Leonardo Santos Paulucio

Categorização automática de produtos utilizando apenas o título e aprendizado profundo

Vitória, ES

2022

Leonardo Santos Paulucio

Categorização automática de produtos utilizando apenas o título e aprendizado profundo

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Orientador: Prof. Dr. Thiago Oliveira dos Santos

Vitória, ES

2022

Leonardo Santos Paulucio
Categorização automática de produtos utilizando apenas o título e aprendizado profundo/ Leonardo Santos Paulucio. – Vitória, ES, 2022-
85 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Thiago Oliveira dos Santos

Dissertação de Mestrado – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Programa de Pós-Graduação em Informática, 2022.

1. Processamento de linguagem natural. 2. Classificação de texto. 3. Inteligência Artificial. 4. Redes neurais profundas. I. Oliveira-Santos, Thiago. II. Universidade Federal do Espírito Santo. III. Categorização automática de produtos utilizando apenas o título e aprendizado profundo

CDU: 004

Leonardo Santos Paulucio

Categorização automática de produtos utilizando apenas o título e aprendizado profundo

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Trabalho aprovado. Vitória, ES, 14 de Fevereiro de 2022:

Prof. Dr. Thiago Oliveira dos Santos
Orientador

Prof. Dr. Flávio Miguel Varejão
Convidado 1

Prof. Dr. Patrick Marques Ciarelli
Convidado 2

Vitória, ES
2022

Dedico este trabalho à minha família por todo o incentivo e apoio que me permitiram continuar e nunca desistir.

Agradecimentos

Primeiramente, a Deus, por ter me concedido saúde, força e disposição que me ajudaram a superar todas as dificuldades encontradas ao longo da graduação. Aos meus pais, Valério de Oliveira Paulucio e Maria da Penha Santos Paulucio, pelo amor, apoio e estímulo que me proporcionaram durante essa jornada. Aos meus irmãos, Lucas Santos Paulucio e Vanderson de Ataide Paulucio, por todo apoio e torcida para meu sucesso. À toda minha família de maneira geral e a meus amigos que sempre torceram por mim. Aos meus colegas de laboratório por todo o companheirismo e apoio durante toda a trajetória. Ao meu orientador, Thiago Oliveira dos Santos, pelo apoio e paciência que demonstrou durante a realização deste trabalho. Agradeço à Universidade Federal do Espírito Santo (UFES) e ao Laboratório de Computação de Alto Desempenho (LCAD) pela oportunidade de cursar o Mestrado em Informática e desenvolver este estudo. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior — Brasil (CAPES) — Código de Financiamento 001, Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), e Fundação de Amparo à Pesquisa e Inovação do Espírito Santo (FAPES). Agradeço também à NVIDIA Corporation pelas unidades de processamento gráfico (GPU's) doadas ao laboratório e utilizadas neste trabalho.

"O homem não teria alcançado o possível se, repetidas vezes,
não tivesse tentado o impossível."
(Max Weber)

Resumo

O Processamento de Linguagem Natural (PLN) tem recebido uma atenção cada vez maior nos últimos anos. Em parte, isso está relacionado ao enorme fluxo de dados disponibilizados todos os dias na internet, o que aumentou a necessidade de ferramentas automáticas capazes de analisar e extrair informações relevantes, principalmente do texto. Nesse contexto, a classificação de textos tornou-se uma das tarefas mais estudadas no domínio do PLN. O objetivo é atribuir categorias ou rótulos predefinidos a textos ou frases. Aplicativos importantes incluem classificação de frases, análise de sentimento, detecção de spam, entre muitos outros. Este trabalho propõe um sistema automático de categorização de produtos utilizando apenas seus títulos. O sistema proposto emprega uma rede neural profunda de última geração como uma ferramenta para extrair recursos dos títulos a serem usados como entrada em diferentes modelos de aprendizado de máquina. O sistema é avaliado no conjunto de dados do Mercado Libre de larga escala, que possui características comuns a problemas do mundo real, como classes desequilibradas, rótulos não confiáveis, além de possuir um grande número de amostras: 20.000.000 no total. Os resultados mostraram que o sistema proposto foi capaz de categorizar corretamente os produtos com uma precisão balanceada de 86,57% na divisão de teste local do conjunto de dados do Mercado Libre. Também ultrapassou o quarto lugar no ranking público do MeLi Data Challenge com 91,19% de precisão balanceada, o que representa menos de 1% da diferença para o vencedor.

Palavras-chaves: PLN, classificação de texto, classificação de frase, categorização de produto, redes neurais profundas, aprendizado de máquina, inteligência artificial.

Abstract

Natural Language Processing (NLP) has been receiving increasing attention in the past few years. In part, this is related to the huge flow of data being made available everyday on the internet, which increased the need for automatic tools capable of analyzing and extracting relevant information, especially from the text. In this context, text classification became one of the most studied tasks on the NLP domain. The objective is to assign predefined categories or labels to text or sentences. Important applications include sentence classification, sentiment analysis, spam detection, among many others. This work proposes an automatic system for product categorization using only their titles. The proposed system employs a state-of-the-art deep neural network as a tool to extract features from the titles to be used as input in different machine learning models. The system is evaluated in the large-scale Mercado Libre dataset, which has the common characteristics of real-world problems such as imbalanced classes, unreliable labels, besides having a large number of samples: 20,000,000 in total. The results showed that the proposed system was able to correctly categorize the products with a balanced accuracy of 86.57% on the local test split of the Mercado Libre dataset. It also surpassed the fourth place on the public rank of the MeLi Data Challenge with 91.19% of balanced accuracy, which represents less than 1% of the difference to the winner.

Keywords: NLP, text classification, sentence classification, product categorization, deep neural networks, machine learning, artificial intelligence

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

PLN	Processamento de Linguagem Natural
NLP	*Natural Language Processing*
IA	Inteligência Artificial
AI	*Artificial Inteligence*
ML	*Machine Learning*
BERT	*Bidirectional Encoder Representations from Transformers*
NN	*Neural Network*
UDA	*Unsupervised Domain Adaptation*
GRL	*Gradient Reversal Layer*
RNA	Rede Neural Artificial

Sumário

1 Introdução

Este capítulo visa introduzir a problemática e a proposta de solução abordadas pelo trabalho intitulado "Categorização automática de produtos utilizando apenas o título e aprendizado profundo". A Seção 1.1 apresenta o problema a ser solucionado; a Seção 1.2 apresenta a solução proposta; a Seção 1.3 destaca os objetivos gerais e específicos deste trabalho; e, por fim, a Seção 1.4 descreve a estruturação do trabalho, auxiliando a leitura dos demais capítulos.

1.1 Problemática e motivação

Todos os dias, uma grande quantidade de dados é gerada e disponibilizada na internet. Um estudo, que foi realizado em conjunto pelo Instituto Gartner e a plataforma de gestão de dados Domo, estimou que aproximadamente 9,1 mil terabytes de dados são gerados a cada seis minutos no mundo[1]. Esse enorme fluxo de dados fez com que surgisse uma necessidade cada vez maior por ferramentas automáticas que sejam capazes de analisar e extrair informações relevantes, principalmente de textos; uma vez que a abordagem tradicional (manual) se tornou impraticável. Nesse contexto, o Processamento de Linguagem Natural (PLN) passou a receber uma atenção cada vez maior nos últimos anos, de forma que hoje é possível encontrar diversas empresas em todo o mundo lançando produtos que contam com alguma das ferramentas de PLN em seu funcionamento, como por exemplo, pode-se citar a Google.

O principal objetivo do PLN é permitir que os computadores processem, manipulem e, mais importante, entendam o texto ou a fala em linguagem natural (CHOWDHURY, 2003), cujos dados estão disponíveis em sua maioria de forma semi ou não estruturada (SHARMA; KAUSHIK, 2017). Muitas tarefas estão relacionadas ao PLN, como: resposta a perguntas, tradução de linguagem, sumarização de textos, similaridades de textos, geração de linguagem natural e classificação de texto, sendo esta última o foco deste trabalho.

A classificação de texto pode ser definida como a tarefa de atribuir categorias predefinidas a textos ou frases, permitindo que sejam organizados, agrupados, estruturados, etc. Esta tarefa pode ser realizada manualmente ou automaticamente. A primeira abordagem pode ser feita por um grupo de pessoas que analisa o conteúdo do texto e o atribui a uma categoria adequada. Apesar da qualidade dessa abordagem de categorização, ela é financeiramente custosa, demorada e, também, está sujeita a erros humanos. Isso a torna

[1] Disponível em: <https://exame.com/carreira/dados-uso-favor/>

impraticável para o processamento de grandes quantidades de dados. Por outro lado, a classificação automática pode ser realizada por um sistema, tornando o processo mais barato e rápido, geralmente ao custo de uma qualidade inferior, permitindo o processamento de um fluxo cada vez maior de dados.

Uma tarefa de classificação de texto que é particularmente interessante é a categorização de produtos. Os produtos podem ser categorizados de várias maneiras e com base em muitos fatores (por exemplo, tamanho, preço, cor, finalidade, etc.). Este trabalho se concentra em categorizar os produtos com base em seus títulos. A princípio pode parecer uma tarefa simples, mas na verdade acaba possuindo alguns difíceis desafios. Primeiro, existem muitas categorias às quais um produto pode ser atribuído. É comum que as lojas, principalmente no *e-commerce*, vendam uma ampla gama de produtos, desde brinquedos a bebidas, alimentos e muito mais. Em segundo lugar, as categorias são estruturadas e os produtos geralmente são atribuídos a categorias e subcategorias. Por exemplo, um carrinho de brinquedo pode receber os rótulos "BRINQUEDOS" (categoria) e "CARROS DE BRINQUEDO" (subcategoria). Terceiro, espera-se que algumas categorias incluam muito mais produtos do que outras, o que representa um desequilíbrio inerente entre as categorias. Por último, os conjuntos de dados relacionados a esta tarefa geralmente são ruidosos, uma vez que podem possuir acentuação, pontos, caracteres especiais, entre outros. Isso é ainda pior para sites de comércio eletrônico que têm como objetivo facilitar as vendas de consumidor a consumidor, que é o foco deste trabalho. Nesse caso, os próprios vendedores atribuem categorias ao colocar um novo anúncio de produto. Independentemente do aparente sucesso desta abordagem, existem dois problemas principais: (i) os vendedores podem cometer erros honestos ao atribuir categorias erradas a determinados produtos, tornando-os menos/mais visíveis para os clientes; e (ii) os vendedores podem categorizar deliberadamente um produto incorretamente com o objetivo de chamar mais atenção para ele, o que pode levar a uma experiência ruim para os usuários e, consequentemente, diminuir as vendas gerais.

Os problemas citados motivam a proposta de um método para categorização automática de produtos nos idiomas espanhol e português. Uma vez que a grande maioria dos trabalhos encontrados na literatura tratam apenas do idioma inglês.

1.2 Proposta

Diante dos problemas citados na Seção 1.1, neste trabalho é proposto um sistema automático de categorização de produtos com base apenas em seus títulos. O sistema emprega a rede neural profunda chamada *Bidirectional Encoder Representations from Transformers* (DEVLIN et al., 2019) - BERT - para extrair as *features* dos títulos. Em seguida, diferentes modelos de aprendizado de máquina são investigados para lidar com

a tarefa de interesse. Posteriormente, o sistema proposto foi avaliado em um conjunto de dados de grande escala, lançado para o MeLi Data Challenge contendo 20 milhões de títulos de produtos e mais de 1.500 categorias. O sistema proposto foi capaz de ultrapassar o quarto lugar no conjunto de testes privado do Mercado Libre, atingindo 91,19% de acurácia balanceada (menos de 1% para o primeiro colocado).

Ao final, também é avaliado uma ideia de adaptação de domínio (*domain adaptation, em inglês*), onde a ideia é se treinar um modelo utilizando dados em que para um idioma fonte (*source domain*) as anotações são conhecidas e, para o outro idioma alvo (*target domain*) não. O objetivo é fazer com que o modelo seja capaz de aprender a extrair *features* que sejam invariantes ao domínio, que neste caso é o idioma. E, assim, ser capaz de funcionar ao ser aplicado em dados do idioma alvo, onde as anotações não são conhecidas. A ideia de se avaliar a adaptação de domínio aplicada no contexto de categorização de produtos é interessante por dois motivos principais: o primeiro é que a maioria dos *datasets* disponibilizados estão no idioma inglês, o que dificulta o treinamento e utilização de modelos em outros idiomas. O segundo é que os modelos mais recentes que obtiveram resultados relevantes em tarefas envolvendo texto possuem muitos parâmetros de treinamento, o que faz com que demorem para serem treinados, além de necessitarem de uma grande quantidade de dados anotados para isto, o que os tornam muito "caros". Sendo assim, é muito relevante a possibilidade de se treinar o modelo em um conjunto de dados fonte, onde se tem uma grande facilidade para se conseguir dados anotados e, aplicá-lo em um conjunto de dados alvo em que existe uma grande dificuldade para se obter dados anotados. Para esse treinamento foi utilizada a técnica chamada *Gradient Reversal Layer* (GANIN; LEMPITSKY, 2015). Os resultados mostraram que o modelo foi cerca de 1% melhor do que o *lowerbound* adotado na comparação.

1.3 Objetivos

Esta seção visa apresentar os objetivos gerais e específicos os quais se almejam alcançar com este trabalho. Na primeira subseção são apresentados os objetivos gerais e na subseção seguinte são apontados os objetivos específicos.

1.3.1 Objetivos Gerais

Este trabalho tem como principal objetivo desenvolver um método para extração de *features* de textos e classificação de produtos que utilize apenas as informações dos títulos dos mesmos e, que possa ser aplicado no âmbito de um problema de mundo real (grande quantidade de dados, dados desbalanceados, etc.) e bilíngue, onde será aplicado aos idiomas português e espanhol.

1.3.2 Objetivos Específicos

Pretende-se atingir o objetivo geral do trabalho por meio dos seguintes procedimentos:

- Investigar métodos para classificação de produtos;
- Investigar formas para categorizar produtos em diferentes idiomas;
- Gerar um conjunto robusto de *features* dos títulos de produtos;
- Avaliar o desempenho em um conjunto de dados de mundo real;
- Avaliar o desempenho do método em um conjunto de dados em português e espanhol;

1.4 Estrutura

O trabalho está estruturado da seguinte maneira, a saber.

No Capítulo 1 foi feita uma introdução do tema do projeto seguida de uma explanação da problemática e motivação. Na sequência é apresentada a proposta do trabalho e os objetivos gerais e específicos que se pretende alcançar.

No Capítulo 2 é feita uma explicação breve de todos os conceitos científicos que foram base para o desenvolvimento deste trabalho. Nele são apresentados os conceitos de Processamento de Linguagem Natural, Redes Neurais Artificiais, *Gradient Reversal Layer* e o modelo BERT.

No Capítulo 3 é descrito todo o desenvolvimento da proposta.

No Capítulo 4 será apresentada a metodologia que será utilizada no desenvolvimento do projeto de pesquisa.

Já no Capítulo 5 estão todas as análises e discussões dos resultados obtidos.

E por fim, o Capítulo 6 traz as considerações finais a serem feitas sobre a relevância e desempenho do método proposto.

2 Referencial Teórico

Desde o desenvolvimento tecnológico iniciado a partir da segunda metade do século XVIII, no período que ficou conhecido como Revolução Industrial, o estilo de vida da humanidade passou a sofrer grandes mudanças com o decorrer dos anos. O surgimento das primeiras máquinas trouxe uma grande rapidez na produção de mercadorias, além de causar várias transformações no processo produtivo da época.

Atualmente, com os recentes avanços tecnológicos observados na área da computação em todo mundo, vê-se o surgimento de máquinas que são capazes de executar suas atividades com o mínimo de intervenções humanas ou, em alguns casos, sem nenhuma necessidade de intervenção. Ou seja, de forma completamente autônoma, chegando a serem capazes de substituírem os próprios humanos. O que é surpreendente, visto que os primeiros maquinários que surgiram tinham que ser operados ou conduzidos, obrigatoriamente, por alguma pessoa no dia a dia. Todos esses avanços só foram possíveis graças ao grande progresso ocorrido nas áreas de engenharia e de computação, principalmente na área chamada de Inteligência Artificial (IA).

Apesar de já existirem máquinas capazes de realizarem tarefas que antes só humanos conseguiam, um grande desafio existente hoje dentro da IA é o chamado Processamento de Linguagem Natural (PLN), que tem, resumidamente, o objetivo de fazer com que as máquinas sejam capazes de compreender, interpretar e responder a linguagem dos humanos. Ainda que essa pareça ser uma tarefa simples ou trivial para o ser humano, não se pode dizer o mesmo para as máquinas.

Muitas pesquisas e estudos começaram a ser realizados na área de PLN a partir da década de 1950. Porém, os maiores progressos só começaram a surgir na última década graças a ascensão das chamadas Redes Neurais Profundas, que trouxeram grandes avanços para a área, conseguindo resultados de estado da arte em várias tarefas comuns de PLN como: classificação, análise de sentimento, sumarização de texto, resposta a perguntas, tradução de texto, etc.

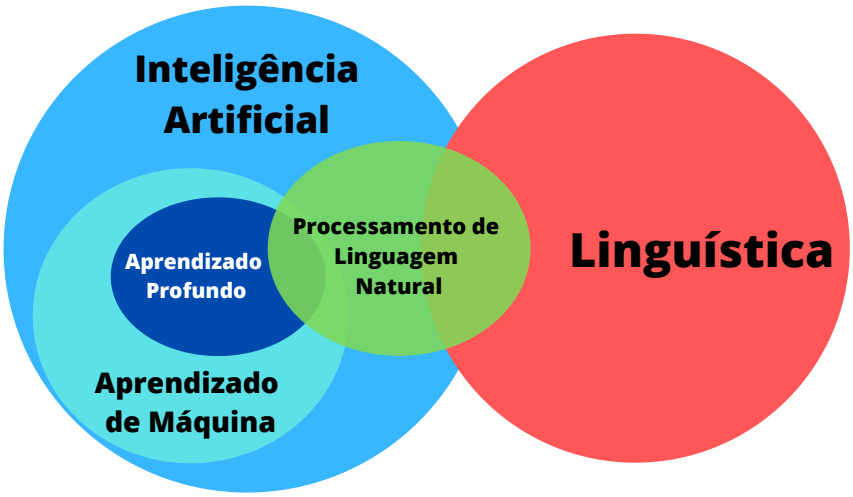
A Seção 2.1 introduz os conceitos de Processamento de Linguagem Natural; a Seção 2.2 apresenta o algoritmo da *Random Forest*; a Seção 2.3 discorre sobre os conceitos de Redes Neurais e Aprendizado Profundo; a Seção 2.4 apresenta a técnica de *Gradient Reversal Layer*; a Seção 2.5 explica como as Redes Neurais podem ser usadas nas tarefas de PLN; e, por fim, a Seção 2.6 relata alguns trabalhos relacionados encontrados na literatura recente com escopos similares ao deste trabalho.

2.1 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN) é uma subárea da ciência da computação, da inteligência artificial e da linguística. O diagrama de Venn representado na Figura 1 mostra a relação entre essas áreas. O PLN estuda os problemas da geração e compreensão automática de línguas humanas naturais, sejam elas nas formas escritas ou faladas.

De acordo com Hutchins (2005), é possível encontrar rastros de ideias sobre processos de tradução desde o século XVII, porém somente no século XX é que possibilidades reais de aplicações começaram a surgir. As primeiras aplicações de PLN foram na tarefa de tradução de texto, que teve uma grande demanda durante a Segunda Guerra Mundial.

Figura 1 – Relacionamento do PLN entre as diferentes áreas de pesquisa.



Fonte: Elaborado pelo autor.

Em 1954, um experimento, feito em conjunto pela Universidade de Georgetown e a IBM, realizou a tradução automática de mais de sessenta frases russas para o inglês (HUTCHINS, 2005). Apesar disso, nos anos que se sucederam poucas pesquisas em traduções automáticas foram conduzidas. A partir da década de 1990, algoritmos de aprendizado de máquina (*machine learning*, em inglês) começaram a ser aplicados em PLN, graças, em parte, ao aumento do poder de processamento oriundos dos avanços tecnológicos da época. Até então, grande parte dos sistemas eram baseados em conjuntos complexos de regras manuscritas. Com a utilização desses algoritmos, os modelos foram se aprimorando e se tornando mais robustos, o que propiciou resultados melhores e mais confiáveis, mesmo quando eram aplicados em dados desconhecidos.

Na última década, a área de PLN sofreu um grande salto. O avanço das Redes

Neurais Profundas e o surgimento de modelos mais avançados baseados em mecanismos de atenção foram surgindo (VASWANI et al., 2017a; DEVLIN et al., 2019; RADFORD et al., 2018) e, auxiliados pelo avanço dos algoritmos de aprendizagem semi-supervisionados e não supervisionados, alcançaram resultados de estado da arte em várias tarefas. Esse grande avanço já é considerado por alguns autores como um divisor de águas para a área. O PLN possui várias aplicações de mundo real e em vários campos de pesquisa, incluindo pesquisas médicas, inteligência de negócio, métodos de pesquisa, etc. Hoje, dentre as principais tarefas onde é aplicado, pode-se citar:

- **Classificação de Texto ou Categorização:** Processo que visa compreender o conteúdo de um texto, à principio não estruturado, e organizá-lo em categorias predefinidas;
- **Sumarização de Documentos:** Condensa o conteúdo de um texto em um conjunto reduzido, porém representativo, de frases;
- **Tradução Automática de Textos:** Processo de tradução automática de textos ou frases para um determinado idioma;
- **Conversão de Texto em Fala e vice versa:** Consiste em gerar um som falado de um determinado texto ou frase. E, analogamente, gerar um texto a partir de um áudio;
- **Análise de Sentimento:** Ato de identificar ou extrair informações de cunhos subjetivos atrelados a frases ou textos;
- **Plataformas de Buscas:** A partir de uma frase o sistema é capaz de encontrar, dentre um vasto acervo de textos, os conjuntos mais relevantes que estejam relacionados à frase buscada;
- **Reconhecimento de Entidade Nomeada:** É uma das tarefas mais populares de PLN. Consiste em localizar entidades dentro de um texto e classificá-las em categorias predefinidas como: nomes de pessoas, lugares, organizações, valores monetários, etc.;
- **Geração Automática de Texto:** Consiste em gerar textos consistentes e bem estruturados a partir de um determinado assunto ou informação que é dado como entrada.

