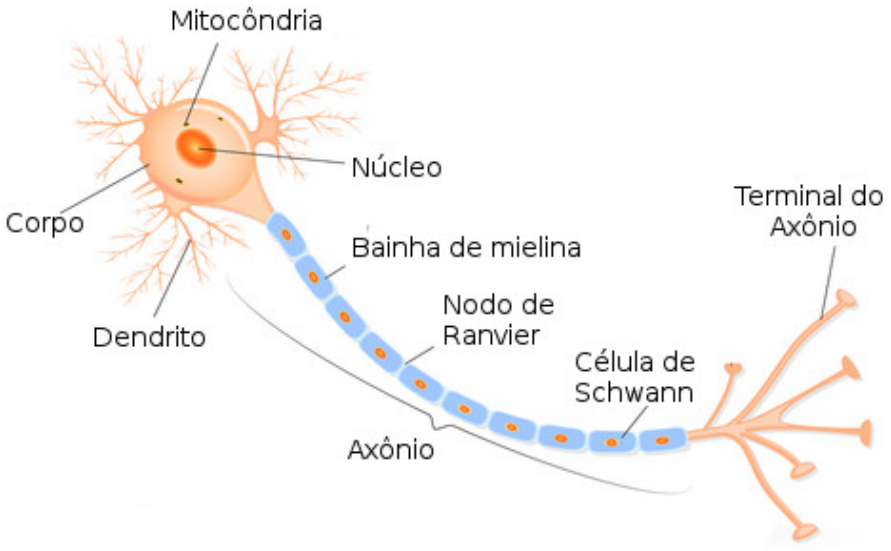
Utilizando os modelos de obtenção de *word embeddings*, é possível conseguir uma representação mais rica e contextualizada das palavras, contribuindo para uma melhor compreensão semântica e melhor desempenho em uma variedade de tarefas relacionadas ao processamento de linguagem natural.

3 Redes Neurais

Os seres humanos, desde o desenvolvimento de sua capacidade racional, têm buscado ampliar seu conhecimento e estimulação intelectual. Diversas crianças, ao atingir certa idade, iniciam suas indagações como “Como a gravidade funciona?” ou “Qual é o formato da Terra?” para compreender melhor o mundo ao seu redor e adquirir um maior senso crítico. Muitas dessas perguntas receberam explicações fundamentadas e teóricas ao longo do tempo.

No entanto, uma questão que permanece como uma incógnita é o funcionamento do cérebro humano. Diversos estudos foram conduzidos, e muitas perguntas ainda estão pendentes sobre esse órgão. Porém, uma afirmação que pode ser feita é a presença de neurônios como a unidade fundamental do cérebro. Conforme a Figura (3.1), o neurônio humano recebe sinais elétricos no “corpo”, processa esses sinais e os transmite para terminações nervosas ou outros neurônios através dos “terminais do axônio”.

Figura 3.1: Exemplo de um neurônio.[1]



Fonte: Recurso informativo da Info Escola

Essa concepção inspirou o trabalho de McCulloch e Pitts (1943), que propôs a criação de uma rede neural artificial. Seu objetivo era replicar, de maneira simplificada, a estrutura do neurônio humano, recebendo informações, processando-as e devolvendo o resultado do processamento.

Ao longo do tempo, múltiplos trabalhos que evoluíram o conceito das redes neurais surgiram, como o *perceptron* proposto por Rosenblatt (1958), entre diversos outros autores, que introduziram novos termos e métodos. Com isso, essas redes neurais artificiais têm se mostrado promissoras em diversas aplicações, oferecendo soluções inovadoras em campos como inteligência artificial e aprendizado de máquina.

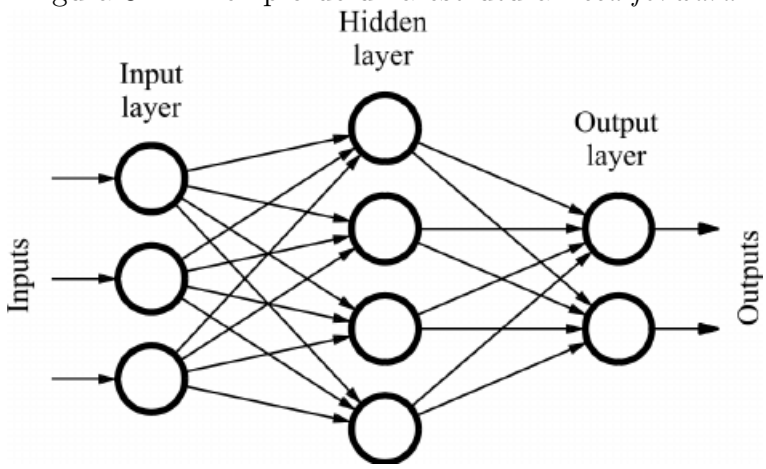
Há vários modelos de arquiteturas de redes neurais que trabalham com tarefas específicas, como redes convolucionais e redes recorrentes, *transformers* e novas estruturas que surgem a cada dia. Algumas delas serão apresentadas com um certo detalhe neste capítulo.

3.1 Rede Feed-Forward

Buscando essa ideia de receber informação, realizar operações e entregar um resultado final, surgem as redes *feed forward* (Rumelhart et al. (1986)), que, combinando múltiplos *perceptrons*, conseguem realizar tarefas mais complexas do que um simples neurônio.

Essa rede recebe valores de entrada (*inputs*) que são recebidos pela camada de entrada (*Input Layer*) e são passados para a camada oculta (*Hidden Layer*) que aplica transformações e, por fim, são entregues a camada de saída (*Output Layer*), gerando os resultados (*Outputs*), tal processo pode ser representado pela Figura (3.2). Há diversas configurações que uma rede neural pode assumir, como adicionar mais neurônios de entrada, de saída ou mais camadas ocultas (Brownlee (2018)).

Figura 3.2: Exemplo de uma estrutura *Feed-forward*.[2]



Fonte: Quiza e Davim (2011)

Uma das características de uma rede neural *feed forward* é a sua total conectividade, sendo ela a ligação de todos os neurônios de uma camada na sua camada subsequente. Essas conexões possuem pesos, ajustáveis, que ajudam na tarefa final da rede, seja ela de classificação ou regressão.

Em uma rede neural, a camada de entrada recebe **j** *inputs*, representados por um conjunto de valores $A_{0,i} = A_{0,1}, \ldots, A_{0,j}$, em que $A_{0,i}$ é um vetor de entrada de *inputs*. Esses *inputs* são transmitidos para cada um dos **m** neurônios nas próximas

k camadas, expressos por $A_{i,l} = A_{1,1}, \ldots, A_{k,m}$. Vale notar que **m** pode variar para cada camada k. Posteriormente, na camada de saída **k** são gerados os *outputs*.

Para a realização do cálculo de cada neurônio, é utilizada as seguintes Equações:

$$Z_{k,m} = \sum_{j=1}^{n} w_{k,m,j} \cdot A_{k-1,m} + b_{k,m}, \tag{3.1}$$

$$A_{k,m} = g(Z_{k,m}), \tag{3.2}$$

em que **k** representa o número de camadas na rede neural, **m** é o número de neurônios na camada, **j** é o número de *inputs* de cada neurônio, **w** são os pesos de cada conexão, **A** os valores de entrada que a camada irá receber e **b** um viés, que é um erro aleatório. Ao multiplicar os pesos pelos valores de entrada e somar o viés, é obtido **Z**, que será aplicado em uma função de ativação **g**.

Na Figura (3.2) o valor de **k** é 2, uma vez que a camada de entrada consiste simplesmente nos valores dos *inputs*. Para a camada de entrada, **j** é 3, resultando em 12 pesos **w**. Na camada oculta, **j** é 4, com 8 pesos **w** e 4 vieses **b**. Por fim, na camada de saída, o valor de **j** e do número de vieses **b** é 2, gerando os *outputs*.

Uma nomenclatura amplamente reconhecida para redes com essa estrutura é o *Multi-Layer Perceptron* (MLP), uma variação das redes *feed forward*. Essa variante, por sua vez, exige a presença de pelo menos uma camada oculta, ao passo que as redes *feed forward* não necessariamente precisam delas. Assim, é possível classificar a rede da Figura (3.2) como um MLP com uma camada oculta ou simplesmente uma rede *feed forward*.

As funções de ativação desempenham um papel fundamental em redes neurais, permitindo a modelagem de relações não-lineares entre os dados (consulte Apicella et al. (2021), uma revisão geral) . Em contextos nos quais a complexidade é desnecessária, como em problemas de regressão com dados linearmente relacionados, é comum empregar uma função de ativação linear. Essa função preserva a linearidade dos dados, facilitando a representação de relações simples e diretas. Nesse caso,

$$g(Z_{k,m}) = Z_{k,m}$$

resultando em,

$$A_{k,m} = Z_{k,m}.$$

Para problemas mais complexos e não lineares, foram desenvolvidas diversas funções de ativação. Algumas delas incluem a ReLU (*Rectified Linear Unit*), introduzida por Fukushima (1975) e popularizada por Nair e Hinton (2010), cuja expressão é detalhada na Equação (3.3). Outras funções são a Sigmóide ou função logística, apresentada por Rumelhart et al. (1986), com sua formulação mostrada na Equação 3.4, e a *softmax* Bengio et al. (2016), descrita na Equação (3.5):

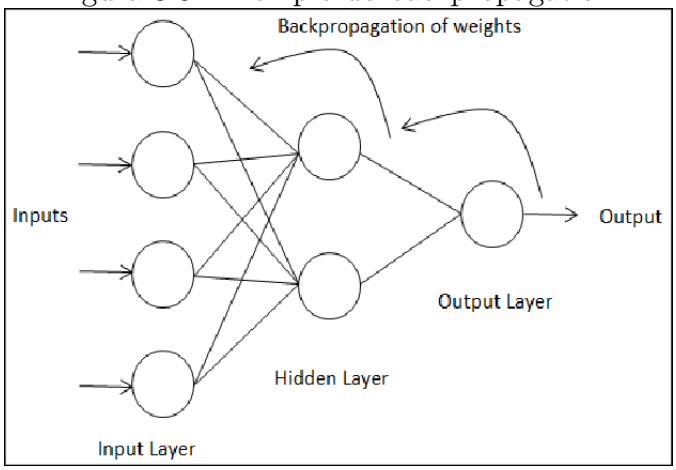
$$f(x) = \max(0, x), \tag{3.3}$$

$$f(x) = \frac{1}{1 + e^{-x}}, \tag{3.4}$$

$$\text{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}}. \tag{3.5}$$

Para que uma rede *feed forward* aprenda e possa fornecer resultados mais precisos, aplica-se um método chamado *backpropagation* ou retro-propagação (Rumelhart et al. (1986)). Esse método ajusta os pesos **w** da rede de modo que a saída prevista seja a mais próxima possível do valor real. Conforme o nome sugere, o *backpropagation* percorre a rede neural no sentido contrário, exemplificado na Figura (3.3), começando pelo erro final e retrocedendo até os pesos da camada de entrada. Esse processo busca atualizar os pesos da rede e assim minimizar os erros de predição.

Figura 3.3: Exemplo do backpropagation.[3]



Fonte: Link para a imagem.

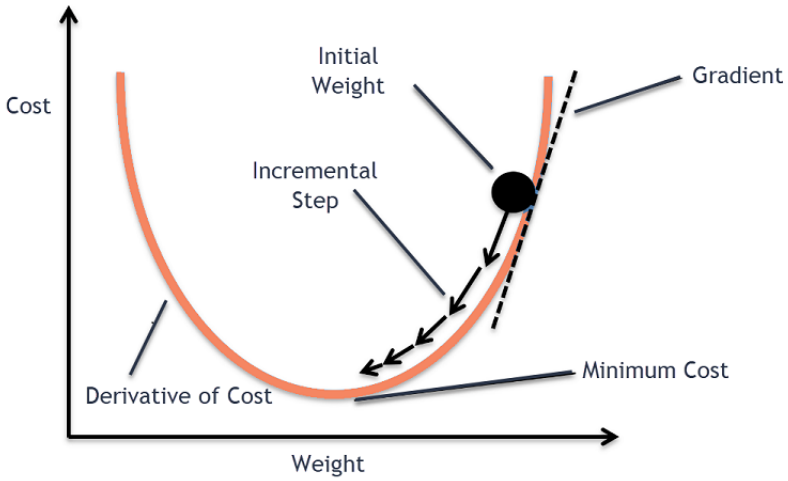
3.2 Erros e ajuste dos pesos do modelo

A função de perda, chamada de *loss*, procura quantificar o erro do modelo, calculando a diferença da predição final com o valor real. Ela deve ser minimizada para que os parâmetros da rede neural sejam os melhores possíveis. Dependendo do objetivo, ela deve ser adaptada. Algumas são:

- **Erro quadrático médio**, para problemas de regressão;
- **Erro médio absoluto**, para regressão;
- **Entropia cruzada**, para classificação binária.

O algoritmo da descida do gradiente (Robbins e Monro (1951)) propõe uma maneira de otimização dos pesos para que os erros sejam minimizados de forma eficiente. A Figura (3.4) mostra o processo de diminuição do erro.

Figura 3.4: Exemplo de uma descida do gradiente.[4]



Fonte: Vidhya (2020)

Conforme ilustrado na Figura (3.4), *Cost* representa o erro de predição e *weight* os pesos. Essa função de descida do gradiente visa encontrar os pesos que minimizam o erro, isto é, o ponto de *minimum cost* (menor erro). Ao longo de sucessivas iterações, o gradiente "caminha" em direção ao ponto de menor custo, resultando na otimização dos pesos. Dessa forma, a descida do gradiente retorna os valores otimizados para cada peso, aproximando ao máximo o resultado final do valor real. A Equação para o cálculo do *backpropagation* é:

$$\delta_{k,m} = \sum_{j=1}^{n} w_{k,m,j} \cdot \delta_{k+1,j} \cdot g'(Z_{k,m}). \tag{3.6}$$

Para todos os neurônios, excluindo a camada de saída, a Equação 3.6 é utilizada. Na última camada, esse cálculo é mais simples, sendo apenas $\delta_{k,m} = A_{k,m} - Y$, em que $\mathbf{Y}$ é o valor real. A partir dos cálculos do *backpropagation*, são atualizados os pesos $\mathbf{w}$ da rede neural utilizando a seguinte Equação:

$$\frac{\partial L}{\partial w_{k,j,m}} = A_{k-1,j} \cdot \delta_{k,m}, \tag{3.7}$$

$$w_{k,j,m} := w_{k,j,m} - \alpha \frac{\partial L}{\partial w_{k,j,m}}. \tag{3.8}$$

Na qual a Equação (3.7) é o gradiente em relação aos pesos, que é aplicado na Equação (3.8) e gera os novos valores de $\mathbf{w}$. Diferentemente, para o viés $\mathbf{b}$, a Equação é a seguinte:

$$\frac{\partial L}{\partial b_{k,m}} = \delta_{k,m}, \tag{3.9}$$

$$b_{k,m} := b_{k,m} - \alpha \frac{\partial L}{\partial b_{k,m}}. \tag{3.10}$$

O parâmetro α em ambas as Equações indica a taxa de aprendizado, que desempenha um papel crucial, determinando o tamanho dos passos durante o processo

de descida do gradiente. Valores muito altos podem resultar em saltos grandes no espaço do gradiente, o que pode levar a não convergência do algoritmo, enquanto que valores muito baixos podem tornar o processo de aprendizado excessivamente lento, permitindo que o modelo fique preso em mínimos locais.

Para realizar a retro-propagação, três métodos são comumente empregados:

- **Lote da Descida Estocástica do Gradiente (LDEG):** Cada observação é utilizada uma por uma para o cálculo dos gradientes e atualização dos pesos;
- **Lote:** O conjunto de dados completo é utilizado para o cálculo dos gradientes e atualização dos pesos;
- **Mini-lote:** Pequenas partes do conjunto de dados são utilizadas para o cálculo dos gradientes e atualização dos pesos.

Normalmente é utilizada a forma de **Mini-lote**, dado que ela combina a eficiência da LDEG com a convergência do Lote.

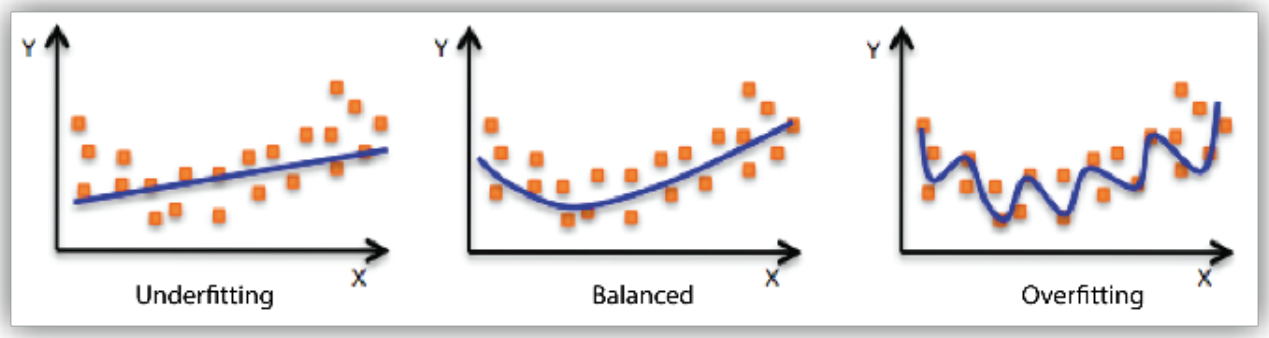
Outras métricas essenciais no contexto de redes neurais são as épocas e os otimizadores. As épocas representam o número de iterações completas em que a rede passa por todo o conjunto de dados durante o treinamento. Em cada época se ajustam os pesos da rede, contribuindo para que o modelo se adapte de maneira mais precisa aos padrões presentes nos dados de treinamento. Os otimizadores, por sua vez, buscam minimizar a função de custo, sendo o mais popular a descida estocástica do gradiente, juntamente ao Adam (Kingma e Ba (2015)) e ao RMSprop (Hinton (ture)).

3.3 Ajuste do modelo

Um dos desafios enfrentados pelas redes neurais surge quando ocorre *overfitting* ou *underfitting*. Esses fenômenos podem comprometer a capacidade da rede de generalizar padrões e fazer previsões precisas. Na Imagem (3.5) é possível observar o que acontece com cada comportamento. Primeiramente, quando ocorre o *underfitting*, os valores esperados não estão bem ajustados. Diferentemente, quando ocorre o*overfitting*, os valores esperados estão ajustados exatamente como os dados estão distribuídos, o que dificulta a generalização. Enquanto isso, o ajuste *balanced* está bem ajustado, sem utilizar uma curva muito simples ou muito complexa.

É essencial encontrar um equilíbrio durante o treinamento da rede para evitar esses problemas e obter um modelo que seja capaz de generalizar de forma eficaz para dados não vistos. Uma técnica comumente empregada para atingir esse objetivo é o *dropout* (Srivastava et al. (2014)). Seu funcionamento se dá na remoção aleatória de alguns neurônios durante o treinamento, impedindo que a rede dependa de forma excessiva de neurônios específicos.

Figura 3.5: Exemplo de *over*, *under* e *perfect fit*.[5]



Fonte: Link para a imagem.

Além disso, a fim de evitar problemas de *overfitting* e *underfitting*, é feita a divisão do conjunto de dados em três partes distintas: o treino, o teste e a validação. Durante a fase de treino, o modelo é exposto a dados específicos, procurando padrões, aprendendo com eles e ajustando seus parâmetros. A etapa de validação ajuda o modelo a evitar *overfitting*, garantindo que não exista um ajuste excessivo aos dados de treino a partir de dados não vistos. Finalmente, a fase de teste avalia o desempenho do modelo em dados não vistos anteriormente, fazendo a rede generalizar as informações. Normalmente, os dados são divididos em 70%, 15% e 15% para treino, teste e validação, respectivamente, variando do objetivo (Brownlee (2020)).

3.4 Métodos de treinamento

Alguns métodos de treinamento foram criados para lidar com dados em um cenário conhecido como *n-shot learning* (Fei-Fei et al. (2006)), que seriam maneiras de treinar a rede neural buscando a otimização de poucos ou nenhum exemplo. Isso ocorre em maior frequência para dados não rotulados, como, por exemplo, nas notas fiscais eletrônicas (NF-e). Nas NF-e, existe um campo de descrição do produto. No cenário ideal, uma pessoa leria a descrição e faria a rotulação manual (indicando o produto daquela nota) de cada uma das NF-e‘s, o que seria inviável. Dessa forma, métodos automáticos são necessários, tornando viável tarefas que de outra forma são consideradas impraticáveis. Essa forma de treinamento automático de dados não rotulados possui três formatos. São eles:

Zero-Shot Learning: Consiste no treinamento da rede neural sem utilizar dados rotulados;

One-Shot Learning: Consiste no treinamento da rede neural com um exemplo rotulado para cada classe;

Few-Shot Learning: Consiste no treinamento da rede neural com alguns exemplos de dados rotulados.

4 Aprendizado Profundo e Processamento de Linguagem Natural

Para enfrentar tarefas de maior complexidade, são utilizados métodos de *deep learning*, que se baseiam em redes neurais artificiais compostas por duas ou mais camadas ocultas (LeCun et al. (2015)). Alguns exemplos de aplicação são a visão computacional, o processamento de linguagem natural e o desenvolvimento de chatbots.

Nesse capítulo serão abordados métodos de *deep learning*, sendo eles as *RNNs, LSTM, Transformer, GPT, BERT, BART, BERTopic* e *LLMs* voltados ao processamento de texto. Além disso, serão apresentados modelos de *machine learning* para texto, como o *Word2Vec* (*CBoW e Skip-gram*) e o *GloVe*.

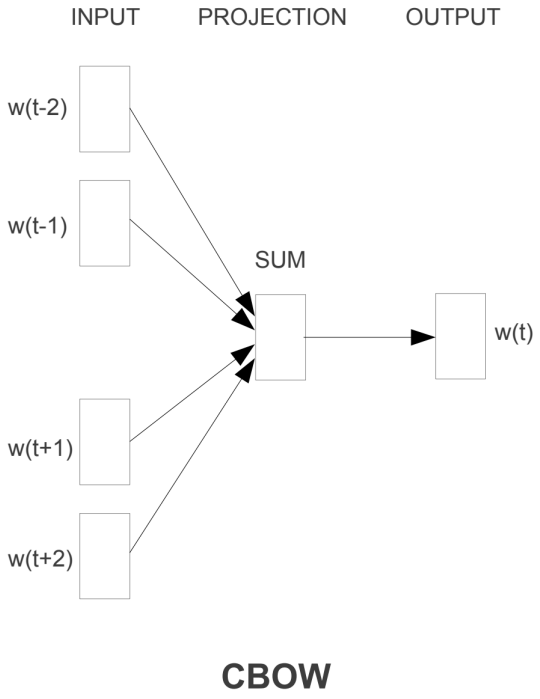
4.1 Word2Vec

No artigo de Mikolov et al. (2013a), são apresentadas duas arquiteturas para obtenção de *word embeddings*, o *Continuous Bag of Words* (CBoW) e o *Skip-gram*. Ambos são amplamente utilizados em modelos de texto.

4.1.1 CBoW

Na arquitetura do *CBoW*, o foco é prever a palavra central a partir das palavras em sua volta. Para isso, o modelo considera uma janela de palavras, no qual essa janela de palavras circundantes, conhecida como contexto, é utilizada para prever a palavra no meio.

Figura 4.1: Exemplo de uma estrutura *CBoW*.[1]



Fonte: Mikolov et al. (2013a)

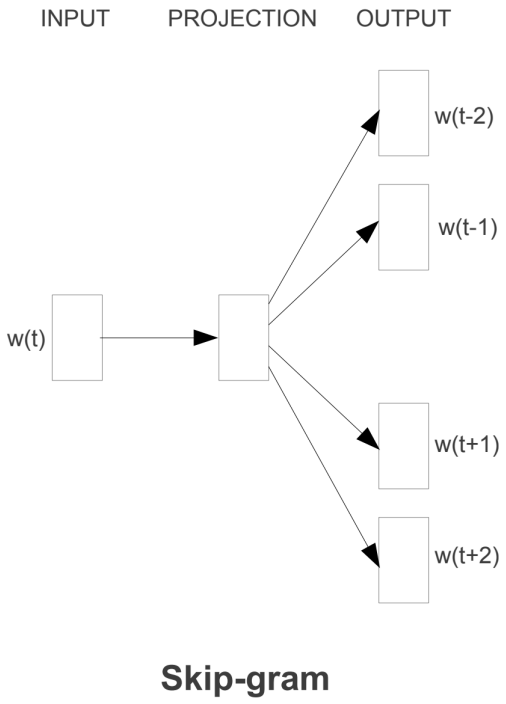
Conforme ilustrado na Figura (4.1), o modelo recebe palavras de contexto, por exemplo, $(w_{t-2}, ..., w_{t+2})$, e tem como objetivo prever a palavra central, w_t. É importante ressaltar que o número de palavras de contexto (ou janela) pode ser ajustado conforme necessário. Uma janela maior pode capturar mais contexto, mas também pode tornar o modelo mais suscetível ao ruído.

A força do *CBoW* reside na sua capacidade de capturar as semelhanças semânticas e sintáticas das palavras (Kulshrestha (2019)).

4.1.2 Skip-gram

Enquanto o *CBoW* procura prever a palavra central a partir de um contexto circundante, o *Skip-gram* inverte essa abordagem e foca em prever palavras de contexto a partir de uma única palavra central. O *Skip-gram* assume que a semântica de uma palavra pode ser usada para inferir as palavras que aparecem ao seu redor.

Figura 4.2: Exemplo de uma estrutura *Skip-gram*.[2]



Fonte: Mikolov et al. (2013a)

Conforme exemplificado na Figura (4.2), o modelo toma uma palavra central w_t e busca prever as palavras circundantes, $(w_{t-2}, ..., w_{t+2})$. Isso é feito para todas as palavras em uma janela deslizante que percorre todo o texto.

No treinamento, o modelo Skip-gram é tipicamente mais lento do que o CBoW, pois precisa atualizar os pesos para cada palavra no contexto. No entanto, o Skip-gram muitas vezes supera o CBoW, especialmente quando a quantidade de dados é limitada (Kulshrestha (2019)).

Uma das vantagens do *Skip-gram* é sua capacidade de capturar relações mais sutis em grandes *corpus*, uma vez que as representações vetoriais de palavras raras ou menos frequentes é mais acurada, fazendo com que ele seja especialmente útil quando há termos mais especializados (Kulshrestha (2019)).

4.2 GloVe

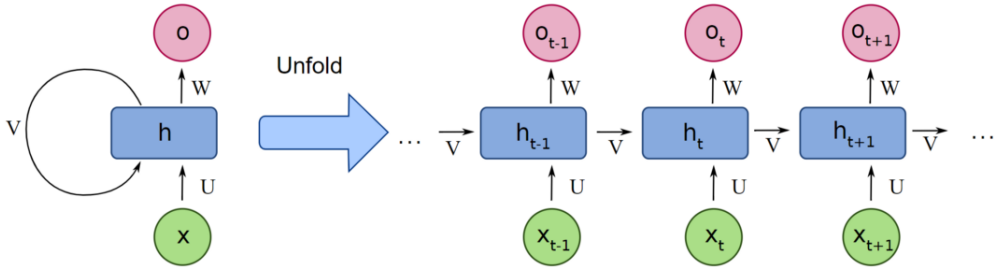
A arquitetura *Global Vectors for Word Representation* (*GloVe*) proposta por Pennington et al. (2014) possui um funcionamento que, à primeira vista, pode parecer similar ao do Word2Vec. No entanto, existem diferenças cruciais em suas abordagens. Sua principal distinção está na forma como ambos os modelos encaram o contexto das palavras no texto. Enquanto o *Word2Vec* trata palavras em uma janela ao redor da palavra-alvo, o *GloVe* busca observar cada palavra em sua ocorrência geral, construindo uma matriz de coocorrência para todo o texto. Essa matriz conta a frequência que duas palavras ocorrem juntas ao longo do texto. Por exemplo, se no livro a frase *"No Brasil amamos futebol"* aparecer e em outro momento surgir a frase *"No Brasil comemos feijão"*, as palavras *"No"* e *"Brasil"* terão uma coocorrência

de 2. Da mesma forma, isso será feito para todos os pares de palavras do texto.

4.3 Rede Neural Recorrente

Para ser possível adentrar nas redes de *transformers*, é preciso entender a funcionalidade de uma rede recorrente, a base para todos os métodos que necessitam de sequencialidade. Ela é representada na Imagem (4.3).

Figura 4.3: Exemplo de uma rede neural recorrente.[3]



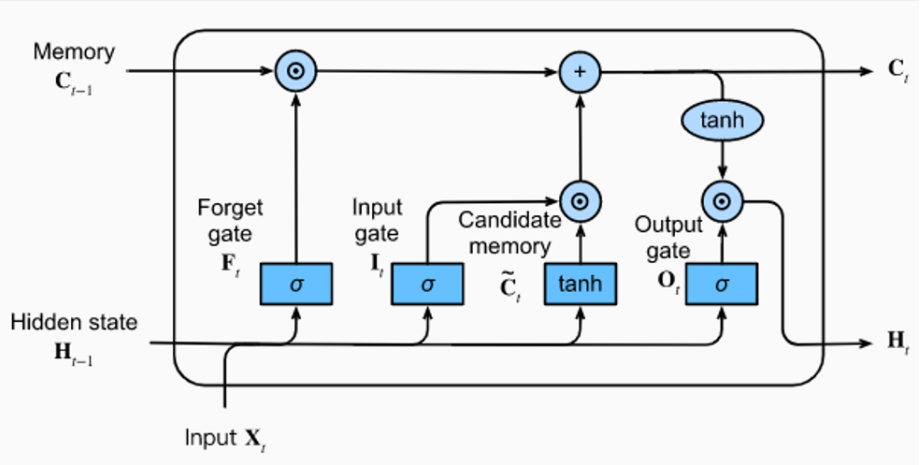
Fonte: Artigo escrito por Debasish Kalita

Observando a Figura (4.3), é possível notar uma clara diferença àquela *feed-forward*. As alterações na camada oculta (representadas por h) são entregues dentro da rede na próxima observação. Por exemplo, em um contexto de séries temporais no qual se deseja prever a temperatura do próximo dia, é possível utilizar tanto a informação do dia atual quanto a dos dias anteriores para essa previsão. Importante ressaltar que o interesse da rede é nas informações processadas pela camada oculta dos dias anteriores, que são utilizadas como informação no modelo do dia de hoje e ajudam a realizar uma previsão, e não em suas saídas diretas, ou seja, *outputs*.

4.4 LSTM

Um outro tipo de rede neural que trabalha com sequencialidade é o LSTM (*Long Short Term Memory*) (Hochreiter e Schmidhuber (1997)). Esse modelo trabalha com portões que controlam as informações do passado e criam pesos para as informações atuais. Sua estrutura está representada na Imagem (4.4).

Figura 4.4: Exemplo de uma rede neural LSTM.[4]



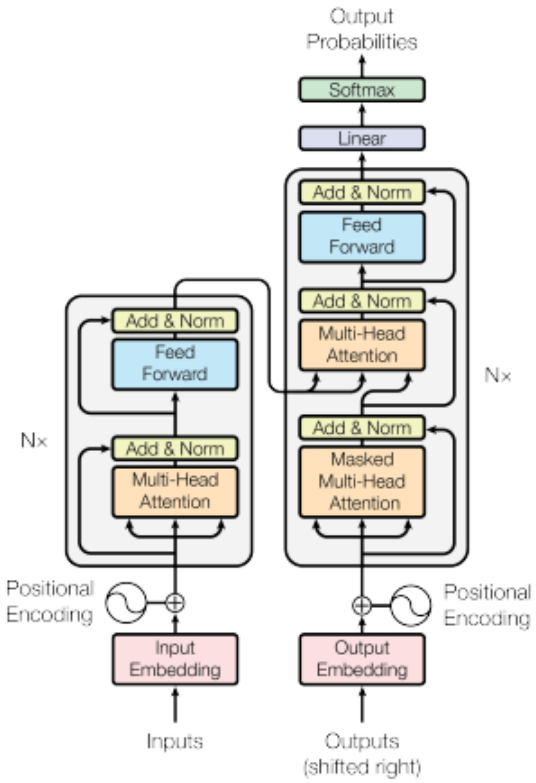
Fonte: Artigo escrito por Ottavio Calzone

Na Figura (4.4), estão representados o *forget gate*, o *input gate* e o *output gate*. Inicialmente, o *forget gate* decide a quantidade de informação que a célula LSTM atual irá esquecer da memória de longo prazo. Subsequentemente, o *input gate* determina a quantidade de informação a ser adicionada à memória de longo prazo. Por fim, o *output gate* regula tanto a quantidade de informação de curto prazo a ser transmitida para a próxima célula quanto a geração do *output* final.

4.5 Transformer

Introduzido por Vaswani et al. (2017), o *transformer* representou uma revolução no campo de *NLP*. Com sua arquitetura baseada em *encoder* e *decoder* exemplificada na Imagem (4.5), ele abriu portas para que uma variedade de tarefas relacionadas a dados de texto possam ser executadas de maneira mais eficiente, sendo base para muitos métodos atuais como o GPT e o BERT. A grande contribuição do método e o que o tornou um marco no campo foi a introdução dos mecanismos de atenção. Esses mecanismos, juntamente com as estruturas de cada componente do *transformer*, além dos *encoder* e *decoder* serão detalhados a seguir.

Figura 4.5: Exemplo de uma estrutura *Transformer*.[5]



Fonte: Vaswani et al. (2017)

4.5.1 Mecanismos de atenção

O *transformer* leva em consideração a ordem em que as palavras são indicadas no texto. Por exemplo, considerando o caso em que há o desejo de traduzir a seguinte frase:

- **"Do you have a pen I can borrow?"**.

Se fosse traduzida literalmente, resultaria em:

- **"Você tem uma caneta eu posso emprestado?"**.

Entretanto, essa tradução está gramaticalmente incorreta. A frase correta seria:

- **"Você tem uma caneta que eu possa pegar emprestado?"**.

A função do mecanismo de atenção é justamente visualizar a frase inteira, permitindo traduções mais precisas e análises contextuais aprofundadas. Ele faz com que uma palavra como "borrow" tenha uma alta atenção com "pegar emprestado", realizando a tradução de forma correta. Outro problema a ser observado é o contexto, por exemplo:

- **"Eu como bolo."**.

A palavra "bolo" possui um significado totalmente diferente de:

- **"Você me deu um bolo."**.

Na língua portuguesa "Bolo" pode indicar um tipo de "torta" quanto o ato de faltar a um compromisso. Essa diferença é avaliada pelo mecanismo de *self-attention* ou autoatenção, que busca captar o contexto da frase através de uma matriz de correlação entre as palavras. Nela, o algoritmo gera pesos maiores, por exemplo, para "como" e "deu" com "bolo", assim, conseguindo distinguir o significado de bolo para as duas sentenças.

4.5.2 Input Embeddings

O primeiro passo de um modelo *Transformer* é receber os *inputs*. Esses *inputs* são transformados em *tokens* e, posteriormente, são atribuídos *word embeddings* a cada *token*, ou seja, um vetor a cada palavra. Em uma frase, cada vetor resultante é organizado em sequência e compõe uma matriz cujo tamanho é determinado pelo comprimento da frase e a escolha do usuário. Vale notar que, à medida que a dimensão cresce, o modelo se torna mais lento e mais complexo. Por exemplo, um vetor de tamanho 10 será mais rápido que um vetor de tamanho 20, mas terá uma representação pior àquele maior. Por outro lado, ao utilizar dimensões muito grandes, há o risco de cometer *overfitting*.

Por exemplo, considerando a frase "Onde está a minha meia", serão gerados 5 *tokens* (n_{tokens}). Após isso, cada palavra recebe um *word embeddings*, resultando em uma matriz de tamanho $n_{\text{tokens}} \times d_{\text{model}}$, em que d_{model} é a dimensão escolhida pelo usuário. Se for escolhido, por exemplo, $d_{\text{model}} = 512$, a matriz resultante será de 5 $\times$ 512.

4.5.3 Positional Encoding

Diferentemente de modelos que trabalham com os dados sequencialmente como os modelos recorrentes e o LSTM, o *transformer* trabalha com as informações em paralelo. Assim, para que o modelo possa realizar esse processamento dessa maneira, o método de *positional encoding* adiciona ordem às palavras dentro da frase de uma maneira numérica, dado que uma mudança de ordem pode mudar totalmente o sentido da frase. Dessa forma, o método consegue trabalhar com os dados em paralelo respeitando uma ordem.

Utilizando o exemplo da Seção dos *Input Embeddings*, no qual o $d_{\text{model}} = 512$, são gerados valores dos *positional encodings* de tamanho 512 para que ele possa ser somado à matriz de *embeddings*. Sua função é dada pela seguintes Equações:

$$\text{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \tag{4.1}$$

$$\text{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right). \tag{4.2}$$

Em ambas as Equações, *pos* representa a posição do *token* na sequência, e i representa a dimensão do *embedding*.

Para evitar a adição de valores como frações e valores extremos, a utilização de funções seno e cosseno ajuda a manter os valores em uma escala controlada, evitando distorções nos *embeddings*.

Nas ondas de cosseno e seno, os valores são limitados pela amplitude da função, uma vez que ela varia dentro de limites fixos no eixo. Isso assegura que os valores permaneçam sempre dentro de um pequeno intervalo.

No entanto, como o intervalo é pequeno, é mais complicado captar diferenças entre os *embeddings*. Por isso, ao variar o valor de "i", o comportamento da função é modificado, ampliando as ondas e tornando valores muito próximos nas primeiras funções facilmente distinguíveis em frequências mais altas. Assim, mesmo para valores consideravelmente grandes, as diferenças continuam perceptíveis.

Para valores pares, são utilizadas as curvas de cosseno, enquanto que para valores ímpares, curvas de seno.

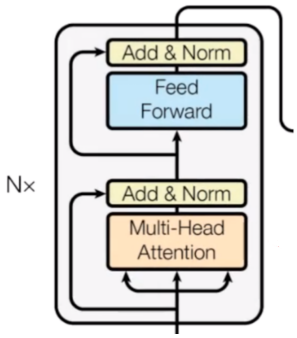
Essencialmente, o que o *positional encoder* faz é adicionar ordem na palavra, como no simples exemplo: **Eu comi bolo hoje = Eu (1), Comi (2), Bolo (3), Hoje (4)**.

