UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Víctor Assunção Ávila

Classificando perguntas sobre produtos do comércio eletrônico quanto à característica usando algoritmos de Inteligência Artificial

Uberlândia, Brasil

2023

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Víctor Assunção Ávila

Classificando perguntas sobre produtos do comércio eletrônico quanto à característica usando algoritmos de Inteligência Artificial

Trabalho de conclusão de curso apresentado à Faculdade de Engenharia Mecânica da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção do título de Bacharel em Engenharia Mecatrônica.

Orientadora: Profa. Dra. Elaine Ribeiro de Faria Paiva

Universidade Federal de Uberlândia – UFU

Faculdade de Engenharia Mecânica

Bacharelado em Engenharia Mecatrônica

Uberlândia, Brasil

2023

Víctor Assunção Ávila

Classificando perguntas sobre produtos do comércio eletrônico quanto à característica usando algoritmos de Inteligência Artificial

Trabalho de conclusão de curso apresentado à Faculdade de Engenharia Mecânica da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção do título de Bacharel em Engenharia Mecatrônica.

Trabalho aprovado. Uberlândia, Brasil, 21 de Novembro de 2023:

Profa. Dra. Elaine Ribeiro de Faria Paiva
Orientadora

Prof. Dr. Marcelo Zanchetta do Nascimento
Membro da Banca

Rodrigo Oliveira Caus
Membro da Banca

Uberlândia, Brasil
2023

Agradecimentos

Este Projeto de Fim de Curso marca o início do fim de uma jornada cansativa e muito desafiadora. A faculdade pública molda seus estudantes em todos os aspectos possíveis. Por variadas vezes, bati de frente com as diversas dificuldades que o curso traz. Definitivamente não é fácil. No entanto, sempre reforcei para mim mesmo que desistir não é uma opção. Sempre reforcei que os meus sonhos (sonhos que criei a partir de experiências na UFU, inclusive) passam diretamente pelo Bacharelado em Engenharia Mecatrônica. Com muito esforço, tenho conseguido prosseguir nessas etapas finais.

Agradeço à minha mãe por sempre ter colocado um degrau a mais na escada da minha carreira. Mesmo quando as realidades pareciam limitadas, ela sempre abriu a porta e me mostrou que havia um caminho a mais para se seguir. E isso se refletiu em mim como uma obsessão, uma determinação infindável por atingir objetivos antes impensáveis.

Pai, vovó, Edmar: a família vai ter seu primeiro membro formado em universidade pública! Obrigado por terem cuidado tão bem de mim desde a infância, cada um da sua forma.

Um beijo pro Pedrinho também, meu irmãozinho, e que eu consiga ajudar ele a conquistar a carreira que ele quiser quando chegar a hora.

Agradeço à Madu, minha namorada, por ter sido minha companhia diária mesmo enquanto eu estava distante.

Vamos! Novas etapas virão por aí.

Resumo

Os *marketplaces* são shoppings virtuais que reúnem ofertas de produtos e serviços de diferentes vendedores. Dentre essas plataformas, o Mercado Livre se destaca pelo seu alto número de vendas no Brasil. Nos últimos anos, os brasileiros passaram a fazer cada vez mais compras *on-line* e com isso a quantidade de perguntas acerca dos produtos aumentou em larga escala, dificultando a operação dos lojistas nos *marketplaces*.

Este trabalho visa apresentar um método de classificação de texto por meio de inteligência artificial para responder as perguntas dos clientes de lojistas do Mercado Livre de forma automatizada. Para isso, dados provenientes do Mercado Livre foram coletados, rotulados, pré-processados e usados para treinar diferentes transformadores, modelos de inteligência artificial de aprendizado profundo que podem ser usados para classificar textos.

A partir dos resultados, notou-se que os modelos treinados são capazes de classificar corretamente a maioria das perguntas quanto ao atributo de produto a que elas se referem e, com isso, responder a dúvida do cliente. Também foi possível perceber que o procedimento de aglutinação de classes melhorou consideravelmente o desempenho dos modelos treinados, pois diminuiu a incidência de situações em que perguntas poderiam ser classificadas em mais de um atributo ao mesmo tempo.

Palavras-chave: Resolução de Dúvidas, Comércio Eletrônico, Classificação de Texto, Transformadores.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

A invenção da *World Wide Web* em 1989 mudou para sempre os hábitos das pessoas. A tecnologia cresceu rapidamente durante a década de 1990, e fontes como a pesquisa publicada por John Quarterman (INTERNET. . . , 1994) e a União Internacional de Telecomunicações da ONU (NUMBER. . . , 2023) relataram que o número de usuários da Internet cresceu de 2,62 milhões de pessoas em 1990 para 414 milhões de pessoas no ano 2000.

Esse cenário de rápido crescimento da Internet motivou empreendedores como Jeff Bezos a buscarem oportunidades nessa área. Inicialmente, ele criou um comércio totalmente online (ou seja, um *e-commerce*) que vendia apenas livros, atuando como intermediário entre as editoras e os leitores (STONE, 2013). Assim surgiu a *Amazon*[1]. Na mesma época, concorrentes como o *EBay*[2], plataforma que permitia que qualquer pessoa vendesse produtos (novos ou usados) num sistema de leilão (FELITTI, 2021), também se destacavam nos Estados Unidos.

Após fazerem uma pós-graduação em administração de negócios na *Stanford University*, nos Estados Unidos, dois argentinos resolveram se basear no modelo de sucesso do *EBay* e criar um site de vendas de produtos através de leilões, levando a experiência de fazer compras em sites como o *EBay* à Argentina e posteriormente ao Brasil (FELITTI, 2021). Dessa forma, o Mercado Livre[3] surgiu em 1999.

Quando os investidores dos Estados Unidos descobriram a Internet, eles a adotaram com devoção e uma bolha (chamada de Bolha PontoCom) começou a inflar. Os capitalistas de risco que investiam nos novos sites fornecedores de diferentes tipos de produtos, como o *Pets.com*, de rações para animais, não necessariamente acreditavam que a Internet era a melhor maneira de vender comida para animais de estimação, mas sabiam que, se não financiassem essas empresas, seus concorrentes iriam financiá-las (CASSIDY, 2003). Essas novas empresas eram avaliadas em milhões de dólares a mais do que corporações tradicionais, como a Sears (loja de departamentos) e a Disney. Após maus resultados, muitas dessas novas empresas faliram.

Ao observar o cenário financeiro desfavorável do ano 2000, a *Amazon* iniciou a modalidade de *marketplace*, ou seja, passou a dispor de vários pequenos comerciantes em sua plataforma. Com isso, o seu catálogo de produtos tornou-se ainda maior, cativando o foco do cliente final no *website* (STONE, 2013). Percebendo a movimentação que a *Amazon* fez para se proteger da crise, o Mercado Livre também decidiu se tornar um agregador de

[1] <https://www.amazon.com>
[2] <https://www.ebay.com/>
[3] <https://www.mercadolivre.com.br>

pequenas lojas (FELITTI, 2021), se assemelhando ao seu concorrente estadunidense.

Com o passar do tempo, grandes lojas e os seus altos volumes de vendas passaram a fazer parte do Mercado Livre. A companhia passou a investir cada vez mais para buscar a liderança no setor e os esforços têm provocado resultado: no segundo trimestre de 2022, por exemplo, o volume bruto de mercadorias foi de US$ 8,551 bilhões, o que representa um aumento de 21,7% em relação ao mesmo período do ano anterior (MercadoLibre, Inc., 2022). Ainda de acordo com o Relatório do Segundo Trimestre de 2022 do Mercado Livre, considerando apenas o Brasil, a receita em dólares aumentou 53% ponderando os mesmos períodos (MercadoLibre, Inc., 2022).

O rápido crescimento em número de clientes demonstrado pelas plataformas de *e-commerce* trouxe consigo uma demanda cada vez maior por atendimento. Ao perguntar desde questões relacionadas à compatibilidade do produto anunciado com um outro item, até questões sobre o frete e tempo de entrega do produto, os clientes demonstram precisar de orientação para concluir a compra. De fato, uma pesquisa feita nos Estados Unidos concluiu que 99% dos consumidores leem o campo de Perguntas e Respostas pelo menos ocasionalmente (Power Reviews, 2022).

Um levantamento divulgado pelo grupo Ebit (EBIT; NIELSEN, 2021) mostra que a quantia gasta em reais nos *e-commerces* disponíveis no Brasil subiu 41% entre os anos de 2019 e 2020, o que indica uma crescente adesão dos consumidores brasileiros ao *e-commerce*. Essa grande demanda por atendimento preocupa os lojistas, que querem vender sem ter custos altos na resolução de perguntas. Surge uma solução: o uso de algoritmos e modelos de aprendizado de máquina, um campo de estudo em recente popularização. Líderes de novas empresas de tecnologia da informação perceberam que grande parcela das perguntas dos clientes de *e-commerces* possuem respostas com métodos padronizados de se encontrar, pois já foram perguntadas por outras pessoas ou porque se referem a atributos daquele produto que já foram descritos pelo lojista ou pelo fabricante. Essas empresas de tecnologia da informação passaram então a fornecer serviços de inteligência artificial (THOMAS, 2022), atendendo à nova demanda do mercado.

A depender das características da pergunta feita pelo cliente, um algoritmo diferente de aprendizado de máquina deve ser usado. Em (ZUCCHI; REIS, 2021), o modelo proposto responde a uma nova pergunta com a resposta dada por um atendente real a uma pergunta similar feita anteriormente. Ele usa aprendizado de máquina para classificar pares de perguntas como similares ou não similares, onde um elemento do par é uma nova pergunta ainda não respondida e o outro elemento do par é uma pergunta respondida existente na base de dados. No entanto, o método proposto em (ZUCCHI; REIS, 2021) não é capaz de resolver o problema apresentado neste trabalho, pois para ser aplicado no problema seria preciso ter uma grande base de dados com pergunta e resposta sobre cada atributo de cada produto disponível no Mercado Livre, o que é custoso em termos

financeiros e de trabalho humano.

Em (SANT'ANNA et al., 2020), o modelo proposto responde a perguntas, usualmente sobre compatibilidade entre dois produtos, recuperando informações em um grafo de conhecimento. Ele usa aprendizado de máquina na etapa que consiste em aprimorar esse grafo de conhecimento, pois a recuperação de informações é feita a partir de pares de perguntas e respostas sobre compatibilidade escritas de forma não estruturada, e se faz necessário fornecer esses pares a um modelo de aprendizado de máquina para que sejam estruturados no formato adequado ao grafo de conhecimento (SANT'ANNA et al., 2020). Entretanto, a solução proposta não é capaz de responder as perguntas que este trabalho responde, pois ela consegue responder apenas questões relacionadas à compatibilidade entre dois produtos, e não questões relacionadas a outros atributos.

Este estudo propõe um novo método de resolução de perguntas de clientes e aplicação em um cenário real. Para isso, técnicas da área de classificação de texto serão aplicadas em uma base de dados de perguntas sobre diferentes atributos de produtos do Mercado Livre. Após aplicar técnicas de processamento de linguagem natural e classificação de texto, será possível entender a qual atributo uma determinada pergunta se refere, e assim poder respondê-la recuperando o valor do atributo em uma tabela preenchida pelo lojista.

O trabalho desenvolvido foi uma solução realizada e aplicada na empresa GoBots como estágio do curso de graduação em Engenharia Mecatrônica.

1.1 Objetivos

1.1.1 Objetivo geral

O objetivo deste trabalho é propor um método de resolução de perguntas de clientes relacionadas a atributos específicos de um determinado produto para plataformas de *e-commerce* atuantes no Brasil. Esse método é baseado em Classificação de Texto, mais especificamente Classificação de Perguntas, e será treinado a partir de uma base de dados de perguntas reais e aumentadas em Português.

Para chegar ao melhor modelo possível, foram avaliadas diferentes formas e ferramentas de pré-processamento de texto, de representação vetorial de texto (*encoding*) e de classificação de texto. Esses componentes formarão uma *pipeline* que classificará perguntas em atributos como "cor", "marca", "tamanho", entre outros. Ao final, os resultados desses experimentos em um conjunto de novas perguntas são discutidos.

1.1.2 Objetivo específico

- Construção de uma base de dados rotulada em relação aos atributos de 1419 perguntas divididas em 28 ou 40 classes, sendo estas classes os atributos mais empregados na descrição dos produtos do Mercado Livre.
- A base de dados é privada e pertence à empresa *GoBots Soluções Inteligentes LTDA*.
- Teste de dois diferentes algoritmos de classificação de texto (BERT refinado em Português e *DIETClassifier*), escolhendo o que se destaca mais com base em métricas conhecidas da literatura. A escolha dos algoritmos foi baseada em trabalhos anteriores da empresa GoBots, na qual este projeto foi desenvolvido.
- Estudo de caso da implementação da solução em um *e-commerce* específico, o Mercado Livre, importante para avaliar se a solução seria útil também em outras empresas semelhantes brasileiras de *e-commerce*.