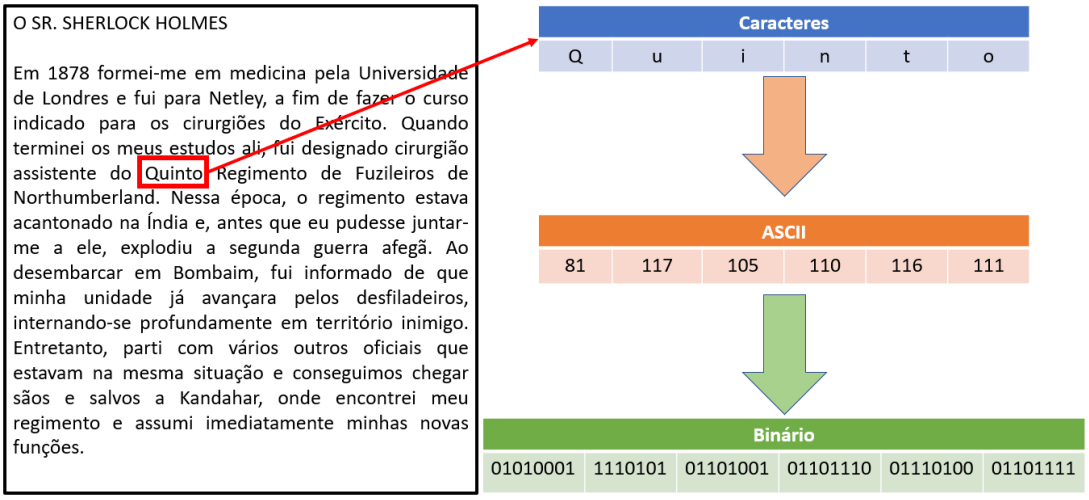
2.1.1 Classificação de Texto

Realizar a leitura de um texto, seja ele de um livro, jornal, revista ou artigo na internet; pode ser considerado algo trivial para os humanos. Porém, entender e interpretar a informação ali contida é uma tarefa um pouco mais complexa do que se aparenta. Isso

porque, muitas vezes, além da simples leitura é necessário compreender o contexto em que ele foi escrito, analisar e interpretar os sentidos das palavras em determinadas situações. Ou seja, existe uma necessidade de realizar toda uma análise morfológica, sintática e semântica do texto. Apesar de tudo isso, para os humanos essas tarefas acabam sendo transparentes devido a experiência e o aprendizado cultural que vão sendo adquiridos durante toda a vida.

Já para as máquinas, compreender essas diferentes relações é uma tarefa que tem se mostrada bastante complexa e desafiadora. Ao se realizar uma análise percebe-se, que para os computadores, as palavras nada mais são do que vários conjuntos de caracteres. E, se for feita uma análise mais aprofundada, vê-se que esses caracteres nada mais são do que números binários. A Figura 2 ilustra essa representação. Diante de tudo isso, nota-se que extrair alguma relação entre palavras se torna algo ainda mais difícil de ser realizado por uma máquina.

Figura 2 – Representação de uma palavra para uma máquina.



Fonte: Elaborado pelo autor.

Perante esta dificuldade, diversas pesquisas foram realizadas no decorrer dos anos buscando encontrar a melhor forma para se representar palavras de maneira a permitir que o computador fosse capaz de assimilar e extrair informações e relações entre palavras.

Dessa forma, hoje existem duas fases principais no contexto de classificação de textos: o pré-processamento de dados e o desenvolvimento do algoritmo e modelo.

Normalmente, os conjuntos de textos possuem uma numerosa quantidade de características que os representam. Contudo, grande parte delas acabam sendo redundantes ou descartáveis. Sendo assim, a fase de pré-processamento trata justamente de uma etapa de limpeza dos dados, de forma que apenas as informações mais relevantes são mantidas. Isso propicia uma redução do vocabulário tornando o conjunto de dados menos esparso, o que é desejável em processamentos computacionais.

Dentre os procedimentos executados nessa fase, pode-se dizer que os principais são: remoção de palavras irrelevantes (*stopwords*), *stemming*, lematização e tokenização. A Figura 3 apresenta os três procedimentos principais executados no pré-processamento.

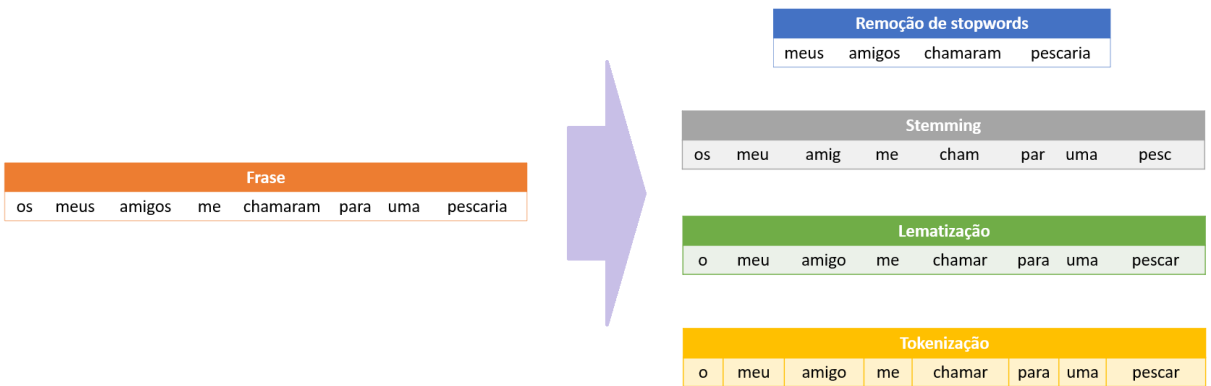
1. **Remoção de palavras irrelevantes**: Como o próprio nome já sugere, esse procedimento consiste na remoção de palavras que não agregam informação ou que não possuem muito significado e, por isso, podem ser consideradas irrelevantes. Esse grupo de palavras é composto por preposições, artigos, conjunções entre outros. Alguns exemplos são: as, e, os, de, para, com, sem, foi.

2. ***Stemming***: É um procedimento que consiste em reduzir a palavra ao seu radical ou raiz através da remoção de seus sufixos. Nem sempre a raiz obtida por esse processo consiste em uma palavra existente no vocabulário. Exemplos: amig: amigo, amiga, amigão; gat: gato, gata, gatos; prop: propõem, propuseram, propondo.

3. **Lematização**: Consiste em um processo para reduzir a palavra para seu "lema", ou para a chamada forma canônica. Diferentemente do *stemming* o processo para se obter a forma canônica de uma palavra dependerá da sua classe gramatical, por exemplo, para um verbo a sua forma canônica será o seu infinitivo. Outro ponto, é que a forma canônica obtida sempre será uma palavra existente no vocabulário. Exemplo: propor: propõem, propuserem, propondo.

4. **Tokenização**: Esse processo também é conhecido como segmentação de palavras. Consiste em quebrar a sequência de caracteres em um texto localizando o limite de cada palavra, ou seja, os pontos onde uma palavra termina e outra começa (INDURKHYA; DAMERAU, 2010). Para fins de linguística computacional, as palavras assim identificadas são frequentemente chamadas de tokens.

Figura 3 – Ilustração dos principais procedimentos executados durante a fase de pré-processamento.



Fonte: Elaborado pelo autor.

Uma vez que os dados passaram pela etapa de pré-processamento chega a etapa de definição do algoritmo que será responsável por realizar o processamento desses dados. Existe um vasto número de diferentes algoritmos que foram criados para essa finalidade, eles normalmente são divididos em dois grupos:

1. **Sistemas Baseados em Regras:** Foi a abordagem inicialmente utilizada no início do desenvolvimento dos primeiros sistemas de classificação mas que ainda encontram algumas poucas aplicações nos dias de hoje. Esses sistemas utilizam regras linguísticas cuidadosamente definidas e projetadas que acabam, na maior parte dos casos, ficando muito complexas dependendo do conjunto de dados utilizados nas definições. Tudo isso acaba gerando um certo vínculo do sistema com as regras, o que acaba fazendo com que ele não consiga bons resultados ao lidar com conjuntos de textos diferentes, pois as regras definidas acabam ficando muito específicas para um conjunto de dados.

2. **Sistemas Baseados em Aprendizado de Máquina:** É a abordagem mais utilizada atualmente. Esses sistemas utilizam algoritmos de aprendizado de máquina que são baseados em métodos estatísticos. Dessa forma, eles são capazes de aprender suas próprias regras baseadas nas tarefas que se deseja realizar e, à medida que mais dados de treinamento são utilizados, os sistemas vão aprimorando seus métodos e se tornando mais robustos. Com o avanço dos modelos de aprendizado profundo, se tornou a abordagem mais pesquisada e usada atualmente.

2.2 *Random Forest*

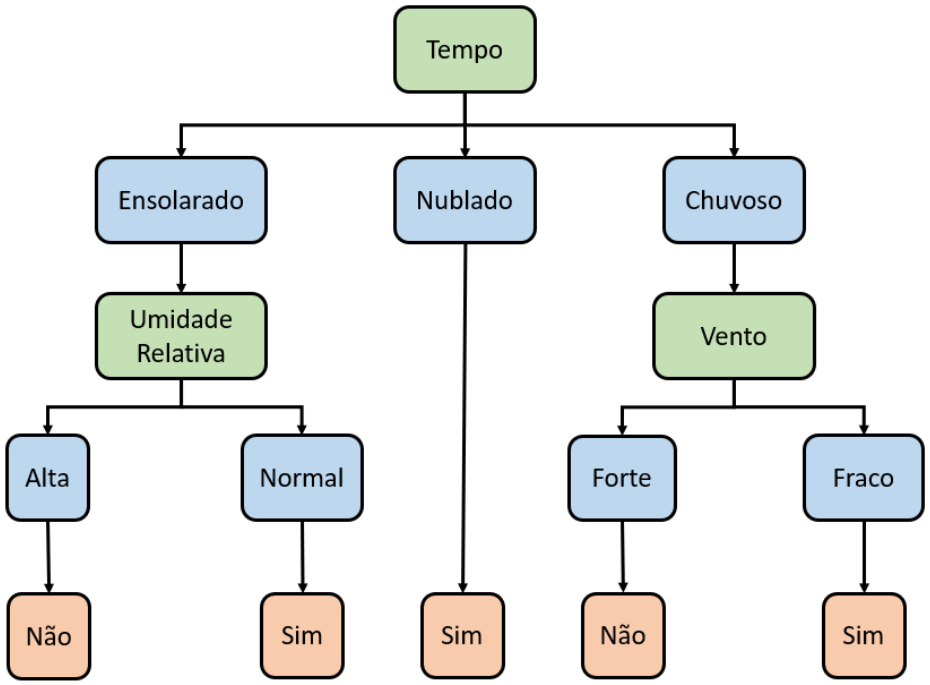
A *Random Forest* é um algoritmo supervisionado de aprendizado de máquina (*machine learning*, em inglês) (HO, 1995) muito utilizado em problemas de classificação e regressão.

O algoritmo baseia-se na utilização de várias árvores de decisão, que nada mais são do que uma forma alternativa de se representar um conjunto ou uma tabela de regras de decisão em uma forma de árvore.

A Figura 4 ilustra um exemplo de árvore de decisão para a atividade de jogar bola. Nele os nós em verde representam os atributos ou as *features* que são testados. Já os ramos em azul são os possíveis valores dos atributos. Por fim, as folhas, em laranja, representam a saída da árvore ou a sua classificação. Dessa forma, em um dia ensolarado e com umidade relativa normal a saída da árvore será “Sim”, ou seja, é um dia em que se pode jogar bola. Já no caso de um dia chuvoso e com vento forte, a árvore dará como saída “Não”, indicando que é não é um dia em que se deva jogar bola.

É importante destacar que pode existir mais de uma árvore de decisão para um mesmo problema, isso irá depender da escolha e ordem dos atributos que formarão a árvore.

Figura 4 – Exemplo de árvore de decisão para jogar bola.



Fonte: Elaborado pelo autor.

Os métodos utilizados na construção da árvore, normalmente, tentam gerar a estrutura mais otimizada possível. Para isso, esses métodos buscam identificar uma relação entre os atributos e as classes dos dados, de forma que aqueles que possuam uma maior relação fiquem sempre no topo da árvore.

Os métodos mais comumente utilizados para se definir essa estrutura são:

- **Entropia:** Esse método faz uma análise da forma como os dados estão distribuídos pelas *features* em relação as suas respectivas classes. Uma maior entropia, indica uma maior desordem dos dados. Já uma entropia baixa, indica uma menor desordem dos dados. Assim, o algoritmo realiza um cálculo que informa o ganho de informação de cada *feature*, aquela que possuir um maior ganho será colocada no topo da árvore e assim sucessivamente até que a árvore esteja montada.
- **Índice Gini:** Da mesma forma que é feito com a entropia, esse método também realiza uma análise da distribuição das *features* com relação as classes, mas utilizando um outro método. Com essa análise é escolhida a *feature* com o menor índice Gini, pois isso indica uma maior ordem na distribuição dos dados.

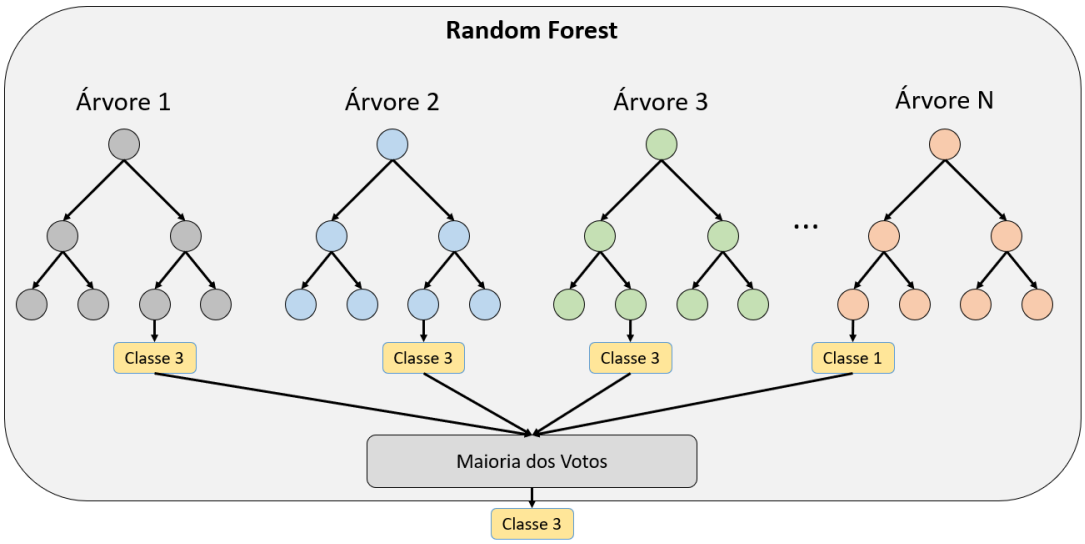
Como mencionado anteriormente a *Random Forest* utiliza um conjunto de várias árvores de decisão em sua composição. Para determinar qual a classe de uma entrada é realizado uma contagem das saídas de cada indivíduo do conjunto, sendo que a classe que obtiver a maior frequência é a classe definida para uma dada entrada. A Figura 5 ilustra

resumidamente sua estrutura.

Apesar de ser composta por várias árvores de decisão, o algoritmo possui duas diferenças principais. A primeira delas é que na criação de uma árvore, que irá compor o conjunto, não são utilizados todos os dados de treinamento fornecidos, mas sim algumas amostras desses dados. É o chamado *bootstrap*, que consiste em um processo de reamostragem com repetição. A partir dessa amostra dos dados é que uma das árvores será criada.

Uma vez que uma amostra é selecionada, a segunda etapa consiste na criação da árvore. Para isso é feita a escolha e seleção das *features* que irão para os primeiros nós. Aqui entra a segunda diferença, diferentemente do visto para árvores de decisão, essa definição de qual *feature* será a primeira da árvore não é feita baseando-se em todas as disponíveis. O algoritmo da *Random Forest* seleciona de maneira aleatória um número de *features* (pré-definido na hora de sua criação, por exemplo 2 *features*) e realiza um dos métodos citados anteriormente (entropia ou índice Gini) para selecionar qual dessas selecionadas irá formar o primeiro nó. Esse processo se repete novamente selecionando-se novas *features* (mas excluindo as já selecionadas anteriormente), realizando os cálculos e selecionando a que irá formar o próximo nó até que a árvore esteja totalmente construída.

Figura 5 – Exemplo da estrutura de uma *Random Forest*. Nesse exemplo a *Random Forest* é formada por N árvores. Cada árvore prediz uma classe para uma dada entrada. Ao final, a classe que obtiver a maior frequência é a escolhida como sendo a predição final para o dado de entrada.

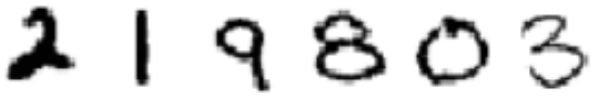


Fonte: Elaborado pelo autor.

2.3 Redes Neurais e Aprendizado Profundo

A maioria das pessoas ao olhar para os números do *dataset* MNIST[1] (DENG, 2012) ilustrado na Figura 6, é capaz de reconhecer e identificar o conjunto de dígitos 219803 sem muitos problemas. Apesar de, à principio, parecer uma tarefa trivial, quando se inicia um projeto para desenvolver um programa que seja capaz de reconhecer esses dígitos percebe-se que a tarefa é muito mais complexa do que aparenta.

Figura 6 – Conjunto de dígitos manuscritos do *dataset* MNIST.

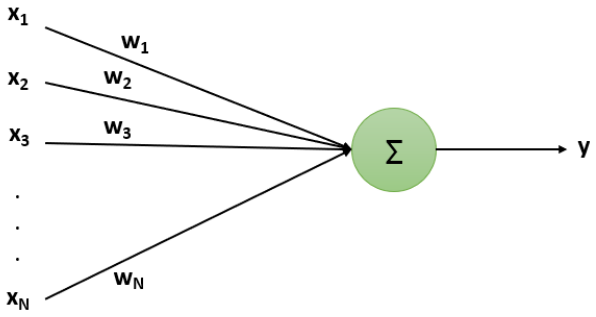


Fonte: Dataset MNIST.

O cérebro humano é formado por milhares de neurônios que são conectados entre si e que são os responsáveis por todo o processamento dos sinais e impulsos elétricos que chegam a ele pelos sentidos (visão, audição, etc.) até o reconhecimento ou extração de alguma informação. E ainda, tudo isso é feito inconscientemente ou, de forma automática, induzindo a todos que é uma tarefa fácil. Pode-se dizer que o cérebro é um supercomputador que vai sendo treinado, ou que vai evoluindo, durante toda a vida.

Diante dessa analogia com o cérebro humano, os pesquisadores Warren McCulloch e Walter Pitts apresentaram um artigo em 1943 (MCCULLOCH; PITTS, 1943) falando como os neurônios devem funcionar e criaram o primeiro modelo simplista de rede neural utilizando circuitos elétricos. Esse estudo foi o pioneiro e abriu portas para o desenvolvimento das pesquisas em redes neurais. Inspirado por esse trabalho, na década de 1950, o cientista Frank Rosenblatt criou o primeiro neurônio artificial capaz de "aprender" chamado *perceptron* (ROSENBLATT, 1958). A Figura 7 exemplifica a estrutura de um *perceptron*.

Figura 7 – Estrutura de um *perceptron*.



Fonte: Elaborado pelo autor.

Como exemplo, imagine que um *perceptron* possua apenas três entradas x_1, x_2, x_3

[1] Disponível em: <http://yann.lecun.com/exdb/mnist/>

(a quantidade de dados de entrada podem variar entre 1 e vários). Para o processamento desses dados são introduzidos pesos w_1, w_2, w_3, que são números reais responsáveis por expressar a importância de cada uma das entradas. Ao final, a saída y é dada pela Equação 2.1. Ela é um valor binário (0 ou 1) que é determinado pela soma ponderada das entradas multiplicadas por seus respectivos pesos, se esse valor ultrapassar um certo limiar (*threshold*), a saída será 1, caso contrário será 0. Esse valor é um número real e um parâmetro do neurônio. Por questões algébricas, ele geralmente é movido para a esquerda da inequação passando a ser chamado de *bias* (**b**).

$$y(\mathbf{x}) = \begin{cases} 0, & \text{se } \sum_{i=1}^{N} w_i x_i + b \leq 0 \\ 1, & \text{se } \sum_{i=1}^{N} w_i x_i + b > 0 \end{cases} \tag{2.1}$$

Dessa forma, escrevendo essa equação da saída na forma vetorial obtêm-se:

$$y = \mathbf{w}^\intercal \mathbf{x} \tag{2.2}$$

Onde:

$$\mathbf{w}^\intercal = \begin{bmatrix} w_1 & w_2 & w_3 & b \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & 1 \end{bmatrix}^\intercal \tag{2.3}$$

Apesar de sua estrutura simples, o *perceptron* apresenta dois principais problemas que fizeram com que sofresse algumas alterações em sua estrutura. O primeiro problema é que devido a sua simplicidade ele acaba não conseguindo modelar problemas mais complexos (de mundo real) onde não-linearidades estão presentes. O segundo deles é que ao se realizar uma pequena variação em alguns dos pesos do neurônio sua saída pode mudar bruscamente de 1 para 0 ou vice e versa. Isso acaba sendo um grande problema para o processo de treinamento que será visto mais adiante.