Com os *positional encodings* adicionados, os *embeddings* entram nos *decoders* e *encoders* com uma representação de posição somada.

4.5.4 Encoder

O *encoder* de um *transformer*, utilizando mecanismos de atenção, é capaz de, ao receber um *word embedding* com *positional encodings*, retornar uma representação vetorial de uma frase com contexto. Ele captura as relações e dependências entre partes da sentença. Sua estrutura base é mostrada na Figura (4.6).

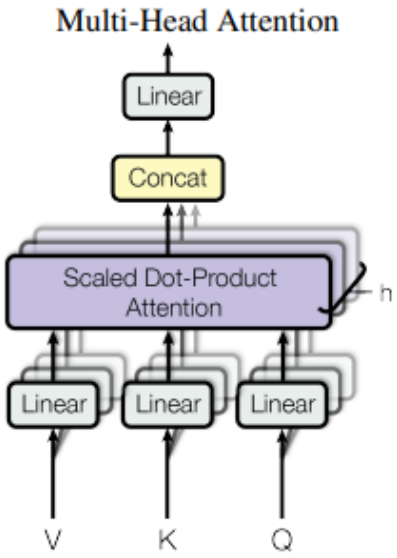
Figura 4.6: Exemplo de uma estrutura *Encoder*.[6]



Fonte: Vaswani et al. (2017)

Após as transformações ocorridas nos *positional encodings*, a frase entra no *encoder* em forma de matriz e passa pelo mecanismo de *Multi-Head Attention*. Nessa etapa, o *transformer* irá realizar o cálculo da atenção, peça chave do modelo.

Figura 4.7: Exemplo da *Multi-Head Attention*.[7]



Fonte: Vaswani et al. (2017)

Nessa parte, os *embeddings* serão divididos em 3 componentes:

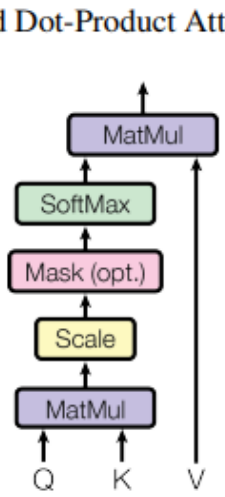
- A ***Query*** pode ser entendida como o elemento procurado ou, em outras palavras, o foco da atenção. Para ilustrar, ao realizar uma busca com a frase "Vídeo comendo chocolate na Espanha", a *query* representa aquilo que é procurado. Ela pode ser representada pela frase completa ou focar em uma palavra central, como "chocolate".

- A ***Key*** representa o que cada elemento oferece à *Query*. Usando o exemplo da *Query*, a *Key* seria a contribuição de cada elemento para o foco da atenção, ou seja, o que "Vídeo", "comendo", "chocolate", "na" e "Espanha" oferecem para a frase completa ou para a palavra central "chocolate".

- O ***Value***, é a representação do significado de cada palavra. No exemplo, seria o que "Vídeo", "comendo", "chocolate", "na" e "Espanha" realmente significam.

Essas representações ocorrem por conta de que os *embeddings* de cada palavra na *key* são multiplicados pelo foco de atenção (*query*), destacando o quão relevante cada palavra da *key* é para a frase ou a palavra central na *query*. No final, os *values* são multiplicados, agregando o significado de cada palavra à frase com os pesos correspondentes. Essa é uma simplificação do *Scaled Dot-Product Attention*, que será aprofundado a seguir.

Figura 4.8: Exemplo da *Scaled Dot-Product Attention*.[8]



Fonte: Vaswani et al. (2017)

Voltando em como esses três componentes são criados, os *embeddings* formados com seus *positional encodings* passam por três transformações lineares distintas que calculam os componentes *query*, *key* e *value* . Essas transformações são representadas da seguinte forma: $W_{Qi} \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_{Ki} \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_{Vi} \in \mathbb{R}^{d_{\text{model}} \times d_v}$, sendo W_{Qi} a matriz de pesos da *query* da cabeça (*head*) "i", W_{Ki} a matriz de pesos da *key* da cabeça "i" e W_{Vi} a matriz de pesos da *query* da cabeça "i". Essas matrizes pertencem ao conjunto dos números reais e possuem dimensão $d_v = d_k = \frac{d_{\text{model}}}{h}$, no qual "h" e "i" são componentes da cabeça, que serão explicados posteriormente. Além disso, vale notar que essas matrizes de pesos são treinadas e melhoradas conforme o algoritmo vai gerando *outputs*, aprimorando a capacidade preditiva da rede.

Para calcular a multiplicação dos *embeddings* com os pesos dos componentes, as fórmulas seguem as Equações descritas em (4.3).

$$\begin{aligned} Q_i &= \text{Embedding}^{n_{\text{tokens}} \times d_{\text{model}}} \cdot W_{Qi}^{d_{\text{model}} \times d_k}, \\ K_i &= \text{Embedding}^{n_{\text{tokens}} \times d_{\text{model}}} \cdot W_{Ki}^{d_{\text{model}} \times d_k}, \\ V_i &= \text{Embedding}^{n_{\text{tokens}} \times d_{\text{model}}} \cdot W_{Vi}^{d_{\text{model}} \times d_v}. \end{aligned} \tag{4.3}$$

Dessa forma, os tamanhos das matrizes de cada componente são: $Q_i^{n_{tokens} \times d_k}$, $K_i^{n_{tokens} \times d_k}$ e $V_i^{n_{tokens} \times d_v}$.

Após essa etapa, os três componentes são introduzidos na *Scaled Dot-Product Attention*, conforme ilustrado na Figura (4.8). Nesse processo, ocorre uma multiplicação entre as operações de *Query* (**Q**) e *Key* (**K**), que é escalonada, seguida pela aplicação da função *Softmax* e, por fim, multiplicada pela matriz de *Value* (**V**). Isso é resumido na Equação (4.4).

$$\text{Attention}(Q_i, K_i, V_i) = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i. \tag{4.4}$$

Inicialmente, a multiplicação da *query* pela transposta da *key* resulta em uma matriz de scores da atenção com dimensão $n_{tokens} \times n_{tokens}$. Por exemplo, se a frase

de entrada for “Sorvete de banana”, a matriz seria $3x3$. Tomando a linha 1 como exemplo: a coluna 1 representa o score da atenção entre a palavra “Sorvete” com ela mesma; a coluna 2 é o score da atenção entre “Sorvete” e “de”; a coluna 3 é o score de “Sorvete” com a palavra “banana”. Esses scores representam os pesos, ou força, que cada palavra na frase tem em relação às outras. Seria a relação de cada elemento com os outros.

No caso do *encoder*, a estrutura de *Mask* não é necessária. Ela é empregada apenas no *decoder*, que receberá um foco maior nesse método.

Após isso, o produto escalar resultante passa por um escalonamento de tamanho $\sqrt{d_k}$. Esse procedimento é realizado para garantir que os pesos da matriz sejam bem distribuídos e não atinjam valores excessivamente altos, dado que valores de d_k muito altos tendem a gerarem gradientes muito baixos para a função *softmax*. Desta forma, no próximo passo, ao aplicar a *Softmax*, as probabilidades resultantes não serão muito altas, permitindo que os scores da atenção sejam distribuídos de maneira equilibrada entre diferentes palavras.

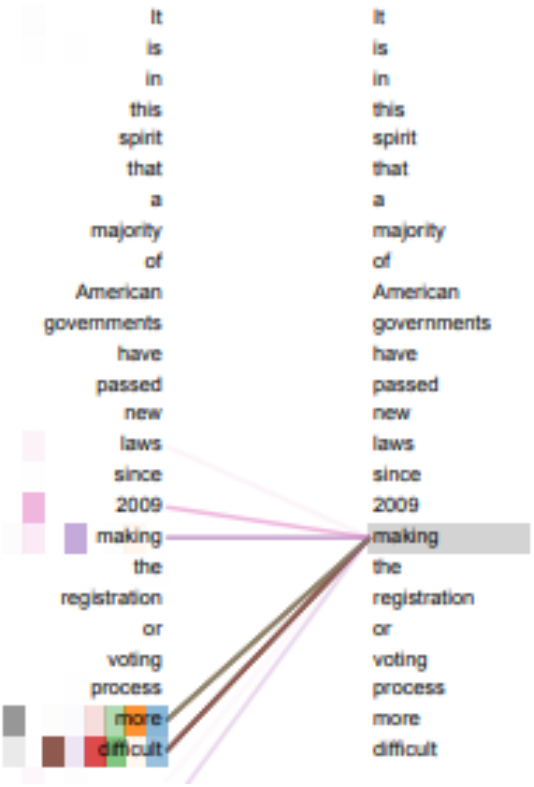
Descrita pela Equação (3.5), a *softmax* é uma função que cria probabilidades para cada elemento da matriz resultante do escalonamento acima. Assim, cada valor nas linhas está distribuído entre 0 e 1, com a linha somando 1. Esse resultado representa o score da atenção de cada palavra com as outras da frase em probabilidades. Com essa abordagem, os pesos mais altos serão mais importantes para cada *embedding*, enquanto pesos baixos terão menos influência nos vetores associados a cada palavra.

Por fim, essa matriz de probabilidades de atenção de dimensão $n_{tokens} \times n_{tokens}$ é multiplicada pelo *value* de dimensão $n_{tokens} \times d_v$, gerando uma matriz de dimensão $n_{tokens} \times d_v$. Note que o tamanho da matriz resultante é o mesmo àquela que havia entrado no *scaled dot-produt attention*. Dessa forma, são atribuídos aos *embeddings* de entrada novos valores que possuem o componente da atenção. Pode-se dizer que o resultado dessa multiplicação é a atenção, ou os *embeddings* com a atenção. Exemplificando, na frase “Sorvete de banana”, a linha 1 é o vetor da palavra “Sorvete” embutido com a atenção, valendo a mesma ideia para as outras linhas e colunas.

Porém, uma frase pode se relacionar de maneira diferente cada vez que ela é processada. Pegando como exemplo a frase “Levei o Kid para passear, ele é meu cachorro”, a palavra “Levei” está relacionada com “Kid”, “ele” e “cachorro”. Propõe-se que, para que o modelo consiga entender melhor as relações na frase, múltiplas cabeças (**h**) sejam processadas em paralelo, sendo cada *head* capaz de capturar relações semânticas de sua maneira específica. Tomando 8 cabeças como exemplo, $Q_i = [Q_1, ..., Q_8]$, $K_i = [K_1, ..., K_8]$ e $V_i = [V_1, ..., V_8]$ são processados em paralelo, realizando 8 vezes a Equação (4.4). Além disso, se $h = 8$ e $d_{\text{model}} = 512$, $d_k = d_v = 64$, gerando Q_i, K_i e V_i com dimensões $9x64$.

Tomando como exemplo a Figura (4.9), é possível observar que, na frase “It is in this spirit that a majority of American governments have passed new laws since 2009 making the registration of voting process more difficult.”, diferentes cabeças, representadas por cores distintas, atribuem diferentes pesos para a palavra evidenciada “making”. Para a *head* em laranja, “making” está fortemente relacionada com “more”, enquanto para a *head* em vermelho essa relação é mais forte com a palavra “difficult”.

Figura 4.9: Exemplo de uma estrutura *Transformer*.[9]



Fonte: Vaswani et al. (2017)

Por fim, cada *head* formada por seus respectivos Q_i, K_i e V_i é concatenada e passada por uma transformação linear. Esse é o fim da *Multi-Head Attention*, que segue para uma camada chamada de *Add & Norm*, como ilustrada na Figura (4.6). Dessa forma, o *input* da camada de *Add & Norm* se torna de dimensão $n_{\text{tokens}} \times d_{\text{model}}$.

A camada de *Add & Norm* realiza inicialmente conexões residuais. Essas conexões são simples adições do *input* ao *output*, ajudando a mitigar problemas associados ao gradiente da rede, como o desaparecimento do gradiente. Após a soma, é realizada uma normalização dos componentes para estabilizar a escala, contribuindo para a performance e estabilidade do modelo durante o treinamento.

Os dados seguem por uma camada *feed forward*. Este passo é realizado para que o modelo aprenda padrões e comportamentos nos *inputs* pela introdução de funções não lineares, além de moldar os dados para o *output* final. Inicialmente, a matriz de dimensão $n_{\text{tokens}} \times d_{\text{model}}$ é alimentada à rede *feed forward* em paralelo, tendo cada *token* processado separadamente. Após isso, ela passa por três diferentes passos, consistindo em duas transformações lineares com uma ativação $ReLU$ no meio. Essa rede é descrita pela Equação (4.5).

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2. \tag{4.5}$$

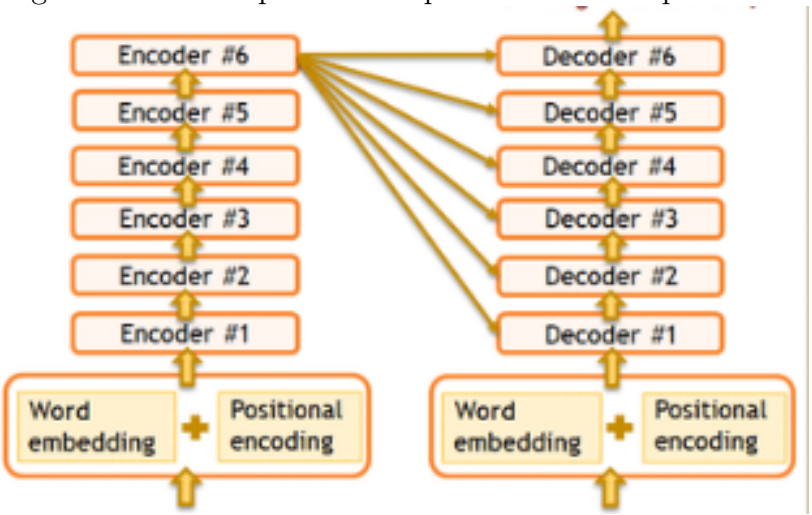
Nessa Equação (4.5), $\mathbf{x}$ representa o vetor associado a cada palavra da frase de tamanho d_{model}, W_1 e W_2 são matrizes de pesos que são aprendidos durante o processamento da rede e possuem dimensões $d_{\text{model}} \times d_{\text{ff}}$ e $d_{\text{ff}} \times d_{\text{model}}$, respectivamente, b_1 e b_2 são vetores de viés de tamanho d_{ff} e d_{model}, respectivamente. Foram definidas

matrizes de comprimento d_{ff} pois esse valor é maior que o d_{model}, podendo capturar maiores comportamentos e complexidades. Para um $d_{\text{model}} = 512$, normalmente é utilizado um $d_{\text{f}} = 2048$.

Os dados são recebidos, separados em vetores e concatenados no final. Dessa maneira, a matriz de *embeddings* com atenção passa por transformações em uma rede neural que contribui na habilidade de perceber comportamentos e padrões nos dados para cada um dos *tokens* da sentença. Finalmente, essa matriz passa por outra camada de *Add & Norm* para gerar o *output* do *encoder*.

Uma última escolha dentro do *encoder* se dá na escolha do "N". Esse hiperparâmetro indica o número de *encoders* que serão empilhados para entregar seus *outputs* ao *decoder*. Tomando um "N" igual a 6, as saídas do primeiro *encoder* servem de *input* ao segundo *encoder*, até que no sexto *encoder* suas saídas são fornecidas aos *decoders*. Exemplificando na Imagem (4.10), é possível observar essa sequência de *outputs* dos *encoders*. Essa abordagem amplia o potencial do modelo para identificar padrões nos dados, uma vez que cada *encoder* é especializado em reconhecer distintos comportamentos.

Figura 4.10: Exemplo de múltiplos *Encoders* empilhados.[10]



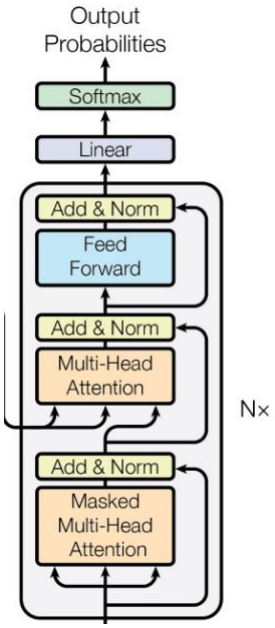
Fonte: Link para a imagem.

O *encoder*, utilizado separadamente, possui um desempenho excelente em tarefas de processamento de linguagem natural, análise de sentimento e de classificação, entre diversas outras.

4.5.5 Decoder

O *decoder* em um *transformer* emprega mecanismos de atenção semelhantes aos do *encoder*, mas com certas distinções. Enquanto o *encoder* se concentra em interpretar e codificar uma sequência de entrada, o papel principal do *decoder* é gerar uma sequência de saída. Essa sequência pode ser derivada das informações fornecidas pelo *encoder* ou diretamente dos inputs. A configuração do *decoder* é apresentada na Imagem (4.11).

Figura 4.11: Exemplo de uma estrutura *Decoder*.[11]



Fonte: Vaswani et al. (2017)

No *decoder*, são recebidos *output embeddings*, que são sequências já geradas pelo *decoder*. Exemplificando, se o decoder retornou a palavra **"Eu"**, essa sequência é incorporada ao *output embedding* como informação para a previsão do próximo termo.

Em seguida, são aplicados *positional encodings* que passam por camadas de atenção e *feed-forward*, mantendo a mesma estrutura explicada anteriormente para o *encoder*. Ao final desse processo, o *decoder* realiza sua predição.

Porém, há uma diferença em seu primeiro mecanismo de atenção, que recebe uma máscara. Essa máscara restringe o contexto da frase, no qual apenas as palavras que já foram observadas são utilizadas. Por exemplo, se a frase de entrada for **"Bom dia para você"** e apenas as palavras **"Bom"** e **"dia"** foram observadas, o mecanismo de atenção concentrará sua análise exclusivamente na atenção entre essas duas palavras, ignorando o restante da sentença. Esse comportamento é diferente de um mecanismo de atenção que não utiliza a máscara, que considera toda a frase.

Além disso, o foco do *decoder* é gerar *outputs* de texto, já que sua finalidade é compor sentenças completas. Durante o processo de geração, o modelo escolhe cada palavra sequencialmente a partir do seu vocabulário, sem considerar o panorama global. A escolha da próxima palavra é feita com base nas palavras anteriores e nas informações fornecidas pelo *encoder* através de probabilidades calculadas. A palavra

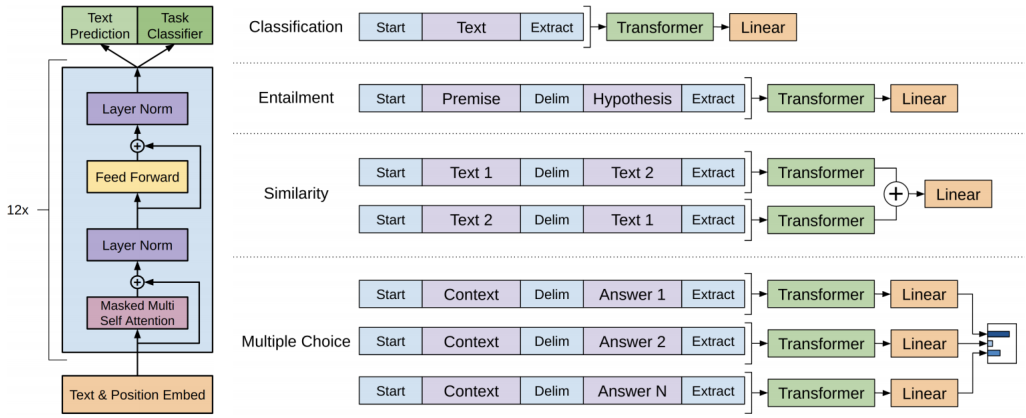
com a maior probabilidade é então selecionada para ser a próxima na sequência. Esse processo se repete iterativamente até que, no final, se a maior probabilidade indica o término da frase, o *decoder* encerra a geração e entrega o *output* final.

A arquitetura *Transformer*, com seus componentes *encoder* e *decoder*, oferece uma abordagem paralelizável e eficaz para capturar relações complexas em sequências de texto. O *encoder*, utilizando mecanismos de atenção, é capaz de interpretar o contexto de diversas sentenças. Por outro lado, o *decoder* utiliza tanto a informação contextual fornecida pelo *encoder* quanto seus próprios inputs anteriores para gerar sequências de saída mais precisas. Juntos, eles formam um sistema que, além de estimar a semântica de uma sequência de entrada, traduz essa compreensão em uma nova sequência de saída, como, por exemplo, em tarefas de tradução. Esta combinação de codificação de contexto e decodificação tem sido a base para muitos avanços no campo de *NLP*, demonstrando a força do *Transformer*.

4.6 GPT

O GPT, que é a sigla para *Generative Pre-trained Transformer*, foi introduzido por Radford et al. (2018). O principal objetivo deste modelo é predizer a próxima palavra em uma sentença, tornando-o especialmente poderoso em tarefas de geração de texto.

Figura 4.12: Exemplo de uma estrutura GPT e um exemplo de seu treinamento.[12]



Fonte: Vaswani et al. (2017)

Esse método utiliza uma variante da arquitetura *transformer*, especificamente baseada no *decoder* desse modelo, como pode ser visto na Figura (4.12). Através das estruturas de atenção, o GPT é capaz de atribuir diferentes pesos às palavras, permitindo que o modelo considere dependências de longo alcance. Com múltiplas camadas do *decoder empilhadas*, cada uma analisando a entrada de uma perspectiva diferente, o GPT consegue processar e gerar texto de forma eficiente.

Essa técnica é caracterizada por sua natureza unidirecional, que foca em prever a próxima palavra em uma sequência com base, exclusivamente, no texto que a precede, permitindo que ele gere sequências de texto de maneira coesa, avançando palavra por palavra, enquanto se baseia no contexto já estabelecido.

Seu treinamento é dividido em duas partes, o pré treinamento e o *fine tuning*. No pré-treinamento, o GPT obtém informações de grandes conjuntos de texto, no

qual ele tenta prever palavras. Por exemplo, ele recebe a frase **"A casa é "** e tenta realizar a previsão da palavra faltante. Após isso, é comparada a palavra predita com a original. No caso, se o GPT completou a frase com **"verde"** e a resposta era **"bonita"** isso resultará em uma perda maior no treinamento. Isso é realizado para que o modelo consiga absorver semântica, gramática e contexto. O ajuste fino é a parte que o método recebe conjuntos de dados menores para realizar tarefas diferentes, como tradução, classificação de texto, entre diversas outras.

Exemplificando alguns modelos de GPT, todos foram apresentados pela *Open AI* em seus diversos artigos. Existem os modelos de GPT1 (Radford et al. (2019)), GPT3 (Brown et al. (2020)) e GPT4 (OpenAI (2023)). Por ser um modelo pré-treinado, suas maiores diferenças estão no número de parâmetros e na escala de seu treinamento, que cresceram muito dos modelos GPT-1 ao GPT-4. Enquanto o GPT-1, por exemplo, tem milhões de parâmetros, o GPT-3 consiste de bilhões de parâmetros. Popularizado, o Chat GPT utiliza uma estrutura de GPT-3 e GPT-4.

4.7 BERT

Diferentemente do GPT, o BERT (*Bidirectional Encoder Representations from Transformers*) possui uma estrutura de diversos *encoders*, que tem como seu principal objetivo realizar tarefas de classificação, análise de sentimento, entre diversas outras. Ele foi introduzido no artigo de Devlin et al. (2018) e desde então recebe diversos variantes para diferentes tarefas.

Por apenas compreender a estrutura do *encoder*, o BERT consegue extrair o contexto da frase e posicionar palavras semanticamente similares em representações próximas. Esse processo é feito através dos mecanismos de atenção, explicados com detalhe na Seção (4.5.4) sobre *encoder*.

Enquanto o GPT é unidirecional, uma característica fundamental do modelo BERT é sua estrutura bidirecional, como indicado pelo nome. Nessa abordagem, o BERT recebe uma frase completa e é capaz de avaliar palavras em toda a sentença para compreender o contexto. Assim, os *word embeddings* fornecidos pelo BERT são contextualizados. Vale notar que toda a arquitetura do BERT é derivada do *transformer* (Vaswani et al. (2017)), com a exceção do *decoder*.

O processamento sequencial no BERT inicia com os *word embeddings* que são atribuidos *positional encoders*. Em seguida, a sequência passa por uma camada de *multi-head attention*, seguida por *add & norm*, *feed forward*, e outra camada de *add & norm*, resultando no *output*. Dado que o BERT utiliza *encoders* empilhados, isso é realizado "N" vezes até que o *output* gerado é o final, ou seja, o último BERT da pilha, ou BERT de número "N".

O BERT é dividido em duas partes: o *pre training* e o *fine-tuning*. Elas consistem em:

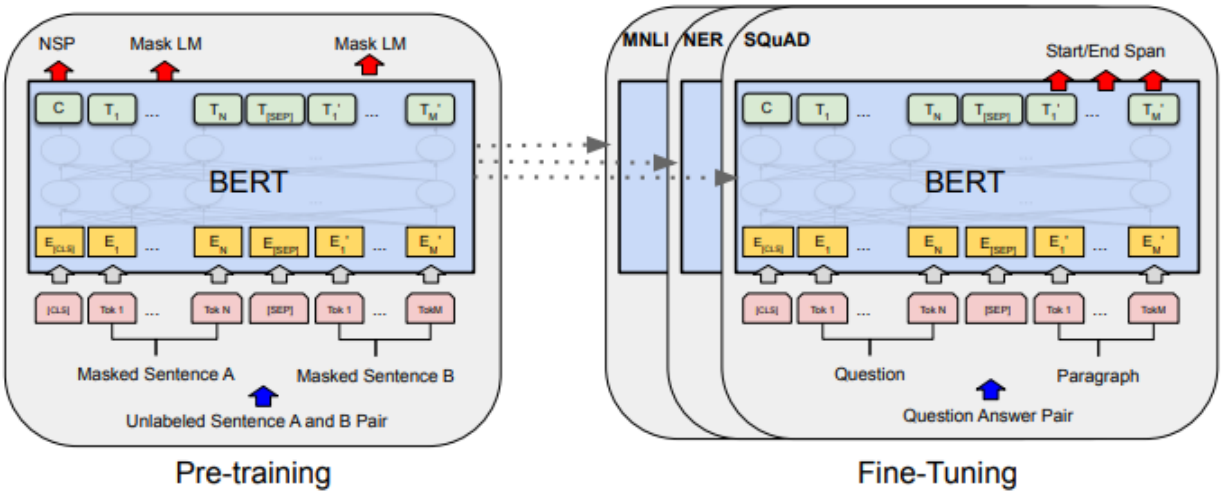
O ***Pre-Training***: nesta etapa, o modelo é exposto a diversos conjuntos de textos para aprender padrões. Em geral, se o modelo BERT é treinado com um banco de dados específico, como no caso de dados esportivos, ele tende a se destacar em tarefas relacionadas a esse domínio. Isso sugere que a eficácia do BERT em uma tarefa específica está diretamente ligada ao conjunto de dados com o qual foi treinado, permitindo que ele adquira conhecimento especializado. Modelos pré treinados de BERT como o *TwHIN-BERT* Zhang et al. (2023), treinado em 7 bilhões de *tweets* de 100 diferentes línguas; o *BERTimbau* Souza et al. (2020), treinado com uma

base de dados em português brasileiro; e o SportsBERT Srinivasan (2020), treinado em diversos artigos esportivos, são exemplos de aplicações de BERT em diferentes domínios. Isso mostra que, com disponibilidade computacional (dado que modelos podem levar dias para serem treinados e precisam de uma gigantesca capacidade de processamento), diferentes bancos de dados podem ser entregues ao BERT para que ele realize tarefas específicas relacionadas ao tema de interesse.

O ***Fine Tuning***: nesta etapa, modelos pré treinados são treinados com bancos de dados específicos sobre algum tema. Por exemplo, pode-se tomar um modelo como o BERTimbau e ajustá-lo a um novo banco de dados de menor tamanho. Isso é feito para que o modelo seja capaz de realizar tarefas específicas. Por exemplo, se o foco do estudo é classificar tipos de bebida e os dados são em português, um modelo como o BERTimbau pode ser selecionado. Ele receberá o banco de dados e ajustará seus parâmetros pré treinados para se especializar na tarefa de classificar tipos de bebida. Se o modelo não recebesse o ajuste fino, ele provavelmente não iria ter um bom desempenho na classificação de bebidas, uma vez que o modelo pode não possuir conhecimento específico sobre esse domínio, mesmo que compreenda a língua portuguesa. Uma analogia que pode ser utilizada é a leitura de um livro sobre culinária. Se uma pessoa que não possui conhecimento na cozinha tentar fazer um *petit gateau*, por exemplo, não irá conseguir. Porém, ao ler um livro de culinária com a receita, o resultado pode não ser perfeito, mas certamente será melhor do que se ela não tivesse lido o livro.

Nas próximas seções, serão detalhadas as etapas de *pre-training* e *fine tuning* realizadas pelo BERT destacadas na Figura (4.13), além de ilustrar a sua *Input Representation*.

Figura 4.13: BERT *Pre-Training* e *Fine Tuning*.[13]



Fonte: Devlin et al. (2018)

4.7.1 Input Representation

O BERT emprega inicialmente um método para representar seus *inputs*, que é capaz de lidar tanto com problemas que envolvem apenas uma frase (como classi-

ficação e análise de sentimento), quanto com problemas que envolvem sequências de texto, como perguntas e respostas. Isso é ilustrado na Figura (4.14), que será detalhada mais adiante.

No BERT são utilizados *Word piece embeddings*, apresentados em Wu et al. (2016). Esse método auxilia no aprendizado do modelo, dado que se o modelo ainda não aprendeu uma determinada palavra, ele consegue a dividir em sub-palavras que conhece. Por exemplo, o modelo pode ser familiarizado com a palavra "come", porém não reconhecer "comeu" e "comendo". Nesse caso, ele irá dividir as palavras em "come" + "##u" e "come" + "##ndo", permitindo a identificação de diversas sentenças ainda não aprendidas. Além disso, é importante destacar que, no modelo de *Word Pieces*, a inclusão do prefixo "##" indica que uma sub-palavra está sendo usada. Isso é especialmente útil para diferenciar palavras derivadas, como "come" + "##ço", que possui um *embedding* totalmente diferente daquelas derivadas de "come" no contexto de comer. Essa abordagem oferece flexibilidade para capturar a diversidade morfológica das palavras.

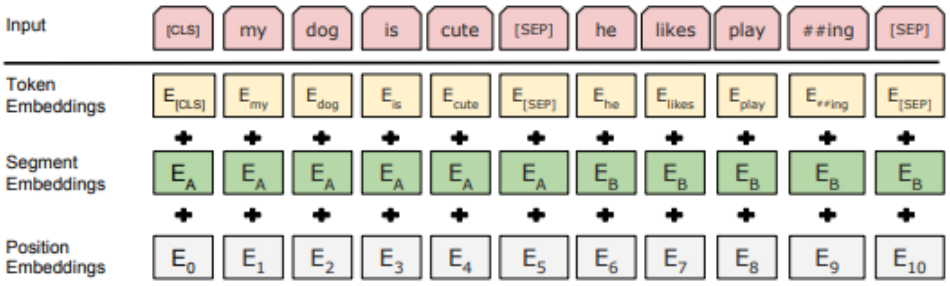
Além disso, os *inputs* do BERT recebem dois tipos de *tokens*:

*O token **[CLS]***. Ele é inserido no início da frase e é uma representação agregada da sentença inteira na forma de um *token*, ótimo para tarefas de classificação.

*O token **[SEP]***. Ele é inserido para separar frases dentro de um conjunto de sentenças.

Uma última adição são os *Segment embeddings*, que são *embeddings* usados para indicar a qual sentença a frase está inserida.

Figura 4.14: Exemplo do *Input Representation* do BERT.[14]



Fonte: Devlin et al. (2018)

Como observado na Figura (4.14), um exemplo de *input* foi fornecido. Ele consiste em 2 frases: *"my dog is cute"* e *"he likes playing"*. Ao enviar a sentença, alguns procedimentos ocorrem:

A sentença recebe o token [CLS] no início da frase e os tokens [SEP] entre as frases A e B e um no final da frase B;

Dado que o modelo possui *"play"* em seu vocabulário mas não tem *"playing"*, essa palavra é dividida em *"play"* e *"##ing"*;

São organizados os *Token Embeddings* que são somados aos *Segment Embeddings* e aos *Positional Embeddings*.

Por fim, essas são as representações finais dos embeddings da frase completa, consistindo em uma matriz de dimensões $n_{\text{tokens}} \times d_{\text{model}}$ que será utilizada como *input* do modelo.

4.7.2 Pre-Training

Na fase de pré treinamento, são utilizados duas tarefas não supervisionadas, o *Masked LM* e o *Next Sentence Prediction*.

Dado que o BERT é bidirecional, o método de *Masked LM* foi introduzido como MLM por Devlin et al. (2018) para um treinamento eficaz. Anteriormente, esse método havia sido apresentado como *cloze* por Taylor (1953). Essa tarefa consiste na utilização de máscaras (*[MASK]*) em palavras selecionadas aleatoriamente dentro de uma sentença, com o objetivo de realizar a previsão dessas palavras mascaradas. Essas máscaras são aplicadas em 15% dos *Word Pieces*, permitindo que o modelo seja treinado para prever todas as partes de uma sentença, seja ela início, meio ou fim. Isso contrasta com abordagens como a do GPT, que se realiza apenas a previsão do final da frase. Dessa forma, o BERT avalia o contexto global da frase ao realizar a predição da palavra mascarada.

Porém, essa abordagem possui um problema. Mesmo que o modelo consiga ser pré treinado com as máscaras, a fase de *fine tuning* do modelo não estaria recebendo os *tokens* de [MASK], indicando uma discrepância no método. Isso prejudicaria a parte de ajuste fino pois o método perderia em capacidade preditiva e em entender o contexto nessa etapa. Para mitigar isso, Devlin et al. (2018) propõe que 80% das máscaras não sejam trocadas por palavras, 10% seja trocado por palavras aleatórias e os 10% restantes mantenham a mesma palavra. Exemplificando, se na frase *"my dog is hairy"* a palavra *"hairy"* foi selecionada aleatoriamente, ela será:

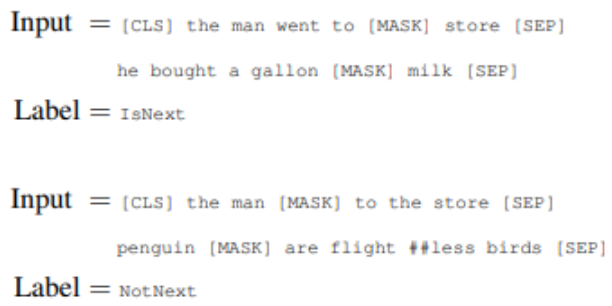
Trocada por [MASK] em 80% dos casos. Assim, a frase resultante será *"my dog is [MASK]"*;

Trocada por uma palavra aleatória em 10% dos casos. Assim, a frase resultante será *"my dog is apple"*;

Trocada pela palavra correta em 10% dos casos. Assim, a frase resultante será *"my dog is hairy"*.

Buscando aprimorar a capacidade de compreensão das relações entre frases, foi desenvolvido o *Next Sentence Prediction*. Esse método envolve o uso de duas frases, denominadas **A** e **B**, em que **B** é considerada a sequência de **A**. No entanto, em 50% dos casos, a frase **B** realmente segue a frase **A**, enquanto nos outros 50%, **B** é escolhida aleatoriamente. Quando a frase subsequente é a correta, ela deve ser classificada como *IsNext*; caso contrário, deve ser classificada como *NotNext*.

Figura 4.15: Exemplo do *Next Sentence Prediction* do BERT.[15]



Fonte: Devlin et al. (2018)

Observando a Figura (4.15), a frase *"the man went to the store"* é designada como

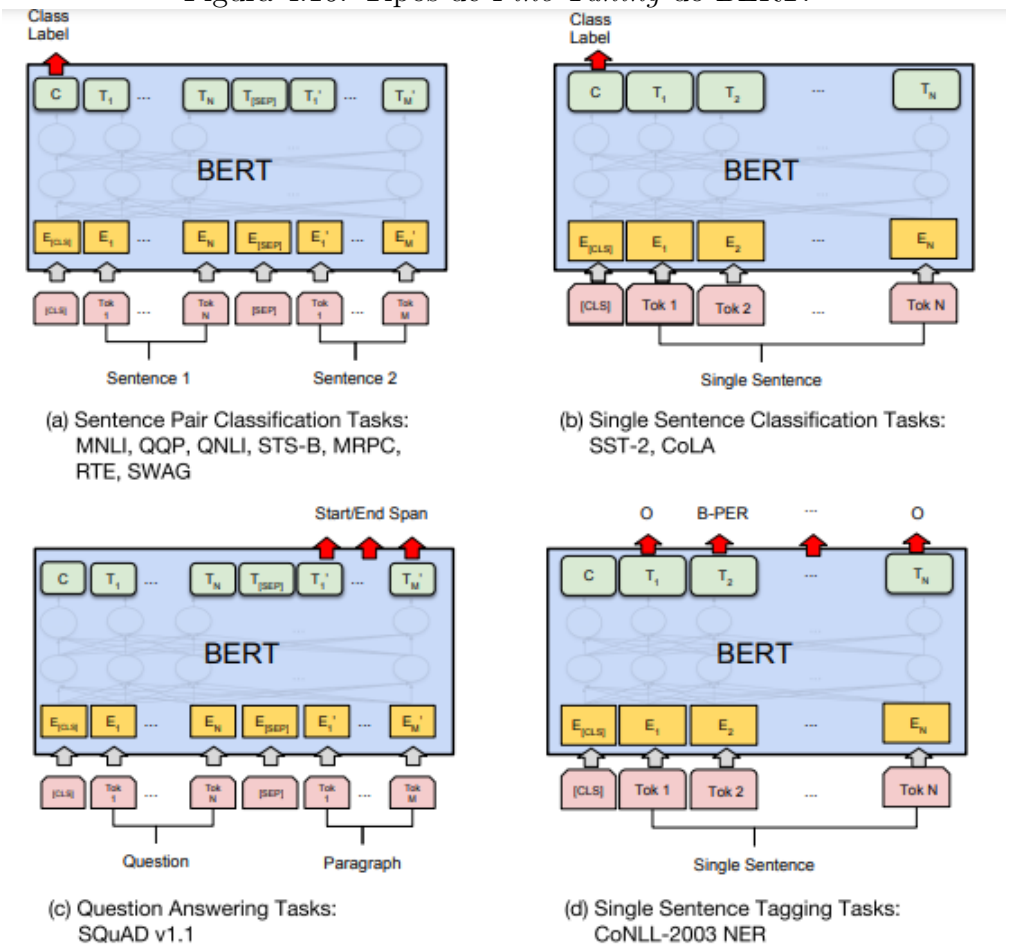
a frase **A**, enquanto a próxima frase, iniciando após o token [SEP], é identificada como a frase **B**. No exemplo apresentado, o modelo classifica corretamente que *"he bought a gallon of milk"* é a frase seguinte correta, ou seja, *IsNext*, enquanto *"penguin birds are flightless birds"* é classificado como *NotNext*.