1.2 Organização do Trabalho

Os outros capítulos deste trabalho estão organizados da seguinte forma:

- **Capítulo 2 — Revisão Bibliográfica:** aprofunda a importância da resolução de perguntas dos clientes no atual contexto do *e-commerce*, assim como explica os conceitos de Processamento de Linguagem Natural que serão usados no decorrer do trabalho. O capítulo também apresenta trabalhos relacionados considerando a aplicação ou a língua dos dados nos quais foram treinados.
- **Capítulo 3 — Método para Avaliação de Classificadores Treinados na Base de Dados do Mercado Livre:** descreve cada um dos passos percorridos para treinar os diferentes modelos de Classificação de Texto apresentados, desde a definição das classes e coleta de dados até o uso da plataforma de treinamento em si, bem como apresenta a forma como foi feita a avaliação dos modelos.
- **Capítulo 4 — Resultados:** compara as métricas encontradas para os modelos experimentados em diferentes situações, analisando pontos positivos e negativos de acordo com a capacidade deles em classificar perguntas quanto a atributos.
- **Capítulo 5 — Conclusão:** resume os resultados encontrados e as análises feitas, assim como levanta pontos a serem aprimorados em trabalhos futuros.

2 Revisão Bibliográfica

Este capítulo tem como finalidade apresentar os conceitos teóricos que serão utilizados durante o desenvolvimento do trabalho, bem como introduzir trabalhos relacionados.

A Seção 2.1 aborda o crescimento acelerado do e-commerce no Brasil, destacando o papel dos *marketplaces* como Mercado Livre e de empresas como a *GoBots* na transformação do varejo *online*. Além disso, enfatiza a importância de sistemas que respondem perguntas para melhorar a experiência do cliente e impulsionar vendas.

A Seção 2.2 discorre sobre a busca das empresas por maior automação no atendimento ao cliente e os benefícios proporcionados para as duas partes envolvidas.

A Seção 2.3 apresenta a tarefa de classificação de texto, que está relacionada ao objetivo deste trabalho, definindo como ela funciona e descrevendo possíveis aplicações. Além disso, especifica dificuldades dessa tarefa quando aplicada ao contexto de responder perguntas sobre atributos de perguntas no Mercado Livre.

A Seção 2.4 explica as variadas etapas que precisam ser seguidas para que seja implementado um modelo de Aprendizado Profundo, que vão desde as etapas que usam métodos clássicos de pré-processamento de texto na área de Processamento de Linguagem Natural (PLN) até etapas que se valem do atual estado da arte que já se consolidou na literatura, como os modelos Transformadores para classificação de texto.

A Seção 2.5 faz um levantamento de trabalhos relacionados que também implementam classificação de texto, e que são semelhantes ao proposto neste trabalho, considerando o contexto em que são utilizados ou a língua dos exemplos disponíveis em sua base de dados.

2.1 Marketplaces

Nos últimos anos, uma intensa mudança de comportamento em relação às compras tem sido vista entre os brasileiros: cada vez mais eles passaram a comprar *on-line*. Um levantamento divulgado pelo grupo Ebit (EBIT; NIELSEN, 2021), que analisa a reputação de lojas virtuais, indica que a quantia gasta em reais nos *e-commerces* disponíveis no Brasil subiu 41% entre os anos de 2019 e 2020. Além disso, 13 milhões de brasileiros realizaram a sua primeira compra via Internet durante esse período.

Ao observarem esse cenário de mudança, que se acelerou com a Pandemia de COVID-19 e continua a crescer, várias empresas do varejo intensificaram seus investimentos no Brasil. Uma modalidade de *e-commerce* que se destaca é a do *marketplace*, que de

acordo com (ROSA, 2019) pode ser definido como um shopping virtual que reúne ofertas de produtos e serviços de diferentes vendedores. Outra característica é que todo o processo de compra, da escolha do produto até o pagamento, é feito sem sair da plataforma. Essa nova modalidade permite que empresas de pequeno, médio e grande porte possam anunciar seus produtos nas plataformas dos maiores varejistas, assim como fazer uso dos seus processos de logística.

A grande concorrência de *marketplaces* existente no Brasil obriga os lojistas a pulverizarem seus anúncios em diferentes plataformas. A dinamicidade das vendas virtuais e o fato delas ocorrerem vinte e quatro horas por dia exigem ainda mais esforço desses comerciantes. Ao mesmo tempo, o cliente final bem atendido virtualmente estreita laços com o lojista, emulando uma situação de compra em ambiente físico (SINGH, 2002). Todos esses fatores criam um cenário perfeito para que apareçam outros atores em cena: as empresas fornecedoras de Inteligência Artificial como um serviço, que podem automatizar aspectos da relação entre clientes e lojistas para que estes consigam despender um tempo maior em outras questões do seu negócio.

2.1.1 Mercado Livre

O Mercado Livre[1] é uma das empresas pioneiras no ramo dos *marketplaces* na América Latina (Mercado Livre, 2020). Fundado em 1999, na Argentina, ele opera em 18 países da América Latina e se considera líder no setor de comércio eletrônico na região, tendo em vista o número de visitantes únicos e de visitas ao site.

Os números anunciados pelo Relatório do Segundo Trimestre de 2022 (MercadoLibre, Inc., 2022) da empresa ao mercado mostram que, por exemplo, o número de usuários únicos ativos subiu de 76 para 84 milhões em comparação ao mesmo trimestre do ano anterior. No mesmo período, o volume bruto de mercadorias cresceu de US$ 7,023 bilhões para US$ 8,551 bilhões.

O ecossistema fornecido pelo Mercado Livre é tão grande que, além dos lojistas e dos clientes finais, há espaço para a participação de outras entidades, como as empresas fornecedoras de *software*. A GoBots[2], empresa que possibilitou a execução deste projeto, atua fornecendo soluções de Processamento de Linguagem Natural para as lojas do Mercado Livre, e será detalhada na seção 2.1.4.

2.1.2 O Campo de Perguntas e Respostas e a sua Influência na Venda

O Campo de Perguntas e Respostas é definido por (NUAI; GHASEMKHANI; KANNAN, 2017) como um sistema de discussões na página de um produto. Um exemplo

[1] <https://www.mercadolivre.com.br/>
[2] <https://gobots.ai/>

desse tipo de sistema é o *"Amazon Answer"*, introduzido em 2012. Esse sistema permite que qualquer usuário da *Amazon* possa perguntar e responder questões disponíveis no *Marketplace*. Uma questão pode ser respondida por mais de um usuário e os participantes podem perguntar e responder sem que tenham comprado o produto. O sistema implementado, no entanto, possui variações a depender da plataforma. No Mercado Livre, por exemplo, as respostas só podem ser dadas pelos lojistas. Logo, esse tipo de sistema supre a necessidade de uma discussão cliente-vendedor análoga à aquela que acontece presencialmente.

Uma pesquisa feita pela *Power Reviews* (Power Reviews, 2022), uma empresa especializada em melhorar a relação e o *marketing* orgânico de clientes com grandes companhias como Calvin Klein e Colgate-Palmolive, entrevistou 7.528 consumidores nos Estados Unidos para entender como eles interagem com os comentários e as perguntas criadas por outros consumidores nos anúncios dos produtos. Entre as principais descobertas feitas, estão: 99% dos consumidores leem o campo de Perguntas e Respostas pelo menos ocasionalmente; 68% dessas pessoas já deixaram uma ou mais questões nesse mesmo campo para que os atendentes da loja respondam.

Esses resultados reforçam a importância crucial que dar a devida atenção a esses questionamentos possui na conclusão da venda, tanto para o cliente que está diretamente fazendo a pergunta quanto para os demais clientes que virão depois e desejam conhecer melhor o produto. Afinal, como (SINGH, 2002) descobriu, as respostas eletrônicas fornecidas possuem valor para os consumidores, e o sucesso de uma companhia é mensurado por quão efetivamente ela interage com os seus clientes.

2.1.3 Migração para o Comércio Eletrônico

Com a popularização dos *smartphones* e do acesso à *Internet* em todo o Brasil (Gov.br, 2022), tornou-se cada vez mais necessário que as vendas também passassem a ocorrer com grande intensidade no ambiente virtual. Isso provocou um crescimento progressivo do *e-commerce* no contexto brasileiro, tendo em vista que a desaceleração e as retrações ocorridas em determinados períodos foram superadas em períodos subsequentes (BAGATINI; LAIMER, 2019).

Essa migração para o comércio eletrônico se deu recentemente: o maior crescimento em número de lojas online até 2019 deu-se entre os anos de 2014 e 2015 (25%) (BAGATINI; LAIMER, 2019). E isso ocorreu em um período em que o varejo em geral estava em retração de -4%. Assim, em momentos de instabilidade e/ou crise financeira, a migração para o digital ocorre de forma mais acentuada, como visto no levantamento que abrange o primeiro ano da Pandemia de COVID-19 (EBIT; NIELSEN, 2021).

2.1.4 GoBots

A GoBots é uma *start-up* situada em Campinas - SP que atua na intermediação entre lojistas e clientes, facilitando a vida das empresas. Além de soluções que envolvem o monitoramento de vendas de diferentes plataformas de forma centralizada, a empresa é especialista na oferta de soluções de Inteligência Artificial, mais especificamente de Processamento de Linguagem Natural relacionado a dados textuais, que atuam no campo de Perguntas e Respostas. A GoBots está sendo apresentada neste projeto porque o autor implementou este trabalho na condição de estagiário da mesma.

O entendimento das perguntas dos clientes é uma tarefa complicada, e portanto exige que a GoBots aplique diferentes formas de raciocínio. Muitas delas são premiadas e já estão em prática: o uso de Grafos de Conhecimento é um mecanismo muito eficiente para responder perguntas relacionadas a peças automotivas e sua compatibilidade com diferentes modelos de carros, como demonstrado em (SANT'ANNA et al., 2020). Além disso, a pesquisa por respostas anteriormente dadas por atendentes reais a perguntas parecidas com a pergunta em questão obteve ótimos resultados, como visto em (ZUCCHI; REIS, 2021).

Existem situações em que a resposta para a pergunta do cliente está disponível de forma estruturada, em uma tabela preenchida pelo lojista, mas ele não foi capaz de encontrá-la no anúncio. Em alguns outros casos essa informação também está anotada, mas não está disponível de forma pública ao cliente final sem que seja feita uma consulta via API ao Mercado Livre. Para esses casos, usaremos a Classificação de Perguntas sobre Produtos Quanto ao Atributo, esmiuçada neste projeto.

2.2 A Necessidade de um Sistema Computacional para Responder Perguntas de Clientes

Uma das práticas mais comuns das empresas para se tornarem mais eficientes é a redução de custos, e uma das áreas que mais demandam gastos nas empresas é o atendimento (BROMBIM, 2021). Na Figura 1 está ilustrado o atendimento das empresas no campo de Perguntas e Respostas do Mercado Livre.

A quantidade de compras feitas *online* tem aumentado cada vez mais, e com isso aumenta o volume de tarefas com que as pessoas que trabalham com suporte nas empresas precisam lidar. Uma solução pra esse dilema pode ter sido encontrada: uma pesquisa feita pela Revista do Call Center (BROMBIM, 2021) descobriu que 94% das empresas concordam que a automação inteligente pode aumentar a produtividade.

Essa mudança de direção para um atendimento mais automatizado também é bem vista pelos clientes: de acordo com a empresa de consultoria Gartner, o autoatendimento

está se tornando a norma, pois os clientes esperam cada vez mais uma experiência fácil e rápida em todas as situações (Callcenter.inf.br, 2019). Em suma, responder cada vez mais perguntas dos clientes por meio de modelos de Inteligência Artificial se mostra bom para as empresas, uma vez que ajuda a reduzir custos, e também é benéfico para os clientes, uma vez que possibilita que eles sejam atendidos de forma mais rápida.

Pesquisas recentes reforçam que cada vez mais clientes buscam o autoatendimento. Esse autoatendimento pode se dar, por exemplo, por meio de *chatbots*, agentes conversacionais que podem interagir com usuários por meio de linguagens naturais e que também podem ser descritos pelo termo "interfaces de usuário conversacionais"(LUO et al., 2022). Uma pesquisa da *Tidio*, uma empresa fornecedora de *chatbots*, apontou que 62% dos consumidores entrevistados prefeririam conversar com um *chatbot*, ou seja, optar pelo autoatendimento, do que aguardar 15 minutos para conversar com um atendente humano (Tidio, 2023). Outra companhia de *chatbots*, a *Botco.ai*, descobriu que 70% dos consumidores de autoatendimento entrevistados tiveram todas ou a maioria de suas perguntas respondidas satisfatoriamente por *chatbots* (Mona Bushnell, 2023).