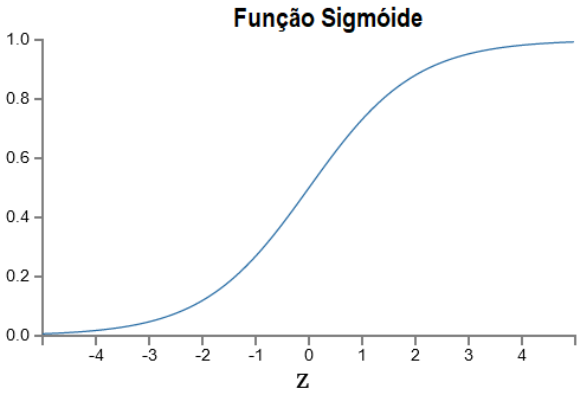
Para tratar essas questões foi usada em sua saída uma função de ativação contínua e não-linear que faz com que a saída do neurônio passe a ser também um número real e não mais uma saída binária. Hoje existem várias funções de ativações, dentre as mais conhecidas pode-se citar: Sigmóide, *Rectified Linear Unit* - ReLU (AGARAP, 2018), Tanh, *Gaussian Error Linear* - GELU (HENDRYCKS; GIMPEL, 2016), etc. Como exemplo, a função sigmóide é dada pela Equação 2.4.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.4}$$

$$\sigma(z) = \frac{1}{1 + exp(-\sum_{i=1}^{N} w_i x_i + b)} \tag{2.5}$$

Ao se introduzir a combinação linear das entradas como sendo a entrada da função de ativação, obtém-se a saída indicada pela Equação 2.5. Perceba que existem algumas similaridades com a saída do *perceptron*. Primeiro, quando se tem um valor de entrada z muito grande $e^{-z} \approx 0$ e assim $\sigma(z) \approx 1$, de forma semelhante ao que se tinha com o *perceptron*. Da mesma forma, quando a entrada é um valor z muito negativo $e^{-z} \approx \infty$ e, assim, $\sigma(z) \approx 0$. Outro ponto é que agora a saída do neurônio passa a possuir uma certa suavidade entre seus valores extremos, nesse caso entre 0 e 1. Com isso, pequenas alterações nos valores de entrada geram pequenas alterações em sua saída, diferentemente do *perceptron* original. Essa ideia fica mais clara ao observar-se a curva da função que está ilustrada na Figura 8.

Figura 8 – Gráfico da Função Sigmóide.



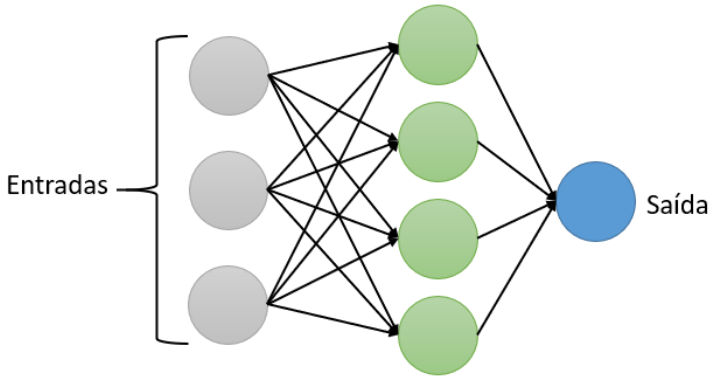
Fonte: Elaborado pelo autor.

Uma Rede Neural Artificial (RNA) nada mais é do que um conjunto desses vários neurônios artificiais conectados entre si. A Figura 9 ilustra a estrutura de uma RNA simples.

Para que uma RNA possa executar alguma tarefa, por exemplo de classificação, ela precisa passar por uma etapa de treinamento que visa realizar os ajustes dos pesos e do *bias* de seus neurônios, de forma que a saída seja a mais próxima possível do valor desejado, para uma dada entrada.

Para isto é utilizado o algoritmo de retropropagação (RUMELHART; HINTON; WILLIAMS, 1986), ou *backpropagation*, em inglês. Este algoritmo visa realizar estes ajustes com base no erro obtido na saída da RNA. Uma vez que este erro é calculado, os pesos são ajustados, começando-se da última camada até as primeiras (de trás para a frente), sempre objetivando a minimização do erro obtido pela RNA. Esse treinamento é realizado

Figura 9 – Estrutura de uma rede neural artificial simples. Ela possui 3 entradas, 4 neurônios em uma camada oculta e 1 neurônio na camada de saída.



Fonte: Elaborado pelo autor.

de forma iterativa e os pesos são atualizados seguindo a Equação 2.6.

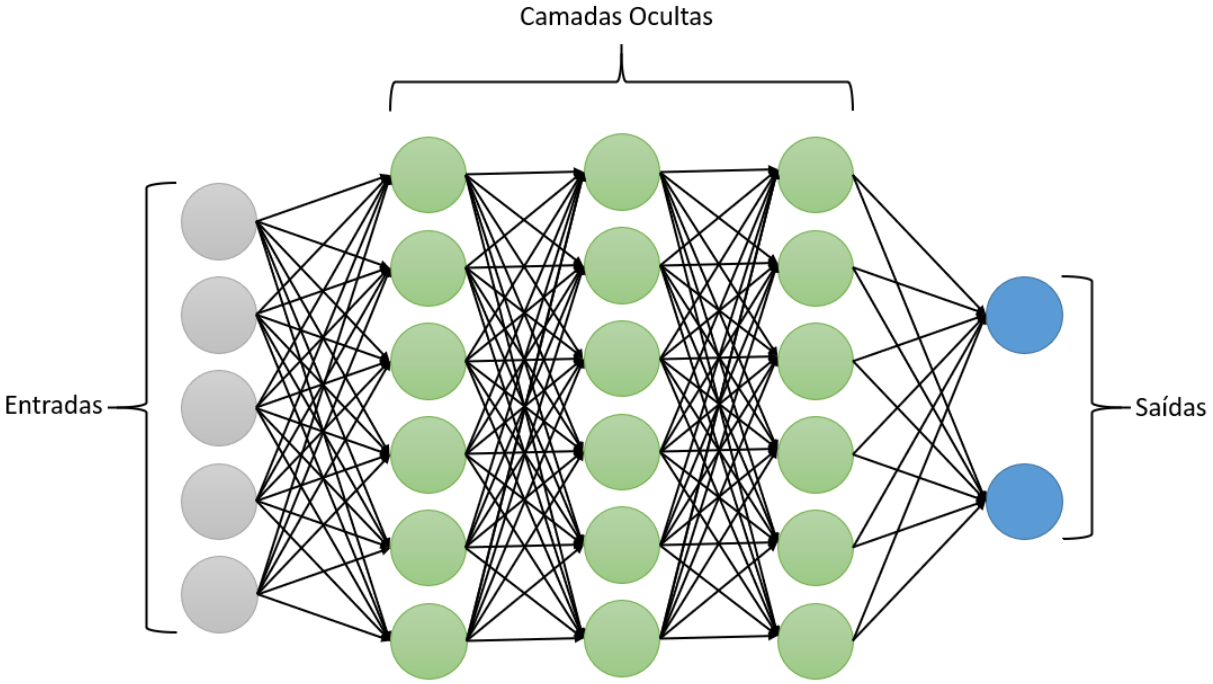
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla C(\mathbf{w}_t) \tag{2.6}$$

Onde:

- w_{t+1} representa o próximo valor do peso;
- w_t representa o valor atual do peso;
- α representa a taxa de aprendizagem (*learning rate*), que determina a taxa com que os ajustes serão realizados.
- $C(\mathbf{w})$ representa a função de custo (ou em inglês *loss function*), que é responsável por calcular o erro entre o valor predito pelo modelo e o valor desejado dado uma entrada;
- ∇ representa o vetor gradiente da função de custo $C(\mathbf{w})$, ou seja, ele diz a direção para onde a função de custo cresce. Porém, o sinal de menos faz com que os ajustes sejam realizados na direção oposta, para onde a função diminui. Assim, o objetivo passa a ser o de minimização da função de custo e, assim, diminuir o erro entre o valor predito e o valor correto (ou desejado);

Apesar de serem capazes de lidar com alguns problemas, as RNAs simples, que possuem apenas uma camada não conseguem modelar problemas mais complexos. Para isso, foram criadas redes neurais profundas (com várias camadas ocultas) que fazem parte do Aprendizado Profundo, ou *Deep Learning*, em inglês. A Figura 10 ilustra a arquitetura de uma rede neural profunda, chamada de totalmente conectada, ou em inglês *fully*

Figura 10 – Rede neural profunda *fully connected*. Ela possui 5 entradas, 3 camadas ocultas com 6 neurônios em cada e 2 saídas.



Fonte: Elaborado pelo autor.

connected. Essa rede consiste em vários neurônios interligados entre si, separados por várias camadas, as chamadas camadas ocultas. Assim, a saída dos neurônios de uma camada são interligados aos neurônios das camadas subsequentes.

Ao se tratar de tarefas que envolvem textos, essas redes *fully connected* acabam se tornando inviáveis. Um dos motivos é que elas acabam precisando de um tempo de treinamento muito grande para o ajuste dos pesos, uma vez que a quantidade de pesos vai crescendo de forma rápida à medida que essas redes vão aumentando. Outro motivo é que essas redes possuem uma certa dificuldade para se conseguir fazer uma modelagem, ou representação, que acaba sendo inerente ao se trabalhar com qualquer tipo de texto.

Dessa forma, surgiram modelos mais complexos que utilizam novos mecanismos focados justamente em tarefas que envolvem texto. Como exemplos podem ser citados: Transformer (VASWANI et al., 2017a), GPT (RADFORD et al., 2018), BERT (DEVLIN et al., 2019), etc.

2.4 *Gradient Reversal Layer*

Geralmente quando um classificador é treinado em um domínio específico (*source domain*), que possui uma certa distribuição em seus dados, ele não consegue obter um bom desempenho em um outro domínio com distribuição diferente (*target domain*). A ideia de diferentes domínios está ilustrada na Figura 11.

Figura 11 – Alguns exemplos de diferentes domínios.

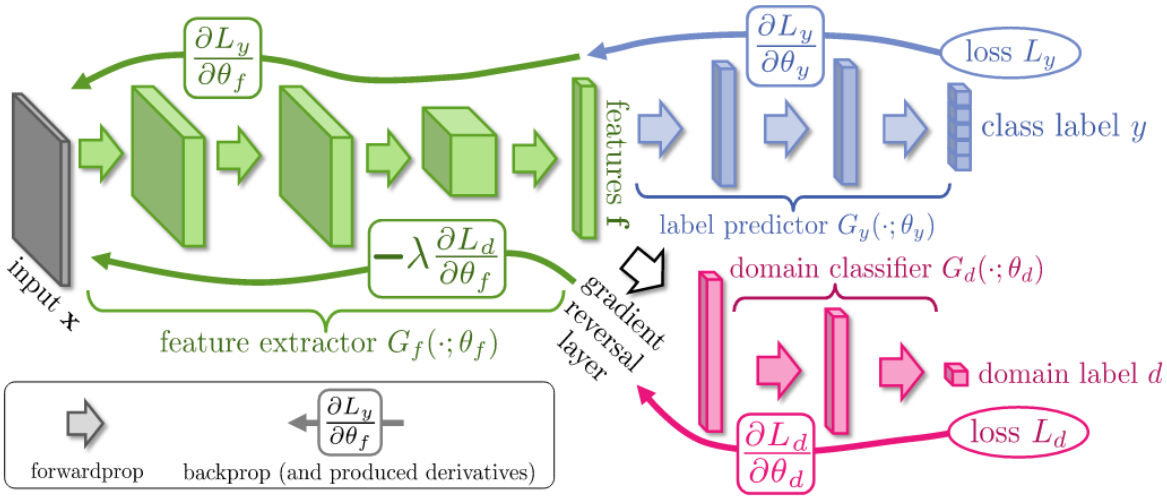
Fonte: (GANIN; LEMPITSKY, 2015).

Diante disso, surge a necessidade de se conseguir treinar um modelo em um domínio de origem (*source domain*) onde os *labels* são conhecidos e utilizá-lo para predizer com precisão os *labels* em um outro domínio alvo (*target domain*) onde não se tem informações dos *labels*. Este processo é conhecido como Adaptação de Domínio Não-Supervisionada (Unsupervised Domain Adaptation - UDA). Uma das técnicas para se resolver este tipo de problema é a *Gradient Reversal Layer*(GRL) (GANIN; LEMPITSKY, 2015). Ela está ilustrada na Figura 12.

O GRL é uma arquitetura que é composta por 3 componentes:

- **Extrator de Características:** É o componente responsável por gerar as *features* dos dados tanto no domínio de origem como no de destino;
- **Classificador de classe:** É o componente responsável por realizar a classificação dos dados no domínio de origem, visto que nesses casos os *labels* são conhecidos;
- **Classificador de domínio:** É o responsável por predizer se as *features* geradas pelo extrator de características correspondem ao domínio de origem ou de destino.

Figura 12 – Arquitetura do GRL.



Fonte: (GANIN; LEMPITSKY, 2015).

O objetivo principal é que o extrator de características consiga transformar os dados de origem (*source*) e destino (*target*) em um conjunto de *features* que "aparentemente" vêm de uma mesma distribuição. Dessa forma, o classificador de domínio não deverá ser capaz de identificar se os dados são do *source* ou *target* e o classificador de classe deverá funcionar para ambos os domínios.

Para isso, durante o treinamento, o extrator de características é treinado para maximizar a *loss* do classificador de domínio, enquanto o classificador de domínio é treinado para minimizar sua *loss* de classificação de domínio. Essa ideia é similar ao método de treinamento chamado *adversarial training*, onde o extrator de características tenta confundir o classificador de domínio fazendo com que as distribuições dos dados de *source* e *target* se aproximem. Já o classificador de classe é treinado para predizer os *labels* dos dados do *source*, uma vez que esses são conhecidos.

Com isso, o extrator de características é treinado para minimizar a *loss* do classificador de classe e maximizar a *loss* do classificador de domínio. Isso faz com que ele seja capaz de aprender a gerar *features* que sejam invariantes ao domínio dos dados de entrada.

2.5 *Bidirectional Encoder Representations from Transformers* - BERT

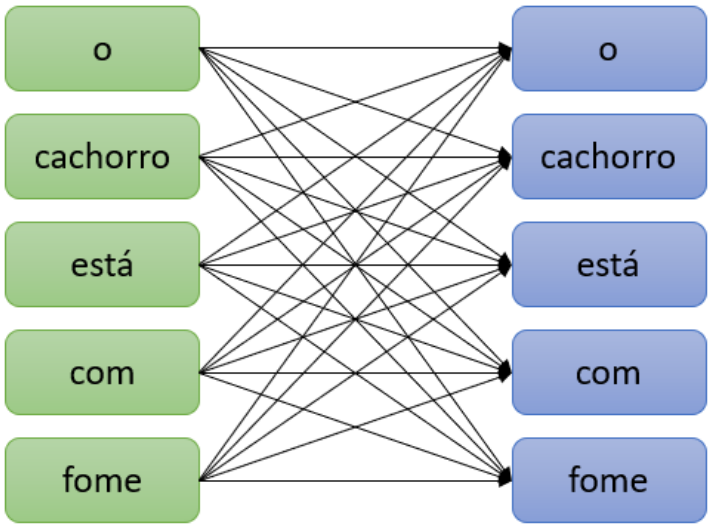
Em 2017 a Google lançou a *Transformer* (VASWANI et al., 2017b), um modelo baseado em mecanismos de atenção que obteve resultados surpreendentes em tarefas de tradução de texto, além de trazer uma maior paralelização, o que diminuiu o tempo necessário para os treinamentos. Esse modelo é considerado por alguns pesquisadores como um marco na área de PLN, pois após o seu lançamento diversos novos modelos começaram a surgir e alcançar resultados de estado da arte em diversas tarefas de PLN, dentre eles está o BERT.

O BERT (DEVLIN et al., 2019), também desenvolvido pela Google, foi pré-treinado em um conjunto de texto de aproximadamente 33 milhões de itens em inglês, alcançando resultados de estado da arte em 11 tarefas de PLN. No final de 2019, a Google passou a adotar o modelo em seu mecanismo de busca, sendo considerado pela própria empresa como um dos maiores saltos na história de seu buscador.

Diferentemente dos modelos tradicionais de PLN que são treinados em uma determinada ordem (esquerda para a direita, ou direita para a esquerda), o BERT foi um dos primeiros modelos pré-treinados que introduziu a ideia de treinamento bidirecional, ou seja, ele emprega todo o conjunto de palavras de uma frase (ou sentença) no treinamento ao mesmo tempo. Isso traz duas principais vantagens. A primeira é que o modelo se torna capaz de extrair as relações entre todas as palavras que compõem a frase, ou seja, consegue extrair informações de contexto. A segunda é que ele consegue construir um modelo da linguagem com um conjunto de dados de texto menor. Isso acaba ocorrendo graças a sua

bidirecionalidade no treinamento já que ele extrai informações mais precisas do contexto das palavras. A Figura 13 ilustra essa ideia.

Figura 13 – Diagrama ilustrando a ideia de treinamento bidirecional. Perceba que todas as palavras possuem uma conexão entre si, o que permite extrair informações de relações e contexto entre elas.



Fonte: Elaborado pelo autor.

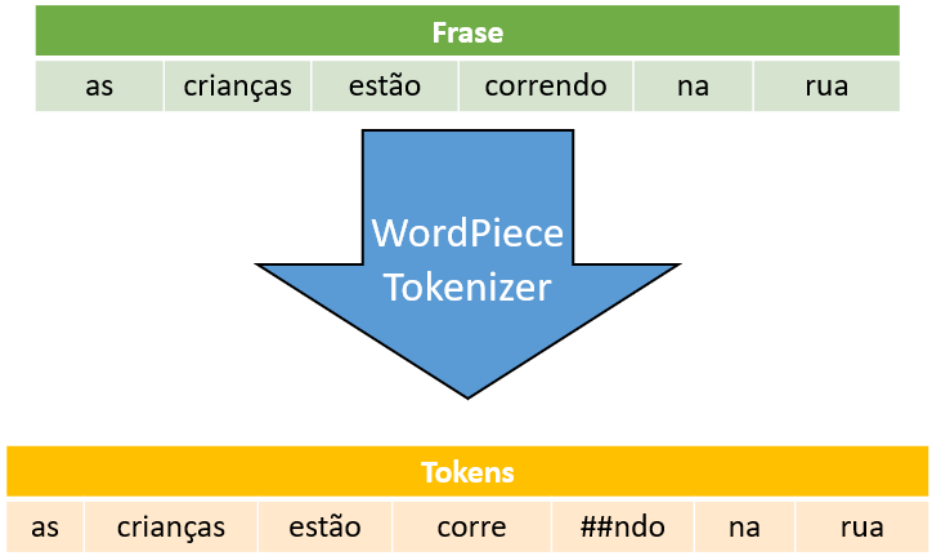
Antes que o modelo possa processar as entradas diretamente (palavras) e extrair as informações relevantes e de contexto, é necessário efetuar alguns procedimentos nos dados brutos para que o modelo possa compreender.

A primeira delas é a tokenização. Para isso o BERT emprega uma estratégia chamada *WordPiece Tokenizer* (WU et al., 2016; SCHUSTER; NAKAJIMA, 2012). Essa abordagem utiliza um algoritmo que verifica se a palavra existe no vocabulário, e caso não exista, ela vai sendo quebrada (ou segmentada) em subpalavras recursivamente até que elas sejam encontradas no vocabulário utilizado. Esse processo é muito eficaz para tratar casos em que a palavra não esteja no vocabulário, pois com o processo recursivo de segmentação as chances de uma parte menor da palavra estar no vocabulário aumentam.

A Figura 14 mostra o processo de tokenização utilizando essa estratégia. Perceba que nesse exemplo a palavra “correndo” não existe nesse vocabulário, e por isso, ela é quebrada em partes menores: “corre” e “ndo”; que por sua vez existem no vocabulário.