Normalmente, para modelos BERT, são apresentados dois tipos: *BASE* e *LARGE*. Sua principal diferença é o tamanho, seja ele em parâmetros ou número de *encoders* empilhados. Por exemplo, Devlin et al. (2018) apresentou o $\text{BERT}_{\text{BASE}}$ com 110 milhões de parâmetros e 12 *encoders* empilhados, e o $\text{BERT}_{\text{LARGE}}$, com 340 milhões de parâmetros e 24 *encoders* empilhados. Diversos outros modelos de BERT como o BERTIMBAU (Souza et al. (2020)) também adotaram essa distinção.

4.7.3 Fine Tuning

Durante a fase de *fine-tuning*, o modelo pré-treinado é submetido a um novo conjunto de dados associado a uma tarefa específica. Nesse contexto, o modelo é capaz de aprender sobre os padrões presentes nos dados dessa tarefa específica, permitindo a execução de tarefas mais especializadas. Durante esse processo, os parâmetros do modelo são ajustados para otimizar seu desempenho na tarefa específica em questão, embora a mudança nos parâmetros não seja muito significativa, uma vez que o modelo já foi pré-treinado anteriormente.

Figura 4.16: Tipos de *Fine Tuning* do BERT.[16]



Fonte: Devlin et al. (2018)

Observando a Figura (4.16), é possível identificar diversas tarefas que podem ser realizadas pelo BERT, a partir de uma mudança de *input* e *output* como exemplificado no artigo Devlin et al. (2018). São elas:

Classificação por sentenças. Nesse tipo de tarefa, o BERT utiliza o *token* [CLS], que contém todo o significado da sentença, para classificar as frases;

Classificação por frases. Nesse método, o BERT classifica a única frase a partir do *token* [CLS];

Tarefas de pergunta e resposta. Nesse caso, o BERT prevê as posições iniciais e finais pelo contexto;

Atribuição de tags. Nessa tarefa, ele identifica e atribui *tags* a partir da utilização de *O* e *B-PER*.

4.8 BART

O modelo BART, introduzido por Lewis et al. (2019), consiste em um *autoencoder* de remoção de ruído. Ele é inicialmente treinado com textos que contém ruído e, a partir dessas sentenças, faz com que o ruído seja removido, obtendo a frase correta.

Esse método utiliza a arquitetura do transformer (Vaswani et al. (2017)), obtendo desempenhos muito bons em compreensão, pergunta e resposta, geração de texto,

sumarização, entre diversos outras técnicas.

4.9 BERTopic

Grande parte dos dados gerados são não estruturados e não rotulados. Textos curtos como *tweets*, mensagens e diversos outros frequentemente não possuem esses rótulos. Dessa forma, transmitir conteúdo a uma rede neural se torna difícil, uma vez que ela precisa, muitas vezes, aprender na tentativa e erro, observando o resultado real e o seu resultado predito.

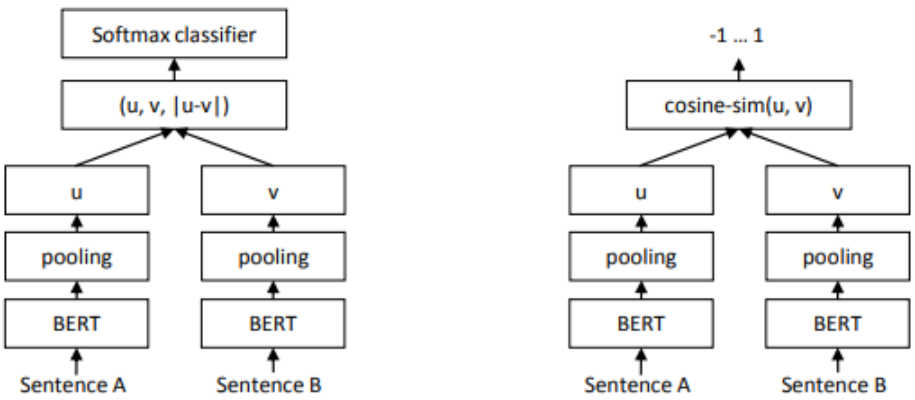
Para lidar com a ausência de rótulos em textos, modelos que realizam tarefas chamadas de *Topic Modeling* são utilizados. Esse método consiste na atribuição de classes para dados não rotulados, permitindo que o modelo consiga ter algum alvo a ser atingido. Entre eles, alguns modelos não são baseados em redes neurais, como o LDA Blei et al. (2003) e o CTM Blei e Lafferty (2007). Porém, eles não consideram a semântica da frase como, por exemplo, um modelo BERT. Partindo dessa ideia, foi criado o BERTopic por Grootendorst (2022) para que os tópicos sejam criados considerando o contexto da frase.

O BERTopic é dividido em três partes, os *Document Embeddings*, os *Document Clustering* e o *Topic Representation*, que serão detalhados em suas seções.

4.9.1 Document Embeddings

Nessa etapa, o BERTopic obtém os *embeddings* das frases utilizando a estrutura conhecida como S-BERT, ou *Sentence-BERT*, conforme proposto por Reimers e Gurevych (2019). Esse método é uma derivação do BERT Devlin et al. (2018) projetado especificamente para criar *embeddings* para sentenças. Dessa forma, sentenças semanticamente similares são posicionadas próximas no espaço vetorial.

Figura 4.17: Representação do S-BERT.[17]



Fonte: Reimers e Gurevych (2019)

Conforme ilustrado na Figura (4.17), esse método utiliza uma camada de *pooling* no *output* do BERT. Essa camada é utilizada para resumir informação, agregá-la. Três formas de agregação são apresentadas:

***Pooling* por [CLS]**: A informação já esta resumida no *token* [CLS]. Portanto, o *pooling* é apenas a obtenção do *embedding* deste *token*;

***Pooling* por Máximo**: Obtém-se o valor máximo dos *embeddings* elemento em elemento;

***Pooling* por Média**: Obtém-se o valor médio dos *embeddings* elemento em elemento.

Após as camadas de *pooling*, são divididos em três métodos dependendo dos objetivos. São eles:

Classificação: Representado pela estrutura à esquerda da Imagem (4.17), aplica-se aos *embeddings* de cada sentença, denominados u e v, uma diferença entre elemento por elemento. Esse resultado será multiplicado a uma matriz de pesos atualizáveis representada por W_{t}. A Equação (4.6) resume essa álgebra.

$$o = \mathrm{softmax}(W_t(u, v, |u - v|)) \tag{4.6}$$

Com W_t tendo dimensões $R^{3_n \times k}$, em que **n** é a dimensão dos *embeddings* da sentença e k o número de rótulos;

Regressão: Representado pela estrutura à direita da Imagem (4.17), é aplicada uma similaridade por cosseno entre os *embeddings* u e v;

Triplet: O modelo recebe três sentenças, uma central a, uma negativa n e uma positiva p. Esse método consiste em deixar as sentenças a e p mais próximas que a e n.

Utilizando o S-BERT, o BERTopic consegue *embeddings* de sentenças que em um espaço vetorial, no qual sentenças próximas são mais similares. Vale notar que o método do S-BERT é utilizado para criar *clusters* dos dados. Dessa maneira, métodos de criação de *embeddings* que realizem a criação de tópicos podem ser utilizados no BERTopic, podendo melhorar ainda mais o algoritmo.

4.9.2 Document Clustering

Ao lidar com dados que possuem muitas dimensões, um dos desafios é representado pela *curse of dimensionality*. O termo foi introduzido por Bellman (1957) e indica que dados representados em um espaço com muitas dimensões possuem uma interpretabilidade e inferência prejudicadas.

Quando os *embeddings* dos documentos saem do modelo, esses valores são representados por vetores muito grandes. Assim, eles são afetados pela *curse of dimensionality*. Para contornar esse problema, foi proposto a utilização do algoritmo de UMAP McInnes e Healy (2018), que realiza a redução da dimensionalidade. Outros algoritmos como o PCA (Pearson (1901)) e o t-SNE (Maaten e Hinton (2008)) também são conhecidos para esse tipo de tarefa, porém o UMAP foi escolhido por conseguir manter de maneira mais representativa os dados em dimensões menores.

Uniform Manifold Approximation and Projection, ou UMAP, consiste na projeção do comportamento de dados de uma dimensão maior para uma menor. Seu objetivo é replicar padrões de *clusters* presentes nas dimensões mais altas, garantindo que esses padrões sejam preservados nas dimensões reduzidas.

Primeiramente, o método calcula as distâncias entre os pontos na dimensão maior. Essas distâncias são então representadas em um gráfico, em que, por exemplo, se a distância de um ponto a para b é de 3 unidades e para c é de 5 unidades, essas relações de distância são preservadas no gráfico. Com esse gráfico, traça-se

uma curva para calcular escores de similaridade de alta dimensão. Vale notar que essa curva varia conforme o número de vizinhos (x) que se considera, sendo ele o formato da curva definido por $Log_2(x)$. Assim, valores próximos são representados por escores altos, enquanto valores distantes são tendem se aproximar de zero. Vale notar que esses escores não são simétricos. Um escore de 2 de a para b pode ser diferente de b para a.

Ao inicializar os dados na dimensão menor, o UMAP busca aproximar dados selecionando aleatoriamente valores que pertencem ao mesmo *cluster* na dimensão alta e busca separar dados aleatoriamente escolhendo valores que não fazem parte do *cluster* de alta dimensão. Valores com escores altos possuem uma chance maior de serem selecionados para aproximarem.

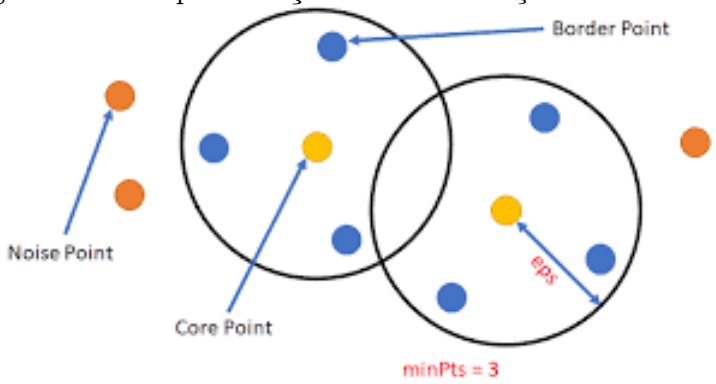
Para que o UMAP saiba quanto ele deve aproximar e separar os pontos, são gerados escores de baixa dimensão a partir de uma distribuição derivada da *t-student*, que é a mesma para todos os escores. Considerando, por exemplo, que b deve se aproximar de a e se afastar de j, um valor a que está no centro de maior densidade da sua distribuição verá b movendo-se em sua direção, enquanto b se desloca para as caudas da distribuição de j, ou seja, as regiões de menor densidade.

Dessa maneira, o UMAP consegue trazer os dados para uma menor dimensão. Após aplicar esse método, é utilizado um outro algoritmo, chamado de HDBSCAN Campello et al. (2013) para *clusterizar* os *embeddings*.

A *clusterização* é o processo de agrupar dados similares em conjuntos não definidos, buscando identificar padrões ou comportamentos semelhantes. O método HDBSCAN opera com o objetivo de agrupar dados, sendo uma adaptação hierárquica do DBSCAN Ester et al. (1996). Primeiramente, será apresentado o DBSCAN, dado que o HDBSCAN é derivado dessa abordagem.

Inicialmente, os dados são posicionados no espaço dimensional. Cada ponto no espaço possui um número de vizinhos. Para que ele seja considerado um *core point* (ponto central), o ponto deve possuir pelo menos (x) vizinhos. Após definir esse valor x, o algoritmo é inicializado, aleatoriamente selecionando um dos *core points*. O ponto central escolhido será o primeiro *cluster* e ele irá agrupar todos os *core points* vizinhos, que farão o mesmo até que todos os pontos centrais vizinhos sejam agrupados. Após isso, todos os pontos não centrais que são vizinhos imediatos de pontos centrais são adicionados ao *cluster*. Posteriormente, outros *core points* que não foram incluídos irão repetir o mesmo processo. Valores que não entraram em nenhum *cluster* são considerados *outliers*, ou seja, pontos que não se encaixam em nenhum grupo. Esse método pode ser observado na Figura (4.18).

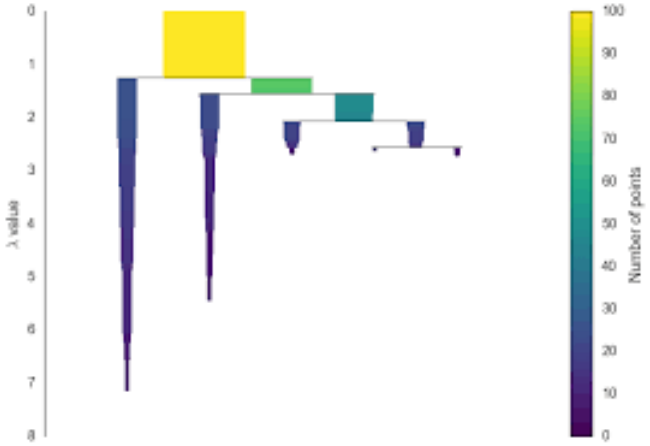
Figura 4.18: Representação da *clusterização* do DBSCAN.[18]



Fonte: https://machinelearninggeek.com/dbscan-clustering/

O algoritmo de HDBSCAN utiliza esse método de forma hierárquica, isto é, deixa de apenas *clusterizar* os dados por densidade e passa a dividi-los através de árvores, como mostra na Figura (4.19). Essa maneira consegue identificar grupos similares dentro de diferentes densidades, com diferentes formatos e é mais robusto em relação a ruídos.

Figura 4.19: Representação da árvore de *clusterização* do HDBSCAN.[19]



Fonte: https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

4.9.3 Topic Representation

Por fim, os tópicos são obtidos a partir dos *clusters* gerados na etapa de *Document Clustering*. Para obter as palavras chave de cada *cluster*, ou seja, os tópicos do grupo, utiliza-se um método TF-IDF modificado, que determina a importância de cada palavra em relação ao tópico.

O método de TF-IDF recebe uma aprofundação maior em sua Seção (2.2.1), dessa forma, apenas a variante desse método sera tratada nesse tópico. Essa variação realiza o cálculo do TF-IDF por *clusters*, ou seja, cada *cluster* calculado terá essa álgebra realizada dentro dele.

$$\boldsymbol{cTF}(\textbf{term}) = \left(\frac{n_{\text{aparição do termo no } \textit{cluster}}}{n_{\text{total de termos no } \textit{cluster}}} \right). \tag{4.7}$$

$$\boldsymbol{cIDF}(\textbf{term}) = \ln\left(1 + \frac{n_{\text{médio de palavras por } \textit{cluster}}}{n_{\textit{clusters} \text{ contendo o termo}}}\right). \tag{4.8}$$

$$\boldsymbol{cTF\text{-}IDF}(\textbf{term}) = \text{cTF(term)} \times \text{cIDF(term)}. \tag{4.9}$$

Observando as Equações 4.8 e 4.7, o cTF-IDF realiza os cálculos por *cluster*, enquanto que o TF-IDF os realiza por documento. Dessa forma, são obtidas distribuições de palavras por tópico para cada *cluster*.

Apresentado por Grootendorst (2022), o *KeyBERTInspired* é um modelo que realiza um ajuste fino ao *topic representation*. O método é uma otimização do *KeyBERT*, que realiza os seguintes passos:

- 1) São sorteados **n** documentos representativos em cada tópico;
- 2) É calculado o c-TF-IDF para esses documentos, no qual os que melhor representam o tópico são obtidos;
- 3) São gerados *embeddings* para cada tópico;
- 4) São comparados os *embeddings* dos tópicos com as palavras;
- 5) Palavras mais próximas a cada tópico são obtidas.

Dessa maneira, o método consegue ajudar o modelo a representar os tópicos.

4.10 Large Language Models

Large Language Models, ou LLMs, são modelos de *deep learning* que possuem estruturas com muitos parâmetros e que são treinados em grande volumes de dados, tendo um poder preditivo muito grande. Os modelos classificados como LLMs são utilizados para diversas tarefas, como predição, geração e entendimento de conteúdo, além de tarefas como geração e tradução de texto, análise de sentimento, entre diversas outras.

Em sua fase de pré-treinamento, corpus com grandes volumes de dados são utilizados no modelo, fazendo com que ele tenha um conhecimento geral sobre diversos domínios. Um exemplo é o Mistral 7B, introduzido por Jiang et al. (2023), que realiza tarefas de matemática, compreensão, conhecimento, código, entre outros. Outros famosos modelos são o LLama 1 (Touvron et al. (2023a)) e 2 (Touvron et al. (2023b)) criados pela *Meta* e, recentemente, o Bode (Garcia et al. (2024)), um LLM em português.

5 Materiais e Métodos

Nesta Seção, serão apresentados:

- Os pacotes e as linguagens de programação utilizados;
- O corpora utilizado;
- Um passo a passo do que foi realizado no modelo BERTopic;
- Os hiperparâmetros do modelo BERTopic;
- A métrica utilizada para avaliar os modelos.

Esses tópicos compõem esta Seção, que apresenta, em detalhes, o que foi feito e utilizado para a análise e obtenção dos resultados.

5.1 Pacotes e Funções

Nesta Seção serão apresentados os pacotes e as funções utilizadas. A linguagem de programação utilizada no trabalho foi o ***Python*** em sua versão **3.11.7** com o uso do ***Miniconda***, ao qual foram empregadas as seguintes bibliotecas:

- ***Pandas*** (McKinney (2010)) para a leitura dos dados em sua versão ***2.2.0***;
- ***Bertopic*** para a modelagem (Grootendorst (2022)) em sua versão ***0.15.0***;
- ***Plotly*** para a visualização dos gráficos em sua versão ***5.18.0***;
- ***Numpy*** para a organização dos dados em sua versão ***1.26.3***;
- ***sklearn*** para o cálculo da métrica em sua versão ***1.4.0***;
- ***Sentence_transformers*** para a utilização de modelos de representação em sua versão ***2.2.2***;
- ***Datetime*** para o cálculo do tempo de execução.

Além do *Python*, foi utilizada a linguagem de programação ***R*** versão **4.3.1** empregando o ***RStudio*** para o pré-processamento do segundo corpus e elaboração dos gráficos. Os pacotes do **R** utilizados foram:

- ***dplyr*** em sua versão **1.1.2**;
- ***ggplot2*** em sua versão **3.4.2**;
- ***dplyr*** em sua versão **0.4.1**.

O processador utilizado nas análises foi um ***Intel(R) Core(TM) i7-10700KF CPU @ 3.80GHz*** em um ***Microsoft Windows 10 Pro 10.0.19045*** com ***32 Giga*** de memória RAM.

Todos os códigos estão disponíveis no repositório do **Github** (https://github.com/AntonioBoll/tcc1).

5.2 Corpora

Os bancos de dados utilizados foram obtidos da Secretaria da Fazenda do Rio Grande do Sul. O primeiro, descrito como *banco_leite_carne_5_por_cento*, foi obtido a partir da realização de processos de Markov (Markov (1906)) em uma *Query* no sistema da SEFAZ-RS, buscando reunir notas fiscais com descrições de produtos relacionados a carne e leite. Apenas 5% do banco foi selecionado, uma vez que essa porcentagem representa 384 mil e 255 observações. Exemplificando o banco, algumas descrições de produtos que compõem o corpus são mostradas a seguir:

- **"granola mae terra 250 g integral castansbaunscoc"**
- **"creme leite piracanjuba 27x200g"**
- **"leite vendfoods 500g"**
- **"leite uht batavo semidesnatado 1l"**
- **"figado bovscongscxg"**
- **"peito resf bd cx pa"**

Diferentemente, no segundo banco (*bancao_1_por_cento*), os dados são obtidos diretamente de uma *Query* no sistema da SEFAZ-RS, sem qualquer pré-processamento. Somente 1% do banco foi selecionado, compreendendo 202 mil e 861 observações. Alguns produtos que integram o banco são mostrados a seguir:

- **"TENIS ESPORTIVO VENUS/689 SKYBLU 34"**
- **"LENTE UNIF AC KODAK POLY NO REFLEX 70 -0250 -075"**
- **"CB NEUGEBAUER BIBS AO LEITE"**
- **"CONJ. 12 CACHEPO DE CERAMICA SORTIDO"**

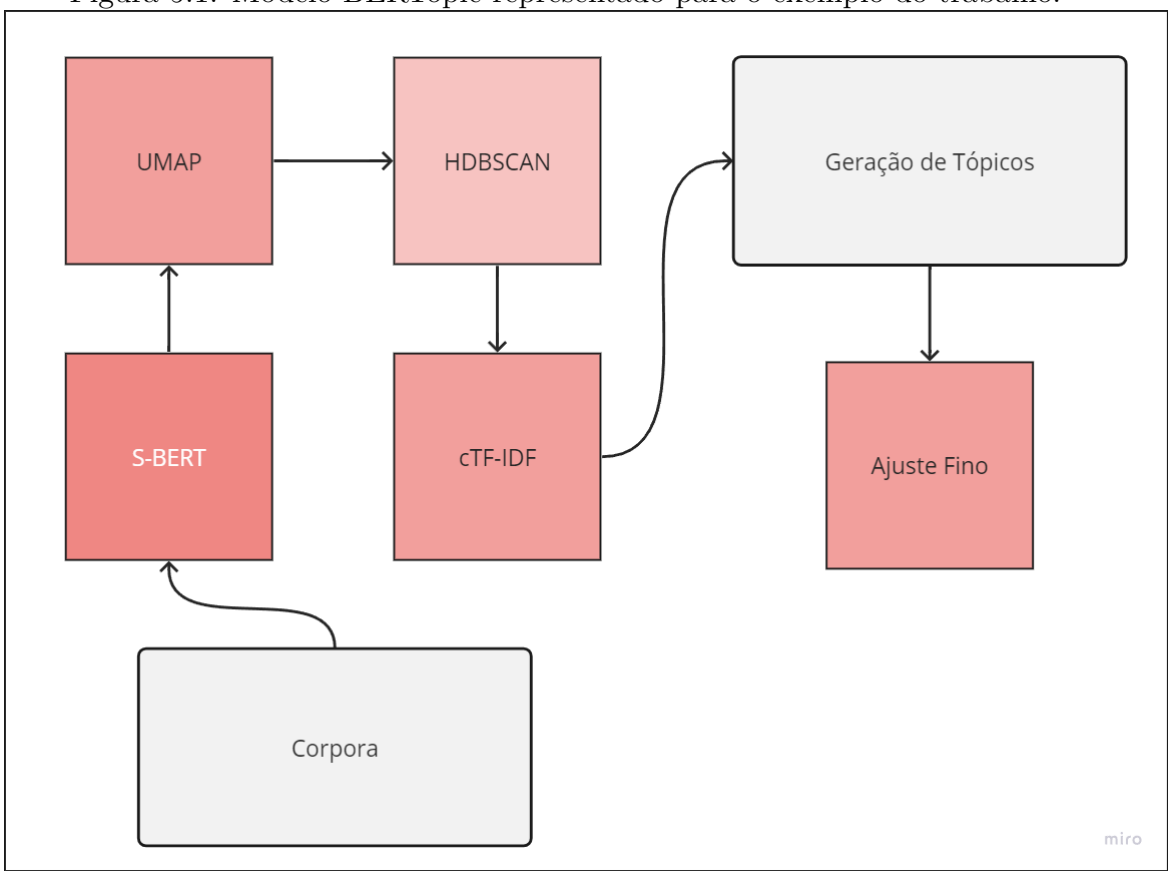
Essas observações consistem em diversas descrições de produtos contidas nas notas fiscais eletrônicas (*NF-e*). Vale lembrar que os dados são sigilosos e, portanto, informações sensíveis não serão apresentadas.

Não será realizado nenhum tipo de pré-processamento no primeiro banco de dados, uma vez que ele já passou pela remoção de acentuação e conversão de letras maiúsculas para minúsculas. O segundo passará pelo mesmo tipo de pré-processamento, visto que esse método ajuda a melhorar o desempenho do modelo Goyal (2020).

5.3 Modelo BERTopic

O modelo possuí três componentes principais, os *document embeddings*, *document clustering* e *topic representation*, além de seu *fine-tuning* que são ilustrados na Figura (5.1). Eles serão exemplificados a seguir.

Figura 5.1: Modelo BERTopic representado para o exemplo do trabalho.[1]



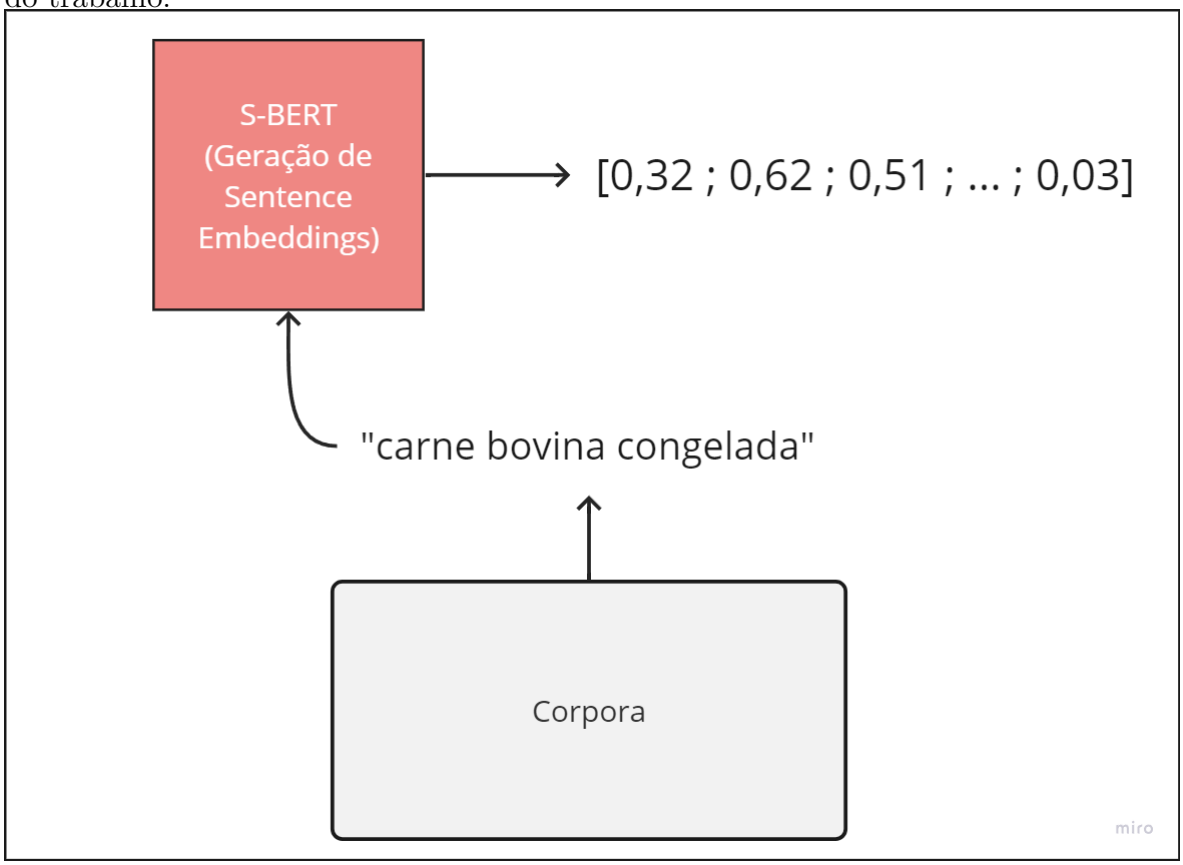
Fonte: O Autor

5.3.1 Document Embeddings

Inicialmente, o corpus passa por uma camada de *S-BERT*, que cria *sentence embeddings* para cada descrição de produto. Essa representação em vetores para cada mercadoria cria um espaço vetorial, no qual os itens do banco se posicionam perto

aos seus semelhantes. Por exemplo, um produto como “carne congelada” provavelmente terá um *embedding* semelhante ao da mercadoria “carne bovina congelada”. Assim, a tarefa de agrupar descrições similares se torna menos complicada. Vale lembrar que os corpus receberam modelagens semelhantes, porém não conjuntas. A Figura (5.2) exemplifica essa etapa.

Figura 5.2: *Document Embedding* do Modelo BERTopic representado para o exemplo do trabalho.[3]

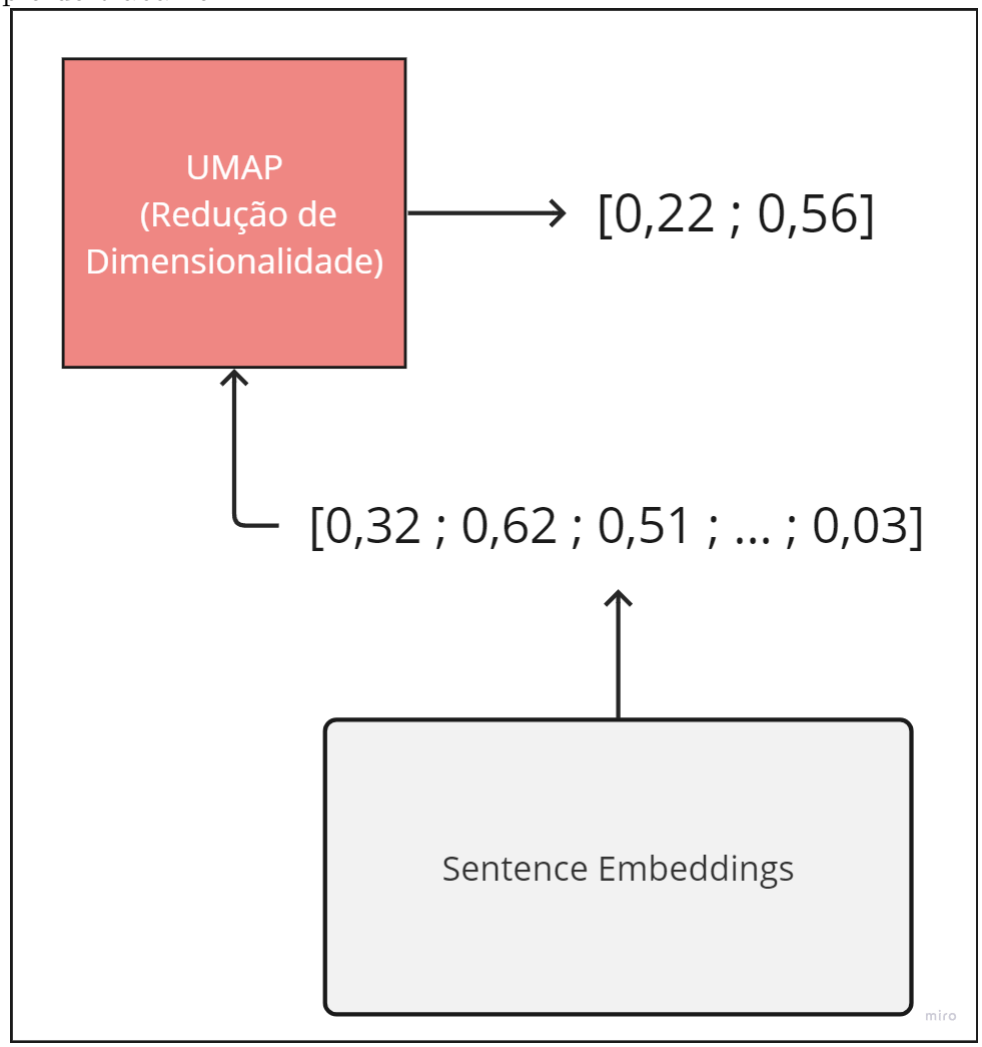


Fonte: O Autor

5.3.2 Document Clustering UMAP

Após a criação dos *sentence embeddings*, é necessário realizar uma redução de dimensionalidade. Isso se deve ao fato de que o espaço vetorial está representado em muitas dimensões, uma vez que esses vetores possuem um comprimento grande. O algoritmo UMAP realiza essa redução, buscando manter os comportamentos originais em uma dimensão 2D. Outros algoritmos, como o PCA, também poderiam ser utilizados para essa redução de dimensionalidade.

Figura 5.3: *Document Clustering UMAP* do Modelo BERTopic representado para o exemplo do trabalho.[5]

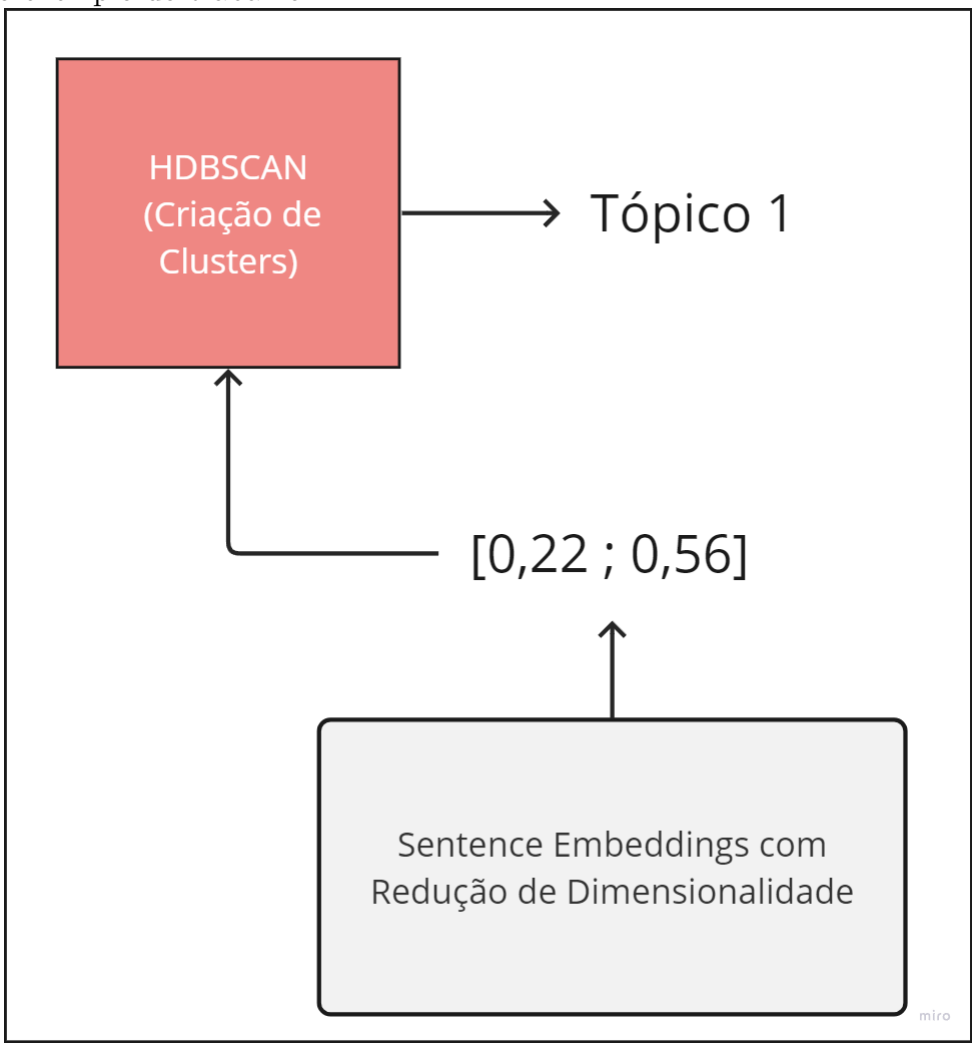


Fonte: O Autor

5.3.3 Document Clustering HDBSCAN

Com a redução de dimensionalidade, o algoritmo de HDBSCAN cria *clusters* para os dados, em que cada *cluster* formado é representado por um tópico. Embora existam outros métodos disponíveis para a clusterização, o HDBSCAN foi escolhido devido à sua capacidade de trabalhar de forma hierárquica. Dessa maneira, os tópicos formados são compostos por produtos que são julgados como semelhantes no espaço vetorial. As mercadorias mais similares possuem probabilidades mais altas de pertencerem ao seu grupo, enquanto os *outliers* normalmente têm esse valor menor.