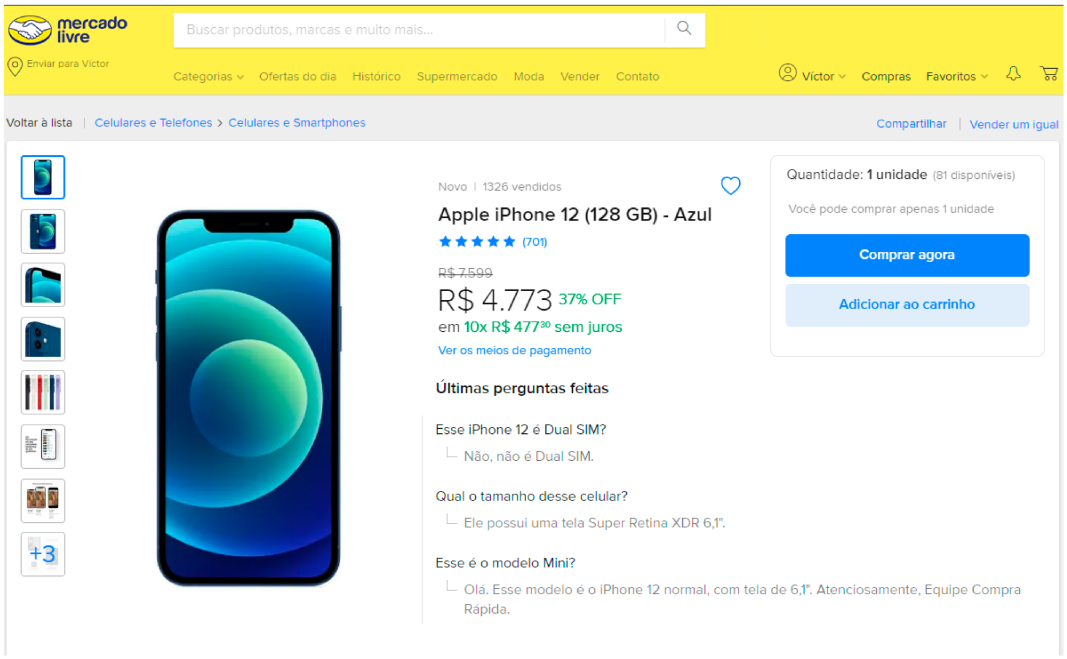

Figura 1 – Exemplos de perguntas com as quais os lojistas dos *marketplaces* precisam lidar diariamente.

2.3 Classificação de Perguntas Sobre Produtos Quanto ao Atributo

O problema a ser estudado neste trabalho, apresentado como classificação de perguntas sobre produtos quanto ao atributo, é definido na literatura como Classificação de Texto (do inglês *Text Classification*). De acordo com a definição apresentada em (AGGARWAL; ZHAI, 2012), a classificação de texto consiste na seguinte situação: dado um

conjunto de instâncias de treino $\mathcal{D} = \{X_1, ..., X_N\}$, onde cada instância X_i é uma sentença textual formada por um conjunto de características (*features*) e está anotada com um valor de classe obtido de um conjunto de k valores discretos indexados como $\{1...k\}$, treina-se um modelo classificador, que relacionará as características de cada instância de treino a uma das classes. Depois que o treinamento é feito, uma instância de teste, cuja classe é desconhecida, é fornecida ao modelo classificador para que este preveja a classe a qual a instância pertence.

Além de atuar como uma parte integrante de um sistema que responde perguntas de clientes no comércio eletrônico, que é a implementação aqui descrita, a classificação de texto também é muito presente em outros contextos, tais como: obtenção de informações dispostas de forma não estruturada em grandes coleções de documentos (RILOFF, 1995), filtragem de informações quanto a sua relevância em *streams* de dados (KHAN et al., 2017), análise de sentimentos para identificar opinião, sentimento e subjetividade em textos (CORRÊA et al., 2017), sistemas de recomendação que sugerem itens baseados no perfil de interesses do usuário e na descrição do item (MOONEY; ROY, 2000), entre muitos outros (KOWSARI et al., 2019).

No contexto de resolução de perguntas de clientes reais, a classificação de textos atua para que a pergunta feita seja entendida. Toda pergunta é um questionamento que um cliente possui sobre uma característica de um determinado produto. No contexto deste trabalho, cada classe que o modelo classificador consegue identificar representa uma característica que o produto tem, isto é, representa um atributo do produto. Assim, se um cliente está interessado em saber a marca de uma chave de fenda, ele pergunta "Qual é a marca da chave de fenda?". Esta pergunta, por sua vez, pode ser classificada como uma pergunta sobre o atributo ou classe "marca". Essa forma de responder perguntas encontra diversas dificuldades, que serão discutidas na seção 2.3.2.

2.3.1 O Conceito de Atributo no Mercado Livre

Os atributos são características dos produtos adicionadas pelos lojistas no formato de pares (nome do atributo, valor do atributo). Eles podem ser acessados por meio de consultas à API do Mercado Livre. Dessa forma, após o modelo classificador prever o nome do atributo ao qual uma pergunta de um cliente real se refere, o valor do atributo pode ser obtido filtrando a resposta da consulta à API e procurando pelo nome do atributo.

Nesse sentido, após a execução da Classificação de Texto pelo modelo classificador, as perguntas feitas na Figura 1 podem ser respondidas filtrando os nomes de atributos ESIMS_NUMBER, DISPLAY_SIZE e MODEL na resposta da consulta à API do Mercado Livre. Os valores dos atributos são 1 (indicando que o celular só possui entrada para um chip), 6,1” (indicando que o celular possui 6,1 polegadas de tamanho), e iPhone 12 (indicando que o celular é do modelo padrão, e não do modelo Mini), respectivamente. A

Figura 2 – Resposta da consulta à API de produtos do Mercado Livre, que retorna nomes e valores dos atributos. Em 2a, informações sobre o produto. Em 2b, a resposta para a pergunta "Esse iPhone 12 é Dual SIM?". Em 2c, a resposta para a pergunta "Qual o tamanho desse celular?". Em 2d, a resposta para a pergunta "Esse é o modelo Mini?".

Figura 2 mostra a resposta da consulta à API do Mercado Livre.

2.3.2 Dificuldades da Classificação de Perguntas quanto ao Atributo

Em Novembro de 2022, o Mercado Livre possuía mais de 7000 atributos únicos, usados para descrever produtos de centenas de subcategorias. Considerando essa condição, estar preparado para responder a perguntas sobre o valor de qualquer atributo, seja ele genérico ou específico, é uma tarefa extremamente difícil, pois ela encontra as limitações citadas a seguir.

- Generalização para atributos não vistos nos dados de treino: manter uma base de dados com exemplos de perguntas para cada um dos 7000 atributos é uma tarefa dispendiosa em termos financeiros, e impossível de se realizar manualmente. Além disso, novas categorias de produtos são lançadas todos os dias, o que tornaria esse conjunto de dados obsoleto. Por isso, outros trabalhos da literatura usam estratégias avançadas, como a concatenação de camadas de modelos que não são classificadores (WANG et al., 2020). A técnica proposta no trabalho citado melhorou consideravelmente o reconhecimento de atributos nunca vistos entre os dados de treino, ou seja, aprimorou a habilidade de *zero-shot learning*.

- Perguntas que se referem à compatibilidade entre produtos: no Mercado Livre exis-

tem dois atributos genéricos, o **COMPATIBLE_BRANDS** e o **COMPATIBLE_MODELS**, que geralmente estão presentes em produtos que são acessórios de objetos maiores, como por exemplo rádios automotivos. É muito comum que as perguntas dessas duas classes diferentes tenham uma estrutura sintática parecida, como exemplificado nas Figuras 3 e 4.

- Esse controle é compatível com Xbox Series S?

Figura 3 – Pergunta que seria classificada como COMPATIBLE_MODELS.

- É compatível com micro retífica black&decker?

Figura 4 – Pergunta que seria classificada como COMPATIBLE_BRANDS.

É possível perceber que a classe a que cada exemplo das Figuras 3 e 4 pertence só pode ser avaliada com base em conhecimento de mundo, que permite distinguir marcas de modelos. Para que um algoritmo de Inteligência Artifical consiga fazer essa distinção com maior assertividade, faz-se necessária uma solução mais específica, como um grande Grafo de Conhecimento com relações entre marcas e modelos de produtos, como explorado em (SANT'ANNA et al., 2020).

- Atributos que fazem parte de uma categoria, mas não de outras: cada categoria possui sua lista de atributos válidos. Alguns atributos são mais genéricos e fazem parte da maioria das categorias, enquanto alguns atributos são mais específicos, compõem categorias restritas. Isso faz com que existam atributos análogos entre categorias diferentes.

  Um disco de freio pode ter o atributo **MATERIAL**, mas uma camiseta só pode ter o atributo **COMPOSITION**. No entanto, não estamos levando em conta a categoria do produto no momento de fazer a classificação, apenas a pergunta. Isso possibilita que uma pergunta sobre essa camiseta seja classificada como **MATERIAL**, o que deixaria o cliente sem resposta.

- Saudações: os clientes frequentemente introduzem palavras extras, como saudações, para estabelecer a cordialidade entre eles e os atendentes. Isso provoca um ruído desnecessário que pode afetar o resultado final, uma vez que esses cumprimentos também serão fornecidos ao modelo classificador se não forem apropriadamente filtrados.

- Variações na ortografia: os clientes possuem diferentes níveis de escolaridade e as perguntas são feitas em um contexto virtual, favorável a abreviações. Esses dois fatores permitem com que uma palavra possa ser escrita das mais variadas formas, o que representa um desafio para os *tokenizadores*, descritos na Seção 2.4.4.

2.4 Etapas da Classificação de Perguntas Quanto ao Atributo

A construção de um modelo de classificação de perguntas quanto ao atributo, que pode ser formalizada como uma tarefa de classificação de texto, exige a realização de diversas etapas, que serão descritas nas subseções a seguir.

2.4.1 Definição das Classes

Não é possível, de forma manual, criar uma base de dados com instâncias de todos os atributos existentes para se classificar um produto em um dado *marketplace*. Isso acontece porque aos produtos anunciados nos *marketplaces* podem ser atribuídas centenas ou até mesmo milhares de atributos diferentes. Apesar de existirem modelos pré-treinados que conseguem identificar atributos não usados no treinamento, como em (WANG et al., 2020), torna-se importante ter uma base de dados construída em um conjunto conhecido de classes a fim de avaliar um modelo.

A API de Categorias do Mercado Livre[3] fornece um recurso **/attributes** que permite encontrar a lista de todos os atributos de uma dada categoria. Além disso, é possível baixar a árvore de categorias do Mercado Livre[4], que relaciona todas as categorias e seu grau de parentesco entre si. Um modelo classificador generalista idealmente teria como dados de treino exemplos de perguntas com os atributos que aparecem no maior número de categorias possível, ou seja, os mais genéricos.

2.4.2 Coleta de Dados

Os dados provavelmente são a parte mais importante de um sistema de aprendizado de máquina. Em contextos mais simplificados, é facilmente possível obter as bases de dados necessárias para resolver um determinado problema, e elas são constituídos de milhares (ou até mesmo milhões) de instâncias (VAJJALA et al., 2020). No entanto, na maioria dos projetos de Inteligência Artificial acadêmicos ou comerciais é preciso fazer uso de técnicas de coleta de dados.

Fontes como (VAJJALA et al., 2020) agregaram as técnicas mais empregadas para concluir esta etapa:

- Usar uma base de dados pública: consiste em pesquisar se existem bases de dados públicas disponíveis que sejam similares e compatíveis com o problema em questão. Agregadores como o Google Datasets[5] e o HuggingFace[6] compilam muitas das opções disponíveis.

[3] <https://developers.mercadolivre.com.br/pt_br/categorias-e-atributos>
[4] <https://developers.mercadolivre.com.br/pt_br/dump-de-categorias>
[5] <https://datasetsearch.research.google.com/>
[6] <https://huggingface.co/datasets>

- *Data Scraping*: consiste em encontrar uma fonte de dados relevante na Internet e fazer uso de técnicas de *Web Scraping*, ou seja, obter esses dados de forma que não seja por meio de um programa interagindo com uma API (nem por interação humana). Geralmente esse objetivo é atingido escrevendo um programa que se comunica com um servidor Web, faz uma requisição de dados (muitas vezes em forma de arquivos HTML) e então formata aqueles dados para extrair a informação necessária (MITCHELL, 2018).

- *Product Intervention*: é uma técnica muito usada por grandes empresas da tecnologia. Quando essas empresas implementam modelos de Inteligência Artificial em seus produtos, os modelos raramente existem de forma isolada do restante do produto. Assim, a técnica de *Product Intervention* consiste em trabalhar juntamente com o time de produto para obter cada vez mais dados diretamente dos usuários, a partir do momento em que estes autorizam. Esses dados são informações relacionadas aos comportamentos que cada usuário teve ao usar aquele produto. Isso acontece por exemplo quando um usuário acessa um *site* de compartilhamento de vídeos, e obtém recomendações de novos vídeos para assistir através de um algoritmo que analisou vídeos previamente assistidos pelo mesmo usuário.

- Aumento de dados: consiste em explorar propriedades linguísticas para criar novos textos que são sintaticamente similares aos textos previamente existentes na base de dados.