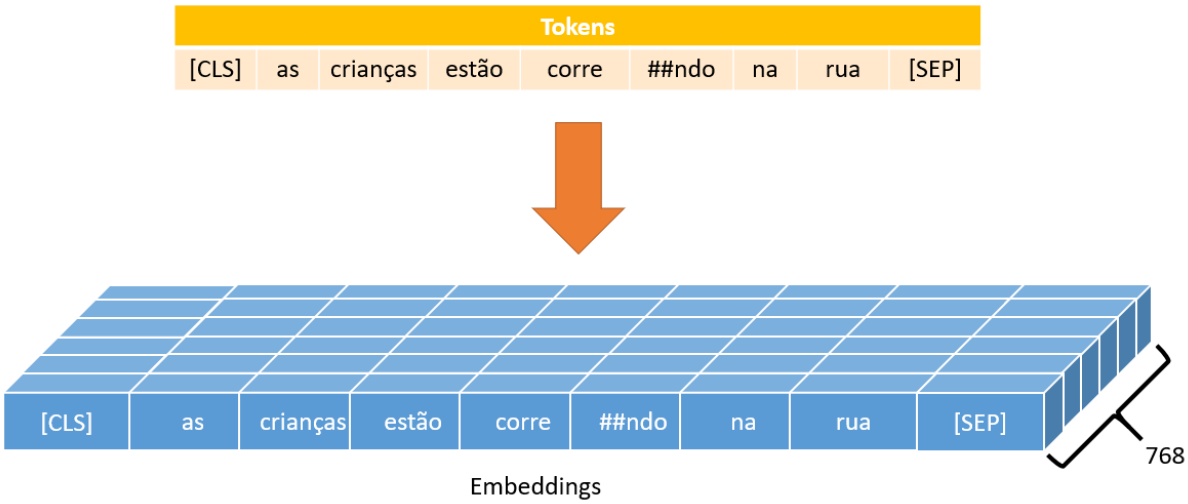
Uma vez que a entrada é convertida em tokens, o BERT possui uma camada onde cada palavra do vocabulário (token) é mapeado para um vetor de números reais de dimensão $\mathbb{R}^{768}$, chamados de *embeddings*. O BERT também inclui em seu vocabulário dois tokens especiais o [CLS] e o [SEP]. O primeiro é utilizado para tarefas de classificação, enquanto que o segundo é utilizado para separar duas sentenças em tarefas que envolvam pares de entrada. O diagrama da Figura 15 ilustra essa ideia.

Figura 14 – Exemplo do processo de tokenização utilizando a estratégia de *WordPiece*.



Fonte: Elaborado pelo autor.

Figura 15 – Ilustração do mapeamento de cada token para um vetor de dimensão 768 (*embedding*) no BERT.



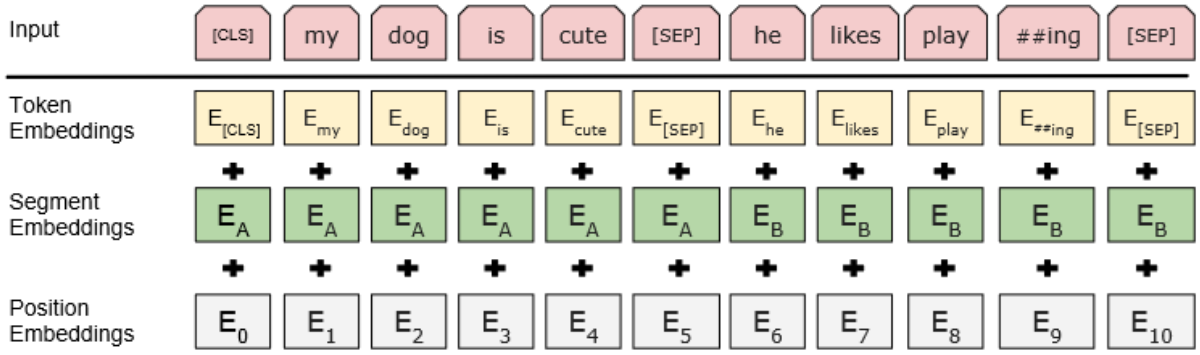
Fonte: Elaborado pelo autor.

Como o modelo do BERT trabalha com algumas tarefas que envolvem a classificação de pares de frases ou textos (por exemplo, para classificar se duas frases tratam de um mesmo assunto, se duas frases são iguais semanticamente, etc.), existe uma camada que é responsável por inserir essa informação aos *embeddings* para que o modelo consiga distinguir cada sentença do par de entrada. Essa camada é formada por dois vetores de inteiros (*embeddings*) e também de dimensão $\mathbb{R}^{768}$. O primeiro vetor é atribuído a todos os tokens que pertencem a primeira frase do par e, de forma análoga, o segundo vetor é atribuído a outra frase (do par) de entrada.

Por fim, como o modelo do BERT processa toda a entrada de forma paralela,

a princípio ele não é capaz de captar a ordem sequencial da entrada, ou seja, ele não consegue identificar a ordem das palavras na frase. Para incorporar e transmitir essas informações ao modelo existe uma outra camada, chamada de *positional embedding*, que é a responsável por essa tarefa. O mecanismo também consiste em uma representação vetorial de dimensão $\mathbb{R}^{768}$ para cada posição. A Figura 16 ilustra resumidamente todas essas camadas de representação utilizadas no BERT. Vale destacar que todos esses *embeddings* de representação são aprendidos durante o treinamento do modelo.

Figura 16 – Diagrama ilustrando a representação das entradas para o modelo BERT.



Fonte: (DEVLIN et al., 2019).

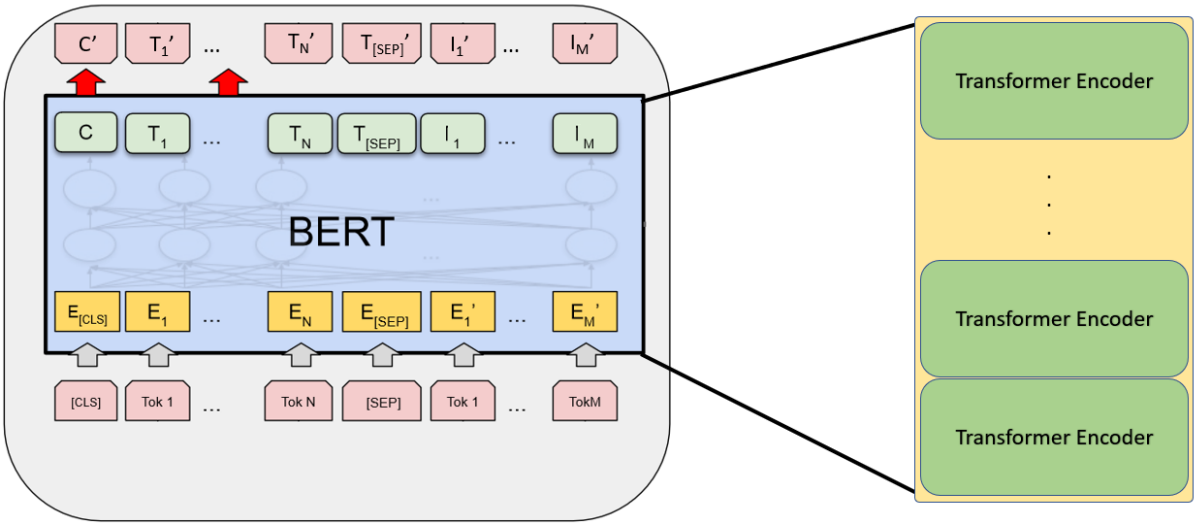
2.5.1 Arquitetura

O BERT utiliza uma das estruturas do modelo *Transformer*, o chamado codificador ou em inglês *Encoder*. Basicamente, sua arquitetura é formada por uma pilha desses *encoders* acoplados, de forma que a saída de um se torna a entrada do outro. Os *encoders* processam toda a entrada que recebem de forma paralela, o que garante uma maior rapidez no processamento, um dos pontos fortes dos modelos baseados na *Transformer*. O BERT é capaz de processar sequências de até 512 tokens por vez. A Figura 17 mostra a arquitetura simplificada do BERT.

O *encoder* é responsável por processar toda a entrada e gerar codificações que consigam armazenar as informações relevantes e do contexto dos dados. Para fazer isso, o *encoder* utiliza mecanismos de atenção que são capazes de calcular o peso ou a relevância de cada palavra em relação as outras. Por exemplo, para uma máquina é difícil conseguir diferenciar entre a palavra "banco" de um contexto onde representa uma agência bancária, de um outro onde diz respeito a um local para se sentar. Em modelos mais simples, a palavra "banco" seria convertida em um mesma representação ou *embedding* independente do seu contexto, porém com os mecanismos de atenção o modelo consegue gerar representações diferentes para uma mesma palavra levando em conta o seu contexto.

A Figura 18 ilustra um exemplo do mecanismo de atenção. Na frase em verde a palavra "banco" possui o sentido de um local para se sentar, já na frase em azul a palavra

Figura 17 – Arquitetura simplificada do BERT.



Fonte: Adaptado de (DEVLIN et al., 2019).

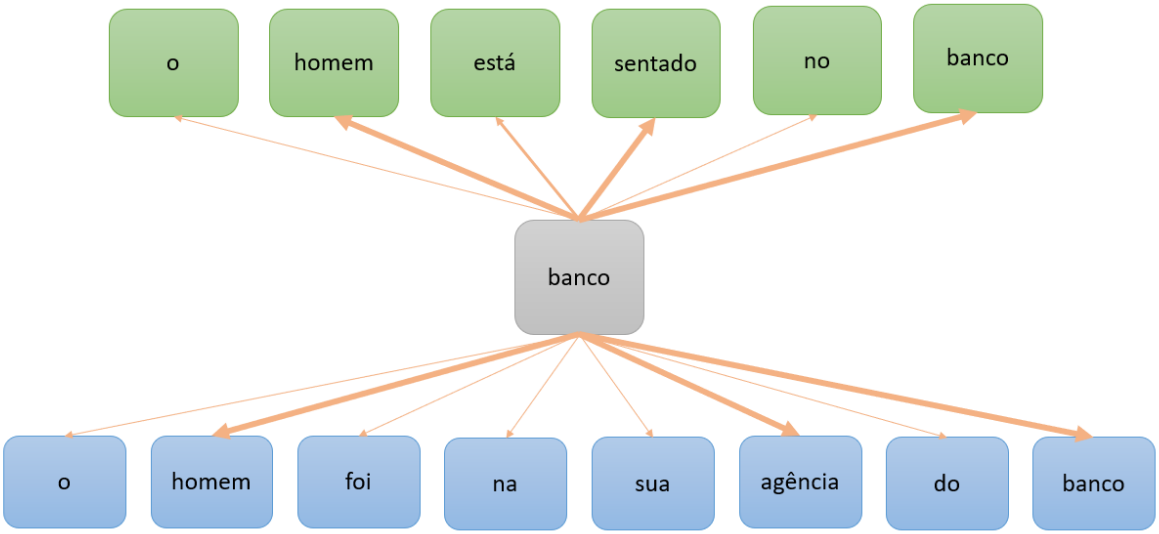
representa a ideia de agência bancária. As setas representam as relações existentes entre a palavra "banco" e todas as outras palavras, de ambas as frases. A espessura da seta representa o grau de importância, ou seja, setas mais grossas indicam que a palavra banco possui uma maior relação, e de forma análoga, setas mais finas indicam que possui uma baixa relação. Percebe-se que na frase em verde a palavra banco possui uma maior relação com as palavras "homem" e "sentado". Enquanto que para a frase em azul as palavras "homem" e "agência", apresentam uma maior ligação. Dessa forma, o modelo é capaz de gerar um *embedding* distinto para a palavra "banco" para cada uma dessas frases, ou contextos.

2.6 Trabalhos Relacionados

O problema de classificação de textos tem sido amplamente estudado na literatura. Vários trabalhos investigaram a aplicação de algoritmos estatísticos e tradicionais de aprendizado de máquina: *K-Nearest Neighbor* (KNN) (TAN, 2005; YANG, 1999), *Naive Bayes* (LEWIS; RINGUETTE, 1994; CHEN et al., 2009), Regressão (FUHR et al., 1991; YANG; CHUTE, 1994), Redes Neurais (WIENER et al., 1995) e *Support Vector Machines* (SVM) (JOACHIMS, 1998; DUMAIS et al., 1998). Contudo, neste trabalho foi realizada uma revisão mais limitada da literatura, focando-se em técnicas que utilizam aprendizado profundo, uma vez que estas têm conseguido resultados expressivos nos últimos anos.

O aprendizado profundo, ou em inglês *Deep Learning*, tem sido aplicado com sucesso a vários domínios de pesquisa, como: em biologia (MAHMUD et al., 2018), em reconstrução de documentos (PAIXãO et al., 2020), em carros autônomos (BERRIEL et al., 2017b; ARRUDA et al., 2019; TORRES et al., 2019), aprendizado multi-domínio

Figura 18 – Ilustração das relações captadas pelos mecanismos de atenção para a palavra "banco" em duas frases com contextos distintos. Flechas mais grossas indicam uma maior relação enquanto flechas mais finas indicam uma menor relação.



Fonte: Elaborado pelo autor.

(BERRIEL et al., 2019), consumo de energia (BERRIEL et al., 2017a), e reconhecimento de expressão facial (Zavarez; Berriel; Oliveira-Santos, 2017; LOPES et al., 2017).

Na última década, modelos de redes neurais profundas começaram a obter resultados surpreendentes em tarefas de PLN (YIH et al., 2011; COLLOBERT et al., 2011; WEI et al., 2018). Motivados pelo sucesso das Redes Neurais Convolucionais (em inglês, *Convolutional Neural Network* - CNNs) em problemas de visão computacional, alguns trabalhos (KIM, 2014; ZHANG; ZHAO; LECUN, 2015; WANG; QU, 2017) também investigaram sua aplicação na classificação de textos.

Li, Zhan e Li (2018) apresentaram uma combinação de *Long short-term memory* (LSTM) e CNN chamada Bi-LSTM-CNN para classificar textos de notícias em grande escala. Lai et al. (2015) propuseram uma Rede Neural Convolucional Recorrente para tarefas de classificação de texto que superaram abordagens anteriores de última geração em três conjuntos de dados. Vaswani et al. (2017b) apresentaram o *Transformer*, um modelo baseado em mecanismos de atenção que dispensam a recorrência e convoluções. O modelo *Transformer* obteve qualidade superior nas tarefas de tradução automática, sendo mais paralelizável e exigindo muito menos tempo para treinar. Em 2018, um codificador bidirecional (BERT) foi introduzido (DEVLIN et al., 2019). Baseado no codificador *Transformer*, o modelo BERT alcançou desempenho de estado da arte em onze tarefas de PLN e, por isso, esse modelo foi o escolhido e utilizado nesse trabalho.

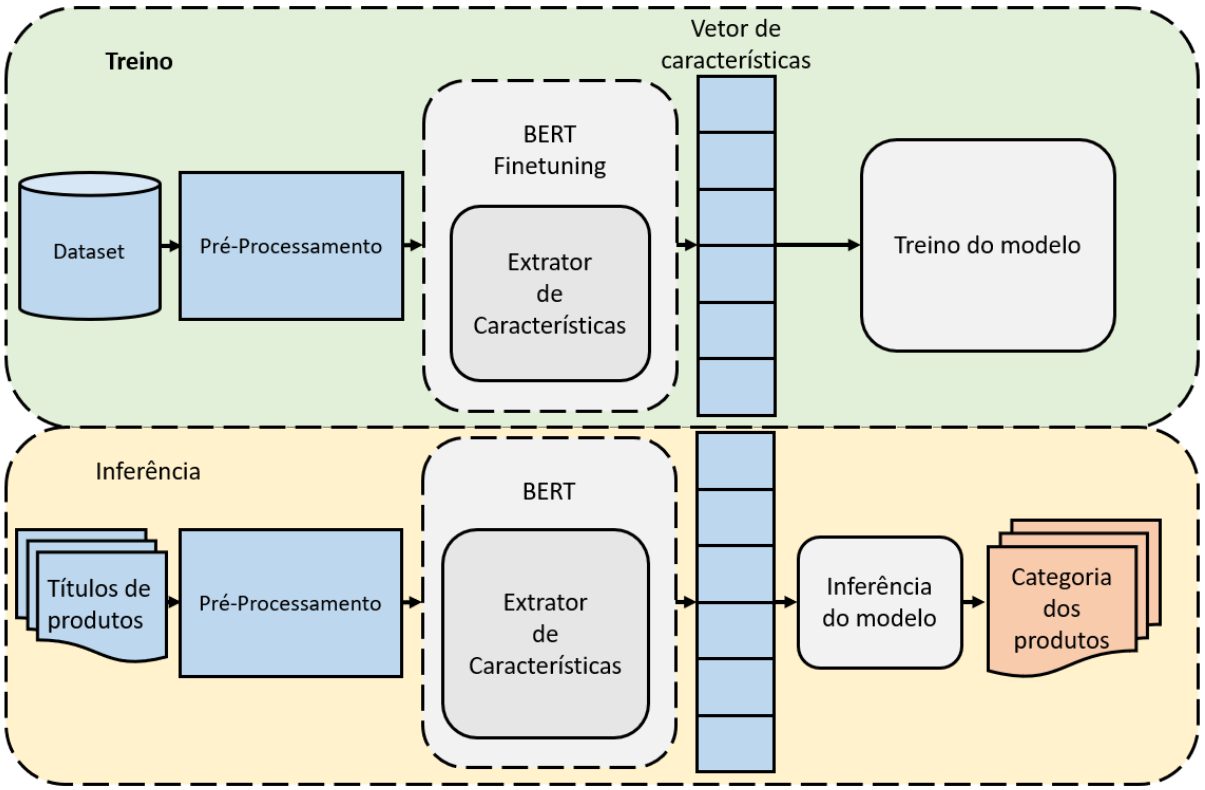
Existem alguns trabalhos na literatura com foco na categorização de produtos. Kozareva (2015) propôs um sistema de categorização automática de produtos usando diferentes recursos como N-gram, Bi-gram, LDA (*Latent Dirichlet allocation*), etc. Para a

categorização, a autora investigou dois algoritmos: um-contra-todos (OAA) e torneio de correção de erros (ECT). Nesse trabalho, em vez disso, concentrou-se em utilizar recursos baseados em aprendizado profundo, porque eles têm se mostrado mais eficazes nos últimos anos. Cevahir e Murakami (2016) usaram uma combinação de *Deep Belief Nets* e *Deep Autoencoders* para categorizar um conjunto de dados de produto japonês de comércio eletrônico em grande escala usando títulos e descrições. Comparados a eles, nesse trabalho foi utilizada uma formulação do problema mais restrita, em que apenas o título está disponível para categorização. Além disso, há interesse na categorização bilíngue, mais especificamente: espanhol e português. Outra abordagem usando a estratégia "dividir para conquistar" foi apresentada em (WIROJWATANAKUL; WANGPERAWONG, 2019). A ideia era combinar três modelos de categorização de produtos, cada um responsável por classificar uma das três informações disponíveis: títulos, descrições e imagens. Em nosso trabalho, como apenas o título está disponível, as abordagens multimídia estão além do escopo de interesse. Além disso, existem dois problemas principais que dificultam uma comparação justa com outros métodos: (i) nenhum deles tem implementações disponíveis publicamente e (ii) a maioria deles está usando conjuntos de dados privados.

3 Categorização de Produtos

O sistema proposto para categorização de produtos está resumidamente ilustrado na Figura 19 e compreende três etapas principais. A etapa inicial de pré-processamento visa remover todo o ruído potencial e padronizar as amostras de entrada. Em seguida, o extrator de características é treinado para que seja capaz de extrair melhores informações dos dados de entrada para a tarefa de interesse que, nesse caso, é a de classificação. Por fim, os modelos são treinados. Uma vez finalizada essa etapa, os modelos estão prontos para a realização das inferências, ou seja, podem receber um título de produto como entrada e produzir a respectiva categoria. Vale destacar que, dado um título de produto o sistema fornece em sua saída apenas uma das categorias dentre todas as possíveis (problema multiclasses), de forma que nunca o sistema irá produzir em sua saída um conjunto de várias categorias possíveis (problema multirrótulo).

Figura 19 – *Overview* do método proposto. Na etapa de treino, o extrator de *features* é ajustado para que produza um conjunto de *features* robusto capaz de representar os dados de entrada. Uma vez que o extrator de *features* está treinado, ele é responsável por produzir as *features* que serão usadas para o treinamento dos classificadores. Por fim, os classificadores treinados são capazes de categorizar os produtos utilizando seus títulos como entrada.



Fonte: Elaborado pelo autor.

3.1 Pré-processamento de dados

Os dados de entrada (títulos de produtos) são disponibilizados em forma de texto bruto, visto que essa informação é preenchida pelos próprios vendedores na criação do anúncio do produto. Essa liberdade no preenchimento faz com que o campo possa conter caracteres desnecessários e indesejados.

Para remover tais caracteres e padronizar a entrada, alguns procedimentos usualmente adotados nas aplicações de PLN são empregados, à saber: Primeiro, pontuação, acentos (os autores removeram os acentos do vocabulário do BERT multilíngue para diminuição do vocabulário[1]) , números e caracteres especiais são removidos. Em segundo lugar, os títulos dos produtos são convertidos para letras minúsculas. Terceiro, as palavras irrelevantes (*stopwords*) nos idiomas de interesse (no caso desse trabalho, espanhol e português) são removidas. Por último, um processo de tokenização é realizado visando converter a entrada resultante em tokens para que o BERT possa processar (o tokenizador multilíngue do BERT foi usado nesta etapa). A Figura 20 exemplifica as etapas de pré-processamento citadas.

Figura 20 – Exemplo das etapas de pré-processamento realizadas.

Pré-processamento	Texto							
Original	Máquina de Lavar ELETROLUX #12kg.							
Remoção de pontuação, acentos, números e caracteres especiais	Maquina de Lavar ELETROLUX 12kg							
Converte para minúsculo	maquina de lavar eletrolux 12kg							
Remove stopwords	maquina lavar eletrolux 12kg							
Tokenização	ma	##quina	lava	##r	ele	##tro	##lux	kg

Fonte: Elaborado pelo autor.