Figura 5.4: *Document Clustering HDBSCAN* do Modelo BERTopic representado para o exemplo do trabalho.[7]

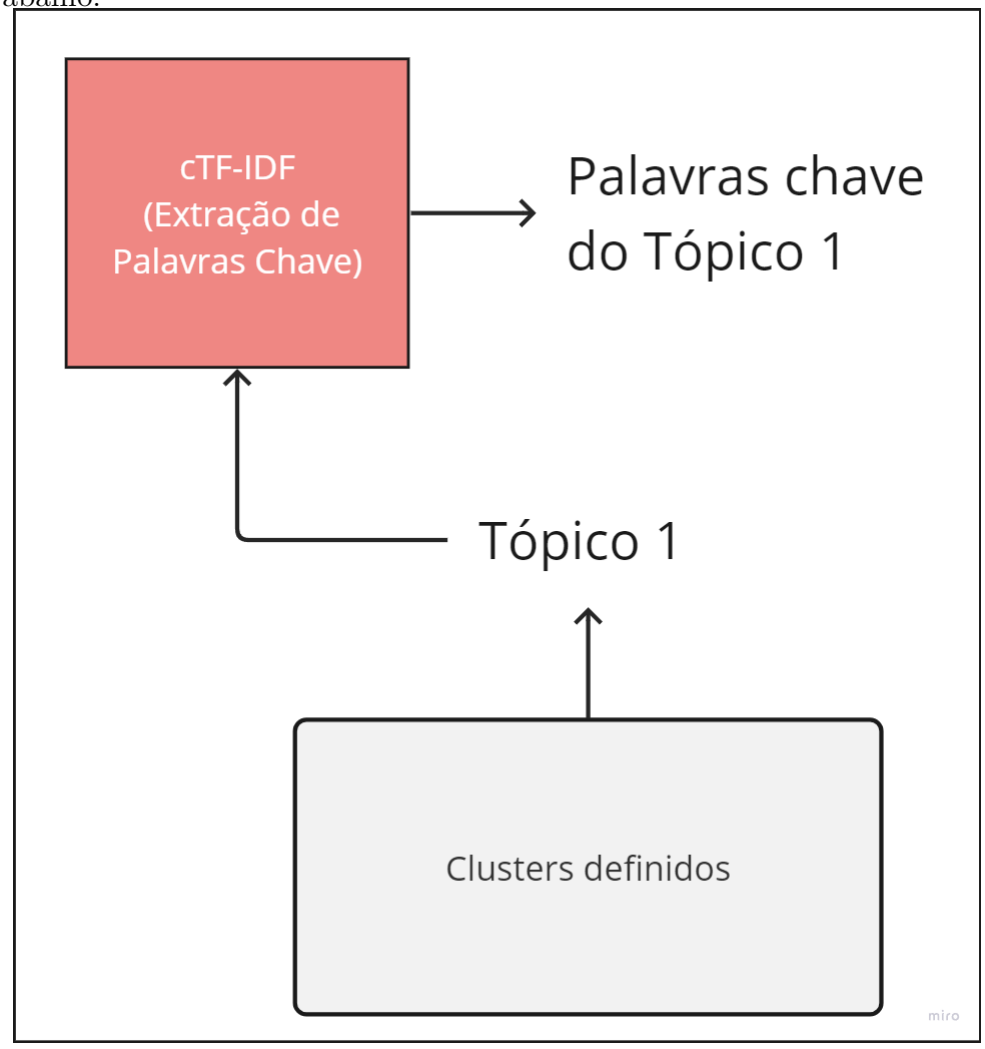


Fonte: O Autor

5.3.4 Topic Representation

Após a criação dos *clusters*, é realizado um cálculo de TF-IDF para cada tópico com o objetivo de extrair suas palavras-chave. Esse método é conhecido como cTF-IDF, em que o c indica o termo *cluster*. Com isso, as principais características que compõem cada um dos agrupamentos são destacadas em seu respectivo tópico.

Figura 5.5: *Topic Representation* do Modelo BERTopic representado para o exemplo do trabalho.[9]



Fonte: O Autor

5.3.5 Ajuste Fino

Após o treinamento do BERTopic, é possível realizar um ajuste fino através de modelos de representação. Eles podem ser métodos de extração, como o KeyBERTInspired ou modelos pré-treinados, como o BART ou LLMs.

Com o modelo BART, métodos de *zero-shot learning* são utilizados, em que são entregues palavras chave juntamente àquelas geradas pelo c-TF-IDF. Se algum tópico for semelhante àquelas palavras chave entregues, ele recebe o nome sugerido. Se não, o nome do c-TF-IDF é mantido. Exemplificando, o modelo BART recebe um input, como [“leite”, “carne”] e nomeia tópicos relacionados a esses temas com base em seu “conhecimento prévio” e as palavras destacadas.

Métodos como o KeyBERTInspired realizam a extração das palavras chave para uma melhor representação de tópicos, enquanto que modelos pré-treinados como o Zephyr Mistral 7B realizam a geração de sumários, nomes, entre diversas outras tarefas para cada tópico.

5.4 Hiperparâmetros BERTopic

O modelo BERTopic foi detalhadamente apresentado na Seção (4.9) sobre sua estrutura e métodos que o compõem. Neste segmento, uma visão geral dos hiperparâmetros da função BERTopic em ***Python* 3.11.7** serão discutidos e apresentados.

- **Modelos de *embedding*** : Foram utilizados dois tipos de *Sentence BERT*, o padrão do algoritmo, chamado de *"all-MiniLM-L6-v2"* e um outro tipo de S-BERT, o *"paraphrase-multilingual-MiniLM-L12-v2"* visando a busca de uma melhor representatividade da língua portuguesa, dado que ele é multilíngue.
- **Palavras chave**: Diversos números de palavras chave foram utilizados para cada tópico. Caracterizado por *"top_n_words"*, números como 5 e 10 foram mais aplicados. Tomando o valor 5 como exemplo, cinco palavras irão representar cada tópico.
- **Tamanho mínimo de componentes do tópico**: Representado por *"min_topic_size"*, indica o menor número de componentes necessários para formar um tópico.
- **Número de tópicos**: Representado por *"nr_topics"*, indica o número máximo de tópicos que podem ser gerados pelo modelo. Vale notar que, ao indicar um valor exemplo de "10", um dos tópicos, indicado por "-1" será gerado, formando um "tópico lixo". Esse tópico agrupa palavras que não foram aceitas em nenhum grupo.
- **Modelos de Representação (*"representation_model"*)**: São modelos que realizam um ajuste fino dos tópicos. Diversos métodos podem ser implementados, como modelos *KeyBERTInspired*, LLMs e BART (para o *zero-shot learning*).

5.5 Métrica

A métrica utilizada foi a de silhueta (Rousseeuw (1987)).Ela consiste em um valor que identifica o quão cada *embedding* é similar ao seu *cluster* comparado à outros *clusters*. Quanto maior o valor da silhueta, melhor o algoritmo realizou as clusterizações. Sua fórmula é descrita a partir da Equação (5.1):

$$\frac{r-s}{\max(s,r)}, \tag{5.1}$$

No qual **s** é a média da distância intra-*cluster* e **r** é a média do *cluster* mais próximo. O resultado final é a média de todos os pontos calculados.

Porém, não seria correto escolher os melhores modelos apenas se baseando na métrica de silhueta, uma vez que modelos com valores muito grandes de tópicos e que realizam partições de milhares de grupos provavelmente sempre possuirão uma silhueta muito alta. Outra métrica enganadora é a porcentagem do banco. Uma amostra que possui produtos bem separados pode ser sorteada, levando a valores altos de silhueta. Por exemplo, se o objetivo de um estudo é identificar os produtos mais frequentes e a população é composta pelos itens $p = \{\text{leite}, \text{leite}, \text{porta}, \text{porta}, \text{leite}, \text{leite}\}$, uma amostra de tamanho 2 pode selecionar os elementos porta e porta, resultando em uma amostra que não informa, de maneira representativa, a população.

Dessa forma, a **Silhueta** foi avaliada juntamente ao **número de tópicos gerados** e a **porcentagem** do corpus, sendo o melhor modelo definido por uma ponderação entre os três conceitos.

6 Resultados

Nessa Seção, os resultados serão apresentados para ambos os corpus, tendo cada um suas discussões em suas determinadas Seções. A primeira (6.1), se refere ao banco de dados refinado, enquanto que o segundo (6.2) indica os resultados para o corpus não refinado.

6.1 Primeira modelagem

Utilizando o primeiro banco de dados (*banco_leite_carne_5_por_cento*), a Tabela (6.1) exibe todos os resultados encontrados considerando diferentes hiperparâmetros e modelos de *embedding*. Os atributos da tabela referentes ao modelo foram identificados na Seção (5.4), com a adição de colunas para o índice; para o tempo de execução do código; para a silhueta e para a porcentagem do corpus utilizada. Vale notar que, ao não fornecer um valor para a coluna "Tamanho Mínimo" (indicado como "X"), são criados tópicos com um mínimo de 10 observações. Para a coluna "Número de Tópicos", se nenhum valor limite for especificado (indicado como "X"), os tópicos são criados sem restrição quanto ao número. Quanto à coluna "Modelo de Representação", o BERTopic não utiliza nenhuma especificação (indicado como "X").

Tabela 6.1: Tabela dos diversos modelos com diferentes hiperparâmetros e seus resultados, além de suas métricas, considerando o corpus dos produtos derivados do leite e da carne (banco refinado).

Modelo	**Modelo de *Embedding***	**Palavras Chave**	**Tamanho mínimo**	**Número de tópicos**	**Modelo de Representação**	**Tempo**	**Silhueta**	**Banco utilizado (%)**
1	"all-MiniLM-L6-v2"	5	500	X (25)	X	41 Min	0,54	50
2	"all-MiniLM-L6-v2"	5	1000	X (16)	X	2h 49 Min	0,43	100
3	"all-MiniLM-L6-v2"	5	500	X (77)	X	1d 4h 36 Min	0,70	100
4	"all-MiniLM-L6-v2"	5	500	10 (7)	X	5 Min	0,52	20
5	"p-m-MiniLM-L12-v2"	5	500	10 (10)	X	13 Min	0,63	20
6	"p-m-MiniLM-L12-v2"	5	1000	10 (5)	X	7 Min	0,61	20
7	"p-m-MiniLM-L12-v2"	5	1000	10 (7)	BART	2h 12 Min	0,66	20
8	"p-m-MiniLM-L12-v2"	5	500	5 (5)	BART	8 Min	0,50	20
9	"p-m-MiniLM-L12-v2"	5	500	10 (10)	BART	14h 49 Min	-0,08	100
10	"p-m-MiniLM-L12-v2"	5	1000	10 (8)	BART	1h 41 Min	0,65	50
11	"p-m-MiniLM-L12-v2"	5	1000	15 (6)	KeyBERT Inspired	10 Min	0,60	20
12	"p-m-MiniLM-L12-v2"	5	500	15 (13)	KeyBERT Inspired	8 Min	0,67	20
13	"p-m-MiniLM-L12-v2"	5	500	10 (10)	Mistral Zephyr 7B	8h 37 Min	0,01	100
14	"p-m-MiniLM-L12-v2"	10	X	X (5446)	X	34 Min	0,93	100
15	"p-m-MiniLM-L12-v2"	10	X	X (5470)	BART	1h 17 Min	0,93	50
16	**"p-m-MiniLM-L12-v2"**	**10**	**1000**	**X (7)**	**BART**	**9 Min**	**0,67**	**20**

Vale ressaltar que na Tabela (6.1), o *modelo de embedding "p-m-MiniLM-L12-v2"* indica o método *"paraphrase-multilingual-MiniLM-L12-v2"*. Além disso, para a coluna *Número de tópicos*, o valor entre parêntesis "()" é o número de tópicos gerado pelo modelo para a configuração de sua respectiva linha, respeitando o limite informado como hiperparâmetro. Ademais, todas iterações de BERTopic que utilizaram o modelo de representação BART receberam *["leite", "carne", "frango"]*, exceto pelo modelo 9 (será explicado abaixo).

Partindo para a análise da Tabela (6.1), alguns modelos parecem apresentaram bons resultados, são eles:

- **Modelo 9**: Ao utilizar um *Zero-Shot Learning* com o modelo BART, um erro foi realizado, em que os tópicos sugeridos foram *["leite", "carne", "frango", ""]*. A ideia de deixar um aberto ("") seria de criar um tópico livre, que não foi compreendida pelo modelo. Isso explica o valor de **-0,08** para a silhueta;
- **Modelo 13**: Foi realizado um teste com uma LLM que não obteve resultados satisfatórios. Talvez seja necessário realizar uma engenharia de *prompt* para maximizar o resultado da LLM. Isso explica o valor de **0,01** para a silhueta;
- **Modelos 14 e 15**: Dado que nenhum valor foi fornecido para o tamanho mínimo, mais de 5440 grupos foram criados para cada modelo (5446 e 5470). Logo, há uma grande quantidade de clusters, todos muito específicos, o que explica os valores de **0,93** para a silhueta.

Dado o contexto fornecido anteriormente na Seção 5.5, optou-se por identificar como os melhores modelos (Tabela 6.1) os que apresentaram os seguintes resultados:

- **Modelo 3**: sem tamanho mínimo, foram gerados 76 grupos. Mesmo que o valor da silhueta seja de **0,70**, esse modelo separa muito os dados.
- **Modelo 12**: Executando o método de *KeyBERTInspired*, a silhueta de **0,67** foi obtida, criando 13 tópicos com 20 % do banco de dados;
- **Modelo 16**: Realizando o método de *zero-shot learning* com o BART, a silhueta resultou em **0,73**, além de 7 tópicos e 20 % do corpus.

Com 6 tópicos a menos que o Modelo 12, o **Modelo 16** será analisado com um maior aprofundamento.

6.1.1 Melhor modelo

A melhor configuração da Tabela (6.1) (Modelo 16) utilizou:

- **Modelos de *embedding*** : *"paraphrase-multilingual-MiniLM-L12-v2"*;
- **Palavras chave**: 10;
- **Tamanho mínimo de componentes do tópico**: 1000;
- **Número de tópicos**: X;

- **Modelos de Representação (*"representation_model"*)** : BART (*bart-large-mnli*) para *zero-shot learning* com os tópicos *["leite", "carne", "frango"]*.

Inicialmente, devido às restrições de tempo de processamento, foi analisado apenas 20% do banco de dados. No entanto, nesta etapa posterior, optou-se por utilizar o conjunto de dados completo para uma análise mais abrangente e precisa. Esse novo modelo, aplicado ao banco completo com as mesmas configurações do Modelo 16 será denominado Modelo 17.

Tabela 6.2: Tabela do melhor modelo considerando o corpus dos produtos derivados do leite e da carne (*banco_leite_carne_5_por_cento*).

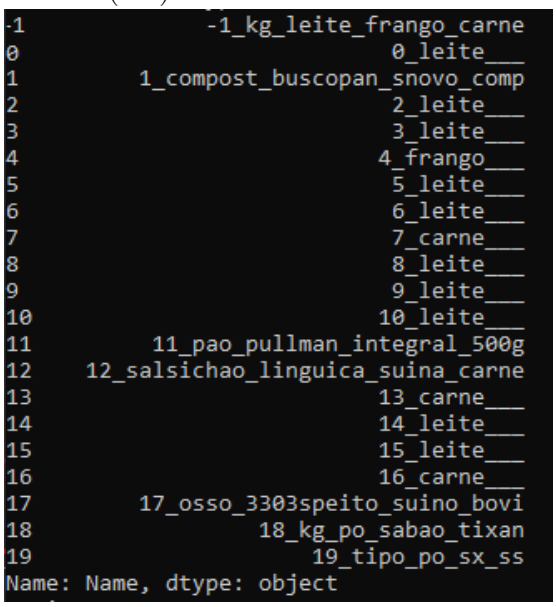
Modelo	**Modelo de *Embedding***	**Palavras Chave**	**Tamanho mínimo**	**Número de tópicos**	**Modelo de Representação**	**Tempo**	**Silhueta**	**Banco utilizado (%)**
17	"p-m-MiniLM-L12-v2"	10	1000	X (21)	BART	8h 1 Min	0,62	100

6.1.2 Resultados do melhor modelo

Partindo para a avaliação do **Modelo 17** da Tabela (6.2), a primeira imagem indica o nome de cada tópico (Figura (6.1)) enquanto que a segunda exibe a quantidade de componentes que os compõem (Figura (6.2)). As Figuras (6.3) e (6.4) indicam as descrições e os *bi-grams* mais frequentes dos tópicos.

Diversos grupos foram gerados baseados no *zero-shot learning* com o BART como observado na Imagem (6.1), além de novos agrupamentos. Ademais, valores dos tópicos estão respeitando o mínimo de 1000, como mostrado na Figura (6.2).

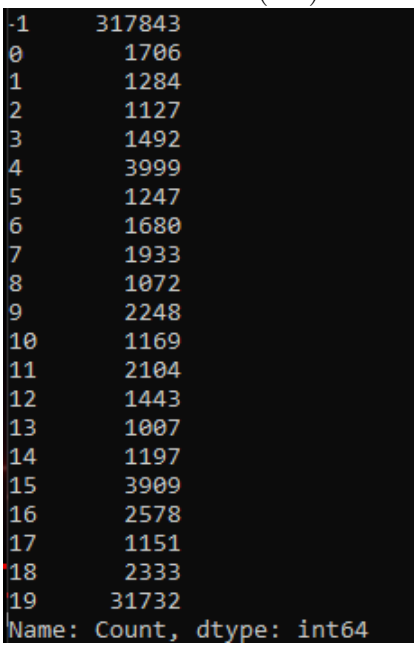
Figura 6.1: Nome de cada um dos 20 tópicos gerados do banco refinado utilizando o Modelo 17 da Tabela (6.2).[2]



```
-1                -1_kg_leite_frango_carne
0                               0_leite___
1          1_compost_buscopan_snovo_comp
2                               2_leite___
3                               3_leite___
4                              4_frango___
5                               5_leite___
6                               6_leite___
7                               7_carne___
8                               8_leite___
9                               9_leite___
10                             10_leite___
11           11_pao_pullman_integral_500g
12      12_salsichao_linguica_suina_carne
13                             13_carne___
14                             14_leite___
15                             15_leite___
16                             16_carne___
17          17_osso_3303speito_suino_bovi
18                   18_kg_po_sabao_tixan
19                       19_tipo_po_sx_ss
Name: Name, dtype: object
```

Fonte: O Autor

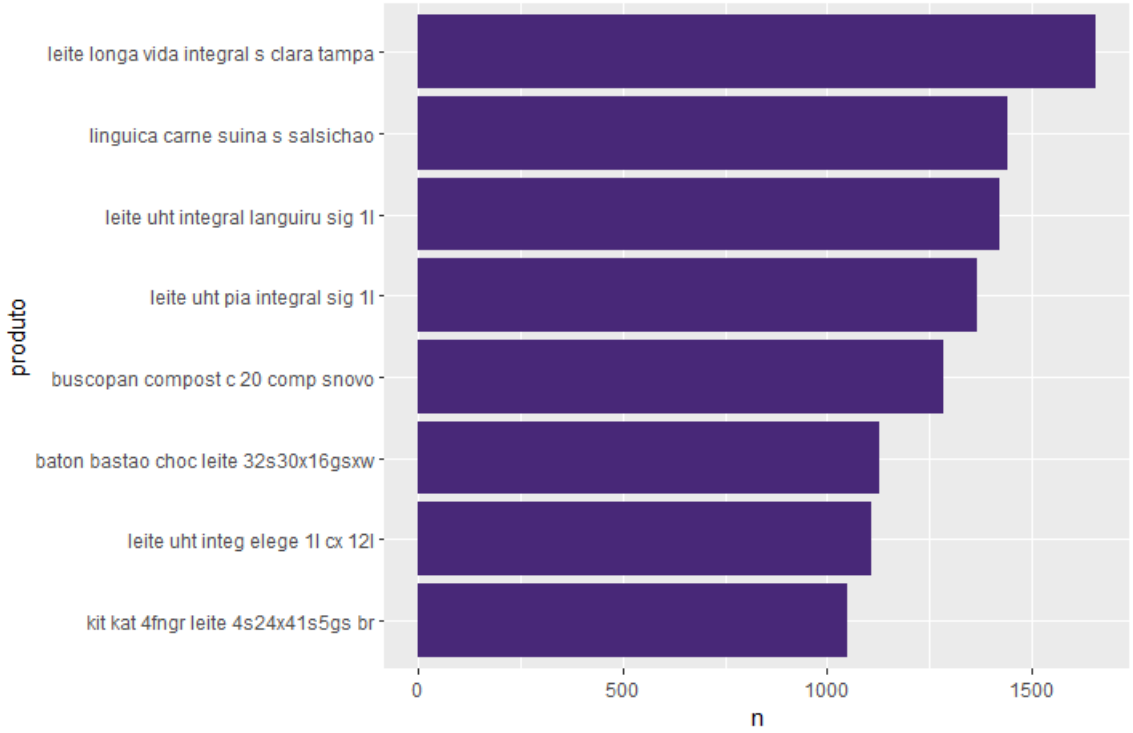
Figura 6.2: Número de palavras (*tokens*) por tópico gerado do banco refinado utilizando o Modelo 17 da Tabela (6.2).[4]



```
-1     317843
0        1706
1        1284
2        1127
3        1492
4        3999
5        1247
6        1680
7        1933
8        1072
9        2248
10       1169
11       2104
12       1443
13       1007
14       1197
15       3909
16       2578
17       1151
18       2333
19      31732
Name: Count, dtype: int64
```

Fonte: O Autor

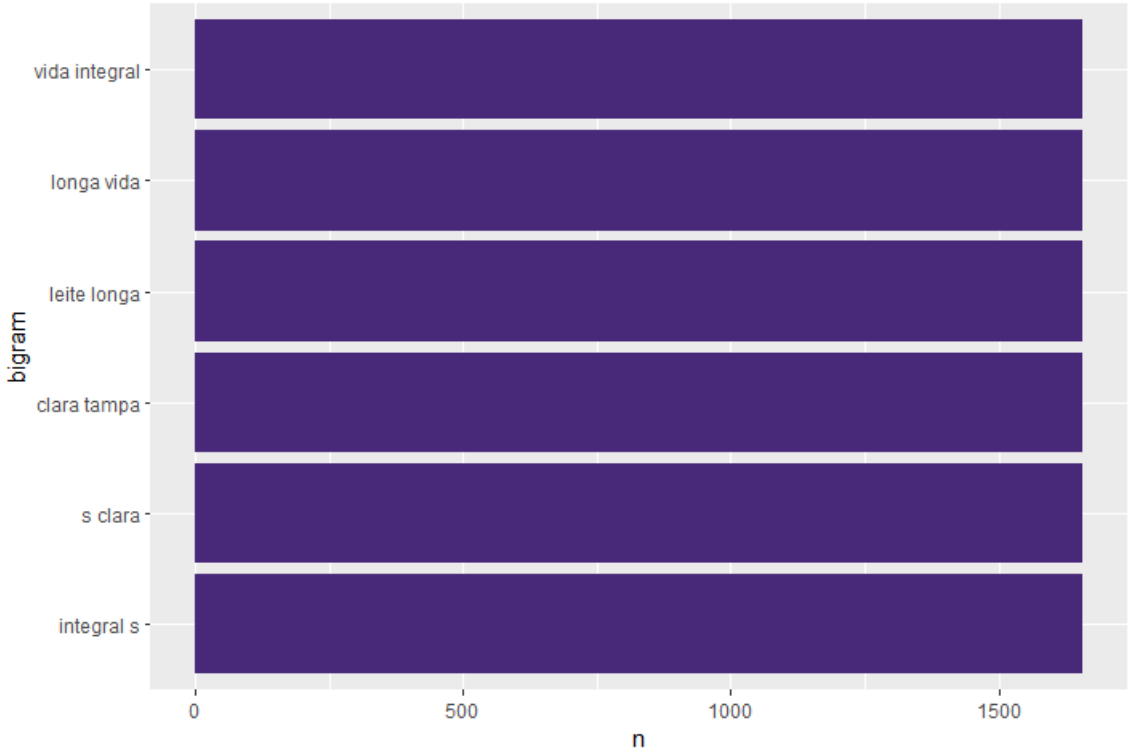
Figura 6.3: Descrições mais frequentes (maior de 1000 aparições) dos tópicos do Modelo 17 utilizando o *banco_leite_carne_5_por_cento*.[6]



Fonte: O Autor

A Figura (6.3) ilustra alguns dos produtos mais frequentes no corpus, incluindo leites, carnes, chocolates e remédios. Já a segunda Figura (6.4) exibe alguns dos *bi-grams* mais comuns, em que predominam *bi-grams* relacionados ao leite e suas marcas.

Figura 6.4: *Bi-grams* com mais de 1450 aparições dos tópicos do Modelo 17 utilizando o *banco_leite_carne_5_por_cento*.[8]



Fonte: O Autor

Os tópicos do Modelo 17 serão avaliados detalhadamente a seguir:

Tópico -1: Caracterizado pelo "lixo", o grupo -1 agrupa os produtos que não foram classificados em nenhum tópico. Apresentado pela Figura (6.5), observa-se um valor de observações alto, chegando a quase 320 mil dos mais variados tipos de produtos. Esse é um comportamento esperado do modelo BERTopic.

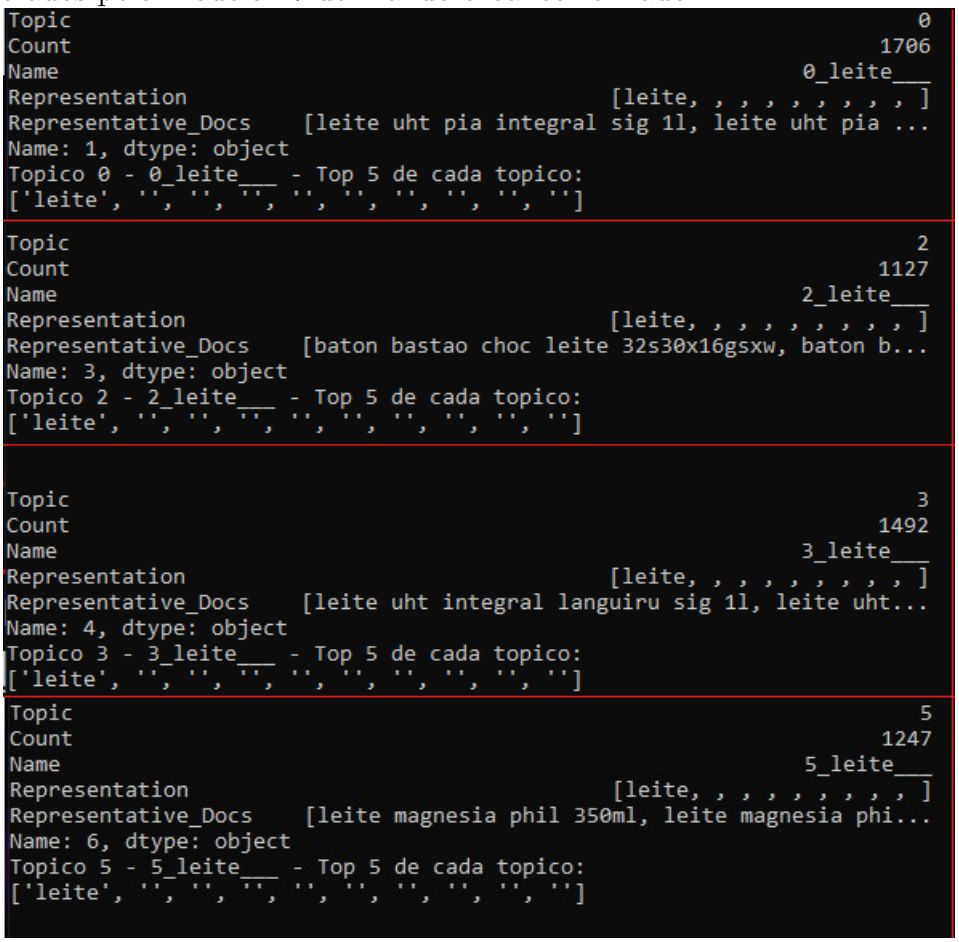
Figura 6.5: Características do tópico -1 do Modelo 17 da Tabela (6.2) utilizando o primeiro banco (carne e leite).[10]

```
Topic                                                              -1
Count                                                          317843
Name                                          -1_kg_leite_frango_carne
Representation          [kg, leite, frango, carne, po, integral, peito...
Representative_Docs     [cortes frango coxa sobrecoxa osso cong pct da...
Name: 0, dtype: object
Topico -1 - -1_kg_leite_frango_carne - Top 5 de cada topico:
['kg', 'leite', 'frango', 'carne', 'po', 'integral', 'peito', 'cong', 'cx', 'pao']
```

Fonte: O Autor

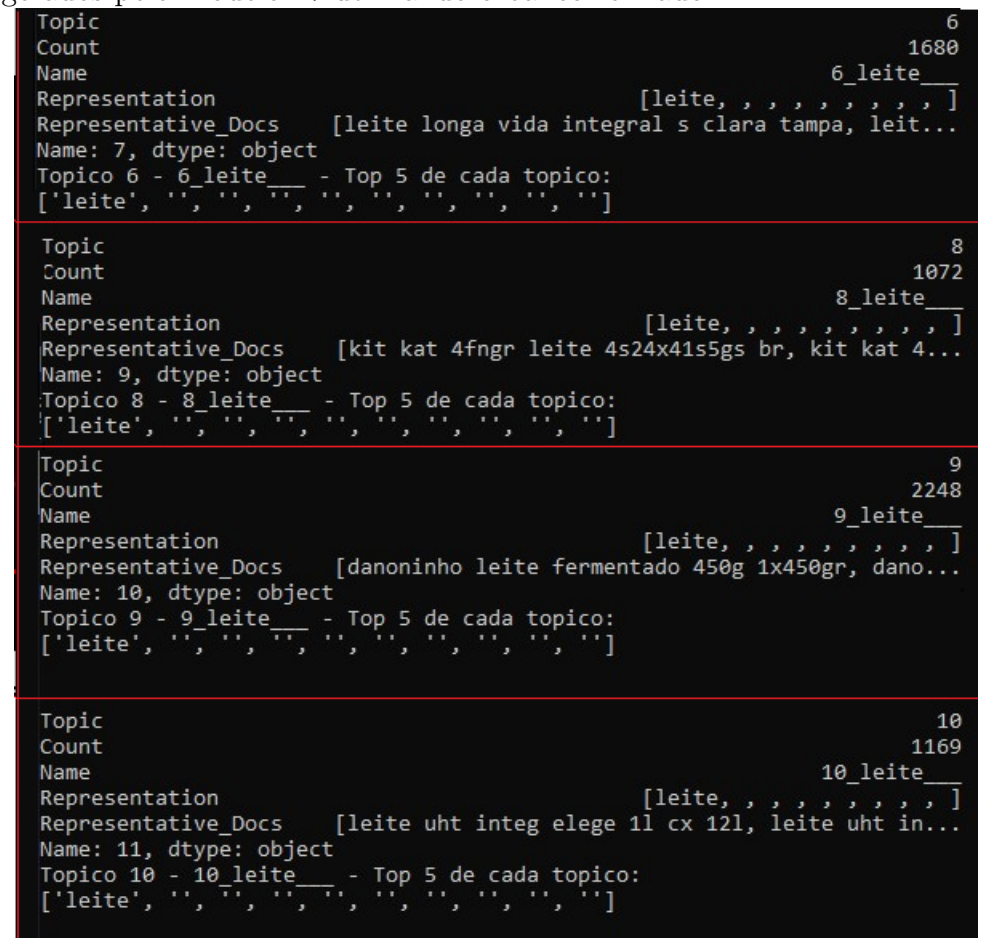
Tópicos relacionados a "Leite": Múltiplos grupos relacionados à Leite foram gerados, como é possível observar nas Imagens (6.6), (6.7) e (6.8). A Figura (6.9) mostra alguns dos principais produtos de cada tópico, enquanto que a Figura (6.10) ilustra alguns *bi-grams*.

Figura 6.6: Detalhes dos Tópicos 0, 2, 3 e 4 relacionados a produtos derivados do leite gerados pelo Modelo 17 utilizando o banco refinado.[12]



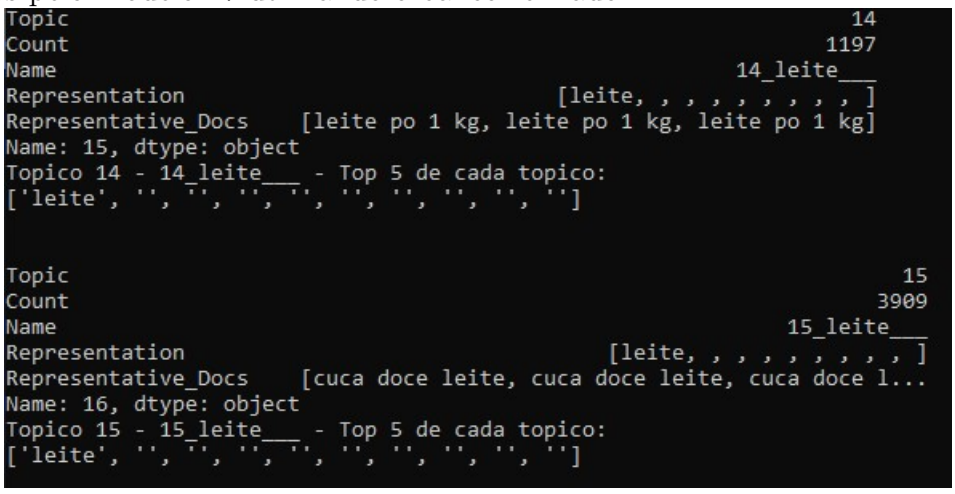
Fonte: O Autor

Figura 6.7: Detalhes dos Tópicos 6, 8, 9 e 10 relacionados a produtos derivados do leite gerados pelo Modelo 17 utilizando o banco refinado.[14]



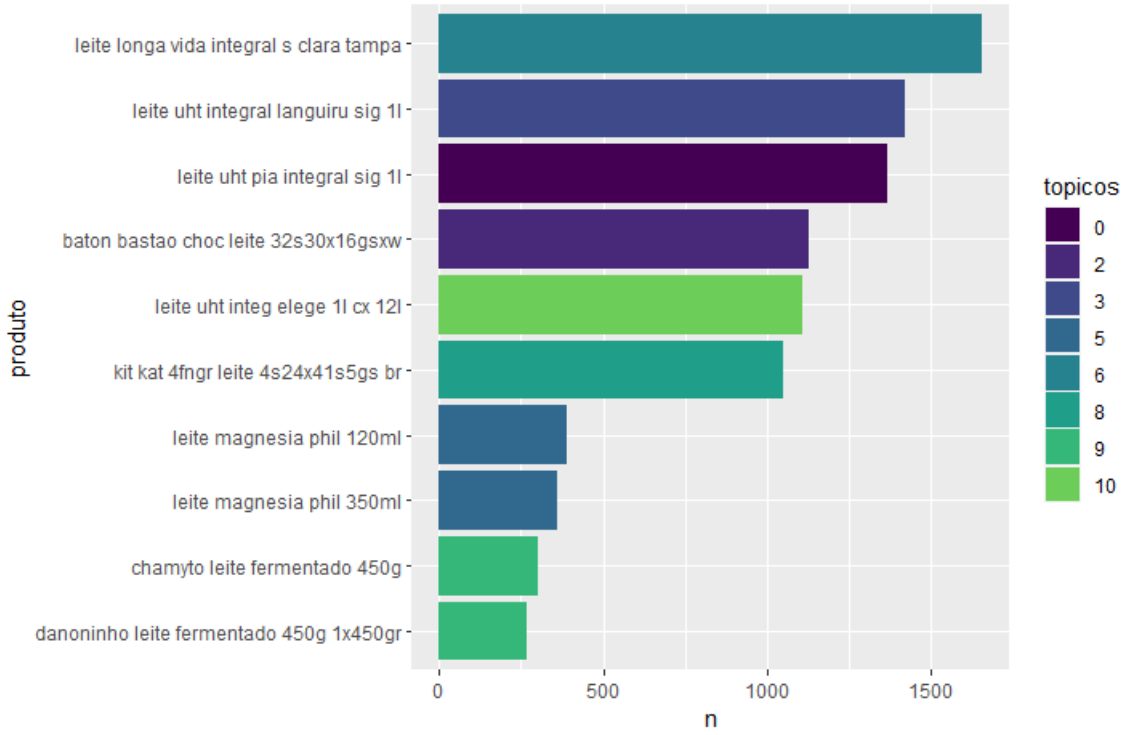
Fonte: O Autor

Figura 6.8: Detalhes dos Tópicos 14 e 15 relacionados a produtos derivados do leite gerados pelo Modelo 17 utilizando o banco refinado.[16]



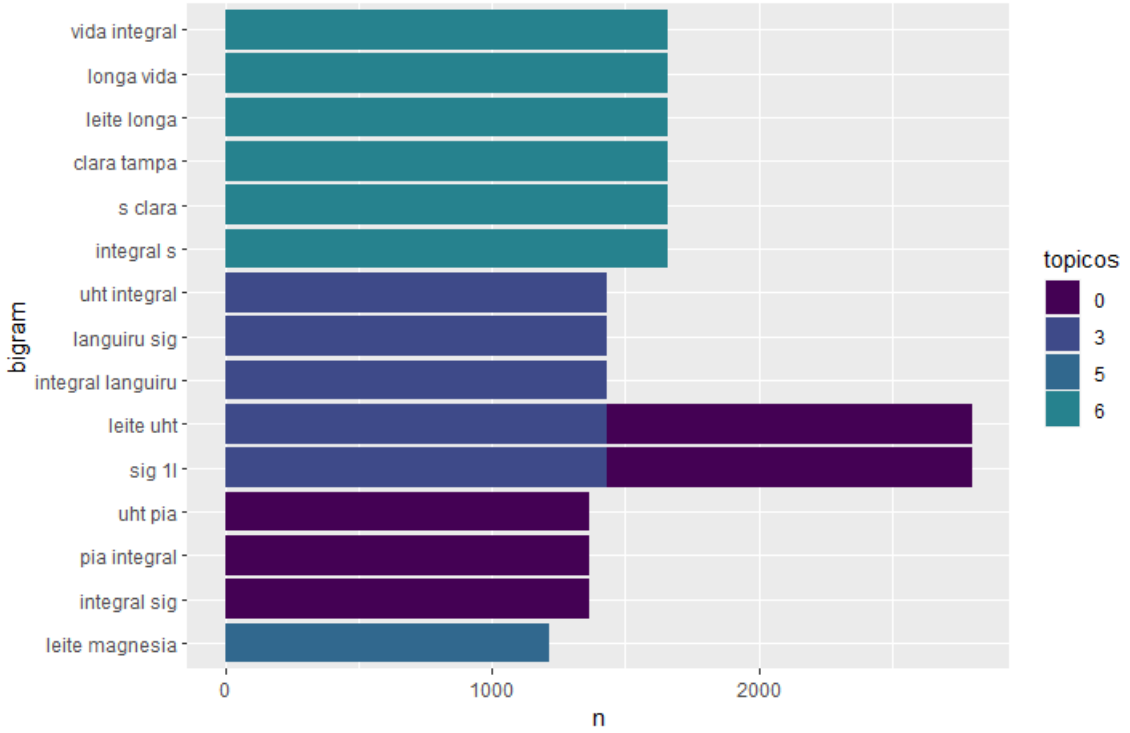
Fonte: O Autor

Figura 6.9: Descrições mais frequentes (mais de 250 aparições) dos tópicos relacionados a leite do Modelo 17 utilizando o banco refinado.[18]



Fonte: O Autor

Figura 6.10: *Bi-grams* com mais de 1200 aparições dos tópicos relacionados a leite do Modelo 17 da Tabela (6.2) utilizando o banco refinado.[20]



Fonte: O Autor

Observando a Figura (6.9), vários produtos relacionados a leite são exibidos, como bebidas e iogurtes. Vale destacar que esse gráfico exibe apenas mercadorias com uma frequência maior que 250 nos grupos relacionados a leite. A segunda (Figura (6.10)) exibe alguns dos *bi-grams* mais frequentes do corpus (com mais de 1200 aparições), possuindo algumas observações compartilhadas por tópico. Um comportamento que pode ser observado é que "Leite" muitas vezes é acompanhado de palavras como "uht" e "longa".