2.4.3 Anotação de Dados

Todas as técnicas de coleta de dados abordadas na Seção 2.4.2, exceto o uso de uma base de dados pública, geralmente levam à compilação de dados não-anotados, ou seja, cada sentença coletada ainda não teve a sua classe correspondente identificada. Como os modelos a serem avaliados neste projeto são modelos de classificação de texto, essa informação se faz necessária para o seu treinamento (AGGARWAL; ZHAI, 2012).

A anotação de dados (ou identificação de classes) é uma tarefa cansativa, demorada e repetitiva, que frequentemente exige o trabalho de uma equipe de anotadores. Além disso, nem sempre é possível saber de forma previsível quantos exemplos anotados serão necessários, uma vez que modelos linguísticos eficazes podem ser construídos com base em uma quantidade de exemplos modesta, desde que os dados de treinamento coincidam com a aplicação desejada (ROSE; HADDOCK; TUCKER, 1997).

Depois que as classes estejam manualmente identificadas, uma prática comum na área de Ciência de Dados é dividir os exemplos em dois conjuntos: o de Treino e o de Teste. O conjunto de Treino é composto por aqueles exemplos que serão consumidos durante as iterações de treinamento, enquanto o conjunto de Teste é composto por outros

exemplos mantidos fora do treinamento, que não influenciarão as propriedades do modelo e servirão para estimar o quão efetivo um modelo é por meio da comparação da previsão com o valor conhecido (VANDERPLAS, 2016). Uma intuição inicial habitualmente tida pelos cientistas é separar 80% dos exemplos para Treino e 20% dos exemplos para Teste, seguindo o Princípio de Pareto (The Data Detective, 2020).

Frequentemente não é possível saber quais dados seriam mais adequados para Treino e quais dados seriam mais adequados para Teste. Para resolver esse problema, pode ser usada a técnica de Validação Cruzada. Essa técnica consiste em fazer uma sequência de treinamentos em que cada subconjunto dos dados é usado tanto como conjunto de Treino quanto como conjunto de Teste. Os subconjuntos devem possuir a mesma quantidade de instâncias, e a quantidade de iterações é definida pela quantidade de subconjuntos. Ao final, pode se combinar as métricas dos modelos treinados em cada iteração da sequência (por exemplo, tomando a média) para obter uma melhor métrica da performance do modelo global (VANDERPLAS, 2016).

2.4.4 Pré-processamento dos Dados

Os *softwares* de NLP tipicamente analisam texto dividindo-o em palavras (*tokens*) e sentenças (VAJJALA et al., 2020). Por isso uma subetapa importante do pré-processamento de dados diz respeito a essas divisões.

Após as divisões, outros passos são frequentemente efetuados por cientistas de dados, a depender da tarefa específica a qual o modelo treinado se dedicará, assim como do domínio do problema. Esses passos são importantes pelo fato de que um grande dicionário dificulta a tarefa de mineração de dados, por exemplo, no caso de redes neurais. Como redes neurais podem ser caras de se treinar, é importante não desperdiçar capacidade computacional (TUNSTALL; WERRA; WOLF, 2022). Alguns passos comuns a esta etapa são:

- *Lowercasing*: consiste em converter todas as letras maiúsculas em minúsculas, deixando o texto todo em caixa baixa (VAJJALA et al., 2020).
- Remoção de *stopwords*: consiste em remover todas aquelas palavras que aparecem com grande frequência no decorrer da base de dados, mas que não contribuem para a semântica da sentença, pois não fornecem nenhuma informação adicional. Ao removê-las, é possível focar nas palavras mais importantes. Alguns exemplos são os artigos, determinantes e pronomes, como “o”, “a”, “os”, “de”, etc. (VAJJALA et al., 2020) (ETAIWI; NAYMAT, 2017).
- Remoção da pontuação e/ou números: a remoção de todo tipo de pontuação, tais como pontos de interrogação, vírgulas, pontos finais, entre outros, afeta o resul-

tado de qualquer abordagem de processamento de texto, especialmente aquelas que dependem da frequência de ocorrência de palavras e frases (ETAIWI; NAYMAT, 2017).

- *Stemming*: é o processo de remover sufixos para reduzir uma palavra de forma que todas as variantes daquela palavra podem se representadas da mesma forma (por exemplo, "carros" e "carro" são ambos reduzidos para "carro"). Isso é feito seguindo uma série de regras. Algoritmos de *Stemming* são específicos para cada língua e diferem em termos de performance e acurácia (VAJJALA et al., 2020) (ETAIWI; NAYMAT, 2017).

- *Lemmatization*: possui algumas similaridades com o *Stemming*, porém consiste em mapear todas as diferentes formas de uma palavra para a sua palavra base. Por exemplo, o adjetivo "melhor" se torna "bom". Esse processo requer um conhecimento linguístico maior do que o anterior (VAJJALA et al., 2020).

2.4.5 Transformação dos Dados

Para que algoritmos de aprendizado de máquina trabalhem com dados textuais, esses dados devem ser convertidos para uma representação numérica. Se textos são representados como vetores de números, a representação numérica é chamada de *Vector Space Model* (VSM). Diversas abordagens algébricas podem ser feitas para se obter um VSM (VAJJALA et al., 2020). Elas serão elaboradas nas Seções 2.4.5.1 e 2.4.5.2.

2.4.5.1 Bag of Words e TF-IDF

Duas abordagens clássicas de representação de dados textuais são a *Bag of Words* e a TF-IDF. Essas abordagens são antigas e de relativa fácil implementação, o que faz com que sejam frequentemente consideradas como linhas de base em artigos científicos. Uma referência inicial a *Bag of Words* em um contexto linguístico está em (HARRIS, 1954), onde o autor argumenta que "língua não é simplesmente uma *bag of words*, mas sim uma ferramenta com propriedades particulares que foram forjadas durante seu uso". Já a definição da abordagem TF-IDF está em (JONES, 1972), onde a autora argumenta que "termos devem ser ponderados de acordo com a frequência em que aparecem no conjunto". Logo, em um contexto onde é feita uma pesquisa usando palavras-chave para encontrar documentos relacionados, a correspondência com termos mais específicos é de maior valor do que a correspondência com termos mais frequentes.

A representação de instâncias usando a abordagem *Bag of Words* pode utilizar o formato de uma tabela atributo-valor, onde cada documento d_i é um exemplo da tabela e cada termo (palavra) t_j é um elemento do conjunto de atributos (MATSUBARA; MARTINS; MONARD, 2003).

Tabela 1 – Representação de documentos

	t_1	t_2	...	t_M	C
d_1	a_{11}	a_{12}	...	a_{1M}	c_1
d_2	a_{21}	a_{22}	...	a_{2M}	c_2
...	...	...	...	...	...
d_N	a_{N1}	a_{N2}	...	a_{NM}	c_N

Cada documento d_i da Tabela 1 é um vetor $d_i = (a_{i1}, a_{i2}, ..., a_{iM})$, no qual o valor a_{ij} refere-se ao valor associado ao j-ésimo termo do documento i. Esse valor pode ser calculado usando diferentes medidas. Se usarmos a medida *tf*, por exemplo, o valor de a_{ij} será a frequência que o termo t_j aparece no documento d_i. No entanto, quando algumas palavras aparecem na maioria dos documentos, eles podem não fornecer informações úteis (MATSUBARA; MARTINS; MONARD, 2003).

A TF-IDF (*term frequency–inverse document frequency*) é outra abordagem de ponderação de termos. Considere $T = \{t_1, ..., t_n\}$ como sendo o conjunto de todos os termos que aparecem em todos os documentos da base de dados. Então, um documento d_i é representado por um vetor de valores reais de n dimensões dado por $x_i = (x_{i1}, ..., x_{in})$ com um componente para cada possível termo do conjunto T (SAMMUT; WEBB, 2010). O peso x_{ij} correspondente ao termo t_j do documento d_i é geralmente um produto de três partes: uma que depende da frequência de t_j em d_i, uma que depende da frequência de t_j no conjunto de documentos como um todo, e uma parte de normalização que depende de d_j. A fórmula mais comum para a ponderação TF-IDF é definida pela Equação 2.1, onde TF_{ij} é a frequência do termo (isto é, o número de ocorrências) de t_j em d_i; IDF_j é o IDF de t_j, definido como o $\log \frac{N}{DF_j}$, onde N é o número total de documentos; e DF_j é o número de documentos onde t_j aparece (SAMMUT; WEBB, 2010).

$$x_{ij} = TF_i \cdot IDF_j \cdot (\sum_j (TF_{ij} IDF_j)^2)^{-1/2} \tag{2.1}$$

2.4.5.2 Tokenizadores

Os Tokenizadores são formas de representação de dados textuais mais recentes, que representam o atual estado da arte. Mais uma vez, a cada unidade em que se divide o texto uma representação numérica relacionada é gerada.

É possível tokenizar em relação a caracteres ou a palavras, mas a forma mais habitual de tokenização empregada pelos cientistas de dados atualmente é a *Subword Tokenization*. Ela combina os melhores aspectos das tokenizações por palavras ou caracteres. Por um lado, é desejável que palavras raras sejam divididas em partes menores, o que per-

mite que o modelo lide com palavras complexas e/ou escritas incorretamente. Por outro lado, é desejável que palavras frequentes sejam mapeadas como entidades únicas para que seja possível manter o comprimento da entrada do modelo em um tamanho controlável (TUNSTALL; WERRA; WOLF, 2022).

O Tokenizador usado por modelos Transformadores, como os que serão descritos na Seção 2.4.6, é o WordPiece (SCHUSTER; NAKAJIMA, 2012) (WU et al., 2016), apresentado pelo Google como um dos componentes do seu modelo BERT, aplicado neste trabalho. Ao oferecer como exemplo a sentença "Tokenizing text is a core task of NLP" ao *framework* Transformers (WOLF et al., 2020), que implementa o WordPiece, é feita a tokenização. Na prática, trechos de palavras são mapeados a números inteiros. A Figura 5 mostra esse mapeamento.

```
encoded_text = tokenizer(text)
print(encoded_text)
```

```
{'input_ids': [101, 19204, 6026, 3793, 2003, 1037, 4563, 4708, 1997, 17953,
2361, 1012, 102], 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}
```

Figura 5 – Mapeamento de *subtokens* a números inteiros únicos. Fonte: (TUNSTALL; WERRA; WOLF, 2022)

Ao converter os números inteiros para tokens, como mostrado na Figura 6, é possível entender como as palavras são divididas. Dois tokens especiais, o [CLS] e o [SEP], foram adicionados ao início e ao fim da sentença. Além disso, as palavras passaram por *lowercasing*, subetapa de pré-processamento explicada em 2.4.4. Outro detalhe que se pode observar é que "tokenizing" e "NLP" foram divididas em dois tokens. Isso faz sentido, uma vez que uma das premissas desse tokenizador é dividir palavras raras em unidades menores (TUNSTALL; WERRA; WOLF, 2022).

```
tokens = tokenizer.convert_ids_to_tokens(encoded_text.input_ids)
print(tokens)
```

```
['[CLS]', 'token', '##izing', 'text', 'is', 'a', 'core', 'task', 'of', 'nl',
'##p', '.', '[SEP]']
```

Figura 6 – Resultado da conversão de números inteiros únicos para tokens. Fonte: (TUNSTALL; WERRA; WOLF, 2022)

2.4.6 Algoritmos Classificadores de Texto

A tarefa de classificação de textos é estudada desde muito tempo, e portanto, diversos métodos de cumpri-la foram implementados ao longo dos anos. Aggarwal e Zhai (2012) e Kowsari et al. (2019) compilaram em suas pesquisas diversas técnicas de aprendizado de máquina para classificação de texto, tais como: algoritmo de Rocchio (ROCCHIO, 1971); *boosting* (SCHAPIRE, 1990) e *bagging* (BREIMAN, 1996), que são algoritmos de votação;

regressão logística (COX; SNELL, 2018); Naïve Bayes (PORTER, 1980); *k-nearest neighbor* (JIANG et al., 2012); *support vector machines* (SVMs) (VAPNIK; CHERVONENKIS, 1964); árvores de decisão (MORGAN; SONQUIST, 1963) e *random forests* (HO, 1995); algoritmos baseados em redes neurais; dentre muitos outros.

No entanto, este trabalho se dedica a explorar algumas das técnicas mais recentes de classificação de texto, especificamente as que são baseadas em *Deep Learning*. Minaee et al. (2021) compilaram em sua pesquisa diversas dessas técnicas, tais como: *transformers* (VASWANI et al., 2017) ou *pre-trained language models*; *feed-forward networks*, como por exemplo o modelo DAN (IYYER et al., 2015); modelos baseados em RNNs, como o MT-LSTM (LIU et al., 2015); modelos baseados em CNNs, como o modelo DCNN (KALCHBRENNER; GREFENSTETTE; BLUNSOM, 2014); *memory-augmented networks*, como a NSE (MUNKHDALAI; YU, 2017); dentre muitos outros. Dentre elas, duas arquiteturas diferentes de Transformadores foram escolhidas para esse projeto, BERT e DIETClassifier.