3.2 Extrator de Características

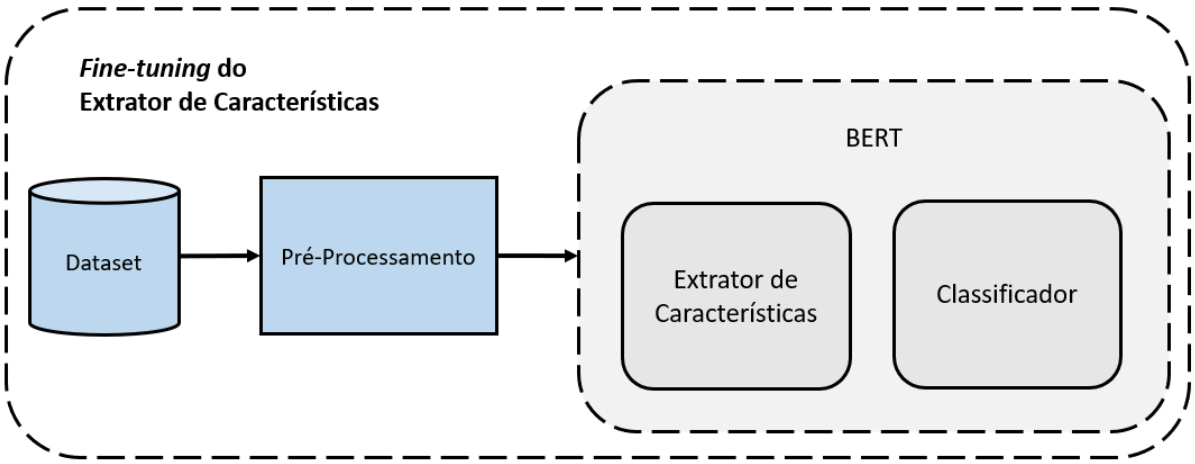
Após o pré-processamento, a arquitetura do BERT é utilizada para extrair os recursos dos títulos processados. Conforme já mencionado na Seção 2.5, o BERT é composto por um conjunto de codificadores da arquitetura do *Transformer* empilhados e usa mecanismos de atenção ao invés de conexões recorrentes, como aquelas vistas em RNNs. O BERT foi escolhido nesse trabalho devido à alta precisão obtida em tarefas de classificação de texto e, também, por ter disponível um modelo pré-treinado em um corpus de texto grande e multilíngue, que será ajustado (*fine-tuning*) para a aplicação de categorização de produtos.

Para realizar o procedimento de *fine-tuning*, uma camada totalmente conectada (camada de classificação) compreendendo C saídas foi adicionada no topo da base de

[1] Disponível em: <https://github.com/google-research/bert/blob/master/multilingual.md>

BERT, onde C é o número de categorias de produtos. A rede resultante é treinada de ponta a ponta como um modelo de classificador, processando os títulos pré-processados e comparando-os com os rótulos de categorias corretas. Após essa etapa, apenas o extrator de características - que projeta os títulos de produtos pré-processados em um espaço de *features* $\mathbb{R}^{768}$ (DEVLIN et al., 2019) - é mantido. A Figura 21 ilustra essa etapa.

Figura 21 – *Fine-tuning* do extrator de características. Nessa etapa, o modelo do BERT é treinado como um modelo de classificador. Ele recebe os títulos dos produtos pré-processados e gera suas respectivas categorias que são comparadas com os rótulos corretos. Ao final, apenas o extrator de características é utilizado.



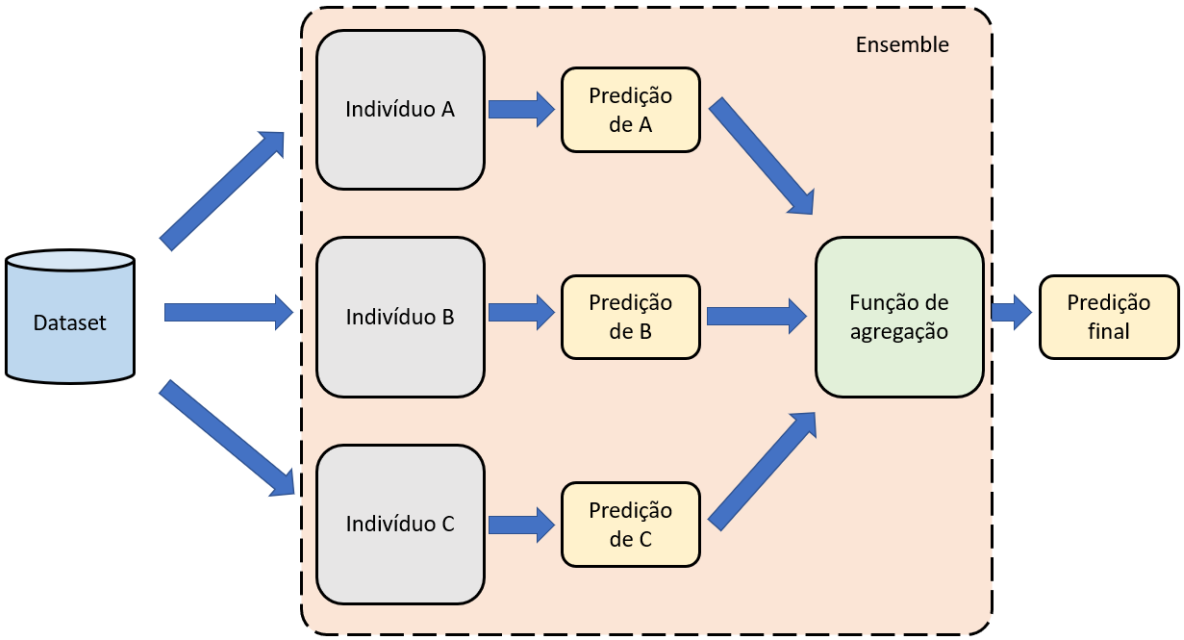
Fonte: Elaborado pelo autor.

3.3 *Ensembles*

Um *ensemble* nada mais é do que um conjunto de indivíduos, geralmente de um mesmo modelo, que combinam suas predições ou saídas com o objetivo de se obter um resultado melhor e mais assertivo do que o alcançado por um único indivíduo. Essa ideia está ilustrada na Figura 22. Estudos mostram que os *ensembles* conseguem reduzir a variância das previsões melhorando a média de acerto em relação a qualquer indivíduo que faz parte do conjunto (KUNCHEVA; WHITAKER, 2003; BROWN et al., 2005; ADEVA; BERESI; CALVO, 2005; GASHLER; GIRAUD-CARRIER; MARTINEZ, 2008).

Optou-se por utilizar *ensembles* neste trabalho devido ao fato de que o *dataset* utilizado possui uma grande quantidade de dados ruidosos e de títulos rotulados sem verificação (não confiáveis). Outro ponto, foi ao fato do custo de processamento dos modelos que necessitavam de uma grande quantidade de recursos computacionais que não estavam disponíveis no laboratório. Dessa forma, foram avaliados dois *ensembles* de modelos de classificadores: um de *Random Forests* e outro de *Redes Neurais*. Sendo que ambos foram formados por 10 indivíduos do modelo de classificação avaliado.

Figura 22 – Exemplo de utilização do *ensemble*. Neste exemplo, o *ensemble* é formado por três indivíduos. As predições dos indivíduos são combinadas através de uma função de agregação que é responsável por computá-las e gerar a predição final.



Fonte: Elaborado pelo autor.

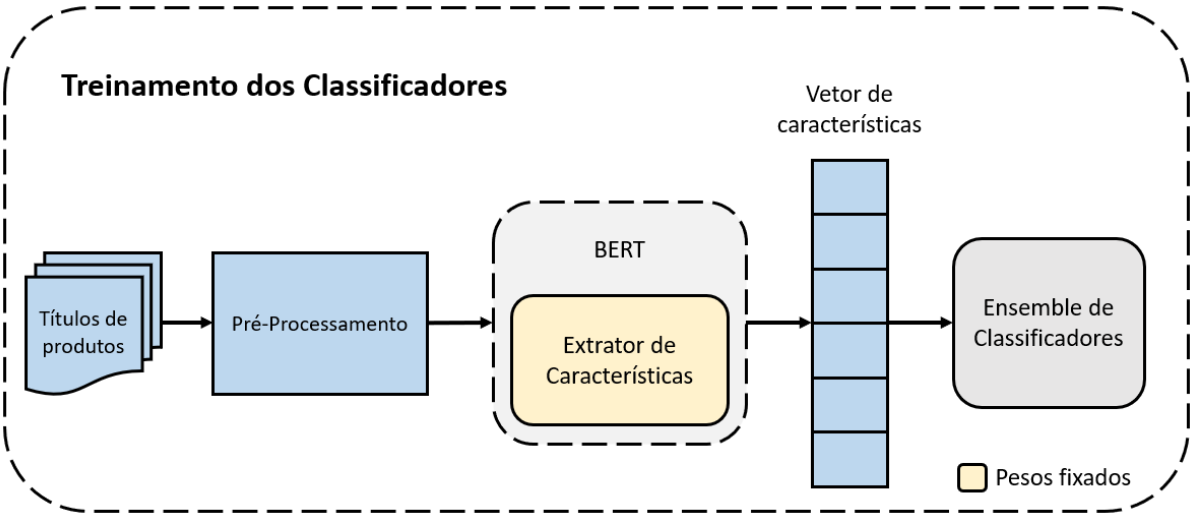
3.4 Treinamento e inferência de modelos

O sistema proposto é treinado em modo supervisionado, ou seja, além das amostras de entrada, deve-se conhecer a categoria correspondente de cada entrada. Durante o treinamento, o sistema recebe como entrada o título do produto em forma de texto livre. Primeiro, o título é pré-processado conforme explicado na Seção 3.1. Em seguida, uma *feature* R^{768} é extraída do título pré-processado pelo extrator de características, que nessa etapa tem seus pesos fixados. Por último, essas *features* são fornecidas a um classificador, no caso desse trabalho um *ensemble* de classificadores, para treinamento. Essa etapa está ilustrada na Figura 23.

Uma vez que os modelos são treinados, eles podem ser usados para atribuir uma categoria aos títulos de um determinado produto, ou seja, o modelo pode prever a qual categoria um determinado produto pertence. Nesse momento de inferência, ilustrado na Figura 24, o mesmo pré-processamento é aplicado aos títulos dos produtos. Como se utilizou *ensembles*, a previsão final é uma combinação das previsões do conjunto de indivíduos para cada modelo de classificador.

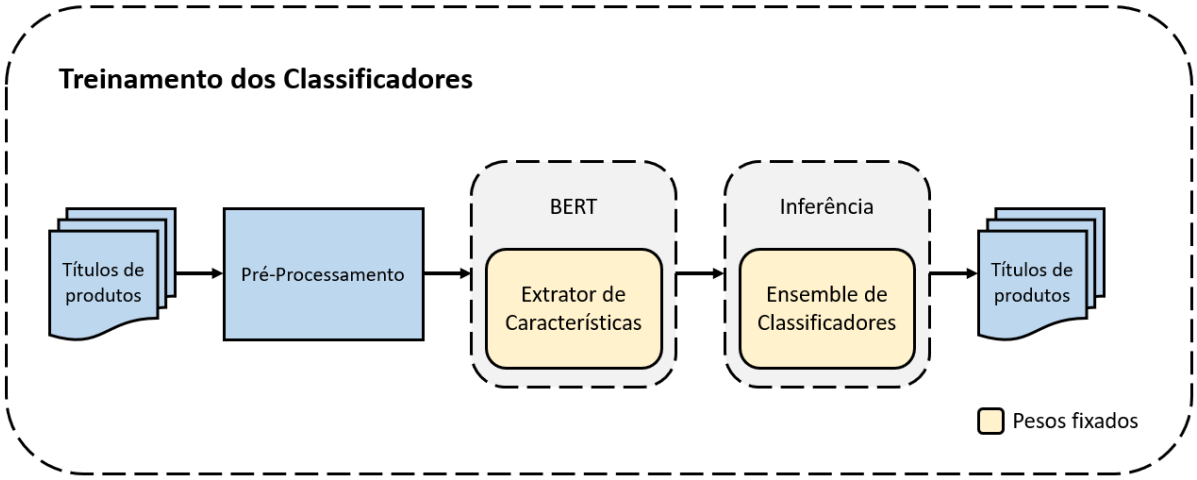
Neste trabalho, os modelos são combinados com base na adição das confianças previstas (ver Equação 3.1). Este vetor possui C posições. Cada posição i representa a confiança de um título de produto pertencer à i-ésima classe. Em seguida, todas as confianças produzidas por cada modelo são somadas para formar o *ensemble*. No final, a

Figura 23 – Ilustração da etapa de treinamento dos classificadores. Nessa etapa, o extrator de características, que já está treinado e com seus pesos congelados, é responsável por gerar as *features* dos títulos dos produtos que serão usadas no treinamento dos classificadores.



Fonte: Elaborado pelo autor.

Figura 24 – Ilustração da etapa de inferência. Nessa etapa, os títulos dos produtos passam pelo pré-processamento e em seguida pelo extrator de características, que está com seus pesos congelados. As *features* geradas são passadas para o *ensemble* de classificadores, que nessa etapa também tem seus pesos congelados, gerando a categoria final do produto.



Fonte: Elaborado pelo autor.

posição com o maior nível de confiança será a categoria predita.

$$\text{Categoria predita} = \arg\max_{c} \sum_{m} \{\hat{y}_c | c \in [1..C]\}^m, \tag{3.1}$$

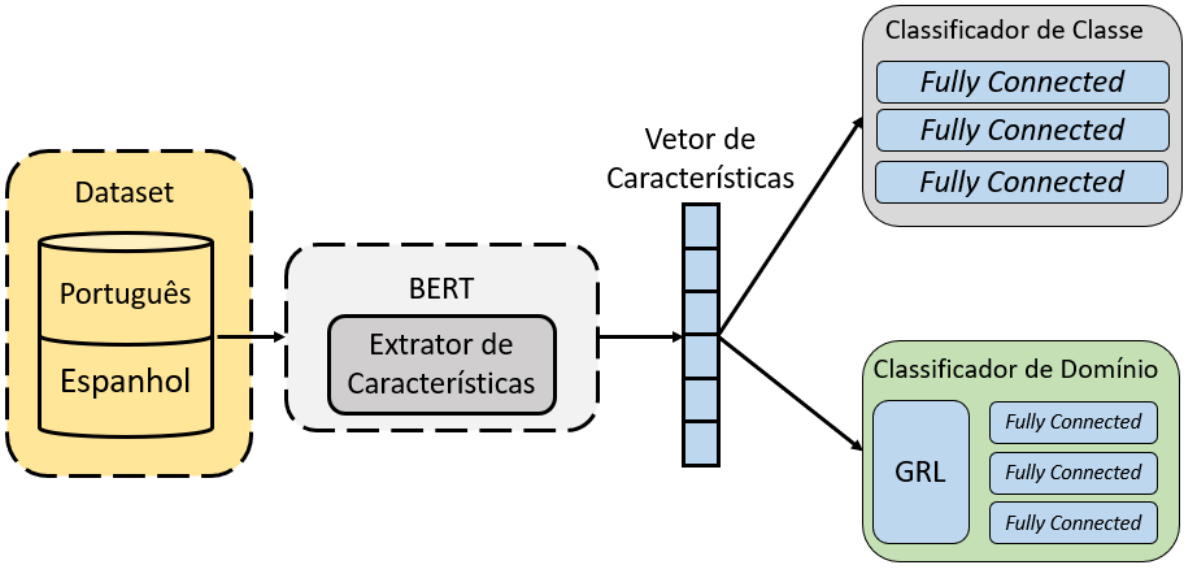
onde $\{\hat{y}_c\}^m$ são as confianças previstas pelo m-ésimo modelo para as C classes dadas o título de um produto.

3.5 Adaptação de Domínio

O sistema proposto está ilustrado na Figura 25, ele é treinado utilizando a mesma ideia apresentada na Seção 2.4, onde apenas os dados da fonte são conhecidos.

Durante o treinamento, o sistema recebe como entrada os títulos dos produtos, já pré-processados conforme descrito na Seção 3.1, de ambos os domínios fonte (espanhol) e alvo (português). Em seguida, as *features* são geradas, a partir dos títulos, pelo extrator de características. Por último, as *features* do domínio de fonte são utilizadas no classificador de classe, visto que seus rótulos são conhecidos. E também são utilizadas no classificador de domínio, juntamente com as *features* geradas dos dados de alvo, que tenta identificar o domínio ao qual cada uma delas pertence. Com a ajuda do GRL, o extrator de características acaba sendo obrigado a tentar gerar *features* que sejam invariantes ao domínio dos dados. Ao final, o modelo é capaz de atribuir uma categoria ao título de um determinado produto que seja do domínio do alvo, sem que tenha sido treinado com informações dos rótulos desses dados.

Figura 25 – Ilustração da proposta de *domain adaptation*. A ideia baseia-se no GRL. O extrator de características será treinado para que consiga gerar *features* dos dados de fonte (espanhol) e do alvo (português) de forma que sejam invariantes ao domínio.



Fonte: Elaborado pelo autor.

4 Metodologia Experimental

Este capítulo tem o objetivo de apresentar toda a metodologia que foi utilizada no desenvolvimento deste trabalho. Serão apresentados: o *dataset*, as configurações experimentais, os recursos computacionais os experimentos que foram realizados e as métricas utilizadas para avaliação.

4.1 Dataset

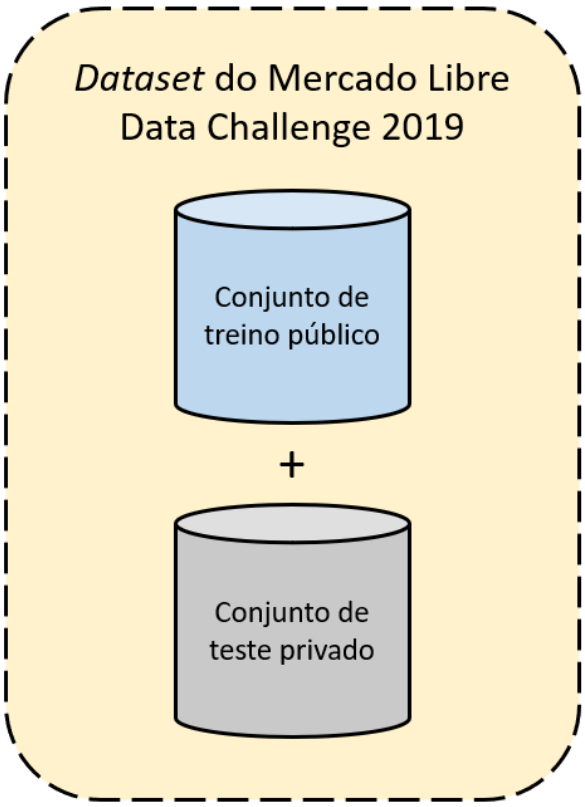
Nesse trabalho foi utilizado o conjunto de dados do Mercado Libre. Esse *dataset* foi lançado para o MeLi Data Challenge 2019 (MERCADOLIBRE, 2019)[1]. O desafio consistia, resumidamente, em treinar um modelo para que fosse capaz de categorizar corretamente anúncios de produtos baseados apenas em seus títulos. O objetivo final era fornecer uma melhor experiência ao cliente, facilitando a busca e descoberta de produtos, melhorando as recomendações, entre outras possibilidades.

Originalmente, os dados são divididos em dois conjuntos: treino e teste. A Figura 26 ilustra este *dataset*.

O conjunto de treino consiste em uma lista de 20.000.000 de produtos. Para cada produto, quatro recursos são fornecidos: o título do produto, o idioma do título, a categoria e a qualidade do rótulo. O título do produto foi fornecido pelo vendedor e está disponível sem nenhum pré-processamento, ou seja, contém pontuação, letras maiúsculas, acentuação, etc. O idioma do título pode ser espanhol ou português, pois a maior parte de seu mercado está na América Latina. A categoria é uma descrição textual única de uma das 1588 categorias do conjunto de dados, incluindo suas subcategorias (por exemplo, impressoras, impressoras 3D, lembranças, câmeras, guarda-chuvas, etc.). É importante observar que uma categoria é atribuída independentemente do idioma do título, ou seja, espera-se que o título de um produto de celular seja categorizado como "CELULARES", independentemente de o título ser em espanhol ou português. Por fim, a qualidade do rótulo é utilizada para especificar se a categoria é confiável, ou seja, verificada pela equipe do Mercado Libre, ou não confiável, cujas categorias foram escolhidas pelos vendedores e não foram revisadas pela equipe do Mercado Libre, caso em que se deve esperar uma taxa de erro maior na atribuição da categoria quando comparada aos confiáveis. No total, apenas 1.184.245 amostras ($\approx 6\%$) foram consideradas confiáveis. Os casos confiáveis, no entanto, não são uniformemente distribuídos pelas categorias, o que significa que algumas categorias têm um número grande de casos confiáveis e outras têm um número baixo (ou mesmo zero). Além

[1] Disponível em: <https://ml-challenge.mercadolibre.com/2019/>

Figura 26 – *MeLi Data Challenge Dataset*: O *dataset* é formado por um conjunto de treino e outro de teste.



Fonte: Elaborado pelo autor.