Assim, é possível observar que alguns dos tópicos se destacam por suas similaridades, como os seguintes grupos:

- Os grupos **0, 3, 6 e 10** das Figuras (6.6) e (6.7) representam produtos relacionados à bebida leite, separando-os em marca, como "pia" (Grupo 0), "languiru" (Grupo 3), "santa clara" (Grupo 6) e "elege" (Grupo 10), além de características como "integral", "1 litro", dentre outras. Eles somam um total de 6047 observações.

  O *BERTopic* fornece a probabilidade das observações pertencerem ao determinado tópico que foram agrupadas. Essas informações são exemplificadas na Tabela (6.3), indicando dois exemplos de cada agrupamento.

Tabela 6.3: Tabela de probabilidades do tópico 0, 3, 6 e 10 do Modelo 17 da Tabela (6.2) utilizando o *banco_leite_carne_5_por_cento*.

Produto	**Tópico agrupado**	**Probabilidade (%)**
“leite uht pia integral sig 1l”	0	1
“leite uht integ elege 1l”	0	0,079
“leite uht integral languiru sig 1l”	3	1
“chuleta bovino ssfile resfriada”	3	0,003
“leite longa vida integral s clara tampa”	6	1
“pt 1l leite condensado”	6	0,004
“leite uht integ elege 1l cx 12l”	10	1
“leite fermentado 75g cj 6x1s1s3s”	10	0,005

Observando a Tabela (6.3), os produtos que possuem probabilidade igual a 1 são os principais de cada grupo, enquanto que valores mais baixos são atribuídos à observações com uma menor relação ao agrupamento, mas que ainda assim foram selecionados. Dessa forma, essa probabilidade é apenas de pertencer ao grupo selecionado. Em alguns casos, os produtos são próximos aos principais, como o ’leite condensado’ e o ’leite fermentado’, mesmo que com probabilidades baixas. No entanto, em todos os tópicos, probabilidades altas (maiores que 50%) só são observadas em produtos que realmente pertencem aos grupos.

- Os grupos **2 e 8** das Figuras (6.6) e (6.7) representam produtos relacionados à chocolates ao leite, como por exemplo “bis” e “kit-kat”, com total de 2199 observações.

Tabela 6.4: Tabela de probabilidades do tópico 2 e 8 do Modelo 17 da Tabela (6.2) utilizando o *banco_leite_carne_5_por_cento*.

Produto	**Tópico agrupado**	**Probabilidade (%)**
“baton bastao choc leite 32s30x16gsxw”	2	1
“kit kat 4fngr leite 4s24x41s5gs br”	8	1
“descsyama po psleite 20g”	8	0,009

Partindo da Tabela (6.4), é possível observar os produtos principais dos tópicos, além de um exemplo de um produto que não se enquadra ao agrupamento. Da mesma maneira que na Tabela (6.3), as observações de baixa probabilidade da Tabela (6.4) são relativamente próximas ao tópico sugerido, contendo a palavra “leite”.

- O grupo **5** da Figura (6.6) representa produtos relacionados ao leite de magnésia, somando 1247 observações.

Tabela 6.5: Tabela de probabilidades do tópico 5 do Modelo 17 da Tabela (6.2) utilizando o *banco_leite_carne_5_por_cento*.

Produto	Tópico agrupado	Probabilidade (%)
"leite magnesia phil 350ml"	5	1

A Tabela (6.5) representa o tópico 5, mostrando o principal e mais frequente produto do seu agrupamento.

- Sendo os últimos agrupamentos de leite, os grupos **9, 14 e 15** das Figuras (6.7) e (6.8) são relacionados a produtos de iogurtes, leites em pó e doces. Ao todo são 7354 descrições de produtos separadas em cada tópico.

Tabela 6.6: Tabela de probabilidades dos tópicos 9, 14 e 15 do Modelo 17 da Tabela (6.2) utilizando o *banco_leite_carne_5_por_cento*.

Produto	Tópico agrupado	Probabilidade (%)
"chamyto leite fermentado 450g"	9	1
"iog whey 21 doce leite vc 250g cx 12"	9	1
"leite po santa clara integral 1 kg"	14	1
"fondant leite gulosina pt 20 un 1300 kg"	14	0,421
"rapadura 350gr leite"	15	1
"biscoito recheado sapeca 120g doce leite"	15	0,421

A Tabela (6.6) representa os tópicos 9, 14 e 15, mostrando alguns produtos de seus agrupamentos. Como dito, o grupo 9 é relacionado à laticínios, enquanto que os tópicos 14 e 15 são formados por produtos derivados do leite, como doces e leite em pó.

Tópicos relacionados a "Frango": Apenas um Tópico para frango foi criado, composto por 3999 observações. Nele, variados itens relacionados à frango foram agrupados, como cortes ("coxa", "sobrecoxa", "peito", "miudos") e outras informações como tipo de produto ("congelado", "pizza", "caldo"). Algumas informações do agrupamento podem ser observadas na Figura (6.11) e na Tabela (6.7).

Figura 6.11: Figura das características do tópico relacionados a frango do Modelo 17 da Tabela (6.2) utilizando o banco refinado.[22]

```
Topic                                                              4
Count                                                           3999
Name                                                     4_frango___
Representation                                [frango, , , , , , , , , ]
Representative_Docs    [frango resfriado s miudos spacotes, frango re...
Name: 5, dtype: object
Topico 4 - 4_frango___ - Top 5 de cada topico:
['frango', '', '', '', '', '', '', '', '', '']
```

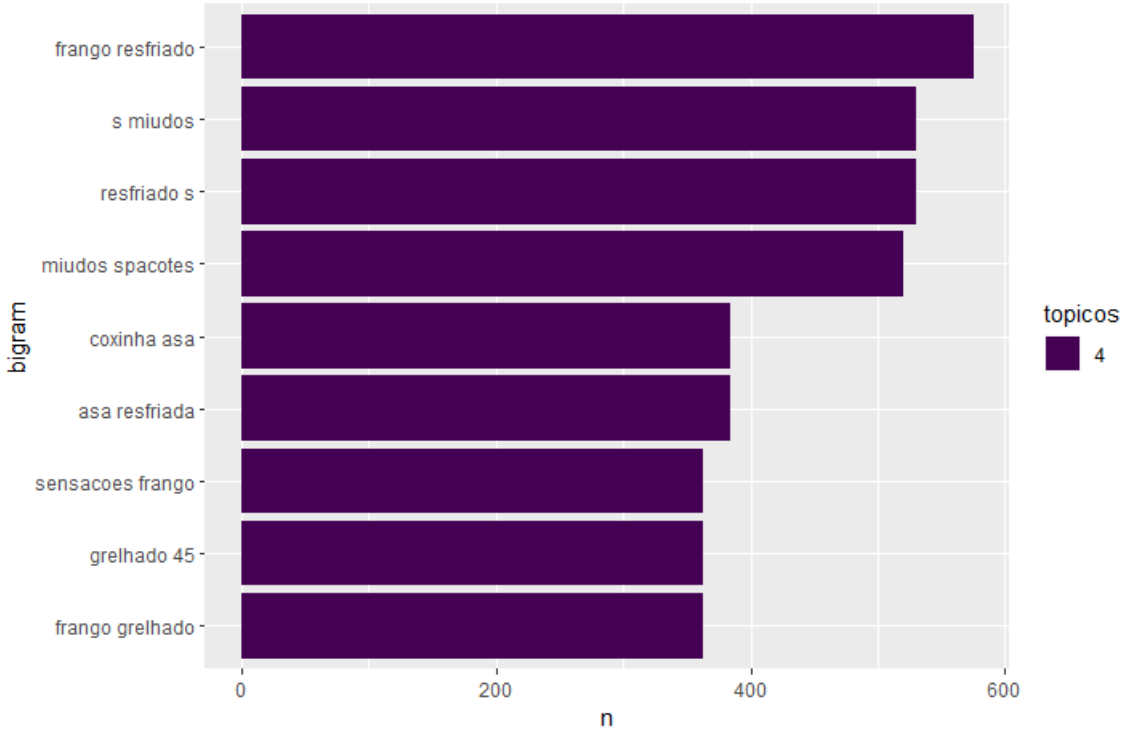
Fonte: O Autor

Tabela 6.7: Tabela de probabilidades do tópico 4 do Modelo 17 da Tabela (6.2) utilizando o *banco_leite_carne_5_por_cento*.

Produto	Tópico agrupado	Probabilidade (%)
“frango resfscsmiudos”	4	0,919
“empada frango grande”	4	1
“mortadela frango tubo lebon”	4	1
“t knorr ideal frango”	4	1
“mscsrs travesseiro frango pt 1k”	4	1

Outra informação relevante é mostrada na Figura (6.14), que mostra os *bi-grams* com frequencia maior que 350. Alguns como “frango resfriado” e “frango grelhado” exibem o tipo de carne, enquanto que “miudos spacotes” e “coxinha asa” mostram alguns tipos de cortes.

Figura 6.12: Figura dos *bi-grams* do tópico relacionado a frango do Modelo 17 da Tabela (6.2) com aparições maiores de 350 utilizando o *banco_leite_carne_5_por_cento*.[24]

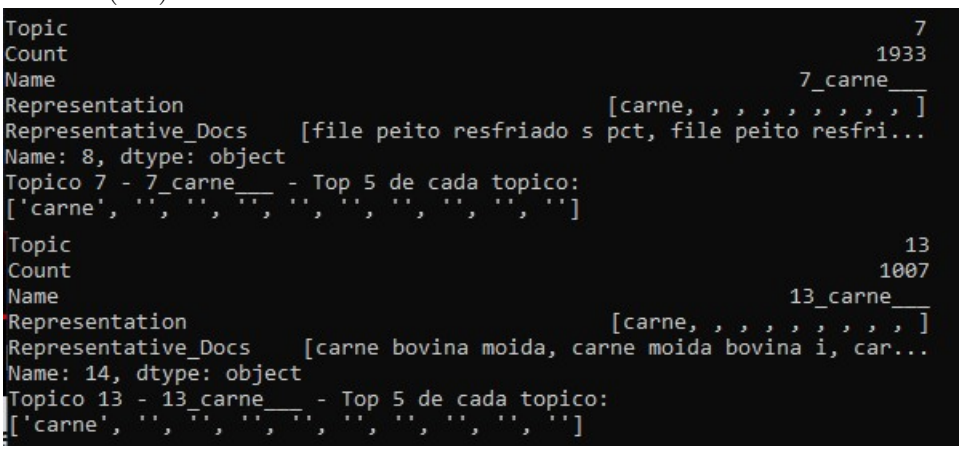


Fonte: O Autor

Tópicos relacionados a “Carne”: Dois tópicos foram obtidos a partir do *zero-shot learning*, rotulados em 7 e 13. O grupo de número 7 é referente a produtos de carne denominados “peito” e temperos saborizados em sua grande parte. Enquanto isso, o agrupamento que leva a categoria 13 é composto por itens mais variados de carne, como cortes, tipos de preparo e outros. A Figura (6.13) exibe as

principais informações de seus tópicos, enquanto que a Tabela (6.8) mostra algumas observações dos mesmos.

Figura 6.13: Figura das características dos tópicos relacionados a carne do Modelo 17 da Tabela (6.2) utilizando o banco refinado.[26]

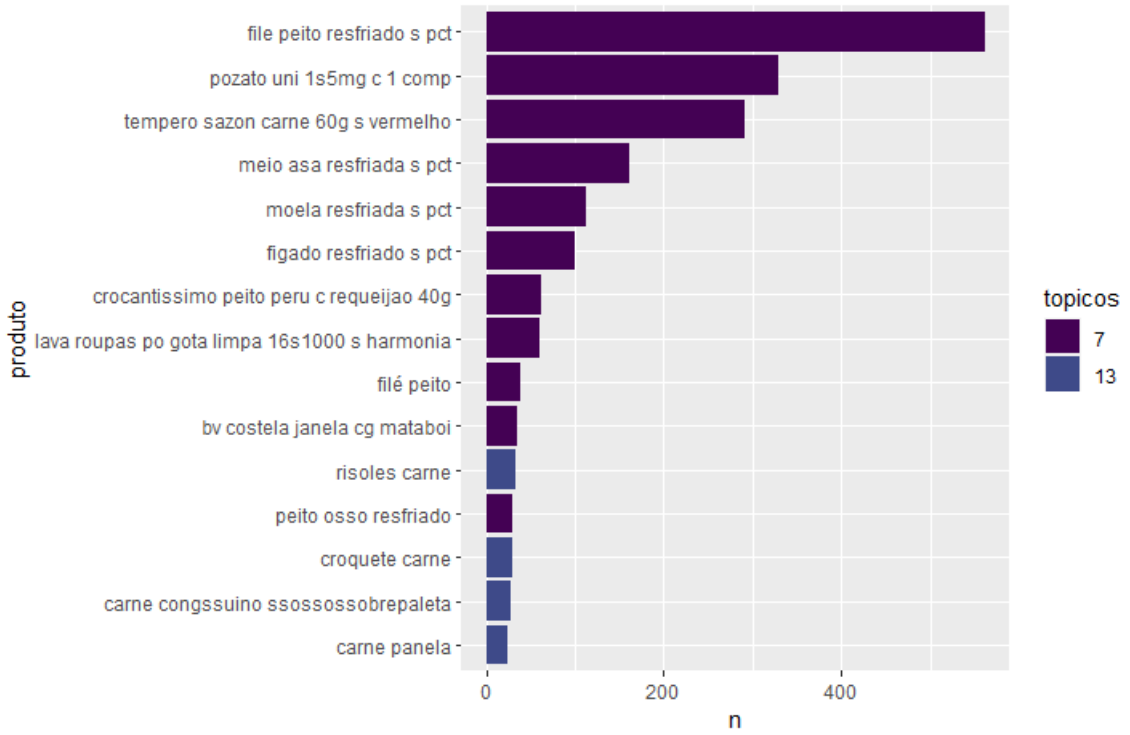


Fonte: O Autor

Tabela 6.8: Tabela de probabilidades dos tópicos 7 e 13 do Modelo 17 da Tabela (6.2) utilizando o *banco_leite_carne_5_por_cento*.

Produto	**Tópico agrupado**	**Probabilidade (%)**
"tempero sazon carne 60g s vermelho"	7	1
"file peito resfriado s pct"	7	1
"carne panela"	13	1
"linguica carne suina embalada"	13	1
"croquete carne"	13	1

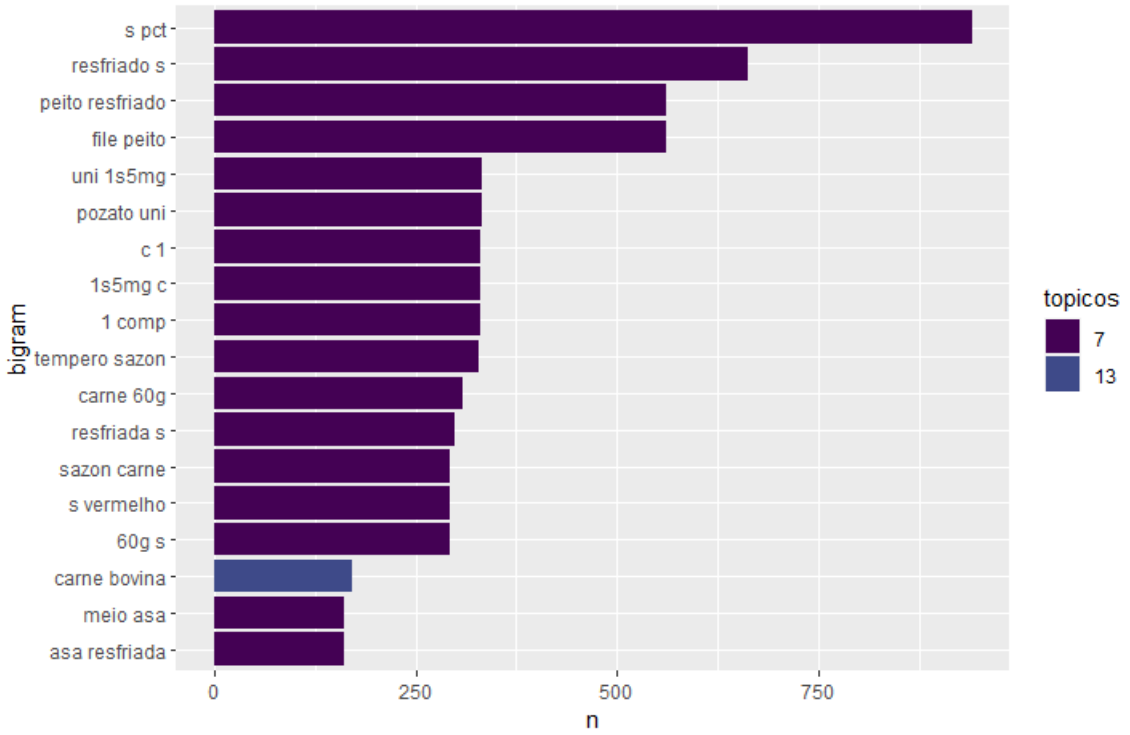
Figura 6.14: Figura das frases mais frequentes dos tópicos relacionado a carne do Modelo 17 da Tabela (6.2) utilizando o banco refinado e considerando um tamanho mínimo de 20.[28]



Fonte: O Autor

A Figura (6.14) mostra algumas das palavras mais frequentes dos tópicos. Aqui, é possível confirmar que o grupo 7 realmente trata de temperos e produtos "peito", enquanto que o agrupamento 13 é preenchido por produtos derivados de carne. Algo curioso de se observar é que a segunda frase mais frequente do tópico é um remédio contraceptivo, o "pozato uni 1s5mg c 1 comp".

Figura 6.15: Figura dos *bi-grams* do tópico relacionado a carne do Modelo 17 da Tabela (6.2) utilizando o *banco_leite_carne_5_por_cento* e considerando um tamanho mínimo de 150.[30]



Fonte: O Autor

Seguindo, a Figura (6.15) apresenta alguns dos *bi-grams* dos tópicos. Diversas palavras que normalmente são empregadas de forma conjunta são exibidas, como "peito resfriado", "tempero sazon" e "carne bovina", entre outros.

Outros tópicos: tópicos de diversos outros produtos, são eles: remédios (Grupo 1), pães (Grupo 11), linguiças (Grupo 12), ossos e carnes (Grupo 17) e produtos diversos (Grupo 18 e 19). Ao todo, esses grupos somam o 40047 descrições, sendo 31732 apenas do tópico 19.

Figura 6.16: Características dos tópicos 1, 11, 12 e 17 de produtos diversos do Modelo 17 utilizando o *banco_leite_carne_5_por_cento*.[32]

```
Topic                                                    1
Count                                                 1284
Name                                  1_compost_buscopan_snovo_comp
Representation         [compost, buscopan, snovo, comp, 20, 7p, polim...
Representative_Docs    [buscopan compost c 20 comp snovo, buscopan co...
Name: 2, dtype: object
Topico 1 - 1_compost_buscopan_snovo_comp - Top 5 de cada topico:
['compost', 'buscopan', 'snovo', 'comp', '20', '7p', 'polim', 'basic', 'disco', 'ceramico']

Topic                                                   11
Count                                                 2104
Name                                  11_pao_pullman_integral_500g
Representation         [pao, pullman, integral, 500g, 1x400gr, rap10,...
Representative_Docs    [pao integral 500g pullman, pao integral 500g ...
Name: 12, dtype: object
Topico 11 - 11_pao_pullman_integral_500g - Top 5 de cada topico:
['pao', 'pullman', 'integral', '500g', '1x400gr', 'rap10', '450g', 'visconti', 'casca', '330g']


Topic                                                   12
Count                                                 1443
Name                                  12_salsichao_linguica_suina_carne
Representation         [salsichao, linguica, suina, carne, salsichaos...
Representative_Docs    [linguica carne suina s salsichao, linguica ca...
Name: 13, dtype: object
Topico 12 - 12_salsichao_linguica_suina_carne - Top 5 de cada topico:
['salsichao', 'linguica', 'suina', 'carne', 'salsichaos', 'dd169', 'borrussia', 'divikrek', 'flocos', 'display']
Topic                                                   17
Count                                                 1151
Name                                  17_osso_3303speito_suino_bovi
Representation         [osso, 3303speito, suino, bovi, palito, bob, c...
Representative_Docs                      [osso kg, osso kg, osso kg]
Name: 18, dtype: object
Topico 17 - 17_osso_3303speito_suino_bovi - Top 5 de cada topico:
['osso', '3303speito', 'suino', 'bovi', 'palito', 'bob', 'cong', 'pernil', 'flex', 'dianteiro']


Topic                                                   18
Count                                                 2333
Name                                  18_kg_po_sabao_tixan
Representation         [kg, po, sabao, tixan, det, ype, maciez, peso,...
Representative_Docs    [sabao po 5 kg, sabao po s kg s, sabao po 1 kg...
Name: 19, dtype: object
Topico 18 - 18_kg_po_sabao_tixan - Top 5 de cada topico:
['kg', 'po', 'sabao', 'tixan', 'det', 'ype', 'maciez', 'peso', 'larg', 'pes']
```

Fonte: O Autor

Figura 6.17: Características do tópico 19 de produtos diversos do Modelo 17 utilizando o *banco_leite_carne_5_por_cento*.[34]

```
Topic                                                   19
Count                                                31732
Name                                  19_tipo_po_sx_ss
Representation         [tipo, po, sx, ss, luva, ma, abracadeira, zb, ...
Representative_Docs    [po sx unc 7s16 chv 5s8 zb, po sx 5s8 unc 1s4 ...
Name: 20, dtype: object
Topico 19 - 19_tipo_po_sx_ss - Top 5 de cada topico:
['tipo', 'po', 'sx', 'ss', 'luva', 'ma', 'abracadeira', 'zb', 'cs', 'polo']
```

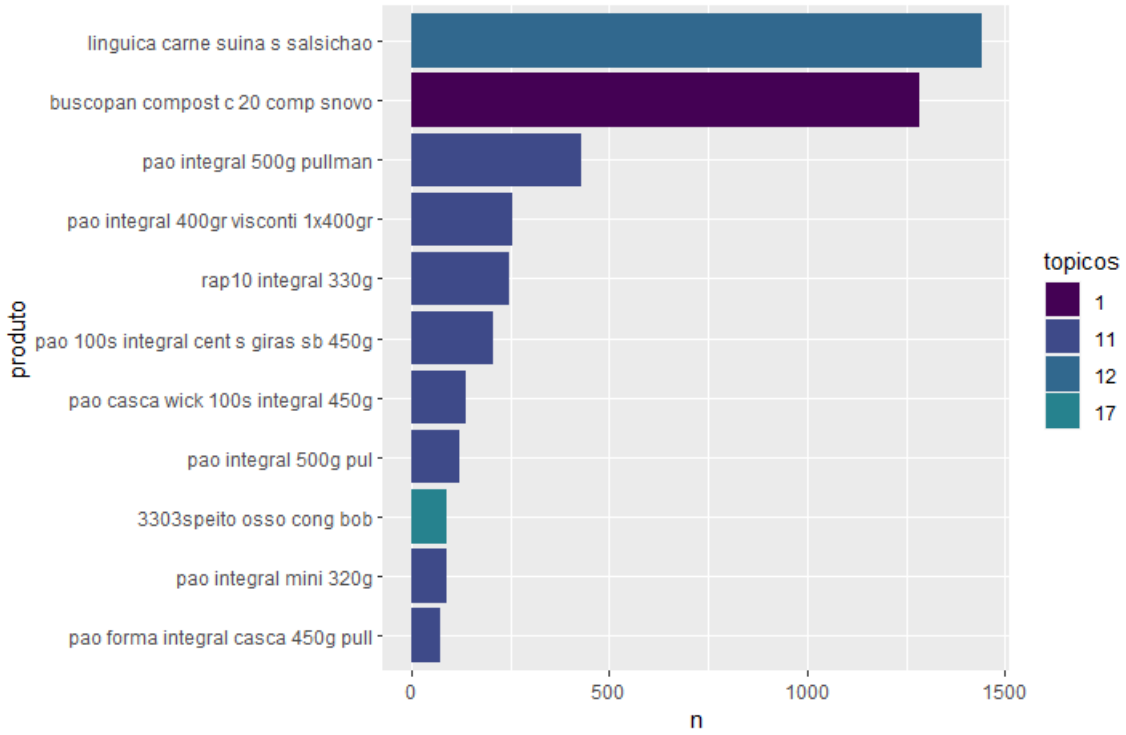
Fonte: O Autor

Tabela 6.9: Probabilidades de observações dos tópicos 1, 11, 12, 17, 18 e 19 do Modelo 17 da Tabela (6.2) utilizando o banco refinado.

Produto	**Tópico agrupado**	**Probabilidade (%)**
“buscopan compost c 20 comp snovo”	1	1
“pao casca wick 100s integral 450g”	11	1
“pao zero s integral 350g pullman”	11	1
“linguica carne suina s salsichao”	12	1
“osso bovi costela kg”	17	1
“carcaca suina osso”	17	1
“sab po tixan 2 kg maciez caixa”	18	1
“cacau po cargill 25 kg”	18	1
“po trav m8 nyl”	19	1
“encosto tipo almofada m at2043”	19	1

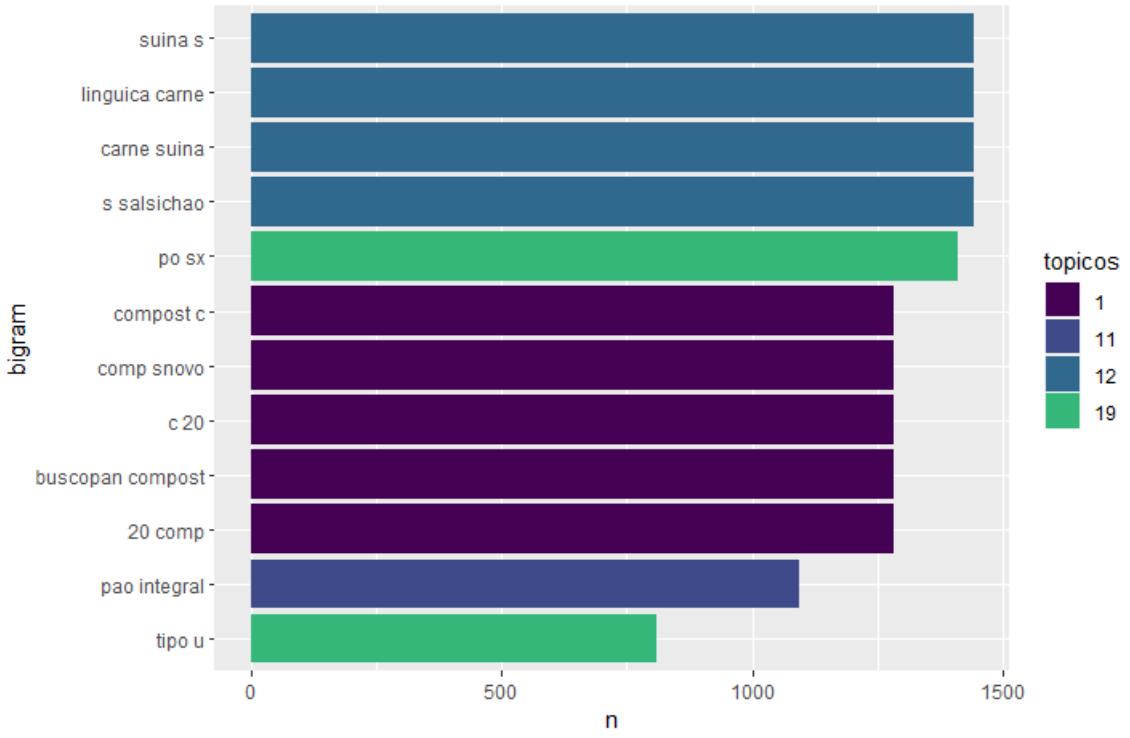
A Figura (6.17) exibe as informações dos tópicos listados, enquanto que a Tabela (6.9) apresenta alguns produtos variados de cada grupo. Desses, os agrupamentos rotulados como 18 e 19 foram os mais variados, ambos apresentando muitos itens com o nome “pó”.

Figura 6.18: Frases mais frequentes dos tópicos 1, 11, 12, 17, 18 e 19 do Modelo 17 da Tabela (6.2) utilizando o *banco_leite_carne_5_por_cento* e considerando um tamanho mínimo de 70.[36]



Fonte: O Autor

Figura 6.19: *Bi-grams* dos tópicos 1, 11, 12, 17, 18 e 19 do Modelo 17 da Tabela (6.2) utilizando o *banco_leite_carne_5_por_cento* e considerando um tamanho mínimo de 800.[38]

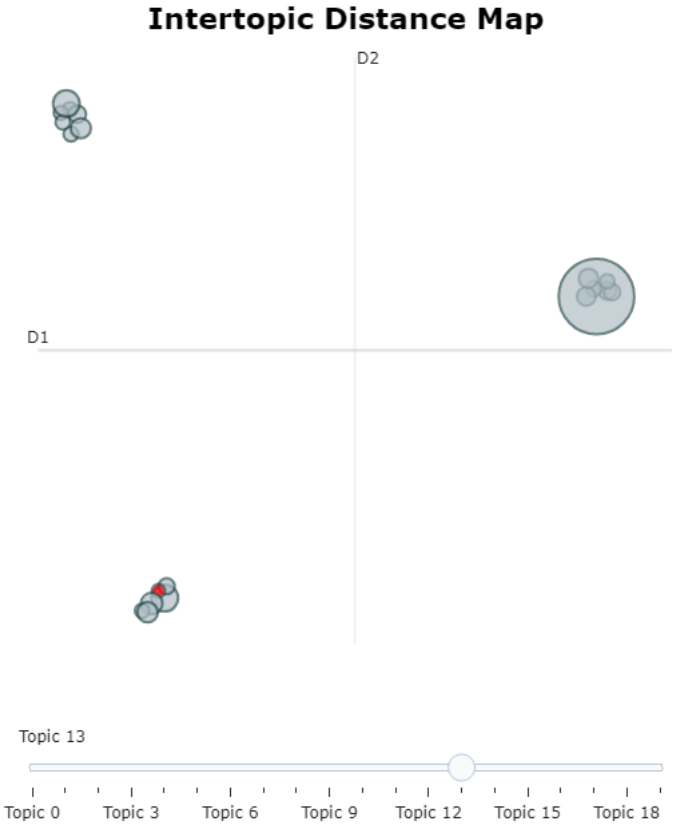


Fonte: O Autor

Por fim, as Figuras (6.18) e a (6.19) indicam as descrições e os *bi-grams* mais frequentes, respectivamente. Nota-se que os tópicos 18 e 19 são compostos por produtos mais diversos. Produtos como remédios ("buscopan compost c 20 comp snovo"), linguiças ("linguica carne suina s salsichao"), pães ("pao integral 500g pullman") e ossos ("3303speito osso cong bob") aparecem com uma maior frequência. Enquanto isso, a Figura (6.19) exibe alguns *bi-grams* relacionados a linguiças e remédios.

Analisando os dados em conjunto, percebe-se que grande parte dos grupos gerados pelo BERTopic não possuem grandes *outliers*, ou seja, possuem muitas descrições similares, exceto pelos Tópicos 18 e 19. A seguir, serão exibidos três gráficos: o primeiro mostra os tópicos no espaço vetorial; o segundo apresenta as principais palavras de cada grupo; e o terceiro exibe uma matriz de similaridade entre os agrupamentos.

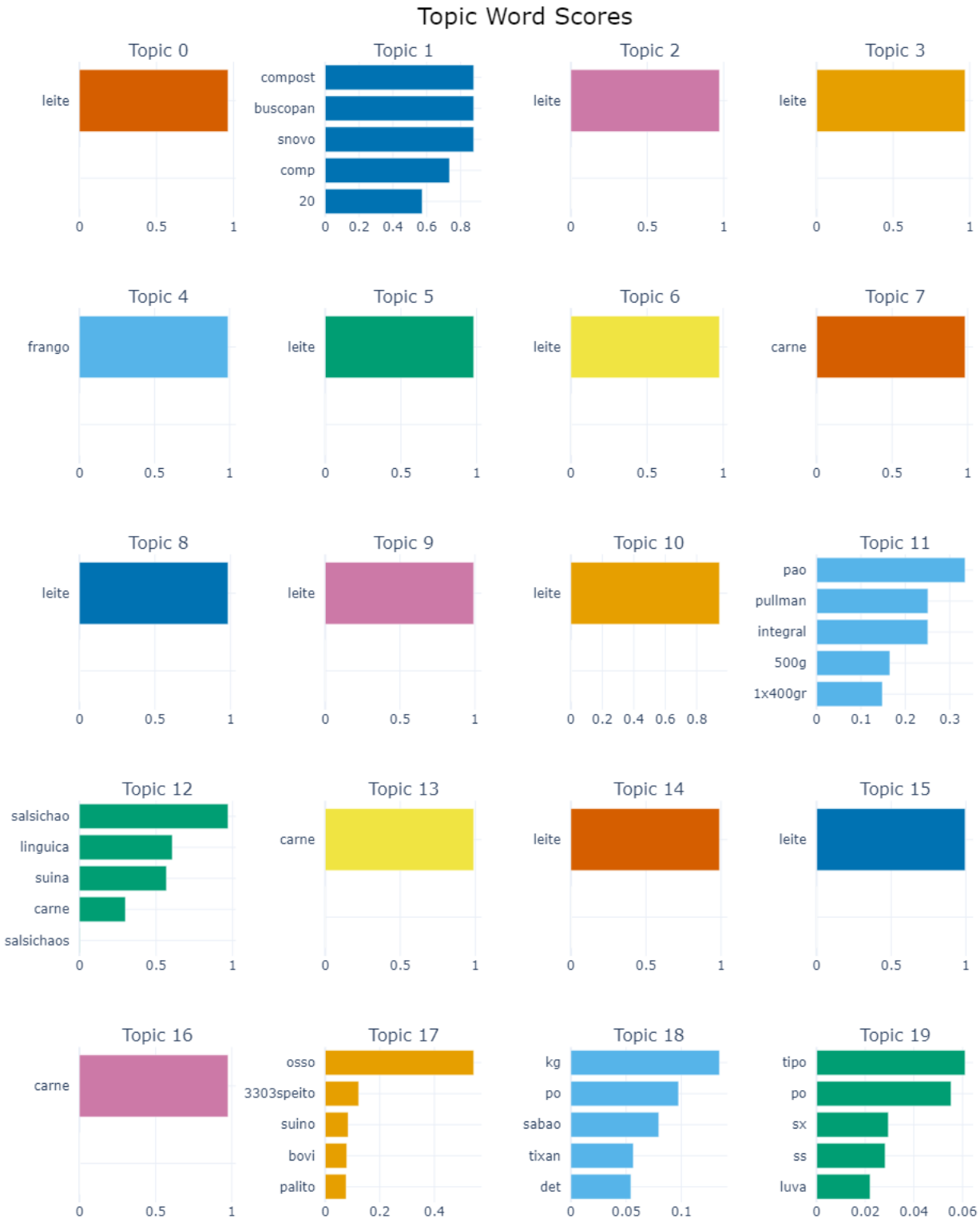
Figura 6.20: Características dos tópicos diversos do Modelo 17 da Tabela (6.2) utilizando o *banco_leite_carne_5_por_cento*. Em vermelho, o grupo 13 é indicado.[40]



Fonte: O Autor

A Figura (6.20) mostra os diferentes agrupamentos no espaço vetorial. No canto inferior esquerdo os tópicos 13 e 16 (Carne), 12 (Linguiças), 4 (Frango), 17 (Ossos) e 18 (Geral) são encontrados. No canto superior esquerdo 7 dos 10 grupos de Leite, sendo eles os tópicos 2 (Leite - Chocolate), 5 (Leite - Magnésia), 6 (Leite - Santa Clara), 8 (Leite - Chocolate), 9 (Leite - Iogurtes), 14 (Leite - Pó e Doces) e 15 (Leite - Doces). Finalmente, o agrupamento na esquerda é composto pelos tópicos 0 (Leite - Pia), 1 (Remédios), 3 (Leite - Languiru), 7 (Carne - Peito), 10 (Leite - Elege), 11 (Pães) e 19 (Geral). É curioso observar que três dos quatro tópicos de leite, que indicam as marcas, estão juntos no espaço vetorial, longe dos outros produtos derivados do leite. Porém, mesmo que alguns tópicos estejam mal posicionados, o comportamento geral parece estar bem definido, com o agrupamento inferior esquerdo representando produtos de carne, o agrupamento superior esquerdo para mercadorias de leite e o direito para produtos diversos.