2.4.6.1 BERT

O BERT é um modelo de Transformador (VASWANI et al., 2017), e foi apresentado em (DEVLIN et al., 2019). Transformadores são algoritmos de *Deep Learning* que possuem como diferencial os mecanismos de auto-atenção, cujo desenvolvimento é explicado em ordem cronológica nos parágrafos seguintes.

Há alguns anos, era comum que tarefas como a tradução fossem resolvidas usando a arquitetura *encoder-decoder*. Isto é, uma frase qualquer em Inglês era convertida para uma representação vetorial, chamada de *last hidden state*, dentro de uma arquitetura chamada de *encoder*, composta de redes neurais recorrentes (RNNs). Na outra ponta, essa representação vetorial era convertida de volta para uma outra língua, como o Alemão, na arquitetura conhecida como *decoder*, feita também de RNNs. Porém, essa solução trazia um problema: a representação vetorial tinha dimensões fixas, o que fazia com que informações fossem perdidas em longas sentenças (TUNSTALL; WERRA; WOLF, 2022). Essa solução era aplicada antes do advento dos mecanismos de atenção.

Os mecanismos de atenção vieram como uma solução para esse problema de perda de informação. A Figura 8 retrata como funciona uma iteração desses mecanismos. Com eles, ao invés de produzir uma única representação vetorial para a frase em Inglês, o *encoder* produz uma representação vetorial diferente para cada palavra da frase. Ao final do Passo 1, todas as representações vetoriais são enviadas para o *decoder*. Após receber as representações, o *decoder* fornece para cada uma delas uma pontuação no Passo 2. Essa pontuação, que é um número inteiro simples, é fornecida como entrada para a função softmax(), que por sua vez é uma função que transforma um vetor de números em uma distribuição de probabilidades, onde o maior elemento do vetor recebe um grande peso (observe a probabilidade associada a cada elemento no Passo 3 da Figura 8). A função

softmax() é descrita pela Equação da Figura 7. Finalmente, no Passo 4, cada representação vetorial é multiplicada pelo seu valor de softmax(), o que faz com que representações vetoriais que tinham grandes pontuações no Passo 2 sejam amplificadas, em detrimento das representações vetoriais que tinham baixas pontuações (ALAMMAR, 2018) (SUTSKEVER; VINYALS; LE, 2014) (CHO et al., 2014).

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Figura 7 – A softmax() de um elemento de um vetor é a exponencial desse elemento de vetor dividida pela soma das exponenciais de todos os elementos do vetor.

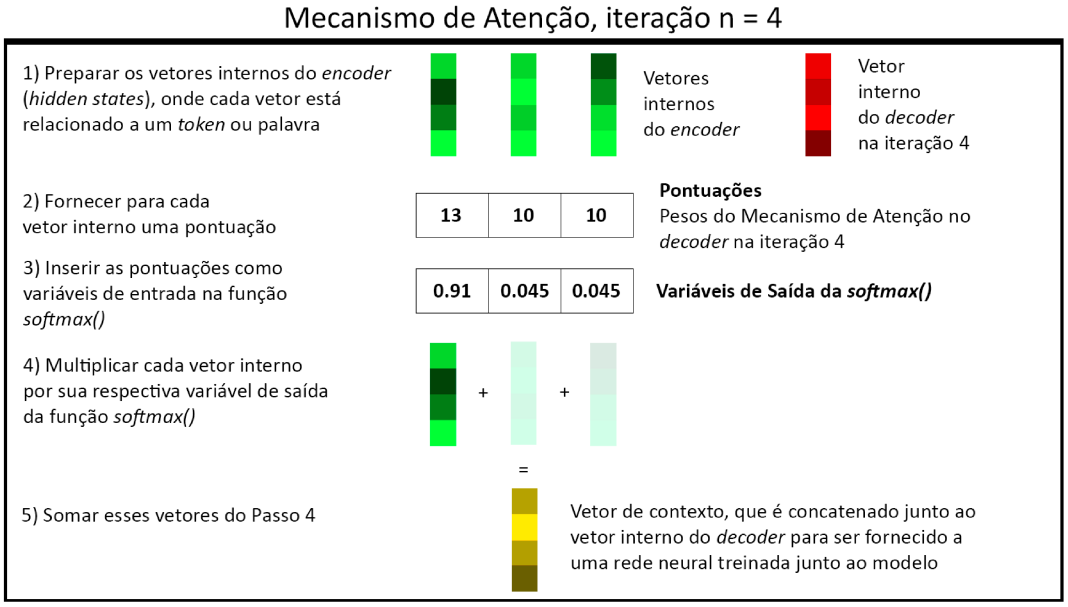

Figura 8 – Mecanismo de atenção em um *decoder*. Fonte: (ALAMMAR, 2018)

Os mecanismos de Auto-atenção são uma evolução dos mecanismos de Atenção, e foram apresentados como uma característica dos Transformadores. A principal ideia por trás deles é que, ao invés de usar uma representação vetorial fixa para cada *token*, eles usam a sequência inteira para calcular uma média ponderada de cada representação vetorial (TUNSTALL; WERRA; WOLF, 2022). Dessa forma, quando uma sentença possui a palavra "manga", por exemplo, é possível diferenciar pelo contexto se a sentença se refere à fruta ou à parte da camisa, e assim produzir uma representação vetorial que leva em consideração esse significado.

A Tabela 2 compila os principais pontos dos mecanismos referenciados nos três parágrafos anteriores: pré-atenção (o que era aplicado antes do avento dos mecanismos de atenção), Atenção, e Auto-atenção. Ela considera uma tarefa de tradução de sentenças, onde o Codificador (*encoder*) recebeu uma sentença em Inglês, por exemplo, e o Decodificador (*decoder*) deve fornecer como saída a tradução da sentença em Português.

Tabela 2 – Os três mecanismos: pré-atenção, Atenção e Auto-atenção

Mecanismo	Principais Pontos
Pré-atenção	O Decodificador tem acesso a apenas um dos vetores (hidden states) gerados pelo Codificador, e esse vetor possui tamanho fixo. Isso faz com que tanto frases simples quanto textos longos sejam representados por um vetor de mesmo tamanho, o que faz muitas informações serem perdidas quando usando textos longos.
Atenção (Attention)	O Decodificador tem acesso a todos os vetores (hidden states) gerados pelo Codificador. Esses vetores são gerados sequencialmente, ou seja, um a cada palavra. Isso significa que na iteração 3, por exemplo, existem três vetores. O Mecanismo de Atenção consiste em fornecer um peso para cada um desses vetores para assim saber quais vetores merecem mais foco, ou mais atenção, a cada iteração. A vantagem é que se torna possível representar adequadamente textos longos, assim como se torna possível criar uma associação entre as palavras "casa" e "house", por exemplo, mesmo que "casa" seja a primeira palavra da frase em Português e "house" seja a última palavra da frase em Inglês.
Auto-atenção/Transformadores (Self-attention/Transformers)	Os pesos dos vetores que definem cada palavra são aprendidos levando em consideração apenas o Codificador. A vantagem é que isso permite obter vetores contextualizados - isto é, obter um vetor diferente para "manga", a fruta, do vetor que seria gerado para "manga", a parte da camisa. Esse aprendizado é feito ponderando outras palavras do contexto daquela sentença.

O BERT, sigla para *"Bidirectional Encoder Representations from Transformers"*, se destaca entre os demais Transformadores pelo fato de criar representações vetoriais profundas bidirecionais, isto é, considerando o contexto à esquerda e o contexto à direita de uma determinada palavra (DEVLIN et al., 2019). Para isso, ele usa apenas o *encoder* da arquitetura dos Transformadores, e é pré-treinado em bases de dados extensas: o *BookCorpus* (ZHU et al., 2015) e a *Wikipedia*[7] em inglês. Essas representações foram aprendidas de forma bidirecional automaticamente ao fornecer uma base de dados de Treino composta em 80% por sentenças com palavras aleatoriamente deixadas em branco (chamadas de palavras mascaradas em Devlin et al. (2019)), como ilustra a Figura 9. Isso fez com que o modelo se tornasse assertivo em predizer qual palavra melhor completaria o espaço em branco fornecido. No exemplo da Figura 9, ao considerar "Eu estou" de um lado e "sorvete" do outro, as opções se tornam restritas: se restringem a palavras como "tomando", "comendo", "lambendo". Não seria possível ter tamanha assertividade

[7] <https://en.wikipedia.org/>

na resposta se o modelo considerasse apenas “Eu estou” e tentasse predizer a palavra restante.

Eu estou tomando sorvete→ Eu estou _____ sorvete

Figura 9 – Como os criadores do modelo BERT tinham a intenção de que ele aprendesse a considerar tanto palavras que estão ao início de uma sentença quanto palavras que estão ao final de uma sentença, para que assim as representações vetoriais das palavras levassem em conta o contexto, 80% da base de dados de Treino era composta de sentenças com palavras mascaradas, isto é, em branco.

Um fator que tornou o BERT muito popular foi a praticidade de sofrer *fine-tuning*, isto é, de receber uma camada de saída a mais e ter a possibilidade de ajustar apenas os pesos dela: isso tornou o modelo eficiente em muitas outras tarefas específicas, como classificação de texto. O BERT ajudou a definir o estado da arte, e por isso foi precursor de vários modelos frequentemente usados nos dias atuais, como os modelos RoBERTa (LIU et al., 2019) (TUNSTALL; WERRA; WOLF, 2022).

2.4.6.2 DIETClassifier

O DIETClassifier (BUNK et al., 2020) também é um modelo de Transformador (VASWANI et al., 2017). Sua característica mais marcante é o fato dele possibilitar que duas tarefas sejam executadas ao mesmo tempo sobre uma entrada de texto: ele faz classificação de intenções e reconhecimento de entidades na mesma iteração, normalmente associadas a uma interface de conversação usuário-máquina, como mostra a Figura 10. Enquanto a classificação de intenções se assemelha muito à tarefa de classificação de texto abordada aqui, e está relacionada a entender qual objetivo o usuário tinha ao enviar uma determinada frase, o reconhecimento de entidades busca categorizar palavras específicas da sentença em classes previamente vistas pelo modelo (MANTHA, 2020).

Outros pontos importantes levantados pelos autores do DIETClassifier são a sua arquitetura modular e seus bons resultados comparados a modelos pré-treinados, como o BERT. A modularização se dá porque é possível usar as representações vetoriais aprendidas por modelos como o próprio BERT como uma de suas camadas, e mesmo assim montar uma arquitetura completa que tenha o classificador DIET como camada de saída. Os bons resultados se devem ao fato de que arquiteturas montadas com o DIET podem ter resultados tão bons quanto os demonstrados por modelos maiores, e mesmo assim chegam a ser até 6x mais rápidas de se treinar (BUNK et al., 2020) (MANTHA, 2020).

2.4.7 Métricas de Desempenho

As matrizes de confusão são um tipo de visualização que permite entender as nuances do desempenho dos modelos treinados, classe por classe. Nesse tipo de gráfico,

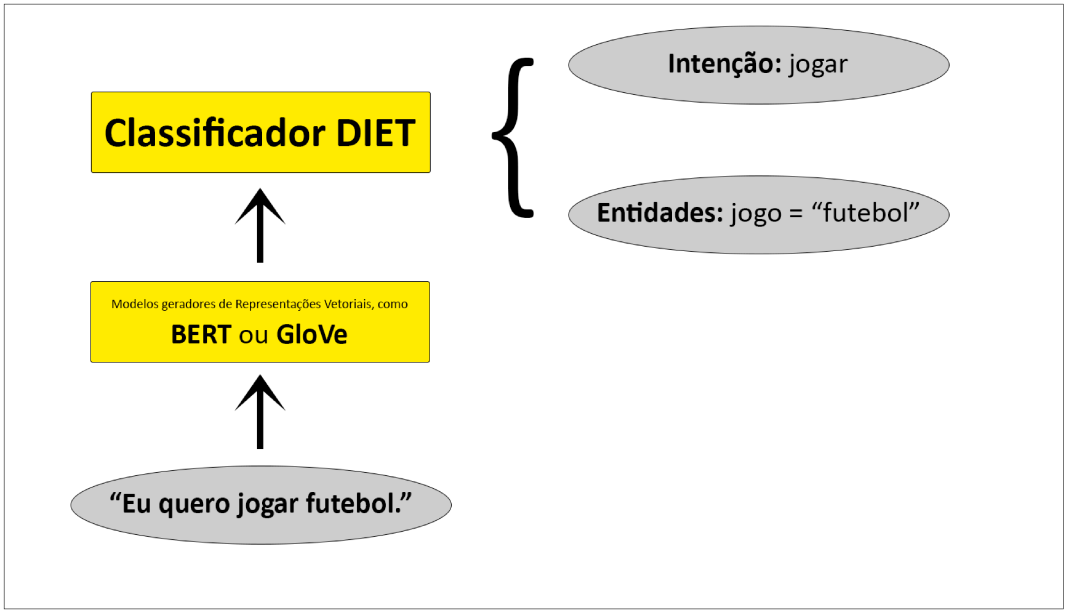

Figura 10 – As duas tarefas que são executadas ao mesmo tempo por um classificador DIET: classificação de intenções e reconhecimento de entidades.

as linhas representam as classes verdadeiras, enquanto as colunas representam as classes previstas pelo modelo em questão (a representação em linha ou coluna varia entre trabalhos).