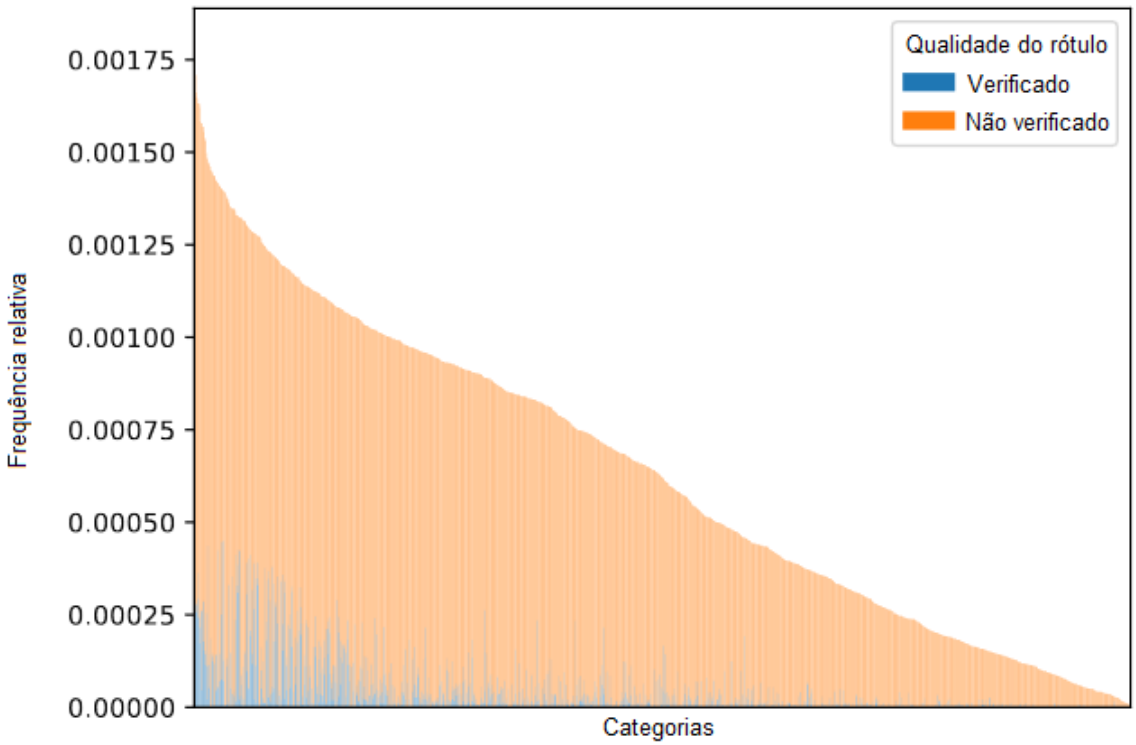
disso, o número de amostras para cada categoria também não são balanceadas. A Figura 27 mostra a distribuição do conjunto de dados em relação às categorias e confiabilidade do rótulo, em que o desequilíbrio de classes e o baixo número de amostras confiáveis podem ser observados. A Tabela 1 apresenta uma pequena análise do conjunto de treino público deste *dataset*, apresentando as seis classes com maior e menor frequência e seus respectivos quantitativos por idioma e por amostras verificadas. Vale destacar que devido ao grande quantitativo de categorias existentes no *dataset* - 1588 no total - a parcela representada por cada uma é bem pequena, sendo que todas possuem uma parcela menor que 1%.

O conjunto de teste é uma lista de 246.955 produtos, cada um contendo 3 recursos: id, título e idioma. O idioma e o título representam os mesmos do conjunto de treinamento. A coluna *id* é útil apenas para identificar cada amostra no sistema de envio privado do desafio. Os participantes do desafio foram avaliados pelo desempenho neste conjunto de teste restrito, portanto, os rótulos corretos, que foram verificados (confiáveis), não estão disponíveis publicamente.

Tabela 1 – Análise do conjunto de treino publico do *dataset* do *MeLi Data Challenge 2019*. São apresentadas informações das seis classes com maior e menor frequência no conjunto e, seus quantitativos por idioma e por amostras verificadas e não verificadas.

Categorias	Português			Espanhol			Conjunto de treino público	
	Verificados	Não verificados	Total de amostras	Verificados	Não verificados	Total de amostras	Total de amostras	Parcela do conjunto (%)
PANTS	3.995	14.694	18.689	4.025	13.259	17.284	35.973	0,1799
COFFEE MAKERS	2.832	14.775	17.607	2.635	14.862	17.497	35.104	0,1755
BABY CAR SEATS	2.747	13.543	16.290	2.897	14.976	17.873	34.163	0,1708
MUSICAL KEYBOARDS	2.557	13.098	15.655	2.603	14.964	17.567	33.222	0,1661
MATTRESSES	2.669	14.641	17.310	2.883	12.774	15.657	32.967	0,1648
PUREBRED DOGS	3.526	14.383	17.909	3.736	11.283	15.019	32.928	0,1646
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
FORCE GAUGES	0	160	160	0	53	53	213	0,0011
CONSTRUCTION LIME BAGS	0	0	0	0	206	206	206	0,0010
COLD FOOD AND DRINK VENDING MACHINES	0	56	56	0	106	106	162	0,0008
PAINTBALL SMOKE GRENADES	0	154	154	0	0	0	154	0,0008
COMMERCIAL POPCORN MACHINES	0	105	105	0	36	36	141	0,0007
HAMBURGER FORMERS	0	23	23	0	86	86	109	0,0005

Figura 27 – Distribuição do *dataset* do Mercado Livre. O *dataset* é altamente desbalanceado e somente $\approx 6\%$ são considerados confiáveis.



Fonte: Elaborado pelo autor.

4.2 Configuração Experimental

Aqui os modelos investigados e os detalhes de implementação associados a cada um são apresentados.

***Fine-tuning* do BERT:** Foi utilizado o modelo *bert-base-multilíngue* que é composto por 12 codificadores empilhados, totalizando 110 milhões de parâmetros. Este modelo foi treinado em 104 idiomas com os maiores conjuntos da Wikipédia. O código-fonte adotado está disponível publicamente (consulte a Subseção 4.3) e foi recomendado pelos próprios autores do BERT como uma alternativa à implementação original. O modelo foi ajustado com um *batch-size* de 128 e uma taxa de aprendizagem fixa de 0,001. O otimizador escolhido foi o *Stochastic Gradient Descent* (SGD) e a função de *loss* selecionada foi a entropia cruzada ponderada (*Weighted Cross-Entropy*, em inglês). Para amenizar o desbalanceamento de classes foi utilizado pesos ponderados para as classes. Os demais hiperparâmetros foram os mesmos do modelo pré-treinado.

Os modelos avaliados nos ensembles foram:

***Random Forest*:** Nos experimentos, foram utilizadas apenas 50 árvores devido à grande quantidade de memória necessária para a construção das árvores. Além do número de árvores, foram usados os valores padrões do *framework scikit-learn* para os outros hiperparâmetros, como o critério usado para a divisão do ramo, profundidade máxima, etc.

Rede Neural: Cada rede neural do conjunto de NNs é uma camada simples totalmente conectada. O modelo foi treinado com um *batch-size* de 128, com uma taxa de aprendizagem fixa de 0,001. A função de *loss* e o otimizador empregados foram a entropia cruzada e SGD, respectivamente.

Classificador de Domínio: É composto primeiramente pelo GRL e em seguida por uma rede neural contendo 3 camadas totalmente conectadas empilhadas, com 1024 neurônios cada, e sendo que entre elas existe uma camada de *batch normalization*. Possui uma saída binária indicando se a entrada é do *source* (0) ou do *target* (1). O modelo foi treinado com um *batch-size* de 512, com uma taxa de aprendizagem fixa de 0,001. A função de *loss* e o otimizador empregados foram a Entropia Cruzada Binária (em inglês, *Binary Cross Entropy*) e SGD, respectivamente.

Classificador de Classe: Composto por uma rede neural contendo 3 camadas totalmente conectadas empilhadas, com 1024 neurônios cada, e contendo uma camada de *batch normalization* entre elas, possui uma saída de dimensão C, onde C é o número de classes existentes. O modelo foi treinado com um *batch-size* de 512, com uma taxa de aprendizagem fixa de 0,001. A função de *loss* e o otimizador empregados foram a Entropia Cruzada (em inglês, *Cross Entropy*) e SGD, respectivamente. Outros detalhes, como as bibliotecas e estruturas usadas, são descritos na Subseção 4.3.

4.3 Recursos Computacionais

O modelo BERT foi treinado em um Intel Xeon X5690 @ 3,47 GHz × 24 com 32 GB de RAM e 1 GPU Titan Xp com 12 GB de memória com NVIDIA CUDA 9.1 e cuDNN 7.0 instalados. O pré-processamento do conjunto de dados e o treinamento dos modelos de aprendizado de máquina foram realizados em um Intel Xeon E7-4850 v4 (2,10 GHz) com 128 vCPU (apenas 100 foram usados) e 256 GB de RAM. O sistema operacional em execução em ambas as máquinas era Linux Ubuntu 16.04.

O *framework* PyTorch (PASZKE et al., 2019) foi adotado junto com a implementação do modelo BERT fornecido por Wolf et al. (2019), que está publicamente disponível [2]. A exceção é a *Random Forest*, na qual a implementação disponível na biblioteca *scikit-learn* (PEDREGOSA et al., 2011) foi usada.

Os treinamentos levaram, em média, 36 horas por época para o ajuste fino de BERT e 2 horas para inferência em conjuntos de teste. O treinamento e a inferência para os modelos *Random Forest* levaram, em média, 10 horas para cada modelo *Random Forest* (foram utilizados 10 modelos no *ensemble*). O conjunto de redes neurais levou, em média, 2,5 horas para treinamento e inferência. O modelo BERT treinado ocupa aproximadamente 1,5 GB de armazenamento.

4.4 Experimentos

Esta seção descreve os experimentos realizados neste trabalho. A primeira subseção apresenta o experimento relacionado ao problema de categorização de produtos. Já a segunda subseção apresenta o experimento voltado para a adaptação de domínio.

4.4.1 Categorização de Produtos

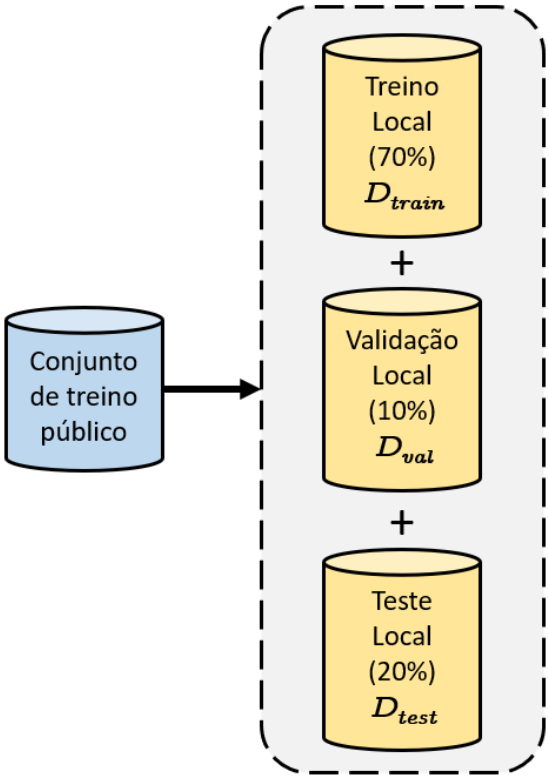
Foram realizados dois experimentos, denominados experimento local e experimento privado. No primeiro, dado que os rótulos do conjunto de teste (privado) não estão disponíveis, o conjunto de dados de treinamento público foi dividido para que se fosse feita a validação dos modelos usando um conjunto de teste *local*. No segundo, os modelos foram avaliados no conjunto de teste privado por meio de um servidor de envio. Antes da avaliação, o conjunto de dados de treinamento público do *dataset* do Mercado Libre ($\mathcal{D}$) foi subdividido aleatoriamente em treinamento ($\mathcal{D}_{train}$), validação ($\mathcal{D}_{val}$), e partições de teste ($\mathcal{D}_{test}$), cada uma compreendendo, respectivamente, 70%, 10% e 20% das amostras de $\mathcal{D}$. A Figura 28 ilustra essa partição do conjunto de treinamento público.

A divisão foi feita de forma que a distribuição de classes permanecesse a mesma de $\mathcal{D}$, o que implica que a distribuição de categorias de cada divisão é igualmente desequilibrada.

[2] <https://huggingface.co/transformers/model_doc/bert.html>

Depois disso, o extrator de características foi treinado em $\mathcal{D}_{train}$ e, ao final de cada época a acurácia balanceada foi medida em $\mathcal{D}_{val}$. O treinamento parou na época em que a acurácia balanceada da validação começou a diminuir. Depois de treinado, o desempenho deste modelo foi usado como um *baseline* para os experimentos local e privado. Além disso, este modelo foi utilizado para extrair as *features* de treinamento dos modelos nos experimentos a seguir. Por causa das limitações de memória, um procedimento diferente (particionamento de dados) foi realizado para treinar os conjuntos (Random Forest e Neural Network).

Figura 28 – Divisão do conjunto de dados públicos disponíveis para treinamento.



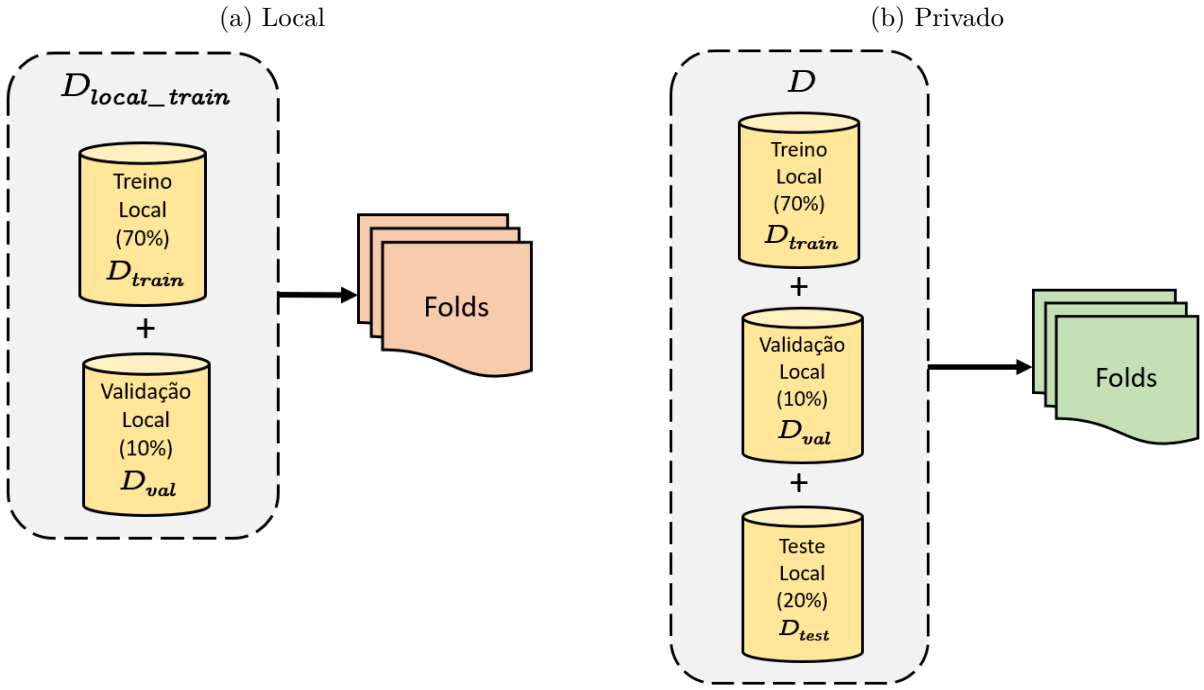
Fonte: Elaborado pelo autor.

Experimento local: Para realizar os experimentos no conjunto de *Random Forests*, 10 diferentes "*folds*" foram criados a partir de $\mathcal{D}_{local\_train} = \{\mathcal{D}_{train} \cup \mathcal{D}_{val}\}$. A Figura 29a ilustra o conjunto de dados de onde os *folds* foram criados. Cada *fold* $\mathcal{D}^{i}_{local\_train}$ foi criado com 315 amostras aleatórias de cada categoria, com um total de 500.220 amostras por *fold*. Devido à existência de amostras não confiáveis, na criação dos *folds* foi adotada a seguinte regra: as amostras para uma classe são primeiramente retiradas dos títulos confiáveis dessa classe. Uma vez que as amostras confiáveis tenham acabado e ainda não tenha sido possível formar o *fold* com 315 amostras dessa classe, o restante das amostras para essa classe são retiradas do conjunto de títulos não confiáveis dessa classe. Ou seja, durante a formação de um *fold*, as amostras para cada classe são primeiramente selecionadas de forma aleatória a partir do conjunto títulos que foram verificados para essa classe, caso uma classe não tenha 315 amostras verificadas o restante é retirado aleatoriamente do conjunto de títulos

não confiáveis (ou não verificados) dessa classe. Vale destacar que dessa forma uma mesma amostra pode compor vários folds.

Em seguida, com um extrator de características fixo, um *ensemble* de *Random Forest* foi treinado para cada *fold*. Para treinar o conjunto de redes neurais (10 no total), $\mathcal{D}^i_{local\_train}$ foi dividido em conjuntos de treinamento (90%) e validação (10%), de modo que o conjunto de validação foi usado para interromper o treinamento seguindo o mesmo protocolo usado no ajuste fino do extrator de características. As métricas de desempenho foram medidas em $\mathcal{D}_{test}$.

Figura 29 – Ilustração mostrando os conjuntos de dados utilizados na criação dos *folds* para os experimentos local (a) e privado (b).



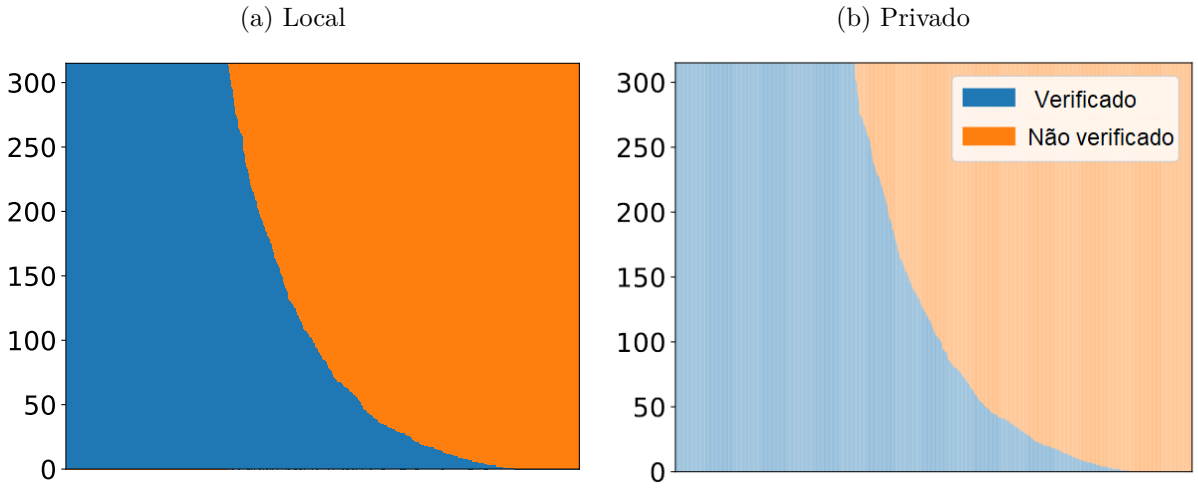
Fonte: Elaborado pelo autor.

Experimento privado: O objetivo principal deste experimento é permitir a comparação com os resultados disponíveis nos resultados finais públicos do *MeLi Data Challenge*. O experimento no conjunto de teste privado seguiu o mesmo protocolo do conjunto de teste local, exceto que os *folds* foram criados a partir de todo o $\mathcal{D}$ original em vez de $\mathcal{D}_{local\_train}$ (a Figura 29b ilustra esse conjunto) e as métricas foram medidas enviando as previsões para o servidor do Mercado Livre (ele foi reaberto após o término do desafio), para que fossem avaliadas no conjunto de teste privado, cujas categorias (*labels*) não são conhecidas, conforme já explicado na Seção 4.1.

Os *folds* também compreenderam 500.220 amostras com 315 amostras de cada categoria, no entanto, amostras mais confiáveis estavam disponíveis em comparação com o experimento local. Portanto, a distribuição dos *folds* usados no experimento local era

ligeiramente diferente (um pequeno deslocamento) dos *folds* usados no experimento privado, como pode ser visto na Figura 30.

Figura 30 – Distribuição das amostras verificadas e não verificadas entre todas categorias dos conjuntos de dados local (a) e privado (b).

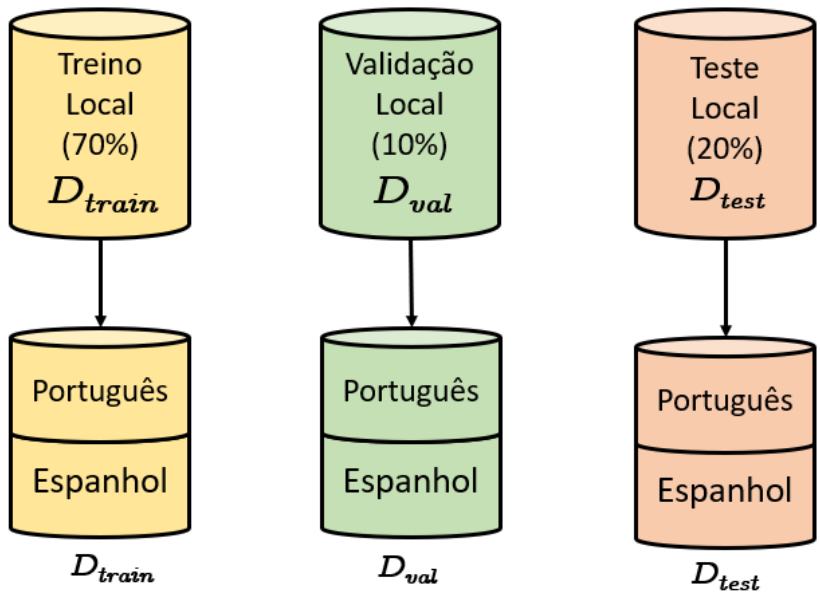


Fonte: Elaborado pelo autor.