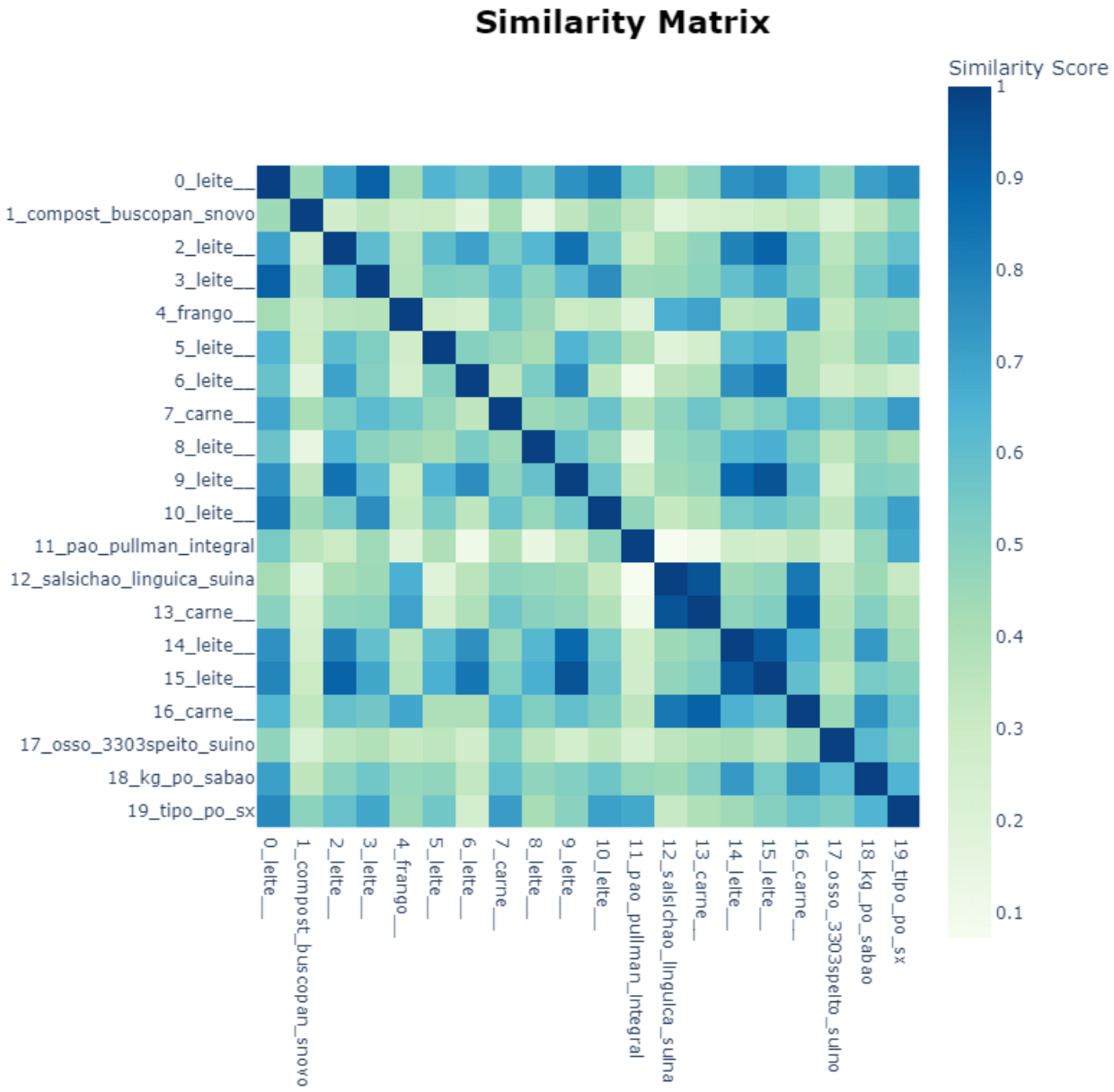
Figura 6.21: Palavras com maior score dos tópicos gerados pelo Modelo 17.[41]



Fonte: O Autor

Na Figura (6.21), escores de palavra por tópico são ilustradas, o que possibilita a identificação das palavras mais influentes e importantes para cada grupo. Maiores valores indicam que os tópicos estão mais específicos. Os agrupamentos 11, 18 e 19 são os menos específicos, tendo suas palavras principais (“pão”, “kg” e “tipo”) com menores valores para os scores (0,33 ; 0,13 ; 0,06, respectivamente). Os tópicos obtidos a partir do *“zero-shot learning”* obtiveram valores maiores que 0,94. Por fim, os grupos gerados com valores altos são o 1, 12 e 17, todos com suas palavras chave (“compost”, “salsichao” e “osso”) possuindo escores elevados (0,88 ; 0,97 ; 0,54, respectivamente).

Figura 6.22: Matriz de Similaridade dos tópicos do Modelo 17.[42]



Fonte: O Autor

Finalizando, a terceira Figura (6.22) exibe uma matriz de similaridade entre os tópicos, em que valores próximos de 1 (Tons de azul mais escuros) indicam maior similaridade, enquanto que valores próximos de 0 (Tons de verde mais esbranquiçados) possuem menor similaridade. Algumas características podem ser observadas:

- **Carnes**: Tópicos associados à carne possuem maior similaridade com agrupamentos relacionados à carne. Exemplificando, o grupo 4 (Frango) é mais similar aos tópicos 7 (Carne - Peito), 12 (Linguiças), 13 e 16 (Carnes), com valores de 0,55, 0,66, 0,70 e 0,69, respectivamente;
- **Leites**: Tópicos associados à leite possuem maior similaridade com agrupamentos relacionados à leite. Exemplificando, o grupo 0 (Leite - Pia) é mais similar aos tópicos: 2 (Leite - Chocolate) com valor de 0,71; 3 (Leite - Languiru) com valor de 0,90; 5 (Leite - Magnésia) com valor de 0,64; 6 (Leite - Santa Clara) com valor de 0,58; 8 (Leite - Chocolate) com valor de 0,58; 9 (Leite - Iogurtes) com valor de 0,75; 10 (Leite - Elege) com valor de 0,83; 14 (Leite - Pó e Doce) com valor de 0,75; e 15 ((Leite - Doces) com valor de 0,79.
- **Outros produtos**: Grupos como o 1 (Remédios), 11 (Pães) e 17 (Ossos) não possuem alta similaridade com outros tópicos. Os agrupamentos 18 e 19 são mais semelhantes à muitos tópicos, uma vez que possuem diversos tipos de produtos.

Dessa forma, foi possível observar que o modelo BERTopic conseguiu realizar um agrupamento muito bom, levando em consideração que criou 21 tópicos a partir de 300 mil observações. Os tópicos 18 e 19 agruparam diversos produtos mais gerais, que foi identificado pelo modelo com escores mais baixos.

6.2 Segunda modelagem

O segundo banco de dados a ser analisado é composto por uma diversidade ainda maior de produtos e busca avaliar a capacidade do BERTopic na construção de tópicos considerando um banco menos específico. A Tabela (6.2) possuí as mesmas colunas que a (6.1), sendo apenas adaptada aos dados do segundo banco. Uma única diferença são os tópicos sugeridos ao modelo de representação BART, exibindo qual foi utilizado em parêntesis.

Tabela 6.10: Modelos considerando diferentes hiperparâmetros, considerando o corpus com produtos mais diversos (*bancao_1_por_cento*)

.

Modelo	**Modelo de *Embedding***	**Palavras Chave**	**Tamanho mínimo**	**Número de tópicos**	**Modelo de Representação**	**Tempo**	**Silhueta**	**Banco utilizado (%)**
1	"p-m-MiniLM-L12-v2"	10	500	X (5)	X	11 Min	0,61	20
2	"p-m-MiniLM-L12-v2"	10	1000	X (6)	X	9 Min	0,63	50
3	"p-m-MiniLM-L12-v2"	10	500	X (13)	X	8 Min	0,47	50
4	"p-m-MiniLM-L12-v2"	10	X	X (2658)	X	7 Min	0,81	50
5	"p-m-MiniLM-L12-v2"	10	X	X (2686)	BART (1)	41 Min	0,82	100
6	"p-m-MiniLM-L12-v2"	10	1000	X (6)	BART (1)	9 Min	0,61	50
7	"p-m-MiniLM-L12-v2"	10	500	X (17)	BART (1)	9 Min	0,66	50
8	"p-m-MiniLM-L12-v2"	10	500	X (10)	X	8 Min	0,57	50
9	"p-m-MiniLM-L12-v2"	10	500	X (10)	KeyBERT Inspired	3h 4 Min	0,57	50
10	"p-m-MiniLM-L12-v2"	10	500	X (36)	BART (1)	26 Min	0,64	100
11	"p-m-MiniLM-L12-v2"	10	1000	X (12)	BART (2)	49 Min	0,42	100
12	**"p-m-MiniLM-L12-v2"**	**10**	**500**	**X (34)**	**BART (2)**	**27 Min**	**0,65**	**100**
13	"p-m-MiniLM-L12-v2"	10	1000	X (10)	BART (3)	48 Min	0,38	100
14	"p-m-MiniLM-L12-v2"	10	500	X (36)	BART (3)	52 Min	0,64	100
15	"p-m-MiniLM-L12-v2"	10	500	22 (22)	BART (4)	36 Min	0,28	100
16	"p-m-MiniLM-L12-v2"	10	500	31 (31)	BART (1)	23 Min	0,59	100

Na Tabela (6.10), os modelos de representação BART possuem diversos tópicos. Para o (1), *[“leite”, “carne”, “frango”]*, enquanto que o (2) *[“leite”, “carne”, “frango”, “shampoo”, “porta”, “linguica”, “pao”]*, o (3) *[“leite”, “carne”, “frango”, “linguica”, “pao”]* e o (4) *[“leite”, “carne”, “frango”, “linguica”, “pao”, “cafe”]*.

Diversos modelos foram gerados, porém alguns se destacam. São eles:

- **Modelos 4 e 5**: Com mais de 2650 tópicos, os dois modelos obtiveram métricas de 0,81 e 0,82 para a silhueta, respectivamente;
- **Modelos 2, 6, 7, 10, 12 e 14**: Todos com um tamanho mínimo de 500 observações, a silhueta foi maior que 0,60 para todos os modelos.

Utilizando o mesmo critério àquele definido na Seção 5.5, o melhor modelo será escolhido. Porém, dado que o banco de dados *bancao_1_por_cento* não passou por nenhum refinamento comparado à cadeia de Markov aplicada para a obtenção do *banco_leite_carne_5_por_cento*, uma maior flexibilidade em relação ao número de tópicos será tomada.

Dessa forma, o **Modelo 12** será analisado com um maior detalhamento, uma vez que sua silhueta é maior que (0,65) e seu processamento foi realizado com 100% do corpus.

6.2.1 Melhor modelo

O melhor modelo, portanto, foi o seguinte:

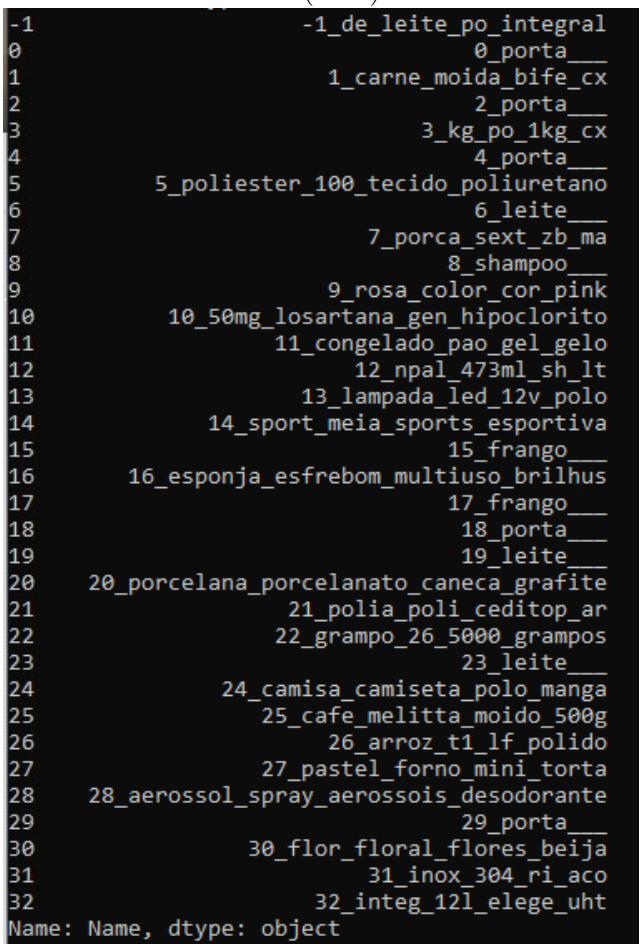
- **Modelos de *embedding*** : *“paraphrase-multilingual-MiniLM-L12-v2”*;
- **Palavras chave**: 10;
- **Tamanho mínimo de componentes do tópico**: 500;
- **Número de tópicos**: X;
- **Modelos de Representação (*“representation_model”*)** : BART (*bart-large-mnli*) para *zero-shot learning* com os tópicos *[“leite”, “carne”, “frango”, “shampoo”, “porta”, “linguica”, “pao”]*.

6.2.2 Resultado do melhor modelo

Partindo para a avaliação do **Modelo 12** da Tabela (6.10), serão mostrados os nomes dos tópicos (Figura (6.23)), o número de componentes (Figura (6.24)), as descrições mais frequentes (Figura (6.25)) e os *bi-grams* (Figura (6.26)).

Utilizando um banco com maior quantidade de observações, grupos relacionados a leite, carnes, roupas, cervejas, entre diversos outros foram gerados como observado nas Figuras (6.23) e (6.24).

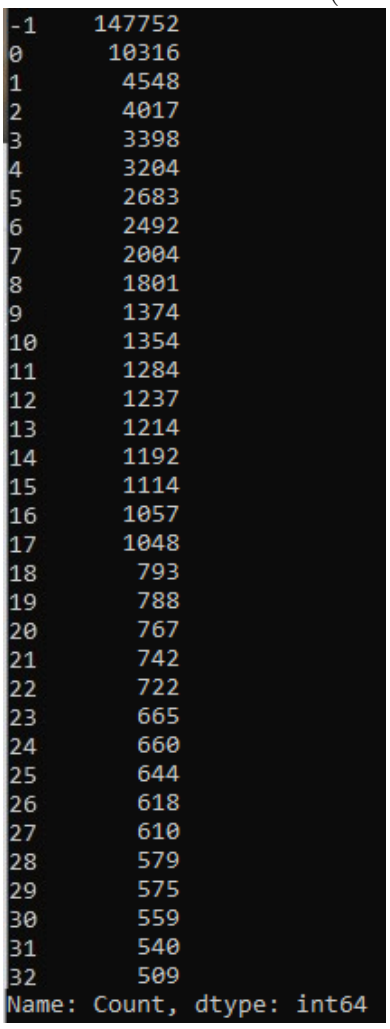
Figura 6.23: Nome de cada tópico gerado do banco *bancao_1_por_cento* utilizando o Modelo 12 da Tabela (6.10).[44]



```
-1                                -1_de_leite_po_integral
0                                              0_porta___
1                                 1_carne_moida_bife_cx
2                                              2_porta___
3                                       3_kg_po_1kg_cx
4                                              4_porta___
5                    5_poliester_100_tecido_poliuretano
6                                              6_leite___
7                                    7_porca_sext_zb_ma
8                                         8_shampoo___
9                                  9_rosa_color_cor_pink
10                    10_50mg_losartana_gen_hipoclorito
11                           11_congelado_pao_gel_gelo
12                                 12_npal_473ml_sh_lt
13                           13_lampada_led_12v_polo
14                       14_sport_meia_sports_esportiva
15                                     15_frango___
16                  16_esponja_esfrebom_multiuso_brilhus
17                                     17_frango___
18                                      18_porta___
19                                      19_leite___
20    20_porcelana_porcelanato_caneca_grafite
21                           21_polia_poli_ceditop_ar
22                         22_grampo_26_5000_grampos
23                                      23_leite___
24                       24_camisa_camiseta_polo_manga
25                          25_cafe_melitta_moido_500g
26                               26_arroz_t1_lf_polido
27                           27_pastel_forno_mini_torta
28    28_aerossol_spray_aerossois_desodorante
29                                     29_porta___
30                        30_flor_floral_flores_beija
31                                31_inox_304_ri_aco
32                             32_integ_12l_elege_uht
Name: Name, dtype: object
```

Fonte: O Autor

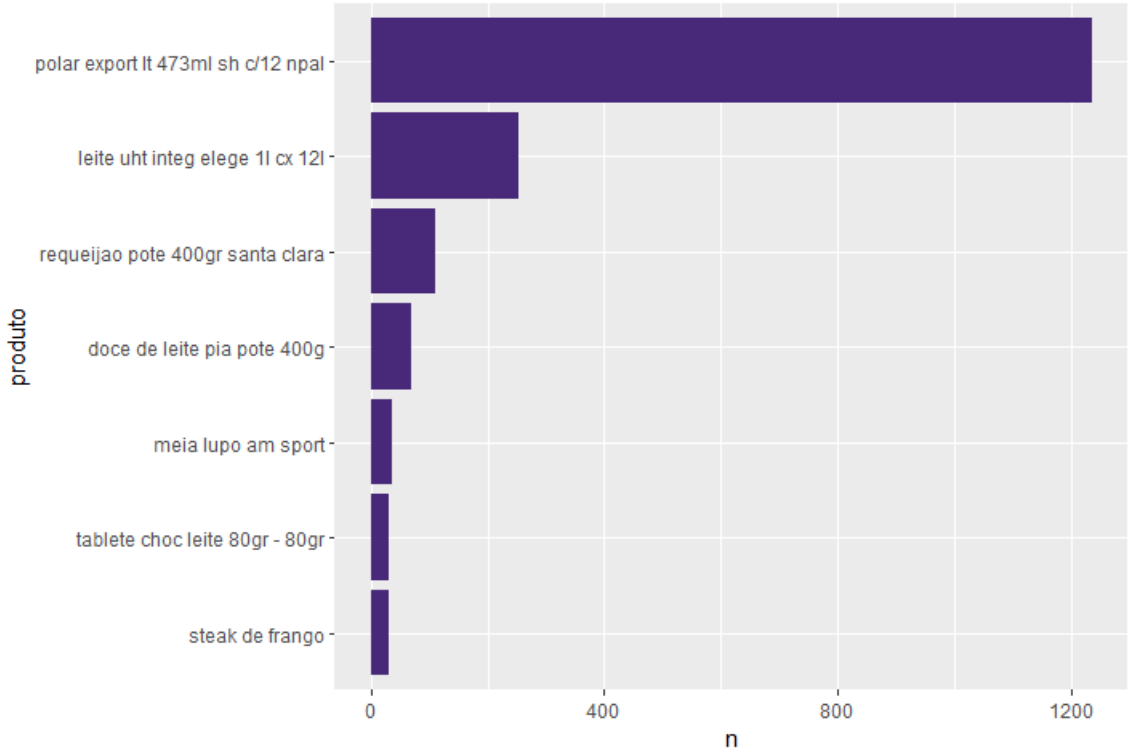
Figura 6.24: Número de observações por tópico do banco *bancao_1_por_cento* utilizando o Modelo 12 da Tabela (6.10).[46]



```
-1    147752
0      10316
1       4548
2       4017
3       3398
4       3204
5       2683
6       2492
7       2004
8       1801
9       1374
10      1354
11      1284
12      1237
13      1214
14      1192
15      1114
16      1057
17      1048
18       793
19       788
20       767
21       742
22       722
23       665
24       660
25       644
26       618
27       610
28       579
29       575
30       559
31       540
32       509
Name: Count, dtype: int64
```

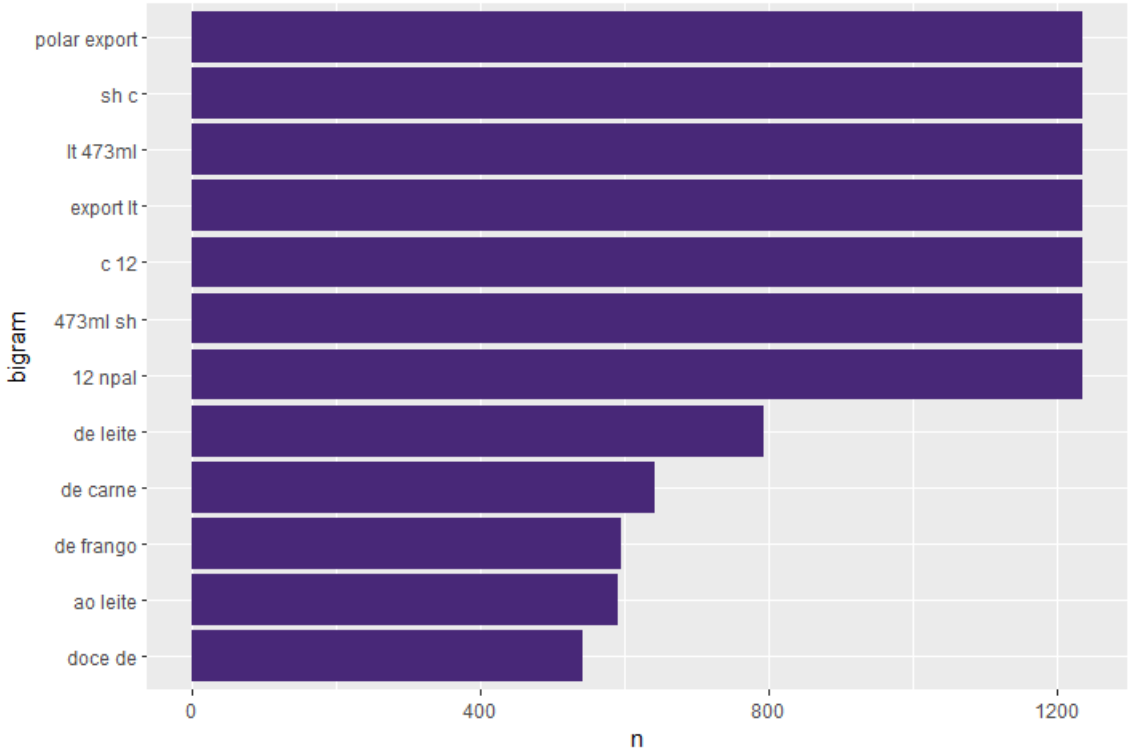
Fonte: O Autor

Figura 6.25: Descrições mais frequentes (maior de 30 aparições) dos tópicos do Modelo 12 utilizando o *bancao_1_por_cento*.[48]



Fonte: O Autor

Figura 6.26: *Bi-grams* com mais de 500 aparições dos tópicos do Modelo 12 da Tabela (6.10) utilizando o *bancao_1_por_cento*.[50]



Fonte: O Autor

Pela Figura (6.25), a cerveja da marca Polar lidera o gráfico, seguido por produtos relacionados a leite, roupas e frango. A Figura (6.26) mostra que as sequencias de duas palavras mais frequentes são da cerveja Polar, seguida de algumas palavras chave acompanhadas da preposição "de", um resultado interessante.

Cada componente do Modelo 12 será avaliado detalhadamente a seguir.

Tópico -1: Apresentado pela Figura (6.27), possui um total de 147752 observações dos mais variados tipos de produtos.

Figura 6.27: Características do tópico -1 do Modelo 12 utilizando o segundo banco (geral).[52]

```
Topic                                                              -1
Count                                                          147752
Name                                          -1_de_leite_po_integral
Representation          [de, leite, po, integral, frango, pote, carne,...
Representative_Docs     [leite uht zero lactose santa clara 1l, cortes...
Name: 0, dtype: object
Topico -1 - -1_de_leite_po_integral - Top 5 de cada topico:
['de', 'leite', 'po', 'integral', 'frango', 'pote', 'carne', 'cx', 'com', 'tipo']
```

Fonte: O Autor

Tópicos relacionados a "Leite": Alguns tópicos foram gerados para leite. Eles são mostrados na Figura (6.28), juntamente aos principais produtos (Figura

(6.29)) e *bi-grams* (Figura (6.30)) dos agrupamentos. Ao todo, os tópicos de leite possuem 4454 observações.

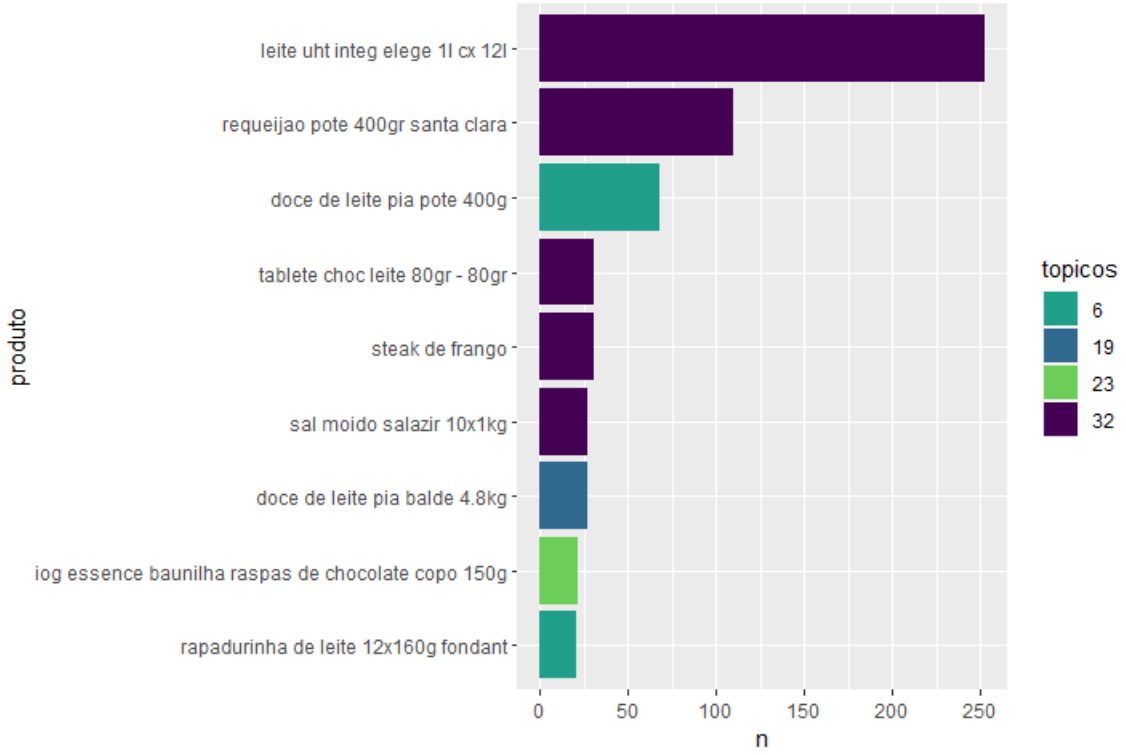
Figura 6.28: Características dos tópicos relacionados a leite do Modelo 12 da Tabela (6.10) utilizando o *bancao_1_por_cento*.[54]

```
Topic                                                                  6
Count                                                               2492
Name                                                       6_leite___
Representation                                  [leite, , , , , , , , , , ]
Representative_Docs    [doce de leite pia 400g, doce de leite 4.5kg, ...
Name: 7, dtype: object
Topico 6 - 6_leite___ - Top 5 de cada topico:
['leite', '', '', '', '', '', '', '', '', '']

Topic                                                                 19
Count                                                                788
Name                                                      19_leite___
Representation                                  [leite, , , , , , , , , , ]
Representative_Docs    [leite uht integral, leite uht integral       ...
Name: 20, dtype: object
Topico 19 - 19_leite___ - Top 5 de cada topico:
['leite', '', '', '', '', '', '', '', '', '']

Topic                                                                 23
Count                                                                665
Name                                                      23_leite___
Representation                                  [leite, , , , , , , , , , ]
Representative_Docs    [chocolate po 200g, chocolate em po 50% 5kg, c...
Name: 24, dtype: object
Topico 23 - 23_leite___ - Top 5 de cada topico:
['leite', '', '', '', '', '', '', '', '', '']


Topic                                                                 32
Count                                                                509
Name                                          32_integ_12l_elege_uht
Representation         [integ, 12l, elege, uht, 1l, 400gr, cx, requei...
Representative_Docs    [leite uht integ elege 1l cx 12l, leite uht in...
Name: 33, dtype: object
Topico 32 - 32_integ_12l_elege_uht - Top 5 de cada topico:
['integ', '12l', 'elege', 'uht', '1l', '400gr', 'cx', 'requeijao', 'santa', '80gr']
```

Fonte: O Autor

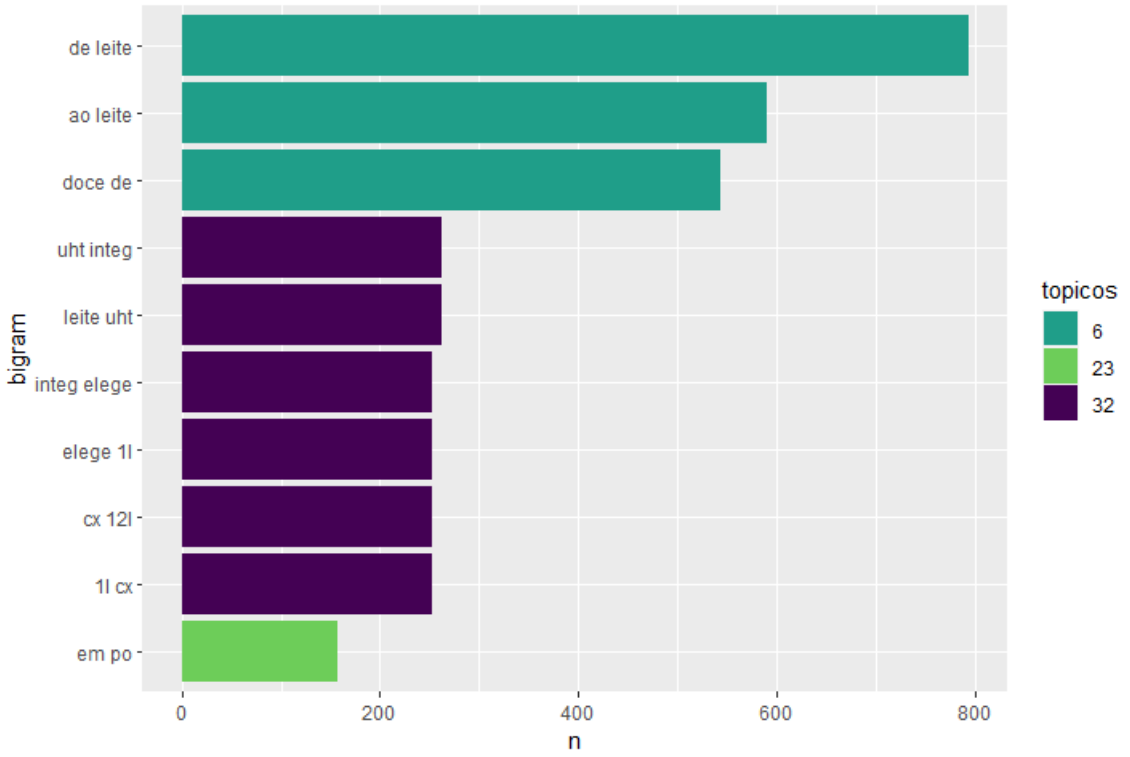
Observando a Figura (6.29), o tópico 32 parece ser composto por produtos mais gerais, como leites e derivados do leite, além de outros tipos de mercadorias. O grupo 6 aparenta possuir mais relação com doces de leite, enquanto que o agrupamento 19 com doces feitos de leite e o 23 com chocolates de leite. Na Figura (6.30), alguns dos *bi-grams* são mostrados, caracterizados por ingredientes (“de leite”) e características (“ao leite”, “doce de”) dos produtos.

Figura 6.29: Descrições mais frequentes (maior de 20 aparições) dos tópicos relacionados a leite do Modelo 12 da Tabela (6.10) utilizando o *bancao_1_por_cento*.[56]



Fonte: O Autor

Figura 6.30: *Bi-grams* com mais de 155 aparições dos tópicos relacionados a leite do Modelo 12 da Tabela (6.10) utilizando o banco não refinado.[58]



Fonte: O Autor

Algumas observações dos tópicos são listadas na Tabela (6.11) Os agrupamentos 23 e 32 possuem mais *outliers*, mas demostram compreender em sua grande maioria chocolates e produtos compostos de leite, respectivamente.

Tabela 6.11: Probabilidades associadas à diferentes descrições dos tópicos 6, 19, 23 e 32 do Modelo 12 utilizando o *bancao_1_por_cento*.

Produto	**Tópico agrupado**	**Probabilidade (%)**
“doce de leite para confeitaria bis 1.8 kg”	6	1
“choc neugebauer ao leite 14x90g”	6	0,171
“pao de mel c/cob. sabor choc. ao leite cx c/12 cxs”	6	0,517
“doce de leite conaprole 3 kg”	6	1
“creme leite ccgl tp”	19	1
“leite uht semi desnatado c/tampa tirol 1 lt”	19	1
“leite l.vida languiru integral cxa 1 x 12 1l”	19	0,344
“rf.creme leite nata pia”	19	0,836
“chocolate quente em po tradicional 250 gr”	23	1
“chocolate em po 1.05kg 50% harald”	23	0,171
“peito frango seara defumado 8338”	23	0,846
“saco lixo preto 060l c/100 - 1 leve (suporta 10kg)”	23	0,985
“leite uht integ elege 1l cx 12l”	32	1
“requeijao pote 400gr santa clara”	32	1
“steak de frango”	32	1
“sal moido salazir 10x1kg”	32	1

Tópicos relacionados a “Carnes” (Frango e Carne): Dos 32 tópicos gerados, apenas o tópico 1 gerou observações para carne. Enquanto isso, os grupos 15 e 17 agruparam produtos de frango. A Figura (6.31) exibe os agrupamentos. Ao todo, foram combinadas 6710 observações de carnes.

Figura 6.31: Características dos tópicos relacionados a carnes do Modelo 12 utilizando o *bancao_1_por_cento*.[60]

```
Topic                                                              1
Count                                                           4548
Name                                                1_carne_moida_bife_cx
Representation          [carne, moida, bife, cx, bov, de, cong, resf, ...
Representative_Docs       [carne moida, carne moida kg, carne moida de 1]
Name: 2, dtype: object
Topico 1 - 1_carne_moida_bife_cx - Top 5 de cada topico:
['carne', 'moida', 'bife', 'cx', 'bov', 'de', 'cong', 'resf', 'bovino', 'suina']

Topic                                                             15
Count                                                           1114
Name                                                  15_frango___
Representation                                 [frango, , , , , , , , , , ]
Representative_Docs     [cortes resf frango file de peito                ...
Name: 16, dtype: object
Topico 15 - 15_frango___ - Top 5 de cada topico:
['frango', '', '', '', '', '', '', '', '', '']
Topic                                                             17
Count                                                           1048
Name                                                  17_frango___
Representation                                 [frango, , , , , , , , , , ]
Representative_Docs     [frango cong.bom frango kg, linguica frango co...
Name: 18, dtype: object
Topico 17 - 17_frango___ - Top 5 de cada topico:
['frango', '', '', '', '', '', '', '', '', '']
```

Fonte: O Autor

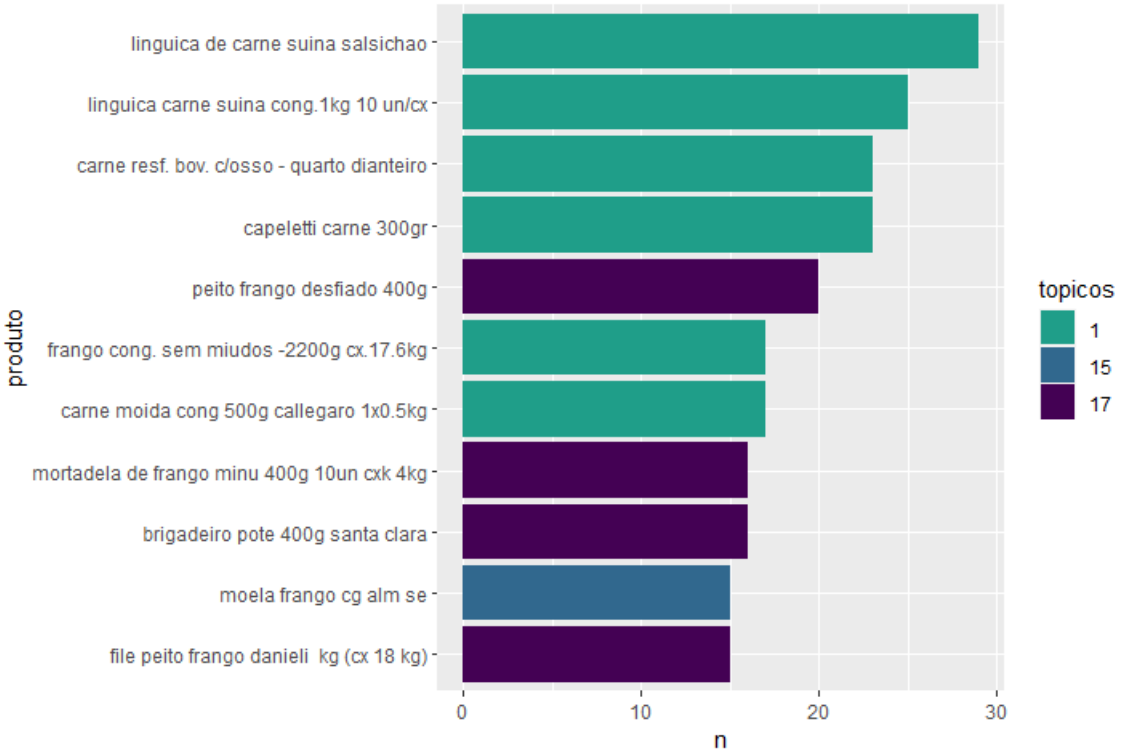
Tabela 6.12: Probabilidades associadas à diferentes descrições dos tópicos 1, 15 e 17 do Modelo 12 utilizando o *bancao_1_por_cento*.

Produto	**Tópico agrupado**	**Probabilidade (%)**
“linguica de carne suina 5kg”	1	1
“bifinho de carne 60 g”	1	1
“sal grosso kitano 1kg”	1	0,114
“figado bovino resf. (2)”	1	0,114
“coracao de frango.”	15	1
“empadinha frango cong.”	15	1
“c lanche croissant pct 10 un frango”	15	0,230
“folhado de frango 10 und/pct - 10 pct/cx”	15	0,233
“peito def frango 2.5kg excelsi”	17	1
“bag strogonoff de frango 1.050kg”	17	1
“whiskas frango 10.1kg”	17	0,110
“brigadeiro pote 400g santa clara”	17	0,095

Na Tabela (6.12), pode-se observar que o primeiro tópico agrupa observações de descrições associadas à produtos de carne de diversos tipos, o que é reforçado pela Figura (6.32). A diferença entre os grupos 15 e 17 parece estar na precisão. Avaliando em geral, o grupo 15 aparenta ser mais robusto, compreendendo somente

mercadorias relacionadas à frango, enquanto que o 17 possui a maior presença de *outliers*.