Suponha que um problema de Processamento de Linguagem Natural foi apresentado, onde um modelo de Inteligência Artificial foi treinado para classificar avaliações dos usuários sobre um determinado filme em **Bom**, **Razoável** e **Ruim**. Esse modelo obteve o resultado retratado na matriz de confusão da Figura 11. Ao olhar o termo da primeira coluna com a primeira linha, percebe-se que 3 avaliações **Bom** foram classificadas como **Bom** pelo modelo treinado. Por outro lado, ao olhar o termo ao lado vê-se que 4 avaliações **Bom** foram classificadas como **Razoável** pelo modelo treinado, o que está incorreto. Dessa forma, um modelo que consegue prever corretamente a opinião dos espectadores em cada avaliação em todas as avaliações de um determinado conjunto teria uma matriz de confusão diagonal (os únicos termos diferentes de zero seriam aqueles da diagonal principal) (GRANDINI; BAGLI; VISANI, 2020).

Soluções para problemas de classificação multi-classe, como o abordado neste trabalho, devem ser avaliadas por um conjunto de métricas específico. Trabalhos como a *survey* conduzida em (GRANDINI; BAGLI; VISANI, 2020) se dedicaram a levantar as características dos indicadores de desempenho mais comuns para esse tipo de tarefa, o que é fundamental para a comparação entre os desempenhos dos diferentes modelos treinados. Algumas métricas permitem avaliar a predição de exemplos como um todo. Outras métricas permitem avaliar o quão boa é a predição de uma determinada classe em comparação com outras classes. O engenheiro de aprendizado de máquina deve entender qual tipo de resultado é mais importante e escolher as métricas adequadas se baseando nisso. Entre as

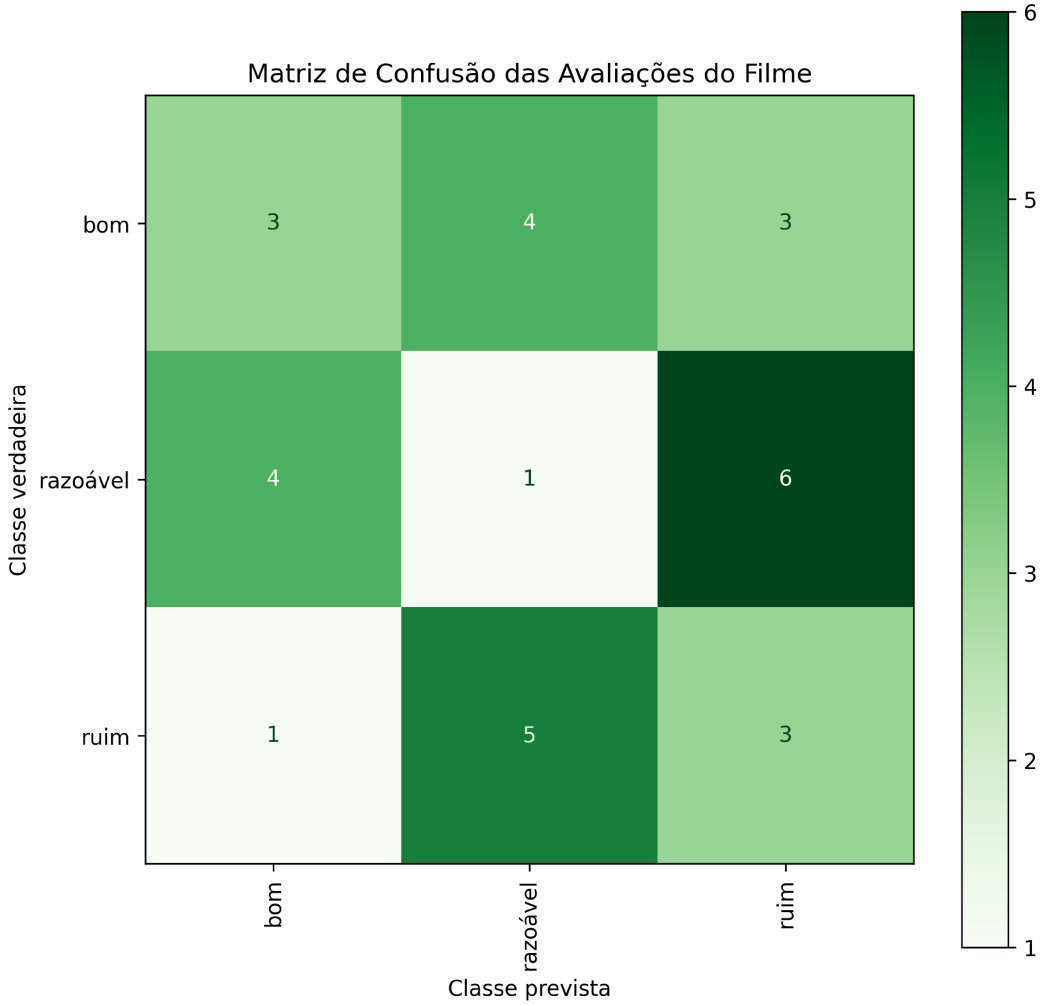

Figura 11 – Exemplo de visualização de Matriz de Confusão, que será esmiuçado em seções posteriores.

métricas mais conhecidas pela literatura, estão:

- **Acurácia**: reflete o quão próximo um dado conjunto de observações (ou previsões, no contexto de Inteligência Artificial) está próximo dos seus valores verdadeiros. Ela considera a soma de todos os elementos classificados corretamente no numerador, enquanto o denominador soma os elementos classificados corretamente com os classificados incorretamente. Assume valores entre 0 e 1 (GRANDINI; BAGLI; VISANI, 2020).

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.2}$$

  Na fórmula de Acurácia:

  - TP: *true positives*, ou verdadeiros positivos, que é quando um exemplo da classe Positiva é classificado como um elemento da classe Positiva.
  - TN: *true negatives*, ou verdadeiros negativos, que é quando um exemplo da classe Negativa não é classificado como um elemento da classe Positiva.

  - FP: *false positives*, ou falsos positivos, que é quando um exemplo da classe Negativa é classificado como um elemento da classe Positiva.
  - FN: *false negatives*, ou falsos negativos, que é quando um exemplo da classe Positiva é classificado como um elemento da classe Negativa.

  A Acurácia é calculada sobre todo o conjunto avaliado, considerando os exemplos de todas as classes. Existem classes com mais exemplos e classes com menos exemplos. Isso indica que classes mais populadas terão um maior peso em comparação com as classes menos populadas no valor de Acurácia (GRANDINI; BAGLI; VISANI, 2020).

- **Precisão**: é a proporção de exemplos que o modelo previu como pertencentes à classe Positiva que realmente são daquela classe, ou seja, ela representa o quanto é possível confiar em um modelo quando ele prevê que um exemplo é de uma dada classe. É uma métrica dada para cada uma das classes no contexto de classificação multi-classes. Assume valores entre 0 e 1 (GRANDINI; BAGLI; VISANI, 2020). A Precisão de uma classe k será igual a (Eq. 2.3):

$$\text{Precisão}_k = \frac{TP_k}{TP_k + FP_k} \tag{2.3}$$

  Ao final, caso seja preciso obter uma métrica de Precisão generalizada para todas as classes, pode-se fazer uma média ponderada considerando cada valor encontrado na (Eq. 2.3), o que tira o efeito do desbalanceamento no número de exemplos entre as classes (PEDREGOSA et al., 2011).

- **Revocação**: considerando todos os exemplos que a classe Positiva possui, indica a porcentagem de quantos foram corretamente identificados como pertencentes a essa classe. É uma métrica dada para cada uma das classes no contexto de classificação multi-classes. Assume valores entre 0 e 1 (GRANDINI; BAGLI; VISANI, 2020). A Revocação de uma classe k será igual a (Eq. 2.4):

$$\text{Revocação}_k = \frac{TP_k}{TP_k + FN_k} \tag{2.4}$$

  Ao final, caso seja preciso obter uma métrica de Revocação generalizada para todas as classes, pode-se fazer uma média ponderada considerando cada valor encontrado na (Eq. 2.4), o que tira o efeito do desbalanceamento no número de exemplos entre as classes (PEDREGOSA et al., 2011).

- ***F1-Score***: é uma métrica que agrega as métricas de Precisão e Revocação em uma fórmula de média harmônica. Assim, torna-se possível encontrar a melhor relação entre as duas quantidades. É uma métrica dada para cada uma das classes no

contexto de classificação multi-classes. Assume valores entre 0 e 1 (GRANDINI; BAGLI; VISANI, 2020). O *F1-Score* de uma classe k será igual a (Eq. 2.5):

$$F1_k = 2 \cdot \frac{\text{precisão} \cdot \text{revocação}}{\text{precisão} + \text{revocação}} \tag{2.5}$$

Ao final, caso seja preciso obter uma métrica de *F1-Score* generalizada para todas as classes, pode-se fazer uma média ponderada considerando cada valor encontrado na (Eq. 2.5), o que tira o efeito do desbalanceamento no número de exemplos entre as classes (PEDREGOSA et al., 2011).

2.5 Trabalhos Relacionados

O estudo da implementação de modelos de aprendizado de máquina para Classificação de Texto é amplo, distribuído entre diferentes tipos de aplicações pelo mundo todo. Alguns dos muitos trabalhos nesse campo se assemelham com o trabalho apresentado por também estarem no contexto do *e-commerce*, enquanto outros são similares porque sua base de dados também é composta por exemplos em Português (AGGARWAL; ZHAI, 2012).

Em (CORTES, 2019), o autor usa uma base de dados em Português para classificar perguntas sobre conhecimentos gerais em relação à classe daquela pergunta, como Localidade, Pessoa, etc. Ele então combina essa técnica com reconhecimento de entidades e recuperação da informação para percorrer milhares de documentos e obter a resposta. O pré-processamento é feito a partir de *Lowercasing* e do uso dos algoritmos TF-IDF e *Word2Vec* (MIKOLOV et al., 2013) individualmente ou em conjunto (chamado pelo autor de Híbrido). Três arquiteturas de aprendizado de máquina são treinadas e comparadas com outras mais simples: *Support Vector Machine* (SVM), *Perceptron* Multicamadas (MLP) e *Long Short-Term Memory* (LSTM) de 4 camadas. A avaliação da Classificação de Perguntas foi feita usando Validação Cruzada (*Cross-validation*) e alterando o tamanho da base de dados, assim como empregando métricas como Precisão, Revocação e *F1-Score*. Além disso, houve a criação de matrizes de confusão para comparar o desempenho dos diferentes algoritmos de pré-processamento (*Bag of Words*, TF-IDF, *Word2Vec* e Híbrido, ou seja, TF-IDF e *Word2Vec* em conjunto) para cada arquitetura de aprendizado de máquina (SVM, MLP e LSTM). Entre os resultados encontrados pelo trabalho está o de que o pré-processamento Híbrido na etapa de classificação de perguntas, usando os algoritmos TF-IDF e *Word2Vec* em conjunto, levou a uma melhor classificação das perguntas do que o pré-processamento individual. Outro resultado encontrado foi o de que a etapa de classificação de perguntas foi importante para o desempenho geral do sistema completo (classificação de perguntas e, após, recuperação da informação).

Uma aplicação da Classificação de Texto voltada para o campo de avaliações de produtos, um espaço comum em sites de comércio *online*, é abordada em (DENIZ; ER-

BAY; COŞAR, 2022). Usando uma base de dados com exemplos em Turco, os autores usaram oito diferentes algoritmos de aprendizado de máquina para classificar não apenas o sentimento do cliente em relação à experiência da compra, mas também aspectos sobre a entrega ser rápida ou se o avaliador recomenda que futuros compradores peçam tamanhos menores. Assim, eles analisaram avaliações de clientes de uma forma orientada a aspectos, e quatro formas de representação vetorial (TF-IDF, *Word2Vec*, *GloVe* (PENNINGTON; SOCHER; MANNING, 2014) e BERT) foram usadas para treinar oito métodos de classificação (*Random Forest* (RF), *Support Vector Classification* (SVC), *Naive Bayes* (NB), *Multi-label k-Nearest Neighbor* (Ml-kNN), *One-versus-Rest Logistic Regression* (OvsR-LR), *One-versus-Rest Stochastic Gradient Descent* (OvsR-SGD), *One-versus-Rest eXtreme Gradient Boosting* (OvsR-XGB) e *One-versus-Rest Support Vector Classification* (OvsR-SVC)). Os sistemas finais criados a partir da combinação de uma das formas de representação vetorial com um dos métodos de classificação, tomando todos eles um a um, foram avaliados a partir de diferentes métricas: *Hamming Loss* (HL); *Micro Averaged Precision* (MicroP); *Macro Averaged Precision* (MacroP); *Micro Averaged Recall* (MicroR); *Macro Averaged Recall* (MacroR); *Micro F1-Score* (MicroF1); *Macro F1-Score* (MacroF1). O trabalho chegou à conclusão de que através dele foi possível criar uma nova perspectiva sobre avaliações de produtos no *e-commerce*, transformando esse problema em um problema multi-classe e multi-rótulo (uma sentença textual pode ser classificada em mais de uma classe simultaneamente). Além disso, os melhores resultados encontrados pelo trabalho possuem métricas superiores a trabalhos relacionados na área de análise multi-rótulo de avaliações de produtos (*multi-label customer review analysis*). Também foi possível criar uma nova base de dados em Turco.