4.4.2 Adaptação de Domínio

Como a ideia da adaptação de domínio é possuir dados cujos *labels* são conhecidos em um domínio fonte (*source*) mas não o do alvo (*target*), foi feita a separação por idioma do *dataset* do Mercado Libre. Com isso, foi escolhido como fonte os dados em espanhol e como alvo os dados em português. A Figura 31 mostra essa separação.

Figura 31 – Separação do *dataset* por idioma.



Fonte: Elaborado pelo autor.

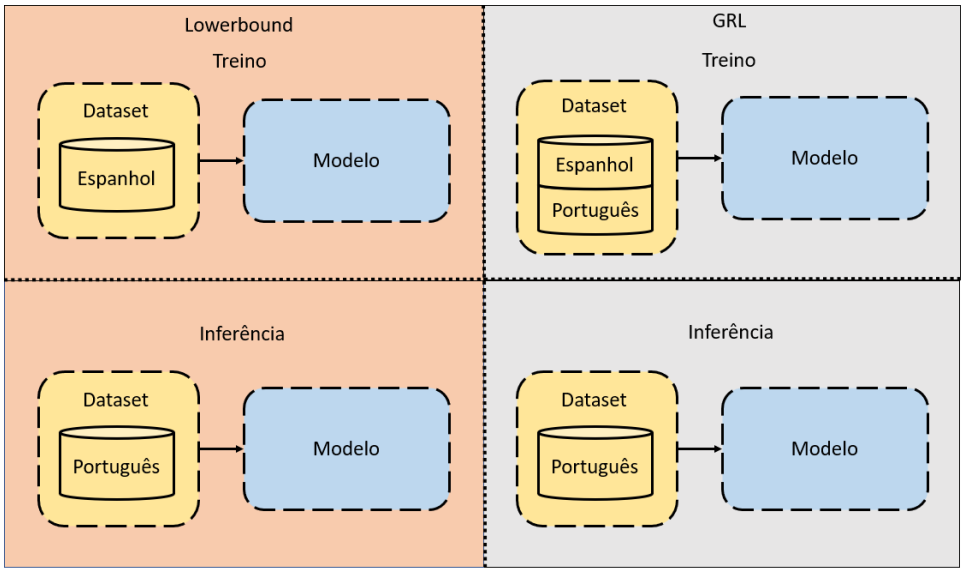
Uma vez que o *dataset* foi separado por idioma, foram realizados dois experimentos, um experimento base que será referenciado neste trabalho como *lowerbound* e outro utilizando a ideia do GRL apresentado na Seção 2.4. Ambos os experimentos foram executados 10 vezes cada para a coleta de estatísticas.

***Lowerbound*:** O objetivo desse experimento é obter um *baseline* para comparação do método de *domain adaptation* proposto. Esse experimento consiste em realizar o treinamento utilizando os dados em espanhol e seus respectivos *labels*, e realizar a inferência nos dados de teste em português. Ou seja, seria um cenário onde um modelo é treinado em um idioma específico, onde se tem os dados anotados, e se aplica esse mesmo modelo em um conjunto de dados de um outro idioma em que, a priori, ele nunca viu e não conhece as respectivas anotações.

Neste experimento, o modelo é treinado utilizando os dados em espanhol de $\mathcal{D}_{train}$. O modelo é validado ao final de cada época no conjunto de espanhol de $\mathcal{D}_{val}$ e a inferência é realizada no conjunto de português de $\mathcal{D}_{test}$.

Adaptação do GRL: Esse experimento consiste em adaptar a ideia do GRL para o problema proposto. Neste experimento, o modelo é treinado utilizando os dados em espanhol como *source*, onde os *labels* são conhecidos, e o português como *target*, onde os *labels* não são conhecidos. Dessa forma, o modelo é treinado com os dados em espanhol e português, sendo que apenas os dados em espanhol são rotulados. E, depois, realiza a inferência nos dados de teste em português. Neste experimento, o modelo é treinado utilizando todos os dados de $\mathcal{D}_{train}$. A validação é realizada ao final de cada época no conjunto de português de $\mathcal{D}_{val}$ e a inferência é realizada no conjunto de português de $\mathcal{D}_{test}$. A forma de treinamento e inferência está ilustrada na Figura 32.

Figura 32 – Treinamento e inferência realizados nos experimentos.



Fonte: Elaborado pelo autor.

4.5 Métricas de Desempenho

O objetivo final do sistema proposto é categorizar corretamente os títulos dos produtos. Nesse contexto, três métricas foram utilizadas para quantificar o desempenho do sistema.

A primeira é a métrica de acurácia e indica uma performance geral do modelo. Ela avalia o percentual de acertos do modelo, ou seja, ela é calculada pela razão do número de predições corretas pelo número total de predições. Esta métrica está definida na Equação 4.1. Pela sua simplicidade, a métrica de acurácia acaba tendo alguns problemas quando é utilizada em *datasets* muito desbalanceados, situação do *dataset* utilizado neste trabalho.

$$\text{ACC} = \frac{\text{Número de predições corretas}}{\text{Número total de predições}} \tag{4.1}$$

Visando atenuar esse problema, a segunda métrica utilizada foi a acurácia balanceada, ou recuperação macro (*macro recall*) como também é chamada, ela está definida na Equação 4.2:

$$\text{BACC} = \sum_{c=1}^{C} \frac{\text{RECALL}_c}{C}, \tag{4.2}$$

Onde:

- $\text{RECALL}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c}$;
- C é o número de categorias; e
- TP_c e FN_c são os verdadeiros positivos (*true positives*) e falsos negativos (*false negatives*) da categoria c, respectivamente.

Essa métrica foi escolhida porque além de levar em consideração o desequilíbrio das categorias, também foi utilizada no *MeLi Data Challenge*. Portanto, nossos resultados no conjunto de teste privado podem ser comparados diretamente aos do placar do desafio público.

A terceira métrica é a acurácia Top-K (Equação 4.3), onde uma previsão é contada como correta se a categoria esperada estiver entre as k maiores pontuações do vetor de saída:

$$\text{Top-K} = \frac{1}{N} \sum_{i=1}^{N} [y_i \in p_i^k], \tag{4.3}$$

Onde:

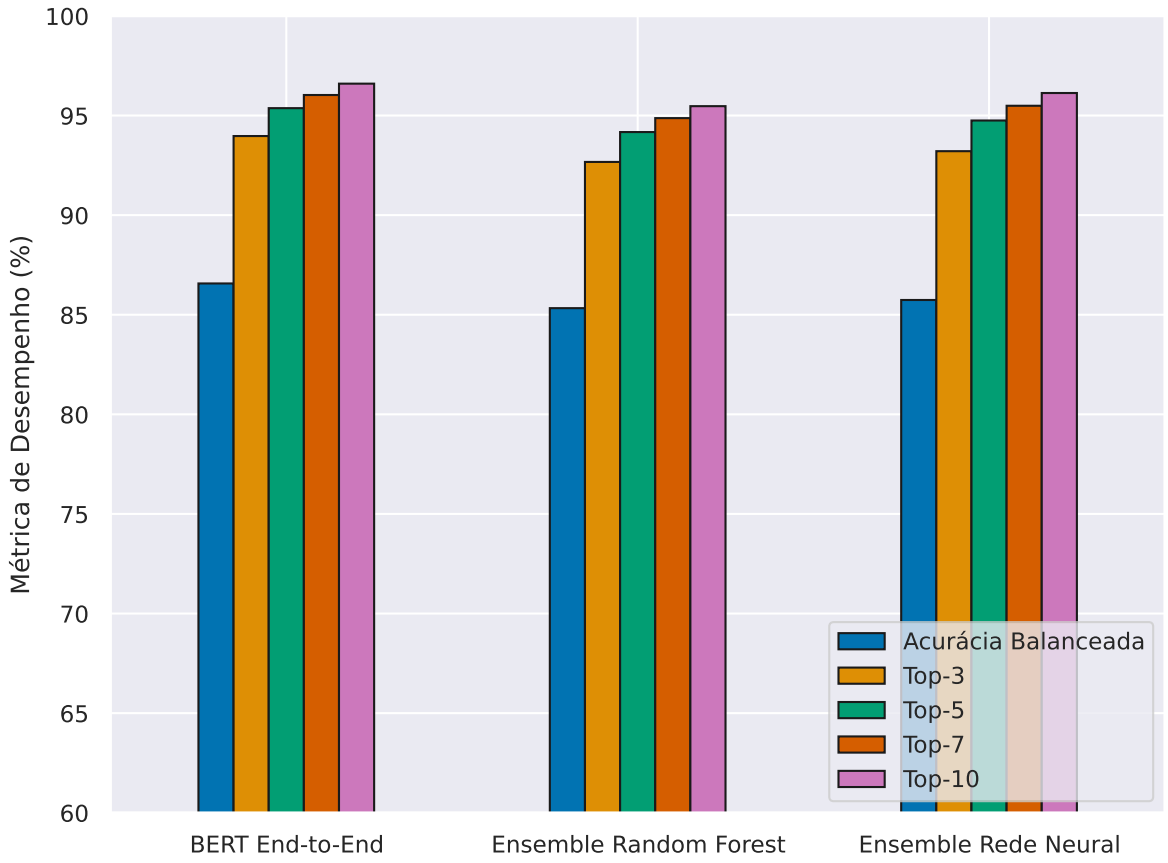
- N é o número de amostras;
- y_i é a categoria de verdade da i-ésima amostra;
- p_i^k é o conjunto das k categorias com o maiores valores de saída;
- $[\cdot]$ é o colchete de Iverson;

Quatro valores de k foram considerados: $\{3, 5, 7, 10\}$. Esta métrica é particularmente útil devido ao fato de que várias categorias são muito semelhantes, principalmente por causa da semântica da categoria-subcategoria (por exemplo, “VIDEO_GAMES” vs. “GAME_CONSOLE”, “INSTRUMENT_AMPLIFIERS” vs. “ ÁUDIO_AMPLIFIERS”, “SHIRT”vs. “T-SHIRT”). Essa métrica também é usada em outros desafios, como o Desafio ImageNet (RUSSAKOVSKY et al., 2015).

5 Resultados e Discussões

A Figura 33 mostra os resultados do experimento local. Como pode ser visto, o modelo de BERT *end-to-end* obteve o melhor desempenho considerando todas as métricas. Em segundo lugar, o conjunto de Redes Neurais obteve um desempenho ligeiramente melhor do que o modelo *Random Forest* em todas as métricas. Usando a métrica Top-3, a precisão de todos os modelos aumentou mais de 5% em comparação com a métrica de acurácia balanceada. Para as métricas Top-5, -7 e -10 (nesta ordem), no entanto, a acurácia balanceada dos modelos não aumentou muito. Isso pode ter acontecido devido ao grande desequilíbrio de classes existentes no conjunto de teste, associado ao grande número de categorias, o que pode ter feito com que algumas categorias de produtos não fossem aprendidas adequadamente.

Figura 33 – Resultados do experimento local. Todas as métricas usadas são apresentados para cada modelo.



Fonte: Elaborado pelo autor.

A Tabela 2 apresenta as cinco categorias que obtiveram os melhores e piores resultados no conjunto de teste local. É possível observar que a categoria que obteve pior

resultado foi a “READY TO DRINK COCKTAILS”, que obteve uma acurácia de 34%. Já a que obteve o melhor resultado foi a “TRAILER HITCHES” com 99% de acurácia. Nota-se ainda que a categoria “DJEMBES” obteve um excelente resultado com apenas 392 amostras no conjunto de teste, indicando que o modelo conseguindo uma excelente precisão para esta categoria. Diferentemente do que é observado para a categoria de pior resultado (READY TO DRINK COCKTAILS) onde o modelo não conseguiu obter uma boa acurácia apesar desta possuir quase o dobro de amostras para a categoria “DJEMBES”.

Tabela 2 – Tabela com informações das cinco categorias que obtiveram as melhores e piores acurácias no conjunto de teste local. São apresentadas informações da acurácia por categoria, número de amostras no conjunto de teste local e a parcela representada por cada categoria no conjunto.

Categorias	Acurácia por categoria	Conjunto de teste local	
		Total de amostras	Parcela do conjunto (%)
TRAILER_HITCHES	0,99	1.306	0,0327
CARD_PAYMENT_TERMINALS	0,98	2.000	0,0500
CYMBALS	0,98	3.893	0,0973
HYDRAULIC_VEHICLE_JACKS	0,98	3.605	0,0901
FOOTBALL_SHOES	0,98	1.751	0,0438
DJEMBES	0,98	392	0,0098
⋮	⋮	⋮	⋮
AIR_FRESHENERS	0,54	4.046	0,1012
HANDICRAFT_BOXES	0,52	3.547	0,0887
BOOKS	0,48	5.892	0,1473
KITCHEN_CABINET_ORGANIZERS	0,43	2.273	0,0568
SOUVENIRS	0,35	4.906	0,1227
READY_TO_DRINK_COCKTAILS	0,34	600	0,0150

Os resultados do experimento privado são mostrados na Tabela 3. O conjunto *Random Forest* obteve o melhor desempenho, embora tenha um desempenho apenas 1% maior do que o método de pior desempenho (BERT *end-to-end*), seguido pelo conjunto Rede Neural. A razão provável para isso é que os conjuntos usados neste experimento foram criados usando toda a divisão de treinamento público (ou seja, $\mathcal{D}_{train}$, $\mathcal{D}_{val}$ e $\mathcal{D}_{test}$), e apresenta amostras mais confiáveis em comparação com $\mathcal{D}_{train}$ (apenas 70%), que foi usado para treinar o BERT uma vez que o treinamento ocorre durante o processo de *fine-tuning*. No entanto, as *features* extraídas com o BERT permitiram que o conjunto *Random Forest* atingisse 91,190% no sistema de inscrições do Mercado Libre, superando o quarto lugar obtido no desafio, e chegando perto dos três primeiros lugares por menos de 1%, conforme mostrado na segunda coluna da Tabela 3.

Apesar da diferença observada entre os resultados obtidos em conjuntos de teste locais e privados poderem parecer estranhos à primeira vista, eles podem ter ocorrido devido à existência de amostras com categorias não confiáveis no conjunto de teste local.

Assim, algumas das previsões erradas feitas pelo modelo podem realmente ser as corretas.

Tabela 3 – Resultados obtidos utilizando a acurácia balanceada para cada um dos modelos nos experimentos locais e privados.

Modelo	**Acurácia balanceada (em %)**	
	Conjunto de teste local	**Conjunto de teste privado**
BERT (*end-to-end*)	86.57	90.19
Ensemble Random Forest	85.33	91.19
Ensemble Rede Neural	85.74	90.89
Rank Final Público do Desafio		
• Top-1	–	91.73
• Top-4	–	91.04

Fonte: Elaborado pelo autor.

Para comparação, o trabalho que obteve o primeiro lugar no desafio do *Mercado Libre* utilizou *ensembles* de diferentes modelos, sendo que cada modelo combinava arquiteturas diferentes, como por exemplo: LSTM+GRU, BiLSTM+GRU, BiLSTM+CNNs, entre outras. Vale destacar ainda que o vencedor treinou os diferentes *ensembles* para cada um dos idiomas, ou seja, para cada idioma (português ou espanhol) o modelo utilizava um *ensemble* que foi treinado especificamente para aquele respectivo idioma. Assim, por exemplo, um título em português era processado pelo *ensemble* treinado apenas com títulos em português, e de forma análoga, um título em espanhol era processado pelo *ensemble* treinado apenas com títulos em espanhol. Essa ideia é diferente da apresentada neste trabalho, uma vez que, aqui, o mesmo *ensemble* processa os títulos tanto de espanhol quanto de português, não havendo *ensembles* distintos para cada idioma.

A Tabela 4 ilustra exemplos do conjunto de teste local, que foram verificados pelo time do Mercado Libre, onde o modelo errou a categorização. Analisando os exemplos é possível notar alguns casos em que apesar do modelo ter errado a categoria predita foi bem próxima e, até poderia ser considerada correta, como por exemplo os itens 1, 10, 11, 13. Esses casos ilustram a questão apresentada na Seção 4.1, sobre a grande quantidade de categorias e subcategorias. Também é possível notar casos onde o modelo realmente realizou uma predição errada, como por exemplo as linhas 7, 9, 12.

Já a Tabela 5 apresenta exemplos de categorizações erradas feitas pelo modelo para dados não verificados pelo time do Mercado Libre. É possível observar casos onde a categoria informada parece não ser a correta (ou a melhor), como os casos apresentados nas linhas 1, 9, 12 e 13. Também é possível identificar casos onde a categoria predita é bem próxima da correta como apresentado nas linhas 2, 3, 11 e 15. Por fim, as Tabelas 6 e 7 apresentam os casos em que o modelo acertou a categoria do produto, para títulos verificados e não verificados, respectivamente.

Tabela 4 – Exemplos de classificações de títulos verificados pelo Mercado Libre onde o modelo errou a classificação.

	Título Original	Categoria informada	Categoria predita
1	Amplificador Condor Gx10 Bivolt - Lacrado/notafiscal/garanti	INSTRUMENT AMPLIFIERS	AUDIO AMPLIFIERS
2	Cascos Motos Just 1 Carbono J12 Dominator Naranja Cross	MOTORCYCLE HELMETS	MOTORCYCLE SUITS
3	Filtro Ar Condicionado Bmw 323i 2.5 Touring E36	CAR DISTRIBUTOR CAPS	CABIN FILTERS
4	Samsung Syncmaster Ta550 (monitor Tv)	TELEVISIONS	COMPUTER MONITORS
5	Toalhinha Felpuda Bordado Snoopy Tam. Visita	SOUVENIRS	WET BABY WIPES
6	Kit 10 Camisetas Básica Slim Fit Camisa Lisa 100% Algodão!!	T SHIRTS	SHIRTS
7	Óculos Bebê Menino Menina Silicone Flexível De 0 até 02 Anos	GLASSES FRAMES	BABY SWIMWEAR
8	Kit Festa Ursa Princesa Rosa	SOUVENIRS	DISPOSABLE PARTY KITS
9	Xarope Antigripal Mel Agriao Kit 42 Unidades Atacado	SUPPLEMENTS	HONEY
10	Notebook Msi Gamer Gs43vr Macbook	ULTRABOOKS	NOTEBOOKS
11	Bebida Uisque Jack Daniels 1 Litro	READY TO DRINK COCKTAILS	WHISKEYS
12	Maquina De Conocer A Pedal Cleve	SEWING MACHINES	BICYCLE PEDALS
13	Kit Seguridad Hd Dvr 4 Camaras Hdmi Exterior Sop Disco 3tb	SURVEILLANCE CAMERAS	SURVEILLANCE MONITORING KITS
14	Vestido Roxo Saída Praia Banho	DRESSES	SARONGS
15	Vino Eugenio Bustos	READY TO DRINK COCKTAILS	WINES

Fonte: Elaborado pelo autor.

Tabela 5 – Exemplos de classificações de títulos não verificados pelo Mercado Libre onde o modelo errou a classificação.

	Título Original	Categoria informada	Categoria predita
1	20 Pares De Conectores Cable Panel Solar Masculino / Femenin	FLOOD LIGHTS	AUDIO AND VIDEO CONNECTORS
2	Organizadora Porta Joias - Bijuterias !! Seja Chic !!	JEWELRY BOXES	JEWELRY DISPLAYS
3	Antiguo Destornillador Celestal / 5	SCREWDRIVERS SETS	MANUAL SCREWDRIVERS
4	Luxca 2 Layers 3 In 1 Protection Cover For Lg Stylo 4 Phone	CELLPHONE TABLET AND GPS SCREEN PROTECTORS	CELLPHONE COVERS
5	Kit Corona Piton Bmw Gs800 Z45/17z 08/10	MOTORCYCLE TRANSMISSION CROWNS	MOTORCYCLE TRANSMISSION KITS
6	Régua Divisória P/ Gaveteiro Ga3 Magus Com 10 Peças Pequena	DECORATIVE BOXES	UNDERWEAR ORGANIZERS
7	Jaqueta De Treino Original Flamengo adidas Preta 2015 2016	FOOTBALL KITS	FOOTBALL JACKETS
8	Compressor Ar Portátil Caneta Bomba Do Spray Mini Tamanho D	AIRBRUSHES	ELECTRIC AIR PUMPS
9	Wool Camp Blanket Stadium Throw Marron Tartun Cabina Alfombr	YOGA MATS	SLEEPING BAGS
10	Durável Survivalist Martelo Multitool Camping Companheiro 9	MANUAL HAMMERS	FITNESS TRAMPOLINES
11	Tensiometro Homedics	BLOOD PRESSURE MONITORS	SPHYGMOMANOMETERS
12	Candy Bar Saludable Personalizado Para 40 Chicos	SOUVENIRS	CANDIES
13	Flamingo Em Cerâmica Fina Resistente Decoração Ousada Rosa	DECORATIVE VASES	STATUES
14	Tampinha Latinha Cerveja	BEERS	BEER BREWING EQUIPMENTS
15	Naxtop 2.5d Protetor De Tela De Vidro Temperado Filme Fronta	CELLPHONE AND TABLET CASES	CELLPHONE TABLET AND GPS SCREEN PROTECTORS

Fonte: Elaborado pelo autor.

Tabela 6 – Exemplos de classificações de títulos verificados pelo Mercado Libre onde o modelo acertou a classificação.