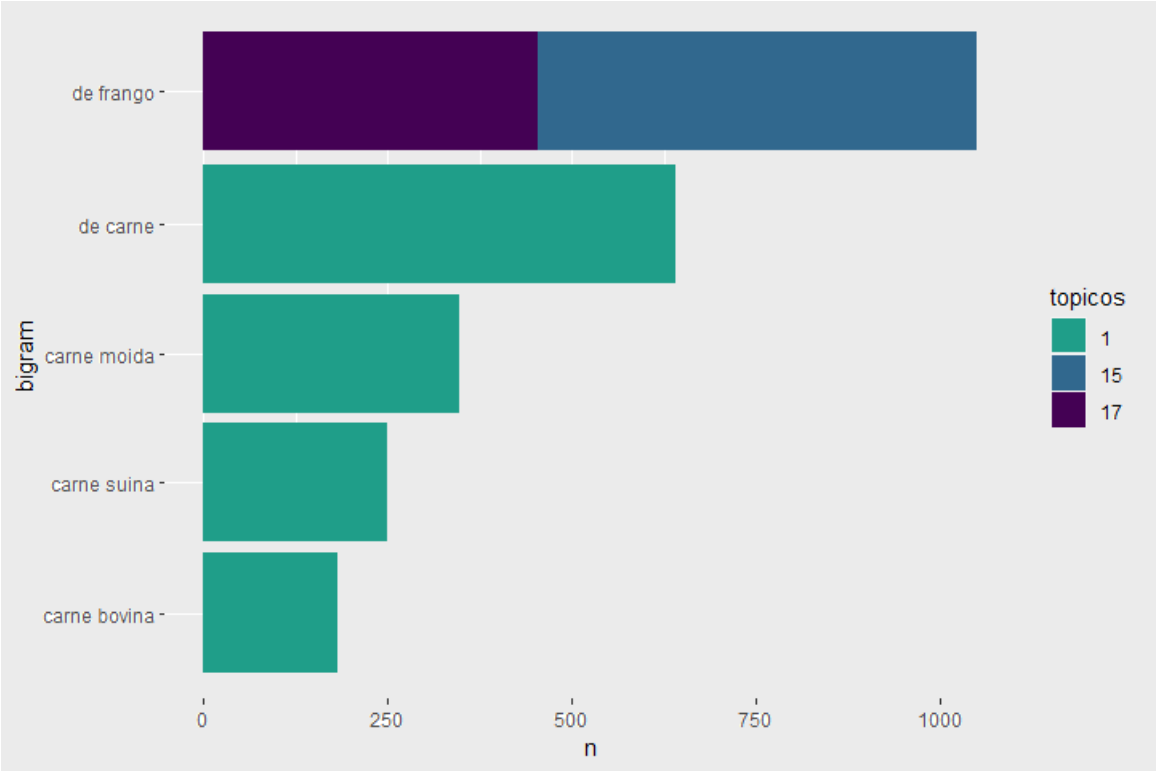
Figura 6.32: Descrições mais frequentes dos tópicos relacionados a carnes do Modelo 12 utilizando o *bancao_1_por_cento* e considerando um tamanho mínimo de 14.[62]



Fonte: O Autor

A última imagem do tópico (6.33) indica a divisão do *bi-gram* "de frango" para os grupos 15 e 17, enquanto que o tópico 1 possui *bi-grams* relacionados a carne.

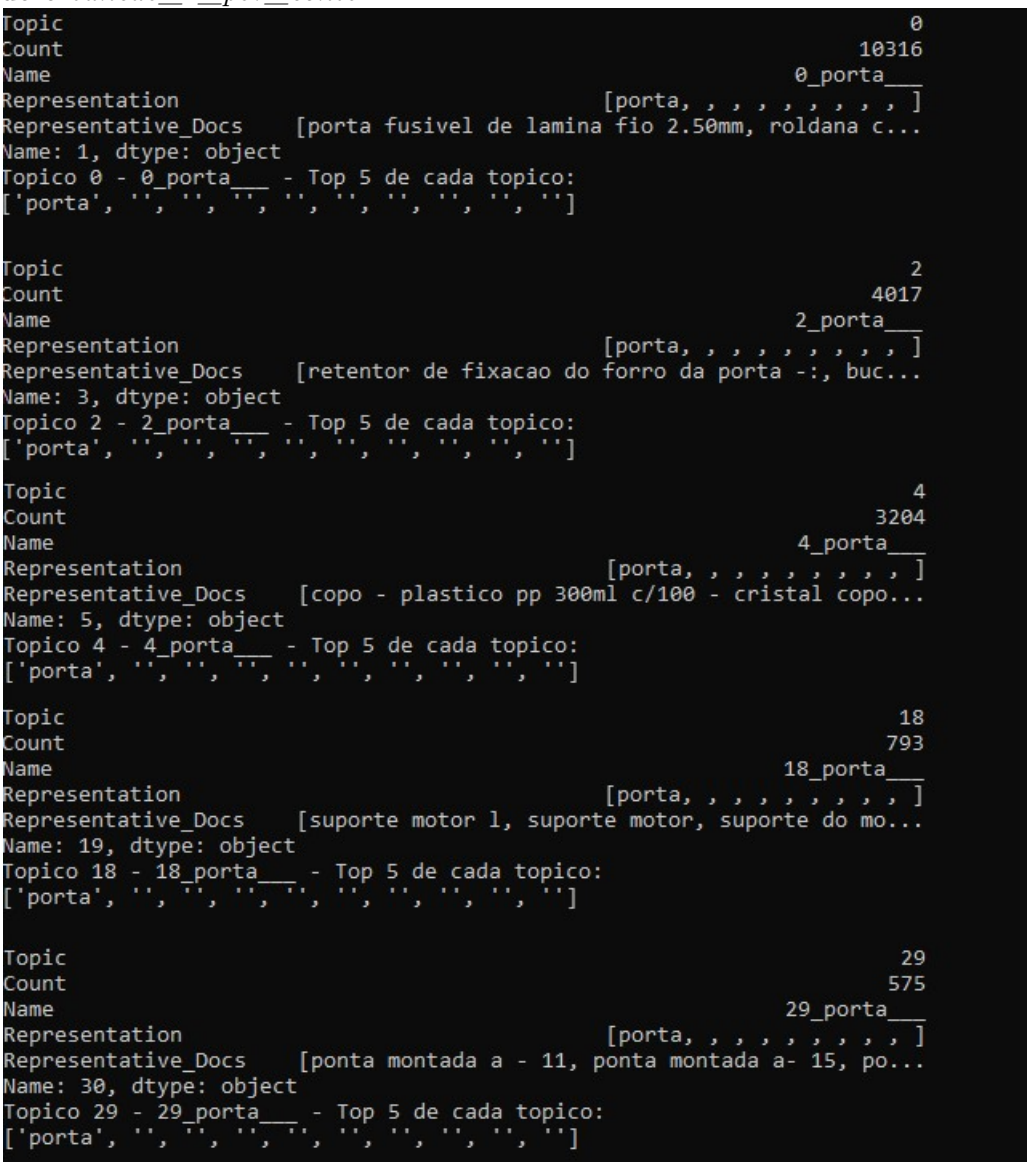
Figura 6.33: *Bi-grams* do tópicos relacionados a carnes do Modelo 12 utilizando o *bancao_1_por_cento* e considerando um tamanho mínimo de 180.[64]



Fonte: O Autor

Tópicos relacionados a "Porta": Observando alguns resultados dos outros Modelos da Tabela (6.10), foi pensado em utilizar como tentativa de *zero-shot learning* a palavra "porta", que acabou gerando o maior agrupamento nessa Seção. Eles somam ao todo 18905 observações. Na Figura (6.34) os tópicos parecem estar bem divididos, com peças, suporte, entre outras.

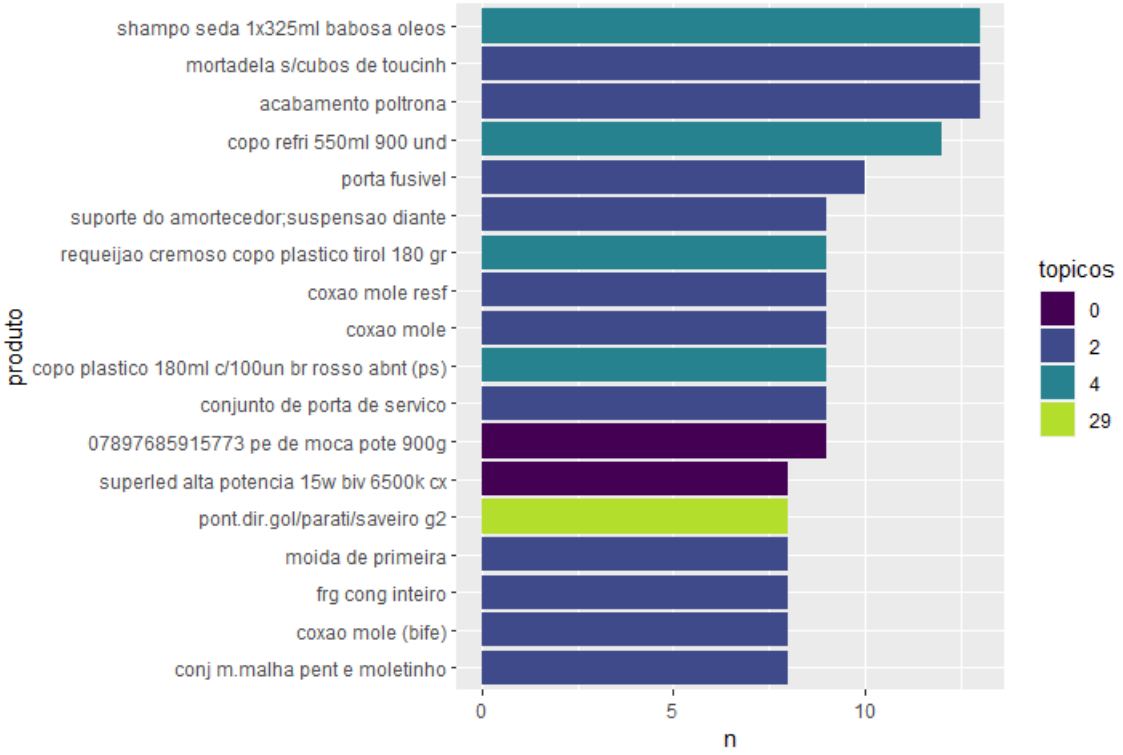
Figura 6.34: Características dos tópicos relacionados a portas do Modelo 12 utilizando o *bancao_1_por_cento*.[66]



Fonte: O Autor

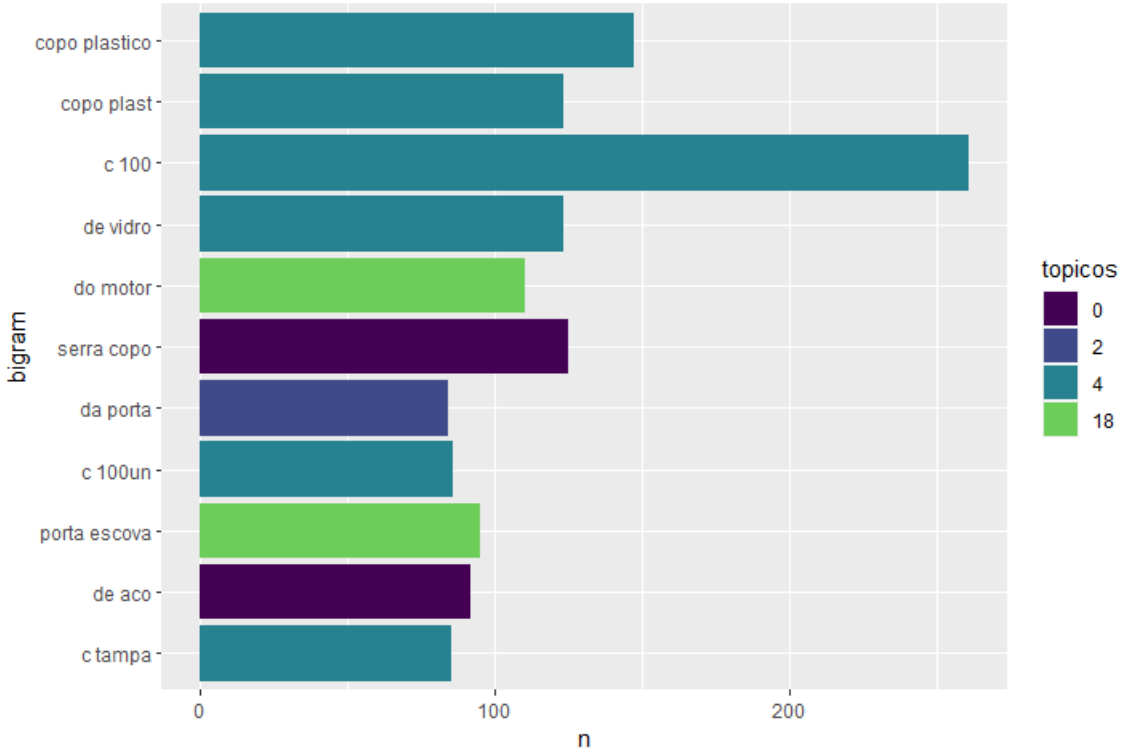
Porém, ao observar as Figuras (6.35) e (6.36) é possível observar que são tópicos muito gerais, sendo o grupo 4 mais relacionado a copos, o grupo 0 e o 18 à ferramentas e motores e o tópico 2 à portas, mesmo que com grande número de *outliers*.

Figura 6.35: Descrições mais frequentes dos tópicos relacionado a portas do Modelo 12 utilizando o *bancao_1_por_cento* e considerando um tamanho mínimo de 7.[68]



Fonte: O Autor

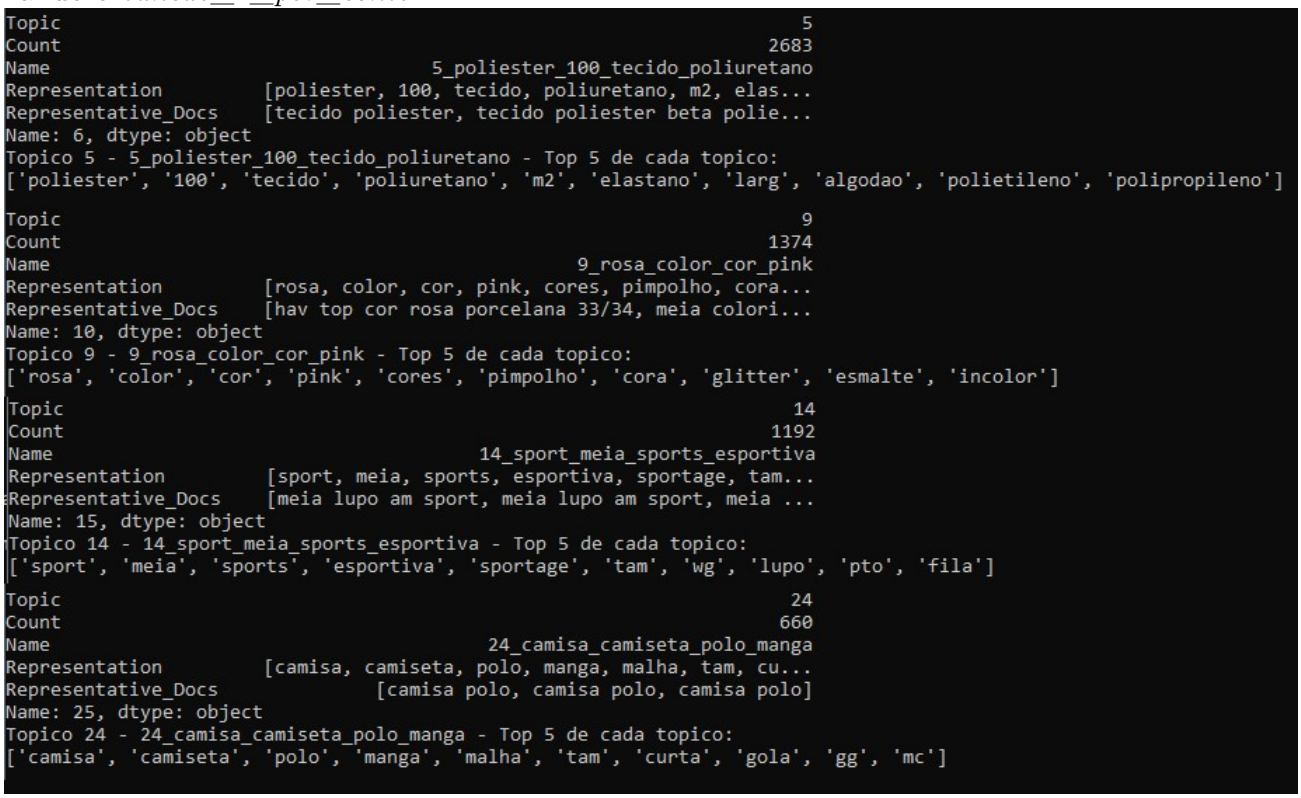
Figura 6.36: *Bi-grams* do tópicos relacionados a portas do Modelo 12 utilizando o *bancao_1_por_cento* e considerando um tamanho mínimo de 150.[70]



Fonte: O Autor

Tópicos relacionados a "Roupas": Três dos quatro tópicos agrupados fazem referência à roupas, como o 5 (poliester), o 14 (meias) e o 24 (camisetas). O agrupamento 9 se refere à cor rosa, sendo o mais diferente do agrupamento. As figuras (6.37), (6.38) e (6.39) ilustram o grupo. Somados, esse tópico compõe 5909 observações.

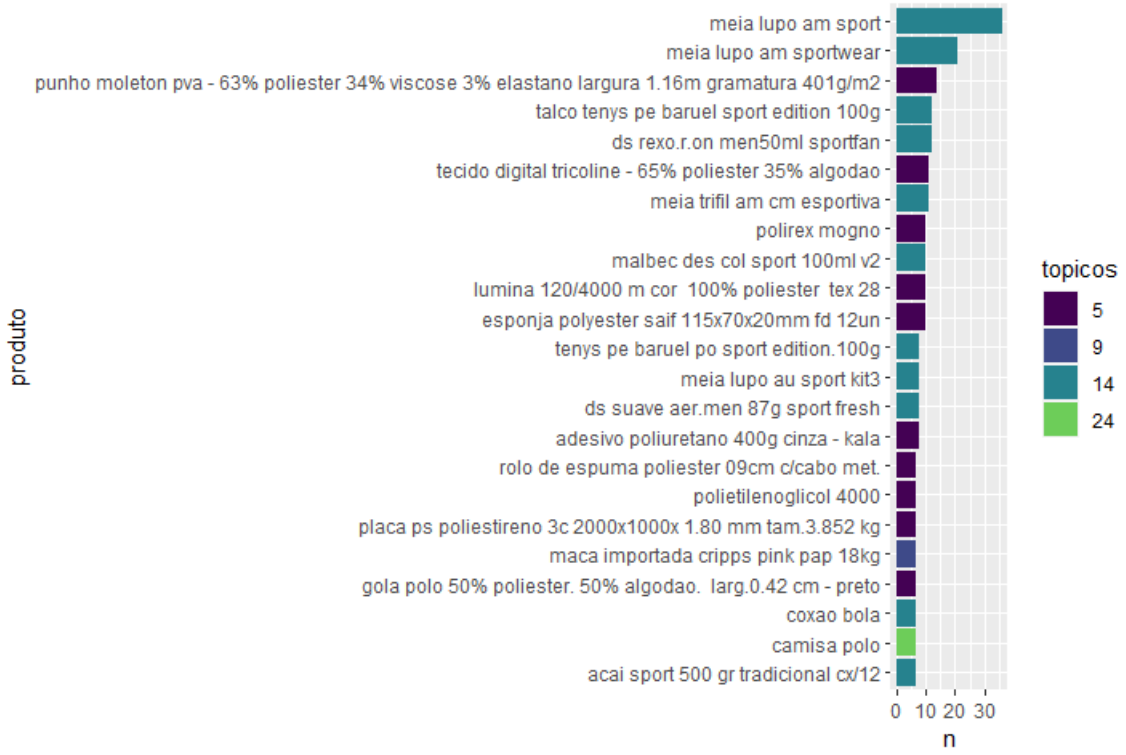
Figura 6.37: Características dos tópicos relacionados a roupas do Modelo 12 utilizando o *bancao_1_por_cento*.[72]



Fonte: O Autor

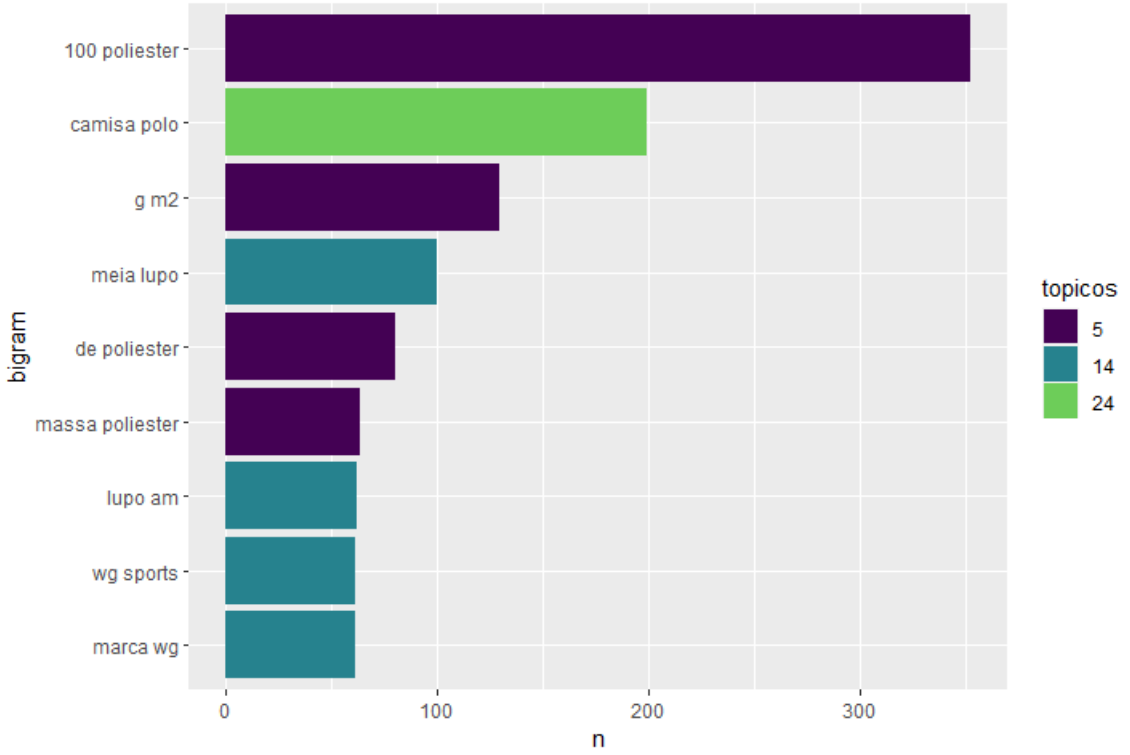
Observando as Figuras (6.38) e (6.39), é possível confirmar as ideias de tópicos discutidas acima. O tópico 5 faz alusão aos materiais das roupas, o 14 à meias e produtos para os pés, o 24 à camisetas e o 9 à cor rosa.

Figura 6.38: Descrições mais frequentes dos tópicos relacionado à roupas do Modelo 12 utilizando o *bancao_1_por_cento* e considerando um tamanho mínimo de 8.[74]



Fonte: O Autor

Figura 6.39: *Bi-grams* do tópicos relacionados a roupas do Modelo 12 utilizando o *bancao_1_por_cento* e considerando um tamanho mínimo de 50.[76]



Fonte: O Autor

Tópicos relacionados a "Comidas": Diversos tópicos relacionados à comida foram encontrados. Eles são compostos pelos grupos 11 (comidas congeladas), 12 (cerveja polar), 25 (cafés), 26 (arroz) e 27 (pastéis), que somam ao total 4393 mercadorias. Para maiores detalhes dos tópicos gerados, a Figura (6.40) exibe algumas das palavras chave.

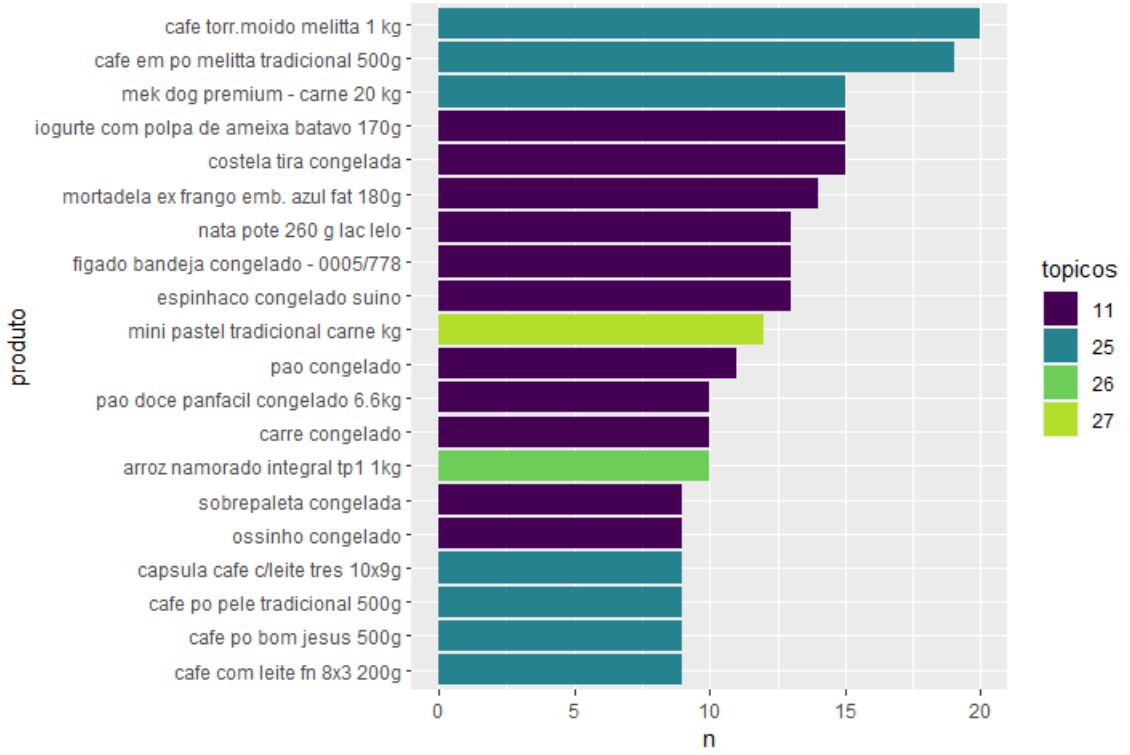
Figura 6.40: Características dos tópicos relacionados a comidas do Modelo 12 utilizando o *bancao_1_por_cento*.[78]

```
Topic                                                          11
Count                                                        1284
Name                                          11_congelado_pao_gel_gelo
Representation          [congelado, pao, gel, gelo, panfacil, refriger...
Representative_Docs          [pao congelado, pao congelado, pao congelado]
Name: 12, dtype: object
Topico 11 - 11_congelado_pao_gel_gelo - Top 5 de cada topico:
['congelado', 'pao', 'gel', 'gelo', 'panfacil', 'refrigerador', 'sorvete', 'domest', 'congelada', 'cl']
Topic                                                          12
Count                                                        1237
Name                                             12_npal_473ml_sh_lt
Representation          [npal, 473ml, sh, lt, export, polar, 12, depi,...
Representative_Docs     [polar export lt 473ml sh c/12 npal, polar exp...
Name: 13, dtype: object
Topico 12 - 12_npal_473ml_sh_lt - Top 5 de cada topico:
['npal', '473ml', 'sh', 'lt', 'export', 'polar', '12', 'depi', 'cv', '16un']

Topic                                                          25
Count                                                         644
Name                                        25_cafe_melitta_moido_500g
Representation          [cafe, melitta, moido, 500g, torr, jesus, trad...
Representative_Docs     [cafe po melitta 500g, cafe em po 500g, cafe c...
Name: 26, dtype: object
Topico 25 - 25_cafe_melitta_moido_500g - Top 5 de cada topico:
['cafe', 'melitta', 'moido', '500g', 'torr', 'jesus', 'tradicional', 'umidas', '2794', 'eletricas']


Topic                                                          26
Count                                                         618
Name                                          26_arroz_t1_lf_polido
Representation          [arroz, t1, lf, polido, tipo, parboilizado, 6x...
Representative_Docs       [arroz tipo 2 5 kg, arroz tipo 1, arroz tipo 1]
Name: 27, dtype: object
Topico 26 - 26_arroz_t1_lf_polido - Top 5 de cada topico:
['arroz', 't1', 'lf', 'polido', 'tipo', 'parboilizado', '6x5', 'benef', 'fino', '30x1']
Topic                                                          27
Count                                                         610
Name                                        27_pastel_forno_mini_torta
Representation          [pastel, forno, mini, torta, carne, bread, de,...
Representative_Docs     [mini pastel carne, mini pastel carne, mini pa...
Name: 28, dtype: object
Topico 27 - 27_pastel_forno_mini_torta - Top 5 de cada topico:
['pastel', 'forno', 'mini', 'torta', 'carne', 'bread', 'de', 'king', 'cordeiro', '2kg']
```

Fonte: O Autor

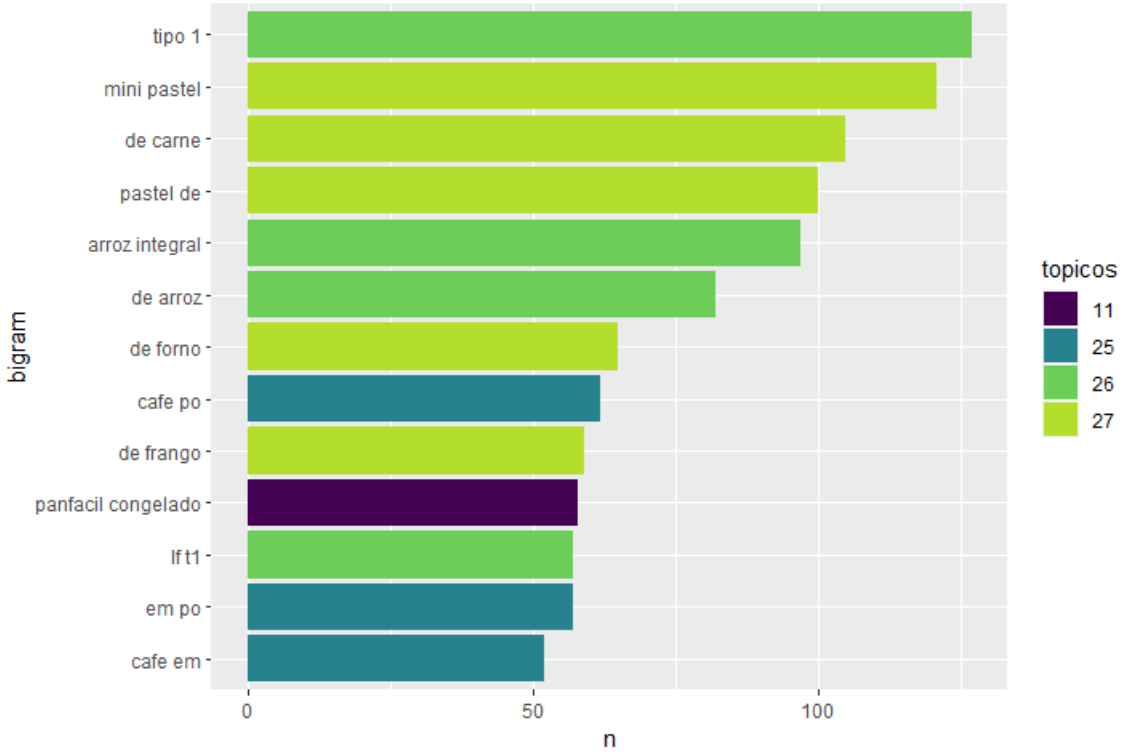
Nas Figuras (6.41) e (6.42) os tópicos parecem se dividir de maneira correta, com cafés, produtos congelados, pastéis e arroz. É importante ressaltar que o tópico 12 não foi inserido nas duas figuras, uma vez que ele possui observações rotuladas "polar export lt 473ml sh c/12 npal" em quase todo o seu conjunto de observações e dificulta a visualização dos demais produtos.

Figura 6.41: Descrições mais frequentes dos tópicos relacionado a comidas do Modelo 12 utilizando o *bancao_1_por_cento* e considerando um tamanho mínimo de 8.[80]



Fonte: O Autor

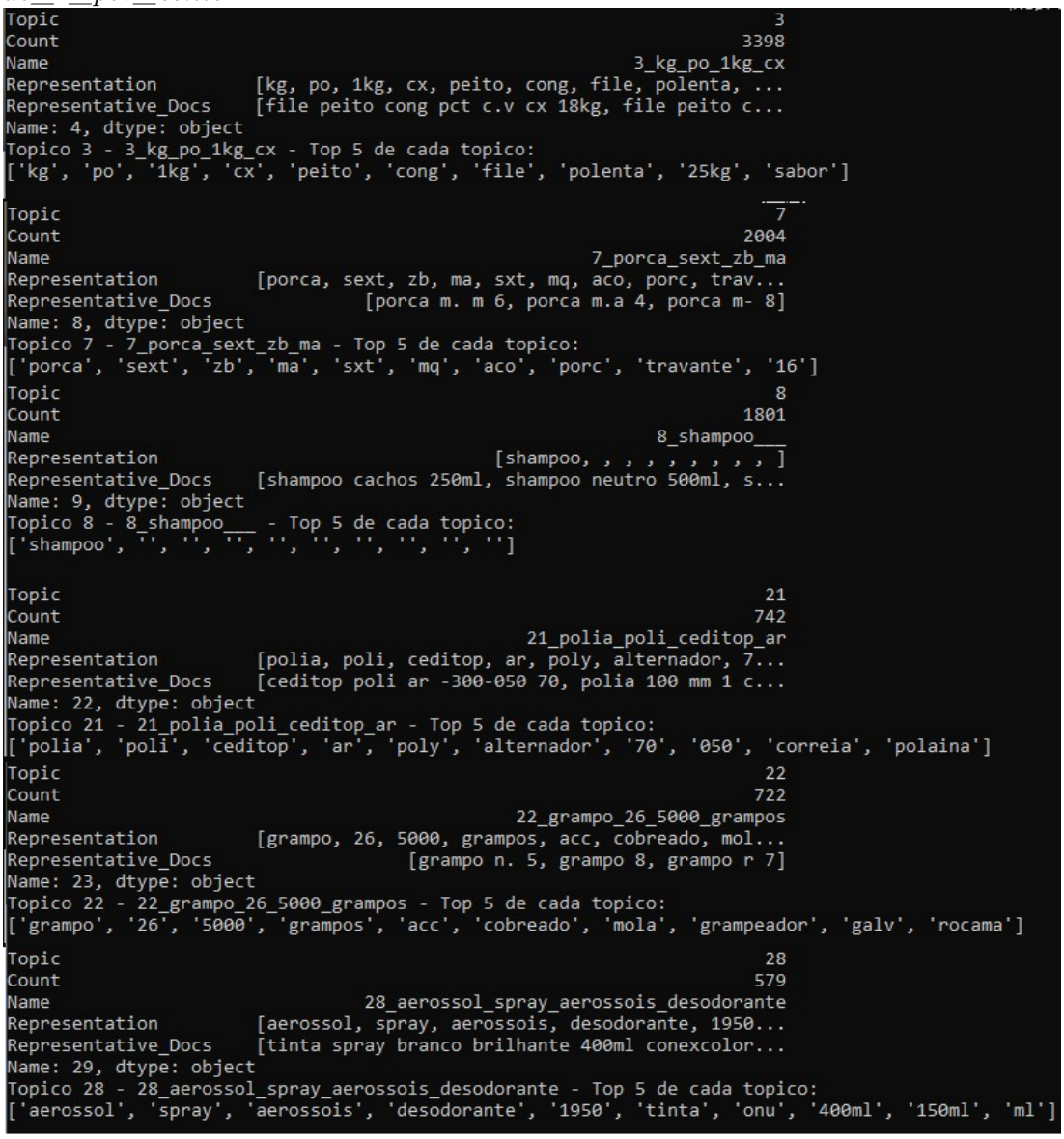
Figura 6.42: *Bi-grams* do tópicos relacionados a comidas do Modelo 12 utilizando o *bancao_1_por_cento* e considerando um tamanho mínimo de 50.[82]



Fonte: O Autor

Tópicos "Gerais" 1: Diversos tópicos não se encaixam em nenhum caso. Dessa forma, eles serão apresentados em conjunto. Os agrupamentos desse item são os grupos 3 (pó e carne), 7 (porcas), 8 (shampoos), 21 (polias), 22 (grampos), e 28 (aerosol e sprays). Na Figura (6.43) é possível observar algumas informações desse agrupamento, que soma 9246 observações.