Com o objetivo de facilitar o trabalho dos lojistas ao anunciarem seus produtos, pesquisadores do *eBay*[8] usaram Classificação de Texto para encontrar os atributos mais importantes de um determinado item (FUCHS; ACRICHE, 2022). Para isso, eles construíram uma base de dados própria com exemplos em Inglês, onde cada entrada possui um título de produto e um par atributo-valor relacionado ao título. Dessa forma, os modelos desenvolvidos classificam um título e retornam um dicionário com os atributos mais importantes contidos naquela sentença, para que assim o anunciante escolha listar esses atributos. O pré-processamento envolveu *Lowercasing*, remoção de *stop-words* e de caracteres não-alfanuméricos para que assim duas redes neurais convolucionais (CNNs) *Seq2Seq* pudessem ser treinadas. O trabalho foi avaliado quantitativamente com as métricas Precisão e *Discounted Cumulative Gain*, usadas para comparar o ranqueamento de atributos das redes neurais convolucionais propostas pelos pesquisadores a outros modelos conhecidos na literatura, o BERT e o ULMFiT. Além disso, foi feita uma comparação entre um modelo de extração de pares atributo-valor associados ao título já em uso pelo *eBay* com o novo modelo proposto. Também houve uma avaliação qualitativa, em que instâncias de

[8] <https://www.ebay.com/>

dados de Teste foram classificadas pelo modelo proposto e os pesquisadores analisaram manualmente os pares atributo-valor retornados. Os resultados obtidos com o trabalho foram o de que o modelo proposto é melhor nessa aplicação em específico do que o BERT e o ULMFiT. Além disso, essa abordagem possibilita que até mesmo pares atributo-valor que não estão explicitamente no título sejam extraídos, uma das provas de que uma abordagem em que há um modelo em que ambas entrada e saída são textos funciona melhor com esse tipo de situação de classificação de texto multi-rótulo com ranqueamento do que as demais abordagens.

Um complexo robô de conversa chinês criado com a intenção de guiar consumidores em suas compras *online* também teve uma de suas funcionalidades desenvolvida usando arquiteturas de Classificação de Texto (YAN et al., 2017). Ao conversar com um usuário, o robô é capaz de classificar se a mensagem enviada pelo usuário possui a intenção de compra de um produto ou apenas a intenção de conversar. Se a intenção de compra existe, quando o nome de um produto e o de um atributo são detectados na fala de um usuário, mas o valor do atributo não, é feita uma pesquisa para se encontrar esse valor no banco de dados de produtos. A base de dados é composta por exemplos em Chinês, e contém tanto relações <produto, atributo, valor> quanto perguntas e respostas vindas de um fórum local. O pré-processamento é feito com a concatenação dos vetores de caracteres de cada palavra, e o algoritmo de aprendizado de máquina empregado especificamente nessa tarefa é uma rede neural convolucional (CNN) classificadora multi-classes. O trabalho foi avaliado por meio da comparação de valores de métricas como Precisão, Revocação e *F1-Score* obtidos pela rede neural proposta contra modelos definidos como linhas de base (BM25, *Naive Bayes* e *Multiple Logistic Regression*). Essas métricas foram usadas para avaliar o desempenho da rede neural proposta tanto em classificação da intenção da fala do usuário (se ele pretende comprar um produto ou apenas conversar) quanto para detecção da categoria do produto que o usuário está interessado (quando ele diz que quer comprar um produto da marca X sem citar explicitamente a categoria do produto). Os resultados encontrados indicaram que o modelo desenvolvido teve dificuldade em classificar instâncias de mensagens dos usuários que não eram relacionadas a compras da forma correta. Isso foi apontado como um problema a ser melhorado, uma vez que 80% das mensagens dos usuários são saudações não relacionadas com a intenção de compra de produtos.

2.6 Considerações Finais

Nos últimos anos, os *marketplaces* se tornaram cada vez mais importantes na economia, com cada vez mais concorrentes marcando presença no comércio *online*. Sendo assim, empregar algoritmos de Classificação de Texto para entender as perguntas dos seus clientes e os orientar melhor quanto a algum produto se tornou imprescindível para o crescimento financeiro das companhias. Existem variados trabalhos sobre Classificação

de Texto relacionados a outros campos de uma página *online* de anúncio de produto, mas nada especificamente sobre a seção de perguntas e respostas ou relacionados a esse contexto em Português.

Para atingir esse objetivo, é ideal conciliar estratégias que são o atual estado da arte, como os modelos Transformadores, com etapas consolidadas de pré-processamento dos dados, como a remoção de *stopwords* e o *Stemming*. Além disso, é fundamental considerar métricas frequentemente usadas na literatura para a etapa de avaliação dos modelos. O Capítulo 3 aborda todos os passos colocados em prática para o treinamento dos modelos de Classificação de Texto, desde a coleta dos dados até o emprego das métricas para avaliar os melhores modelos treinados.

3 Método para Avaliação de Classificadores Treinados na Base de Dados do Mercado Livre

Este capítulo expõe a metodologia utilizada para o desenvolvimento deste estudo, que visa classificar as perguntas de clientes reais de plataformas de *e-commerce*, como o Mercado Livre, em relação ao atributo ao qual a pergunta se refere.

A Seção 3.2 retrata como foi feita a execução de requisições para a API do Mercado Livre com o objetivo de obter os atributos que aparecem no maior número de categorias de produtos. Além disso, mostra a definição dos atributos mais importantes e genéricos para se tornarem as classes.

A Seção 3.3 expõe o procedimento de importação de perguntas feitas por clientes reais do Mercado Livre através do banco de dados da GoBots. Além disso, são expostas a extração manual de mais perguntas de classes pouco presentes na base de dados e a criação manual de mais perguntas para que seja atingido um número mínimo de 15 perguntas por classe.

A Seção 3.4 explica a criação e execução de um *script Python* que coleta uma pergunta aleatória das que foram extraídas do banco de dados da GoBots e mostra para o usuário rotular manualmente, de forma iterativa.

A Seção 3.5 descreve a definição das etapas de pré-processamento dos dados que foram executadas antes do treinamento dos algoritmos de classificação. Para o classificador BERTimbau, foi feita apenas a tokenização com o seu tokenizador. Para o classificador DIETClassifier, foram adicionadas outras etapas.

A Seção 3.6 apresenta o treinamento dos algoritmos de classificação, que foi baseado em diferentes configurações de pré-processamento dos dados, em diferentes hiperparâmetros de treino e em diferentes quantidades de classes. Além disso, a Seção 3.6 define o método para separação de dados em treino e teste.

A Seção 3.7 discorre sobre a criação de uma função responsável pelo cálculo das métricas acurácia, precisão, revocação e F1-score, bem como sobre a geração de matrizes de confusão. Além disso, a Seção 3.7 também explica como foi aplicada a ponderação sobre as métricas para conter o efeito do desbalanceamento das classes.

3.1 Visão Geral do Método para Avaliação

A Figura 12 apresenta as etapas de desenvolvimento deste estudo, que serão detalhadas nas próximas seções.

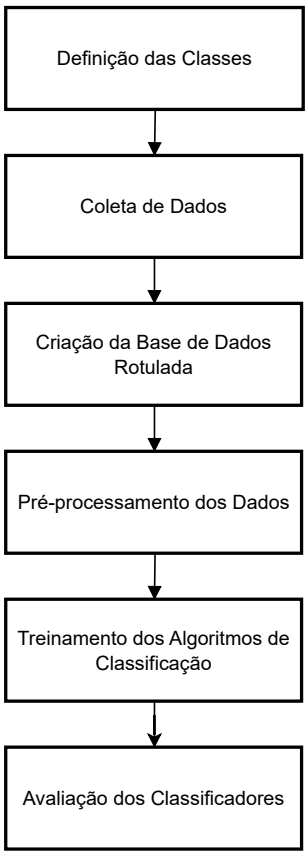

Figura 12 – Etapas do desenvolvimento deste estudo, que consiste em um método para avaliação de classificadores treinados na base de dados do Mercado Livre.

3.2 Definição das Classes

Antes de coletar os dados que compuseram a base de dados, foi necessário definir quais atributos seriam considerados como classes. No Mercado Livre[1], os produtos estavam organizados hierarquicamente em categorias. Essas categorias, por sua vez, estavam organizadas em um formato de Árvore de Categorias, que podia ser acessado com requisições via API[2] usando o comando da Figura 13.

```
$ curl https://api.mercadolibre.com/sites/MLB/categories/all > f.gz
```

Figura 13 – Comando para fazer *download* de um arquivo compactado que possui a Árvore de Categorias completa do Mercado Livre brasileiro.

[1] <https://www.mercadolivre.com.br/>
[2] <https://developers.mercadolivre.com.br/pt_br/categorias-e-atributos>

O Mercado Livre brasileiro possuía 32 categorias principais, e cada uma das categorias principais era dividida em dezenas de categorias secundárias, que por sua vez também podiam possuir outras subcategorias. Por exemplo, um carregador de celular era um produto que se encontrava em Celulares e Telefones > Acessórios para Celulares > Carregadores e Acessórios > Carregadores, de acordo com a estrutura proposta pelo Mercado Livre.

Como a GoBots fornece serviços para lojistas que vendem produtos das 11 categorias principais dispostas na Tabela 3, essas foram as categorias de produtos escolhidas para compor a base de dados.

Tabela 3 – Categorias escolhidas para serem representadas na base de dados

Categoria	
Acessórios para Veículos	Construção
Beleza e Cuidado Pessoal	Eletrodomésticos
Calçados, Roupas e Bolsas	Eletrônicos, Áudio e Vídeo
Câmeras e Acessórios	Esportes e Fitness
Casa, Móveis e Decoração	Ferramentas
Celulares e Telefones	

Para definir quais atributos serão considerados como classes, foi feito o *download* da Árvore de Categorias usando o comando da Figura 13. Em seguida, foi desenvolvido um código em *Python* para percorrer essa Árvore de Categorias. Foram tomadas como base as categorias principais escolhidas e especificadas na Tabela 3, e, após sucessivas iterações, juntou-se em uma lista todas as subcategorias dessas categorias principais, mantendo-se apenas as subcategorias que não possuem divisões.

Com a lista de todas as subcategorias desejadas disponível, novas requisições para a API do Mercado Livre foram feitas. Dessa vez, usando por milhares de vezes o comando da Figura 14, que retorna todos os atributos de uma determinada categoria. Com o auxílio de códigos em *Python*, foram feitas transformações nesses dados: uma nova lista foi feita, descrevendo junto a cada subcategoria todos os seus possíveis atributos.

```
$ curl https://api.mercadolibre.com/categories/<id_cat>/attributes
```

Figura 14 – Comando para fazer *download* de todos os atributos de uma determinada subcategoria do Mercado Livre brasileiro. `<id_cat>` é um código único que identifica a subcategoria no conjunto de todas as categorias.

Em seguida, foi criada uma nova lista com todos os atributos das subcategorias desejadas e o número de ocorrências em que esses atributos estão descritos nessas subcategorias. A Figura 15 mostra os 10 atributos que mais aparecem nesse conjunto.

```
{
  "BRAND": 4892,
  "MODEL": 4892,
  "PACKAGE_HEIGHT": 4892,
  "PACKAGE_WIDTH": 4892,
  "PACKAGE_LENGTH": 4892,
  "PACKAGE_WEIGHT": 4892,
  "ITEM_CONDITION": 4892,
  "GTIN": 4890,
  "MPN": 4890,
  "SELLER_SKU": 4890,
  ...
}
```

Figura 15 – Os 10 atributos que são mais aceitos entre as subcategorias sem descendentes das categorias escolhidas e o número de subcategorias em que cada um aparece.

Por último, foram escolhidos os atributos de maior interesse comercial da lista da Figura 15 para serem as classes. O critério principal para a escolha foi selecionar aqueles atributos que descrevem o maior número de subcategorias diferentes. Após essa primeira etapa de escolha, foram feitos alguns refinamentos de forma manual para atingir o objetivo:

- Remover atributos que raramente são questionados pelos clientes, tais como PACKAGE_HEIGHT (altura do pacote);
- Remover atributos internos de uso privado do lojista e do Mercado Livre, tais como GTIN (código identificador de produtos interno do Mercado Livre);
- Dar preferência a atributos frequentemente perguntados pelos clientes;
- Dar preferência a atributos genéricos, ou seja, que se aplicam a uma grande variedade de produtos de diferentes categorias.