	Título Original	Categoria informada	Categoria predita
1	Luminária Solar 60w Publica Led Poste Sensor Controle S Juro	FLOOD LIGHTS	FLOOD LIGHTS
2	Marcador De Paintball Tm 15 Le + Rotor R1	PAINTBALL MARKERS	PAINTBALL MARKERS
3	Play Station 2 + Volante Hooligans.	GAME CONSOLES	GAME CONSOLES
4	Carrinho Bebe Gemeos	BABY STROLLERS	BABY STROLLERS
5	Potencia Jahro C600 Monoblock (similar Sound Magus Dk600)	AUTOMOTIVE AMPLIFIERS	AUTOMOTIVE AMPLIFIERS
6	Barra Estabilizadora S10 Lt 2016 Abcdesmonte	SWAY BARS	SWAY BARS
7	Hoverboard Scooter 10 Plgds Alpha Board Com Led E Bluetooth	HOVERBOARDS	HOVERBOARDS
8	Tocadisco Pasta Radio Rca Victor Subasta	TURNTABLES	TURNTABLES
9	Reloj De Mesa En Porcelana Decoraciones Florales Funciona	TABLE CLOCKS	TABLE CLOCKS
10	Capacete Moto Articulado Helt Branco Azul Vermelho	MOTORCYCLE HELMETS	MOTORCYCLE HELMETS
11	Panela De Pressão Valência 6 Litros Vermelha Tramontina	KITCHEN POTS	KITCHEN POTS
12	Calça Jeans Feminina Cintura Alta Barata Com Elastano Ref. 9	PANTS	PANTS
13	Cartão Sdhc Sandisk Extreme Classe 10 32gb 90mb/s 600x 4k	MEMORY CARDS	MEMORY CARDS
14	Lectora Grabadora De Dvd Notebook Lenovo Oferta Envios	DVD RECORDERS	DVD RECORDERS
15	Tapete Capacho 120x60 Churrasqueira + Frete Grátis	CARPETS	CARPETS

Fonte: Elaborado pelo autor.

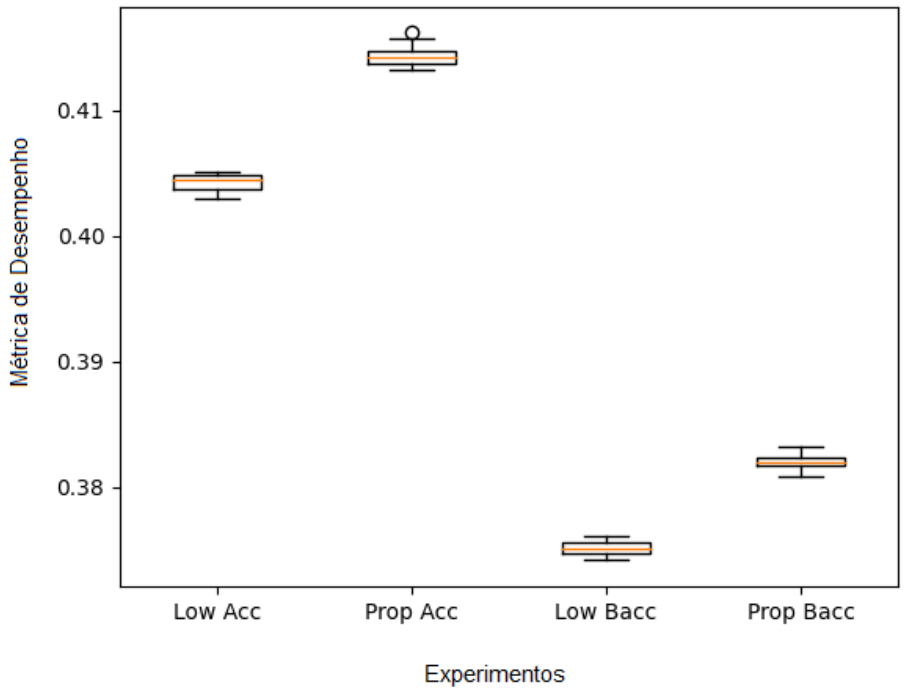
Tabela 7 – Exemplos de classificações de títulos não verificados pelo Mercado Libre onde o modelo acertou a classificação.

	Título Original	Categoria informada	Categoria predita
1	Filtro Ar Bonanza 1984/1990 Sar3589	AUTOMOTIVE AIR FILTERS	AUTOMOTIVE AIR FILTERS
2	Gatito Lunchera Neoprene	LUNCHBOXES	LUNCHBOXES
3	Condensador Bosch Vw Fusca 1.300 De 1968 à 1974	CAR AC CONDENSERS	CAR AC CONDENSERS
4	Touca Porco Com Cachecol E Luva	HAIRDRESSING CAPS	HAIRDRESSING CAPS
5	Maleta Para Guardar E Transportar Medicamentos – Promoção	MAKEUP TRAIN CASES	MAKEUP TRAIN CASES
6	15 Lanterna Cabeça Triplo T6 3led Cree Profissional Tática	HEADLAMPS	HEADLAMPS
7	Set De Latas Mate - Modelo Alicia -oliverta- Gemma Deco	FOOD STORAGE CONTAINERS	FOOD STORAGE CONTAINERS
8	Apc 5litr Ternnova Limpiador Multiproposito Tapizado Y Motor	CLEANING CLOTHS	CLEANING CLOTHS
9	Roupa De Mergulho Em Neoprene 5mm Cressi Medas Lady	WETSUITS	WETSUITS
10	Cruza Palabras Juego De Mesa Palabras Cruzadas Fichas Madera	BOARD GAMES	BOARD GAMES
11	Andador Celeste Usado Buddy	BABY WALKERS	BABY WALKERS
12	Home Theater Lg Novissimo	HOME THEATERS	HOME THEATERS
13	Mini Ems Dezenas Elétrica Massageador Máquina 5 Modos Muscle	PORTABLE ELECTRIC MASSAGERS	PORTABLE ELECTRIC MASSAGERS
14	Pulseira Para Mormaii Mo1011 - Qualidade Garantida	WATCH BANDS	WATCH BANDS
15	Practicuna Piedi	BABY PLAYARDS	BABY PLAYARDS

Fonte: Elaborado pelo autor.

A Figura 34 mostra os resultados do experimento de adaptação de domínio. Como se pode observar, o método proposto (resultados Prop Acc e Prop Bacc) conseguiu obter resultados melhores do que o *lowerbound* (resultados Low Acc e Low Bacc), tanto na métrica de acurácia (Acc) quanto na de acurácia balanceada (Bacc). Apesar da melhora observada ser pequena, cerca de aproximadamente 1%, a estratégia conseguiu superar o de um cenário normal, que é o representado pelo *lowerbound*. Isso é algo interessante pois, atualmente, grande parte dos *datasets* existentes são na sua maioria em inglês. O que dificulta a aplicação de modelos em outras linguagens, visto que a maioria dos modelos são muito pesados e demoram para treinar (são "caros"), além de necessitarem de grandes quantidades de dados anotados. Esse resultado indica que podem ter formas de se conseguir melhores resultados em um domínio cujos *labels* não são conhecidos usando dados de um domínio (*source*) onde se conhecem esses rótulos. O que traria um grande avanço na aplicação de modelos de PLN em dados de diferentes idiomas que não estão anotados. Vale ressaltar que os idiomas português e espanhol são um pouco parecidos, o que pode ser um fator favorável nesta tarefa de adaptação de domínio.

Figura 34 – Resultados dos experimentos de adaptação de domínio. Os experimentos "Low Acc" e "Low Bacc" são os resultados obtidos para o *lowerbound*, usando as métricas de acurácia e acurácia balanceada, respectivamente. Os experimentos "Prop Acc" e "Prop Bacc" são os resultados obtidos para o método proposto de adaptação de domínio para as métricas de acurácia e acurácia balanceada, respectivamente.



Fonte: Elaborado pelo autor.

6 Conclusões e Trabalhos Futuros

A classificação de texto é um grande desafio no domínio da PLN. Apesar da grande quantidade de dados disponíveis, principalmente na internet, a abordagem manual é inviável, o que torna a automação de tais tarefas extremamente relevante. Nesse contexto, esse trabalho propôs um sistema de categorização de produtos que prevê uma categoria com base apenas no título do produto. O sistema emprega um modelo de rede neural profunda, o BERT, para extrair características de títulos que são utilizadas para treinar modelos de aprendizado de máquina que também são comparados com o BERT *end-to-end* (*baseline*).

O sistema de classificação proposto foi avaliado em um conjunto de dados do mundo real em grande escala com mais de 20 milhões de amostras, o conjunto de dados Mercado Libre, lançado como parte do *MeLi Data Challenge*. No conjunto de testes local, o conjunto de Redes Neurais teve um desempenho melhor do que o da *Random Forest*. Este último, no entanto, obteve os melhores resultados no conjunto de teste privado: 91,19% de precisão balanceada. Este desempenho é melhor que o quarto lugar no ranking público e tem menos de 1% de diferença para o vencedor. Os resultados mostram que o modelo BERT foi capaz de extrair características relevantes dos títulos dos produtos permitindo que outros modelos de aprendizado de máquina obtivessem um desempenho comparável ao BERT *end-to-end*.

Além disso, foi proposta a utilização de uma técnica de adaptação de domínio visando a obtenção de um modelo que fosse capaz de classificar produtos em um idioma diferente do que foi treinado inicialmente. Os resultados obtidos mostraram que o método proposto conseguiu superar o de um cenário normal (*lowerbound*) por cerca de 1%. Apesar da melhora ser pequena, ela pode indicar que é possível conseguir treinar um modelo em um idioma, onde se tem as anotações, e aplicá-lo em outro onde estas não são conhecidas, o que pode trazer um avanço na aplicação de modelos em idiomas que possuem uma deficiência de dados anotados.

Trabalhos futuros incluem uma investigação mais abrangente de técnicas adicionais de pré-processamento de PLN, como *stemming* e lematização, bem como do uso de outros *embeddings* de palavras pré-treinadas, como fastText (GRAVE et al., 2018) e GloVe (PENNINGTON; SOCHER; MANNING, 2014), que são *embeddings* bem conhecidos e utilizados na literatura. Sugere-se também que se realizem experimentos utilizando somente as categorias com a maior quantidade de amostras confiáveis e, também, utilizando outras métricas como *coverage*. Como o custo computacional para se trabalhar com esses modelos maiores (BERT, GPT, etc.) é elevado, estudos futuros podem investigar

uma forma de diminuir a quantidade de parâmetros do modelo sem que se tenha uma perda significativa nos resultados, na literatura essa técnica é conhecida como *knowledge distillation* (HINTON et al., 2015). Outros trabalhos também incluem um estudo de outros métodos utilizados no âmbito de problemas de adaptação de domínio que possam ser aplicados no contexto de PLN. Por fim, outros estudos podem verificar como melhorar os *embeddings* do modelo através do uso de outras abordagens como *metric learning* (por exemplo, redes siamesas (CHOPRA; HADSELL; LECUN, 2005), *triplets* (HOFFER; AILON, 2015), *constellation loss* (MEDELA; PICON, 2019)).

Referências

ADEVA, J. J. G.; BERESI, U.; CALVO, R. Accuracy and diversity in ensembles of text categorisers. *CLEI Electronic Journal*, v. 9, n. 1, p. 1–12, 2005. Citado na página 53.

AGARAP, A. F. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018. Citado na página 38.

ARRUDA, V. F. et al. Cross-Domain Car Detection Using Unsupervised Image-to-Image Translation: From Day to Night. In: *International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2019. Citado na página 47.

BERRIEL, R. et al. Budget-Aware Adapters for Multi-Domain Learning. In: *The IEEE International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2019. Citado na página 48.

BERRIEL, R. F. et al. Monthly Energy Consumption Forecast: A Deep Learning Approach. In: *International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2017. Citado na página 48.

BERRIEL, R. F. et al. Automatic Large-Scale Data Acquisition via Crowdsourcing for Crosswalk Classification: A Deep Learning Approach. *Computers & Graphics*, v. 68, p. 32–42, 2017. ISSN 0097-8493. Citado na página 47.

BROWN, G. et al. Diversity creation methods: a survey and categorisation. *Information fusion*, Elsevier, v. 6, n. 1, p. 5–20, 2005. Citado na página 53.

CEVAHIR, A.; MURAKAMI, K. Large-scale Multi-class and Hierarchical Product Categorization for an E-commerce Giant. In: *Proceedings of the International Conference on Computational Linguistics (COLING) Technical Papers*. [S.l.: s.n.], 2016. p. 525–535. Citado na página 49.

CHEN, J. et al. Feature Selection for Text Classification with Naïve Bayes. *Expert Systems with Applications*, v. 36, n. 3, p. 5432–5435, 2009. Citado na página 47.

CHOPRA, S.; HADSELL, R.; LECUN, Y. Learning a similarity metric discriminatively, with application to face verification. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. [S.l.: s.n.], 2005. v. 1, p. 539–546 vol. 1. Citado na página 76.

CHOWDHURY, G. G. Natural Language Processing. *Annual Review of Information Science and Technology*, v. 37, n. 1, p. 51–89, 2003. Citado na página 25.

COLLOBERT, R. et al. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research (JMLR)*, v. 12, n. Aug, p. 2493–2537, 2011. Citado na página 48.

DENG, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, IEEE, v. 29, n. 6, p. 141–142, 2012. Citado na página 37.

DEVLIN, J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. [S.l.: s.n.], 2019. Citado 8 vezes nas páginas 26, 31, 41, 43, 46, 47, 48 e 53.

DUMAIS, S. et al. Inductive Learning Algorithms and Representations for Text Categorization. In: *Conference on Information and Knowledge Management (CIKM)*. [S.l.: s.n.], 1998. p. 148–152. Citado na página 47.

FUHR, N. et al. AIR/X - a Rule-Based Multistage Indexing System for Large Subject Fields. In: *RIAO'91: Intelligent Text and Image Handling – Volume 2*. [S.l.: s.n.], 1991. p. 606–623. Citado na página 47.

GANIN, Y.; LEMPITSKY, V. Unsupervised domain adaptation by backpropagation. In: PMLR. *International conference on machine learning*. [S.l.], 2015. p. 1180–1189. Citado 2 vezes nas páginas 27 e 42.

GASHLER, M.; GIRAUD-CARRIER, C.; MARTINEZ, T. Decision tree ensemble: Small heterogeneous is better than large homogeneous. In: IEEE. *2008 Seventh International Conference on Machine Learning and Applications*. [S.l.], 2008. p. 900–905. Citado na página 53.

GRAVE, E. et al. Learning Word Vectors for 157 Languages. In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*. [S.l.: s.n.], 2018. Citado na página 75.

HENDRYCKS, D.; GIMPEL, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. Citado na página 38.

HINTON, G. et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, v. 2, n. 7, 2015. Citado na página 76.

HO, T. K. Random decision forests. In: IEEE. *Proceedings of 3rd international conference on document analysis and recognition*. [S.l.], 1995. v. 1, p. 278–282. Citado na página 34.

HOFFER, E.; AILON, N. Deep metric learning using triplet network. In: SPRINGER. *International workshop on similarity-based pattern recognition*. [S.l.], 2015. p. 84–92. Citado na página 76.

HUTCHINS, J. The history of machine translation in a nutshell. *Retrieved December*, v. 20, n. 2009, p. 1–1, 2005. Citado na página 30.

INDURKHYA, N.; DAMERAU, F. J. *Handbook of natural language processing*. [S.l.]: CRC Press, 2010. v. 2. Citado na página 33.

JOACHIMS, T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In: *European Conference on Machine Learning (ECML)*. [S.l.: s.n.], 1998. p. 137–142. Citado na página 47.

KIM, Y. Convolutional Neural Networks for Sentence Classification. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [S.l.: s.n.], 2014. p. 1746–1751. Citado na página 48.

KOZAREVA, Z. Everyone Likes Shopping! Multi-class Product Categorization for e-Commerce. In: *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. [S.l.: s.n.], 2015. p. 1329–1333. Citado na página 48.

KUNCHEVA, L.; WHITAKER, C. Measures of diversity in classifier ensembles machine learning 51. 2003. Citado na página 53.

LAI, S. et al. Recurrent Convolutional Neural Networks for Text Classification. In: *Twenty-ninth Conference on Artificial Intelligence (AAAI)*. [S.l.: s.n.], 2015. Citado na página 48.

LEWIS, D. D.; RINGUETTE, M. A Comparison of Two Learning Algorithms for Text Categorization. In: *Third Annual Symposium on Document Analysis and Information Retrieval*. [S.l.: s.n.], 1994. p. 81–93. Citado na página 47.

LI, C.; ZHAN, G.; LI, Z. News Text Classification Based on Improved Bi-LSTM-CNN. In: *IEEE International Conference on Information Technology in Medicine and Education (ITME)*. [S.l.: s.n.], 2018. p. 890–893. Citado na página 48.

LOPES, A. T. et al. Facial expression recognition with convolutional neural networks: Coping with few data and the training sample order. *Pattern Recognition*, v. 61, p. 610 – 628, 2017. ISSN 0031-3203. Citado na página 48.

MAHMUD, M. et al. Applications of deep learning and reinforcement learning to biological data. *IEEE transactions on neural networks and learning systems*, IEEE, v. 29, n. 6, p. 2063–2079, 2018. Citado na página 47.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 37.

MEDELA, A.; PICON, A. Constellation loss: Improving the efficiency of deep metric learning loss functions for optimal embedding. *arXiv preprint arXiv:1905.10675*, 2019. Citado na página 76.

MERCADOLIBRE. *MeLi Data Challenge 2019 - Mercadolibre*. 2019. Disponível em: <https://ml-challenge.mercadolibre.com/>. Citado na página 57.

PAIXãO, T. M. et al. Fast(er) Reconstruction of Shredded Text Documents via Self-Supervised Deep Asymmetric Metric Learning. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2020. Citado na página 47.

PASZKE, A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: *Advances in Neural Information Processing Systems (NeurIPS))*. [S.l.: s.n.], 2019. p. 8024–8035. Citado na página 61.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research (JMLR)*, v. 12, p. 2825–2830, 2011. Citado na página 61.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. GloVe: Global Vectors for Word Representation. In: *Empirical Methods in Natural Language Processing (EMNLP)*. [S.l.: s.n.], 2014. p. 1532–1543. Citado na página 75.

RADFORD, A. et al. Improving language understanding by generative pre-training. 2018. Citado 2 vezes nas páginas 31 e 41.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na página 37.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *nature*, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986. Citado na página 39.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015. Citado na página 67.

SCHUSTER, M.; NAKAJIMA, K. Japanese and korean voice search. In: IEEE. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2012. p. 5149–5152. Citado na página 44.

SHARMA, A. R.; KAUSHIK, P. Literature Survey of Statistical, Deep and Reinforcement Learning in Natural Language Processing. In: IEEE. *International Conference on Computing, Communication and Automation (ICCCA)*. [S.l.], 2017. p. 350–354. Citado na página 25.

TAN, S. Neighbor-weighted K-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, v. 28, n. 4, p. 667–671, 2005. Citado na página 47.

TORRES, L. T. et al. Effortless Deep Training for Traffic Sign Detection Using Templates and Arbitrary Natural Images. In: *International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2019. Citado na página 47.

VASWANI, A. et al. Attention is all you need. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2017. p. 5998–6008. Citado 2 vezes nas páginas 31 e 41.

VASWANI, A. et al. Attention Is All You Need. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2017. p. 5998–6008. Citado 2 vezes nas páginas 43 e 48.

WANG, Z.; QU, Z. Research on Web text classification algorithm based on improved CNN and SVM. In: *IEEE International Conference on Communication Technology (ICCT)*. [S.l.: s.n.], 2017. p. 1958–1961. Citado na página 48.

WEI, F. et al. Empirical study of deep learning for text classification in legal document review. In: *IEEE International Conference on Big Data (Big Data)*. [S.l.: s.n.], 2018. p. 3317–3320. Citado na página 48.

WIENER, E. et al. A Neural Network Approach to Topic Spotting. In: *Symposium on Document Analysis and Information Retrieval (SDAIR)*. [S.l.: s.n.], 1995. v. 317, p. 332. Citado na página 47.

WIROJWATANAKUL, P.; WANGPERAWONG, A. Multi-Label Product Categorization Using Multi-Modal Fusion Models. *arXiv preprint arXiv:1907.00420*, 2019. Citado na página 49.

WOLF, T. et al. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv preprint arXiv:1910.03771*, 2019. Citado na página 61.

WU, Y. et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. Citado na página 44.

YANG, Y. An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval*, Springer, v. 1, n. 1-2, p. 69–90, 1999. Citado na página 47.

YANG, Y.; CHUTE, C. G. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems (TOIS)*, ACM, v. 12, n. 3, p. 252–277, 1994. Citado na página 47.

YIH, W.-t. et al. Learning discriminative projections for text similarity measures. In: *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL)*. [S.l.: s.n.], 2011. p. 247–256. Citado na página 48.

Zavarez, M. V.; Berriel, R. F.; Oliveira-Santos, T. Cross-database facial expression recognition based on fine-tuned deep convolutional network. In: *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. [S.l.: s.n.], 2017. p. 405–412. Citado na página 48.

ZHANG, X.; ZHAO, J.; LECUN, Y. Character-level Convolutional Networks for Text Classification. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2015. p. 649–657. Citado na página 48.

Apêndices

APÊNDICE A – Repositório do Projeto

Todo o código utilizado no projeto, bem como o *dataset* do Mercado Libre se encontra disponível publicamente através do link abaixo:

- <https://github.com/lspaulucio/product-categorization-ijcnn-2020>

O repositório inclui:

- os código dos modelos descritos neste trabalho;
- os pesos dos modelos treinados;
- os *datasets* do desafio do Mercado Libre;