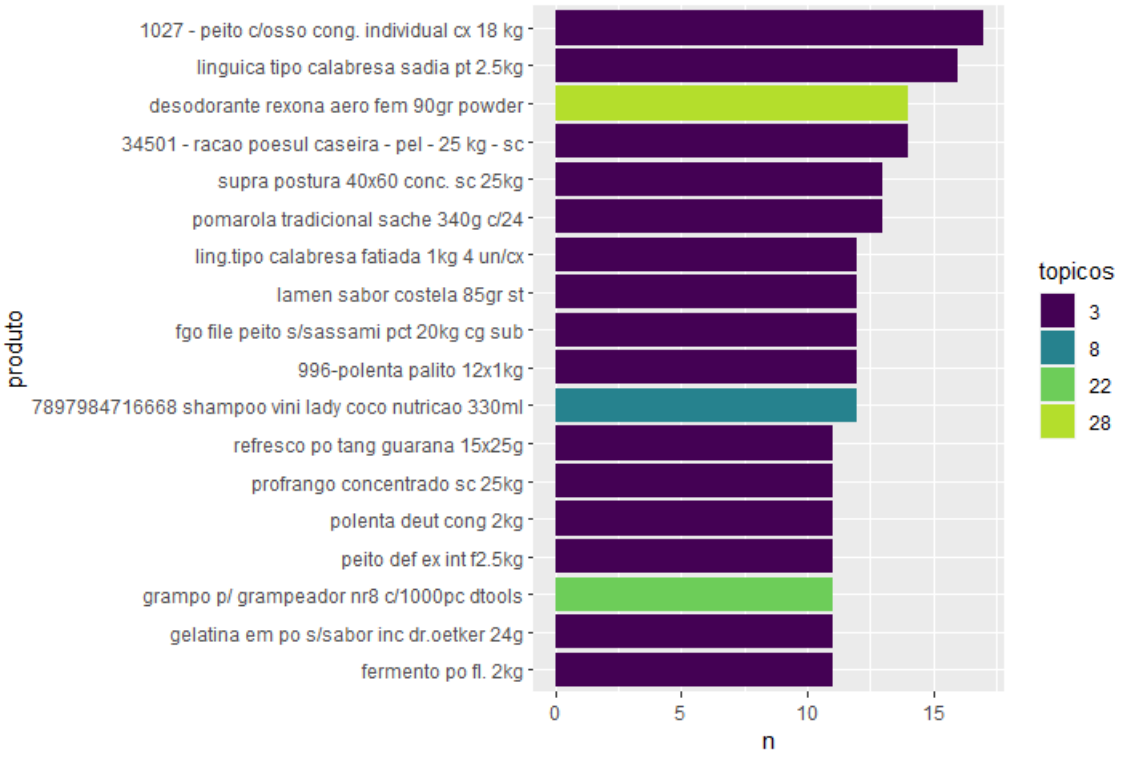
Figura 6.43: Características dos tópicos gerais 1 do Modelo 12 utilizando o *bancao__1__por__cento*.[84]



Fonte: O Autor

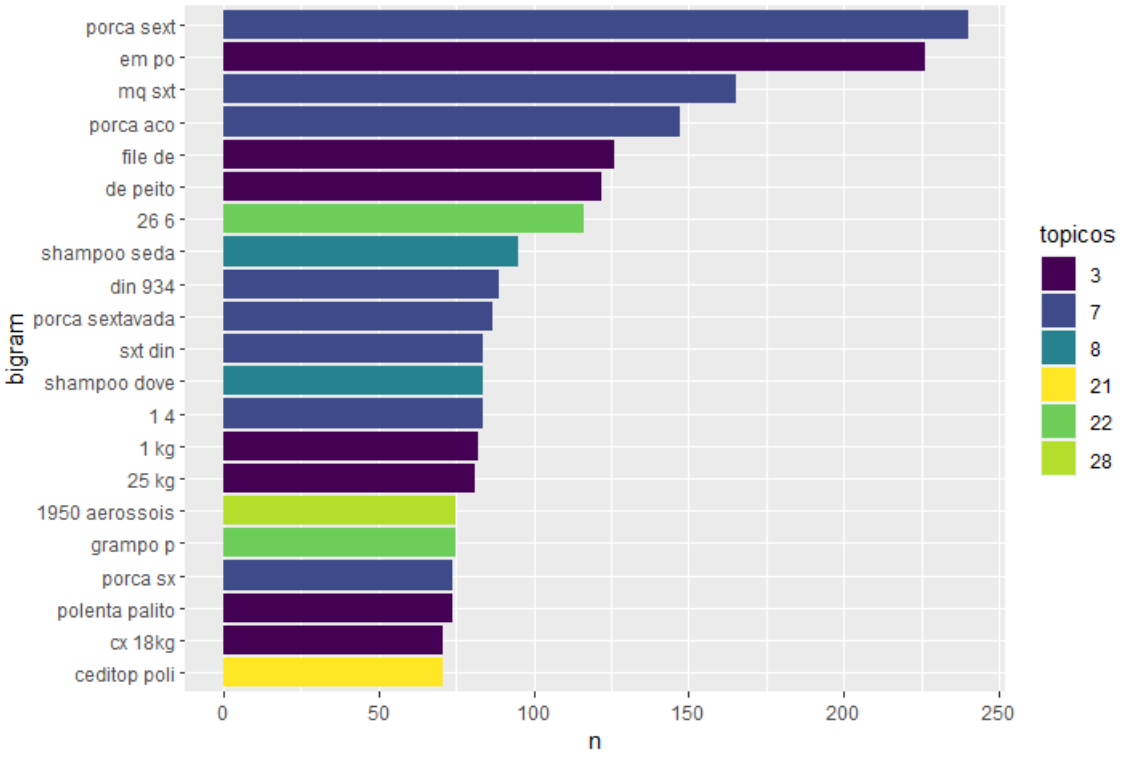
Observando a Figura (6.44), os tópicos 3, 8, 22 e 28 são bem representados em seus produtos. A Figura (6.45) confirma ainda mais os temas de cada agrupamento, mostrando *bi-grams* de todos os grupos, em especial o 7 e o 21 com seus produtos de porcas e polias.

Figura 6.44: Descrições mais frequentes dos tópicos gerais 1 do Modelo 12 da Tabela (6.10) utilizando o *bancao_1_por_cento* e considerando um tamanho mínimo de 10.[86]



Fonte: O Autor

Figura 6.45: *Bi-grams* do tópicos gerais 1 do Modelo 12 da Tabela (6.10) utilizando o *bancao_1_por_cento* e considerando um tamanho mínimo de 70.[88]



Fonte: O Autor

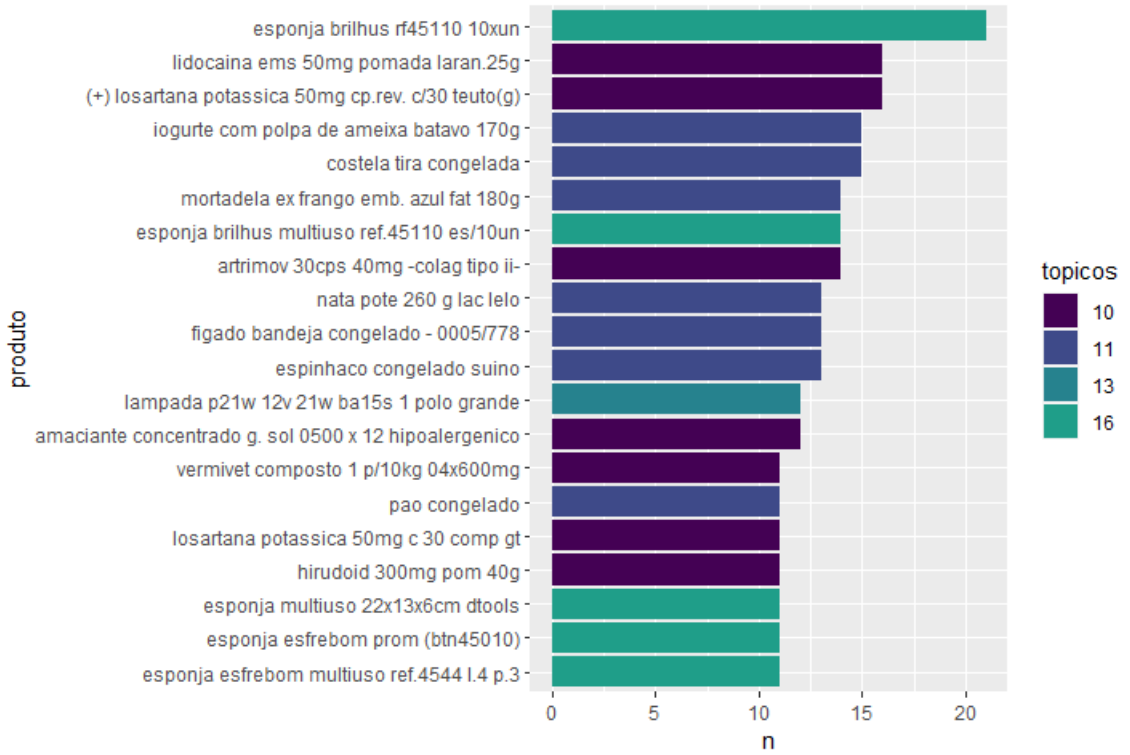
Tópicos "Gerais" 2: Novamente, de maneira semelhante ao tópico "Gerais" 1, os agrupamentos que compõem este item não são similares o suficiente entre si para se tornar um grande grupo de um determinado assunto. Observando as Figuras (6.46), (6.47) e (6.48), pode-se concluir que os tópicos 10, 11, 13, 16, 20, 30 e 31 são agrupamentos de produtos de remédios, produtos congelados, lâmpadas, esponjas, porcelanas, flores e produtos de inox, respectivamente, com 6775 itens ao total.

Figura 6.46: Características dos tópicos gerais 2 do Modelo 12 da Tabela (6.10) utilizando o *bancao_1_por_cento*.[90]

```
Topic                                                          10
Count                                                        1354
Name                                  10_50mg_losartana_gen_hipoclorito
Representation         [50mg, losartana, gen, hipoclorito, 30, sodio,...
Representative_Docs    [g. losartana potassica 50mg c/30 cp (+) teuto...
Name: 11, dtype: object
Topico 10 - 10_50mg_losartana_gen_hipoclorito - Top 5 de cada topico:
['50mg', 'losartana', 'gen', 'hipoclorito', '30', 'sodio', 'hipolabor', '5mg', 'generico', 'mg']


Topic                                                          11
Count                                                        1284
Name                                     11_congelado_pao_gel_gelo
Representation         [congelado, pao, gel, gelo, panfacil, refriger...
Representative_Docs        [pao congelado, pao congelado, pao congelado]
Name: 12, dtype: object
Topico 11 - 11_congelado_pao_gel_gelo - Top 5 de cada topico:
['congelado', 'pao', 'gel', 'gelo', 'panfacil', 'refrigerador', 'sorvete', 'domest', 'congelada', 'cl']
Topic                                                          13
Count                                                        1214
Name                                       13_lampada_led_12v_polo
Representation         [lampada, led, 12v, polo, 24v, lamp, lanterna,...
Representative_Docs    [lampada 1 polo, lampada 2 polo, lampada 1 pol...
Name: 14, dtype: object
Topico 13 - 13_lampada_led_12v_polo - Top 5 de cada topico:
['lampada', 'led', '12v', 'polo', '24v', 'lamp', 'lanterna', '21w', '5w', '6500k']


Topic                                                          16
Count                                                        1057
Name                                16_esponja_esfrebom_multiuso_brilhus
Representation         [esponja, esfrebom, multiuso, brilhus, banho, ...
Representative_Docs                          [esponja, esponja, esponja]
Name: 17, dtype: object
Topico 16 - 16_esponja_esfrebom_multiuso_brilhus - Top 5 de cada topico:
['esponja', 'esfrebom', 'multiuso', 'brilhus', 'banho', 'uso', 'aco', '1un', 'bombril', 'limpa']
Topic                                                          20
Count                                                         767
Name                             20_porcelana_porcelanato_caneca_grafite
Representation         [porcelana, porcelanato, caneca, grafite, cera...
Representative_Docs    [caneca em porcelana, caneca em porcelana, can...
Name: 21, dtype: object
Topico 20 - 20_porcelana_porcelanato_caneca_grafite - Top 5 de cada topico:
['porcelana', 'porcelanato', 'caneca', 'grafite', 'ceramica', 'esmaltado', 'e27', 'plafon', 'cm', 'branco']
Topic                                                          30
Count                                                         559
Name                                    30_flor_floral_flores_beija
Representation         [flor, floral, flores, beija, pipoca, campo, a...
Representative_Docs    [pipoca beija flor 100g, pipoca beija flor 100...
Name: 31, dtype: object
Topico 30 - 30_flor_floral_flores_beija - Top 5 de cada topico:
['flor', 'floral', 'flores', 'beija', 'pipoca', 'campo', 'ajax', 'do', 'limp', 'saponaceo']


Topic                                                          31
Count                                                         540
Name                                       31_inox_304_ri_aco
Representation         [inox, 304, ri, aco, pol, ma, sx, pa, polido, ...
Representative_Docs    [esponja de aco inox, esponja de aco inox, cop...
Name: 32, dtype: object
Topico 31 - 31_inox_304_ri_aco - Top 5 de cada topico:
['inox', '304', 'ri', 'aco', 'pol', 'ma', 'sx', 'pa', 'polido', 'esponja']
```

Fonte: O Autor

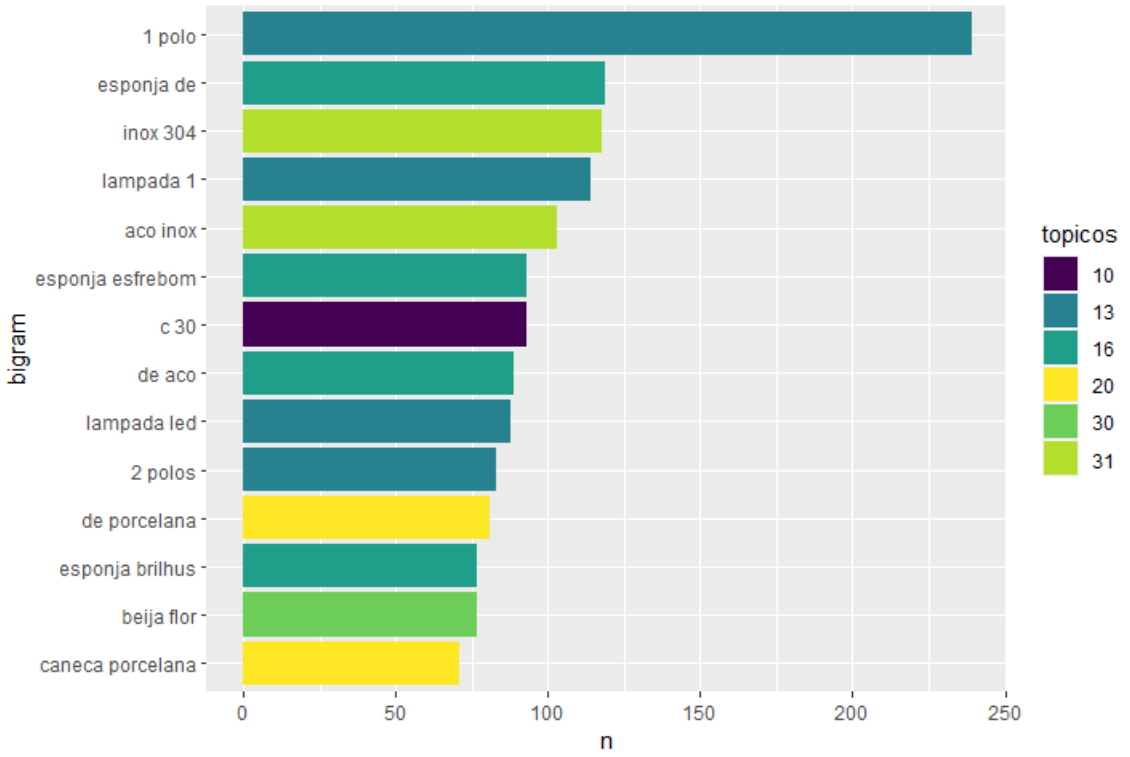
Figura 6.47: Descrições mais frequentes dos tópicos gerais 2 do Modelo 12 da Tabela (6.10) utilizando o *bancao_1_por_cento* e considerando um tamanho mínimo de 10.[92]



Fonte: O Autor

As Figuras (6.47) e (6.48) exibem alguns produtos, além dos *Bi-grams* mais frequentes dos agrupamentos.

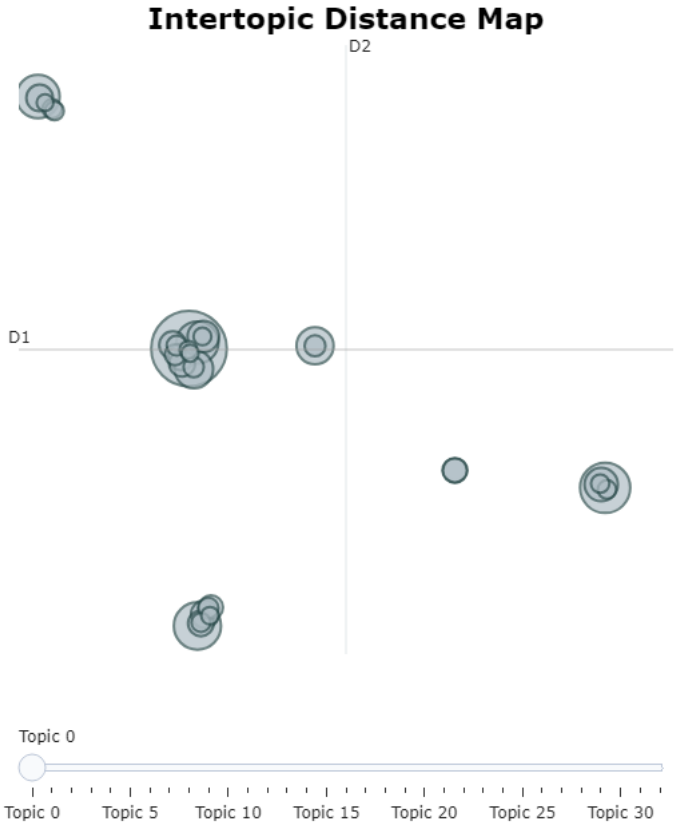
Figura 6.48: *Bi-grams* do tópicos gerais 2 do Modelo 12 da Tabela (6.10) utilizando o *bancao_1_por_cento* e considerando um tamanho mínimo de 70.[94]



Fonte: O Autor

Abaixo, algumas características gerais dos tópicos formados pelo Modelo 12 da Tabela (6.10) utilizando o *bancao_1_por_cento* serão mostradas.

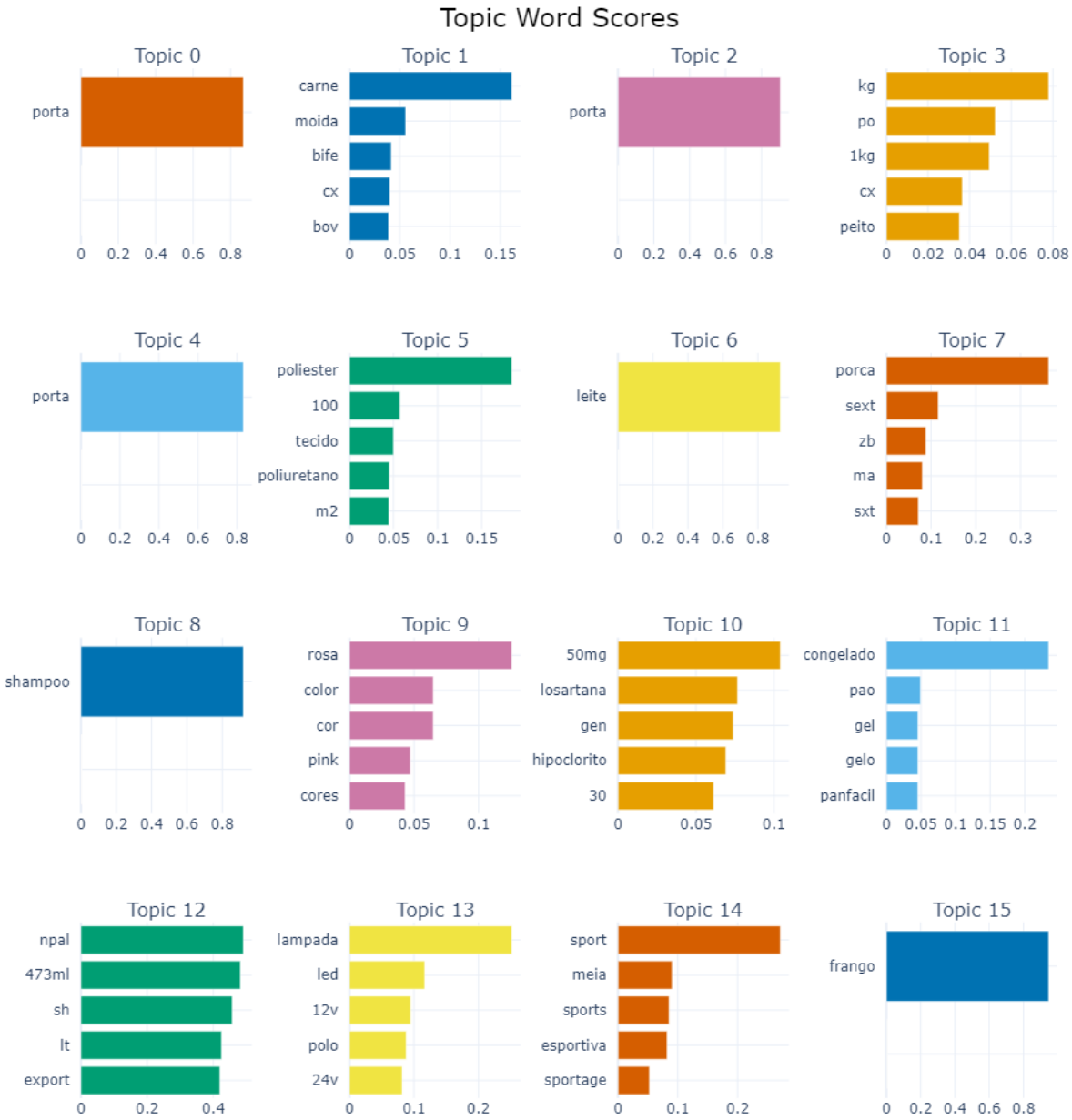
Figura 6.49: Figura das características dos tópicos diversos do Modelo 12 da Tabela (6.10) utilizando o *bancao_1_por_cento*.[96]



Fonte: O Autor

Na Figura (6.49), um comportamento mais randômico pode ser observado. Alguns como os agrupamentos no canto superior esquerdo, na região central e no canto inferior direito parecem ser mais coerentes. O primeiro é composto por grupos como o 3 (pós e carnes), 11 (produtos congelados), 23 (chocolates de leite), 25 (cafés) e 32 (produtos gerais de leite), enquanto que o segundo é formado pelos tópicos 6 (doces de leite) e 19 (doces feitos de leite), e, finalmente, o terceiro é integrado pelos agrupamentos 1 (carnes gerais), 7 (porcas), 26 (arroz) e 27 (pastéis).

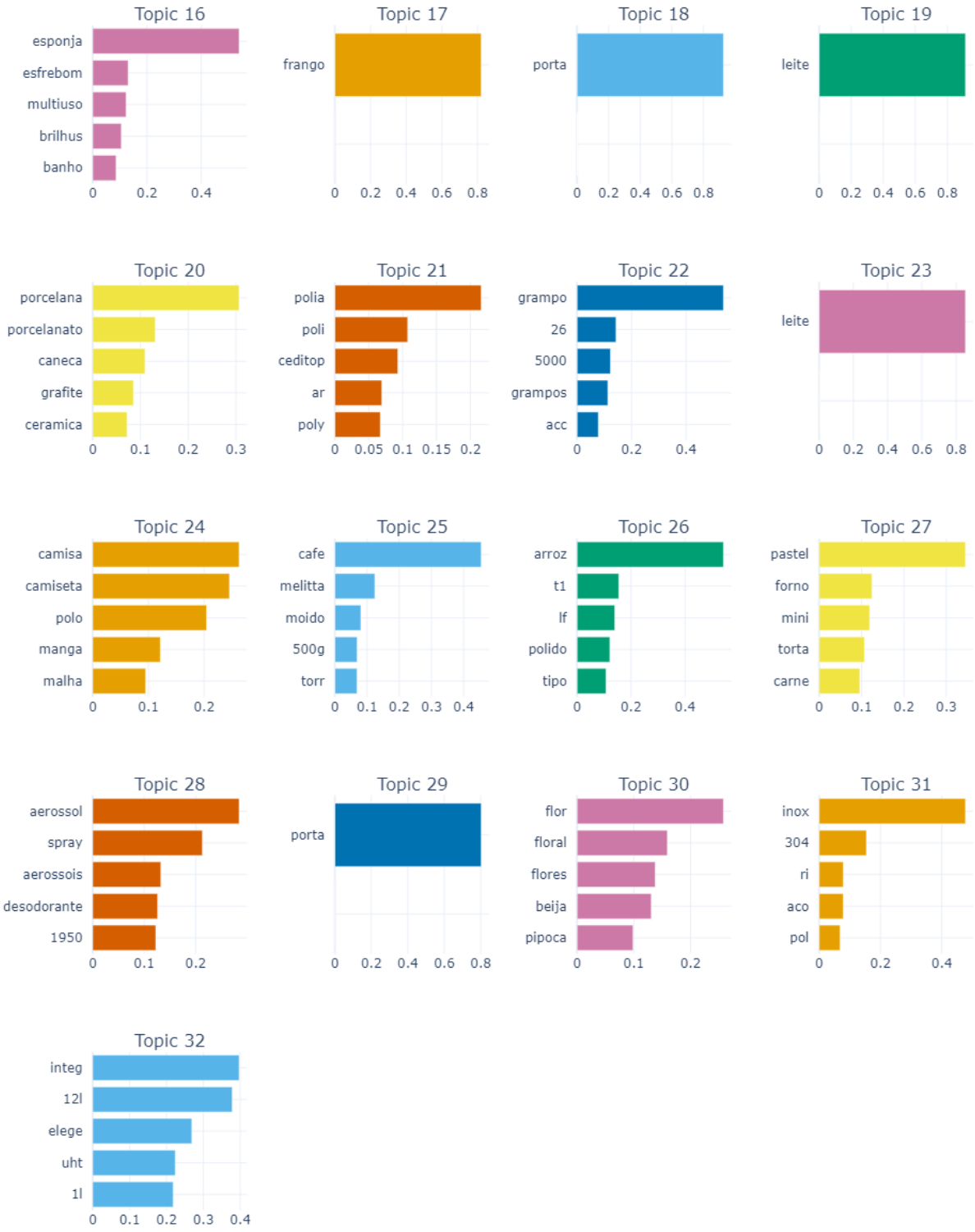
Figura 6.50: Maiores escores de palavras que compõem os tópicos do Modelo 12 utilizando o *bancao_1_por_cento*.[98]



Fonte: O Autor

Nas Figuras (6.50) e (6.51), todos os tópicos obtidos a partir do *"zero-shot learning"* obtiveram escores maiores que 0,80. Enquanto isso, somente os agrupamentos 16, 22 e 26 obtiveram escores maiores que 0,50, mostrando que os agrupamentos não estão muito específicos.

Figura 6.51: Matriz de similaridade dos tópicos do Modelo 12 utilizando o *bancao_1_por_cento*.[100]

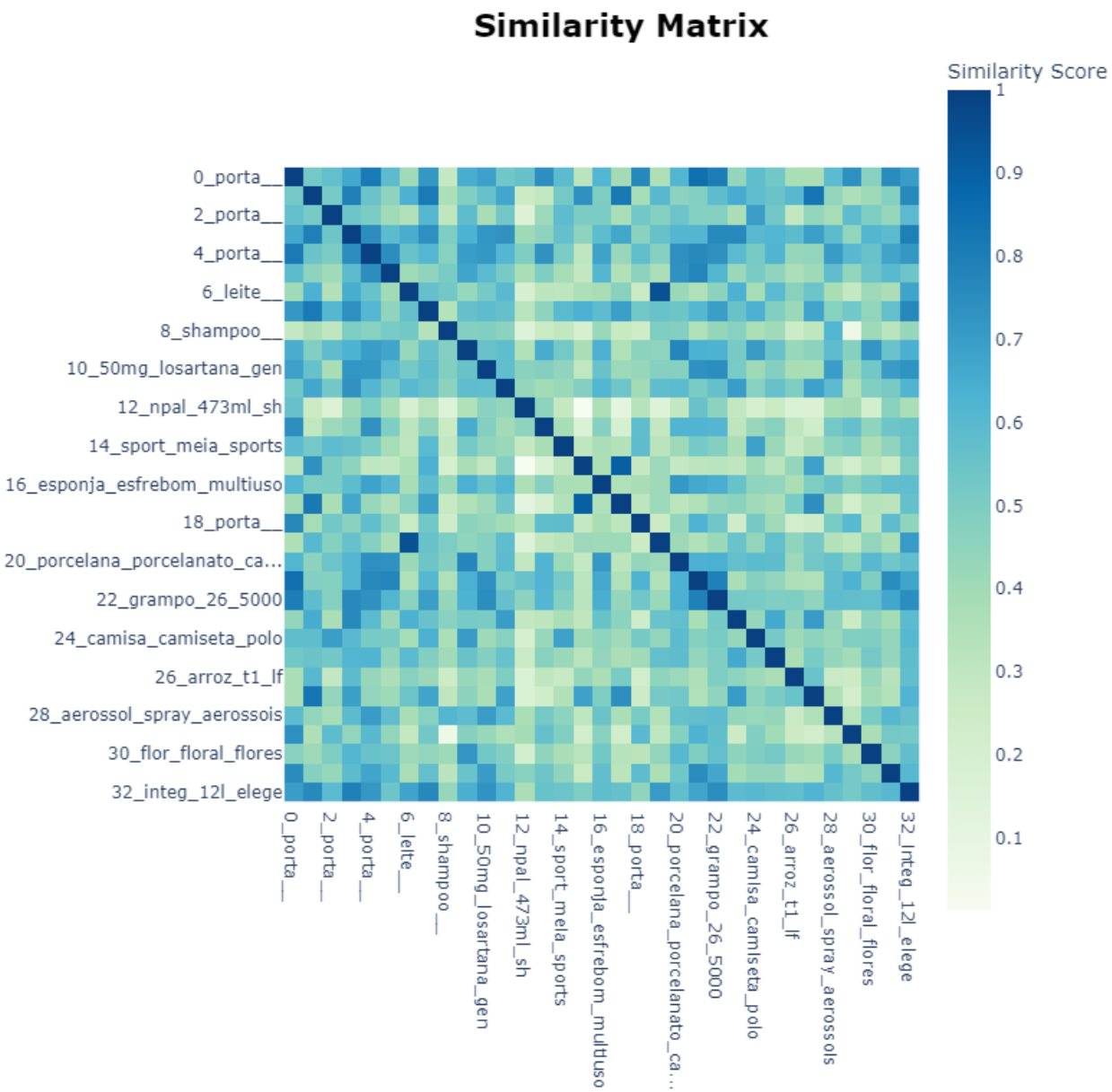


Fonte: O Autor

Por fim, a Figura (6.52) mostra a matriz de similaridade, que indica um compor-

tamento mais aleatório. Existem algumas exceções, como, por exemplo, os grupos 6 (doces de leite) e 19 (doces feitos de leite), que possuem uma similaridade muito alta (0,94); e os grupos 1 (carne) e 27 (pastel), que possuem uma similaridade de 0,83.

Figura 6.52: Figura das características dos tópicos diversos do Modelo 12 da Tabela (6.10) utilizando o *bancao_1_por_cento*.[102]



Fonte: O Autor

Uma diferença entre os resultado do primeiro corpus (banco_leite_carne_5_por_cento) e o do segundo (bancao_1_por_cento) é que os valores dos tópicos e *bi-grams* diminuíram na utilização do segundo banco. Por exemplo, na Figura (6.10) foi utilizado um tamanho mínimo de 1200, enquanto que na Figura (6.29) é utilizado o valor de

20. Outro exemplo são as Figuras (6.50) e (6.51), que resultaram em escores muito menores para muitos grupos, em oposição à Figura (6.21). Isso indica uma menor certeza para os agrupamentos do segundo corpus, além da presença de mais *outliers*, que é um resultado esperado, uma vez que os dados não passaram por nenhum afinamento de conteúdo.

7 Conclusão

Utilizando dois corpus obtidos da Secretaria da Fazenda do Rio Grande do Sul (SEFAZ-RS) compostos de Notas Fiscais Eletrônicas, modelos de BERTopic foram aplicados, variando diversos hiperparâmetros e obtendo múltiplos resultados. Modelos de representação como o BART, Mistral Zephyr 7B e KeyBERTInspired foram empregados, juntamente aos modelos de *embedding "paraphrase-multilingual-MiniLM-L12-v2"* e *"all-MiniLM-L6-v2"*, buscando uma melhora nos resultados. Esses resultados foram avaliados a partir da métrica de silhueta e do número de tópicos gerados.

Em geral, os modelos de representação conseguiram capturar as características dos corpus e fornecer resultados mais confiáveis, com menos grupos e maior valor para a medida silhueta. Além disso, como mostrado nas Seções (6.1.1) e (6.2.1), os agrupamentos foram muito bem realizados, dividindo produtos derivados de carne, derivados de leite e outros. Dessa forma, é possível concluir que o BERTopic conseguiu realizar agrupamentos de uma maneira eficiente, além de mostrar que, em bancos muito diferentes, tópicos análogos foram gerados, destacando a robustez do modelo para dados não rotulados e não estruturados.

Com esses resultados, a rotulação de corpora não estruturada da SEFAZ-RS pode ser obtida, atribuindo tópicos aos diversos produtos. Para a SEFAZ-RS, algo a se considerar é a obtenção, em cada tópico, dos produtos com maior e menor preço, possibilitando a identificação de possíveis fraudes. Outra questão a se considerar é a comparação dos preços praticados no Estado do RS àqueles do Sistema de Registro de Preços (SRP).

Em trabalhos futuros, uma ideia proposta é estudar o que poderia ser retirado nos corpus, eliminando, por exemplo, números irrelevantes e palavras sem sentido. Além disso, com o constante avanço do Processamento de Linguagem Natural (PLN), muitos modelos são desenvolvidos, visando a melhora de antigos e evolução de métodos. Dessa maneira, utilizar modelos de representação atualizados e que possam compreender ainda mais os dados podem melhorar o refinamento feito no BERTopic. Um exemplo é utilizar o Bode (Garcia et al. (2024)), um LLM em português. Ainda no contexto dos modelos, buscar variar e melhorar *prompts* para os LLMs pode trazer um aumento no desempenho do método. Por fim, buscar avaliar os modelos que estratificaram muito o banco de dados e observar se as informações contidas neles são realmente relevantes.

Referências Bibliográficas

Apicella, A., Donnarumma, F., Isgrò, F., e Prevete, R. (2021). A survey on modern trainable activation functions. *Neural Networks*, 138:14–32.

Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ.

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

Bengio, Y., Goodfellow, I., e Courville, A. (2016). *Deep Learning*. MIT Press.

Bird, S., Loper, E., e Klein, E. (2023). *Natural Language Toolkit (NLTK) Stop Words*. NLTK Project.

Blei, D. M. e Lafferty, J. D. (2007). Correlated topic models. *Advances in Neural Information Processing Systems*, 20:147–154.

Blei, D. M., Ng, A. Y., e Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Brown, M., Ryder, S., Kaplan, D., e Amodei (2020). Language models are few-shot learners. *OpenAI Technical Report*.

Brownlee, J. (2018). How to configure the number of layers and nodes in a neural network. *Machine Learning Mastery: Vermont, Australia*.

Brownlee, J. (2020). Train-test split for evaluating machine learning algorithms. *Machine learning mastery*, 23(7).

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.

Campello, R. J., Moulavi, D., e Sander, J. (2013). Density-based clustering by means of core samples. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 160–168. SIAM.

Chomsky, N. (1957). *Syntactic Structures*. Walter de Gruyter.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., e Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

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*.

Ester, M., Kriegel, H.-P., Sander, J., e Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231. AAAI Press.

Fei-Fei, L., Fergus, R., e Perona, P. (2006). One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611.

Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 20(3):121–136.

Garcia, G. L., Paiola, P. H., Morelli, L. H., Candido, G., Júnior, A. C., Jodas, D. S., Afonso, L. C. S., Guilherme, I. R., Penteado, B. E., e Papa, J. P. (2024). Introducing bode: A fine-tuned large language model for portuguese prompt-based task.

Goyal, K. (2020). Data preprocessing in machine learning: 7 easy steps to follow. *upGrad blog*, 22.

Grootendorst, M. (2022). Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*.

Hinton, G. (Year of the lecture). Neural networks for machine learning - lecture 6a: Overview of mini-batch gradient descent. Coursera Video Lecture.

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

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

Hvitfeldt, E. e Silge, J. (2021). *Supervised Machine Learning for Text Analysis in R*.

Ivakhnenko, A. G., Lapa, G., e Valentin (1967). *Cybernetics and Forecasting Techniques*. American Elsevier Pub. Co.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., e Sayed, W. E. (2023). Mistral 7b.

Kingma, D. P. e Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Kulshrestha, R. (2019). Nlp 101: Word2vec—skip-gram and cbow. *Towards Data Science*.

LeCun, Y. (1989). Generalization and network design strategies. In *Proceedings of the 1989 Connectionist Models Summer School*, pages 143–155.

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

LeCun, Y., Bottou, L., Bengio, Y., e Haffner, P. (1998). Learning efficient classification procedures and their application to handwritten character recognition. *Neural Computation*, 1(4):882–897.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., e Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Maaten, L. v. d. e Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Markov, A. (1906). Rasprostranenie zakona bol'shih chisel na velichiny, zavisyaschie drug ot druga. *Izvestiya Fiziko-matematicheskogo obschestva pri Kazanskom universitete*, 15:135–156.

McCulloch, W. S. e Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4):115–133.

McInnes, L. e Healy, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction.

McKinney, W. (2010). Data structures for statistical computing in python. *Proceedings of the Python for High Performance and Scientific Computing*, pages 56–61. https://pandas.pydata.org/.

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.

Moreira, V. P., Silva, A. S., e de Oliveira, E. S. (2003). Rslp stemming algorithm for portuguese: Implementing the text mining technique. *ISSI*, 4:471–476.

Nair, V. e Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814. JMLR. org.

OpenAI (2023). Gpt-4 technical report. *OpenAI Technical Report*.

Panchal, A. (2019). Nlp—text summarization using nltk: Tf-idf algorithm. *Towards Data Science. June*, 10:2019.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572.

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.

Quiza, R. e Davim, J. (2011). *Computational Methods and Optimization*, pages 177–208.

Radford, A., Narasimhan, K., Salimans, T., e Sutskever, I. (2018). Improving language understanding by generative pre-training. *OpenAI Technical Report*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., e Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Technical Report*.

Reimers, N. e Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks.

Reinsel, D., Gantz, J., e Rydning, J. (2018). The digitization of the world from edge to core. White Paper US44413318, IDC. IDC White Paper.

Robbins, H. e Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

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

Salnikov, M. (2018). Text clustering with k-means and tf-idf.

Schwab, K. (2016). *The Fourth Industrial Revolution*. Crown Business.

Souza, F., Nogueira, R., e Lotufo, R. (2020). BERTimbau: pretrained BERT models for Brazilian Portuguese. In *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*.

spaCy Contributors (2023). *spaCy Stop Words*.

Srinivasan, P. (2020). Sportsbert: Domain-specific pretraining of a bert-based transformer model for sports articles. https://huggingface.co/microsoft/SportsBERT. Language: English, Pipeline Tag: fill-mask, Email: prsrini@microsoft.com.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., e Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Taylor, W. L. (1953). Cloze procedure: A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., e Lample, G. (2023a). Llama: Open and efficient foundation language models.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., e Scialom, T. (2023b). Llama 2: Open foundation and fine-tuned chat models.

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.

Vidhya, A. (2020). How does the gradient descent algorithm work in machine learning? Acessado: 2/12/2023.

Werbos, P. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhang, X., Malkov, Y., Florez, O., Park, S., McWilliams, B., Han, J., e El-Kishky, A. (2023). Twhin-bert: A socially-enriched pre-trained language model for multilingual tweet representations at twitter.