Ao todo, foram selecionados 39 atributos para serem as classes do problema, descritos pela Tabela 4. Uma classe especial composta de exemplos que não pertencem à nenhuma das outras classes, a *out_of_scope*, também foi escolhida. Portanto, a base de dados será composta por 40 classes.

3.3 Coleta de Dados

Como os modelos a serem treinados são modelos de Aprendizado Supervisionado, faz-se necessária a criação de uma base de dados rotulada. Neste trabalho, as etapas de

Tabela 4 – Classes escolhidas para serem representadas na base de dados

Classe		
BRAND	HEIGHT	SIZE
DETAILED_MODEL	ALPHANUMERIC_MODEL	MODEL
LINE	DIAMETER	ACCESSORIES_INCLUDED
COMPATIBLE_BRANDS	IS_KIT	UNITS_PER_PACKAGE
DEPTH	MAIN_MATERIAL	SHAPE
COLOR	ORIGIN	FABRIC_DESIGN
THICKNESS	SALE_FORMAT	WIDTH
RELEASE_YEAR	POWER	MATERIALS
MAX_WEIGHT_SUPPORTED	LENGTH	WEIGHT
COMPOSITION	WAIST_CIRCUMFERENCE	CHEST_CIRCUMFERENCE
PART_NUMBER	QUANTITY	PATTERN_NAME
VOLUME_CAPACITY	COMPATIBLE_MODELS	MATERIAL
VOLTAGE	TOTAL_LENGTH	HIP_CIRCUMFERENCE
out_of_scope		

Coleta de Dados e de Criação da Base Rotulada foram feitas de forma intercalada, isto é, enquanto não se atingiu um limiar mínimo de 15 exemplos para cada classe não foi finalizada a etapa de Coleta de Dados.

É importante esclarecer que todas as perguntas foram coletadas de acordo com a Lei Geral de Proteção de Dados Pessoais (LGPD) (BRASIL, 2018). Dessa forma, as perguntas coletadas são dados impessoais, isto é, não contém informações sobre pessoas identificáveis, e portanto não permitem a identificação do autor ou de funcionários que trabalham no atendimento das lojas.

As perguntas foram coletadas de três diferentes fontes:

- Em sua maioria, a partir do banco de dados da GoBots.

  Como a empresa fornece o serviço de resolução de dúvidas de clientes sobre produtos em plataformas de *e-commerce* há alguns anos, muitos exemplos de perguntas feitas por clientes reais estão armazenados no banco de dados interno da empresa, do tipo *MongoDB*[3]. Essas perguntas foram consideradas na criação da base de dados usada neste trabalho por serem perguntas de clientes reais. Por isso, espera-se que a qualidade da base de dados (e consequentemente, dos modelos treinados fazendo uso dela) aumente. Usando o *MongoDB*, foi feita a exportação desses dados para um arquivo CSV.

  As perguntas coletadas por meio dessa fonte são dados de empresas que contrataram o serviço de automação da GoBots. Portanto, as empresas consentiram a coleta dessas perguntas para processamento posterior, o que inclui o processamento realizado neste trabalho. Uma parcela dessas perguntas pode ter sido feita em outros sites de *e-commerce* além do Mercado Livre e incorporada ao banco de dados da GoBots em processamentos de dados anteriores.

[3] <https://www.mongodb.com/pt-br>

- A partir de perguntas de clientes no Mercado Livre que estavam disponíveis de forma pública, mas que não faziam parte do banco de dados da GoBots.

  Algumas das classes selecionadas, como a COMPOSITION, são usadas apenas como atributos de categorias como Vestuário, que são constituídas de produtos que não são vendidos por clientes da GoBots. Por esse motivo, não existem exemplos dessas classes no banco de dados da GoBots. Para contornar esse problema, foi preciso pesquisar por produtos de Vestuário e outras categorias subrepresentadas no próprio site do Mercado Livre, e coletar perguntas reais que poderiam servir de exemplos para a criação da base de dados deste trabalho de forma manual.

  As perguntas coletadas por meio dessa fonte também foram coletadas como dados impessoais e seguindo os Termos e Condições do Programa de Desenvolvedores do Mercado Livre[4].

- A partir de criação manual, feita pelo próprio autor deste trabalho.

  A criação manual possui vieses pessoais, pois não representa as variadas formas de se perguntar que clientes reais podem criar, e isso pode afetar negativamente a qualidade da base de dados. No entanto, foi uma medida necessária para garantir que todas as 40 classes atingissem o limiar mínimo de 15 exemplos.

3.4 Criação da Base de Dados Rotulada

A Criação da Base de Dados Rotulada foi feita individualmente pelo autor deste trabalho. As perguntas que vieram do banco de dados da GoBots foram rotuladas por meio da execução de um *script Python* disposto em um *notebook Jupyter*[5] hospedado na plataforma *Google Colaboratory*[6]. Os *notebooks Jupyter* são uma forma interativa de executar programas escritos em *Python*. Com eles, é possível executar trechos específicos de um programa ao invés do programa inteiro. O *Google Colaboratory* é uma plataforma que fornece acesso gratuito à computação em nuvem e possui fácil integração com o servidor de hospedagem de arquivos *Google Drive*[7], onde ficou armazenado o arquivo da base de dados.

O *script* funciona da forma representada no Algoritmo 1.

[4] <https://developers.mercadolivre.com.br/pt_br/termos-e-condicoes>
[5] <https://jupyter.org>
[6] <https://colab.google>
[7] <https://drive.google.com>

Algoritmo 1: Seleção e Rotulação de Exemplos

Dados : Arquivo CSV 'dados.csv'

Resultado: Base de dados rotulada em JSON

```
1  Abra o arquivo CSV 'dados.csv'
2  Conte o número de linhas no arquivo CSV e armazene em total_linhas
3  Gere um número aleatório, num_aleatorio, entre 0 e total_linhas - 1
4  while True do
5      Leia a linha no índice num_aleatorio do arquivo CSV
6      Escreva "Exemplo a ser anotado: " e o conteúdo da linha selecionada
7      Leia "Escolha um número entre 0 e 39 para classificar a sentença: "
8      if classe_escolhida está entre 0 e 39 then
9          Adicione a linha {"exemplo": linha selecionada, "classe": classe_escolhida} ao
            objeto JSON
10         Salve o objeto JSON em 'dados_rotulados.json'
11     else
12         Escreva "Classe inválida"
13     end
14 end
15 Retorna
```

A geração de um número aleatório se faz importante porque os dados do arquivo CSV foram coletados em ordem cronológica, do mais antigo ao mais novo. Por isso, houve a intenção de proporcionar uma maior variabilidade nos dados que poderiam ser rotulados a partir dessa fonte.

O terceiro, o quarto, o quinto e o sexto passos foram repetidos indefinidamente até que foi observado um acúmulo de exemplos de classes como BRAND e COLOR. Ao mesmo tempo, percebeu-se que classes como COMPOSITION e HIP_CIRCUMFERENCE estavam subrepresentadas com menos de 15 exemplos. Neste instante, a rotulação de exemplos a partir do banco de dados da GoBots foi finalizada, e prosseguiu-se com a rotulação de exemplos a partir dos dados coletados via coleta manual no Mercado Livre e a partir dos dados coletados via criação manual.

Com a intenção de realizar uma segunda rodada de experimentos, na qual o desempenho dos modelos classificadores quando classes semelhantes são agrupadas é avaliado, foi feita uma aglutinação de classes. A aglutinação de classes foi pensada como um procedimento para diminuir o número de situações em que mais de uma classe poderia ser atribuída corretamente à mesma pergunta. Os modelos treinados não são capazes de classificar uma pergunta em mais de uma classe ao mesmo tempo, o que prejudica as métricas de avaliação.

Uma situação em que uma pergunta poderia ser classificada em mais de uma classe é quando, por exemplo, é feita a pergunta "Este é o valor do par ou da unidade?". Essa pergunta poderia ser classificada tanto como ***UNITS_PER_PACKAGE*** quanto como ***SALE_FORMAT***. A classe correta só poderia ser descoberta através de informações extras que os modelos não possuem acesso, como a subcategoria daquele produto em questão.

A Figura 16 retrata as aglutinações de classes que foram feitas. O novo arranjo propiciou que perguntas parecidas fizessem parte da mesma classe.

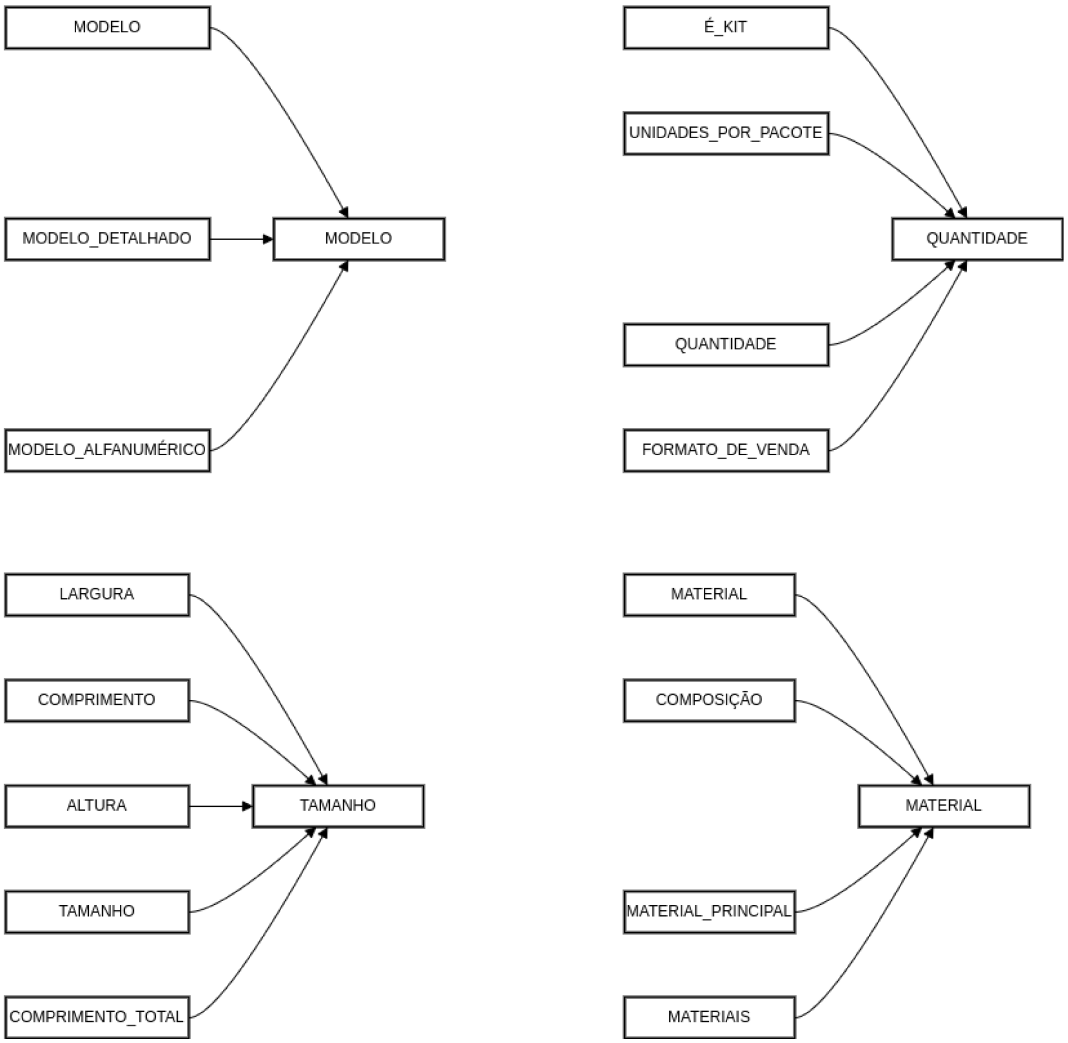

Figura 16 – Ao início de cada seta, as classes originais que compunham a base de dados com 40 classes. Ao final de cada seta, as classes resultantes da aglutinação, que passaram a compor a base de dados com 28 classes.

3.5 Pré-processamento dos Dados

A etapa de pré-processamento dos dados foi feita de forma diferente a depender da arquitetura de Transformador utilizada. Para a arquitetura que usa o modelo BERT para

fazer a classificação, essa etapa foi feita apenas com o Tokenizador do modelo BERTimbau Base[8], uma variação do modelo BERT pré-treinada em um grande corpus de dados da Internet escritos em Português do Brasil. Para a arquitetura que usa o modelo DIETClassifier para fazer a classificação, essa etapa foi feita com o Tokenizador do modelo BERTimbau Base ou com o Tokenizador do modelo BERT Multilingual Base[9], e também com outros componentes extras, explicados na Subseção 3.5.2. O BERT Multilingual Base é uma versão do BERT original desenvolvida pela *Google* que apresenta suporte a 104 idiomas, incluindo o Português.

As etapas de pré-processamento citadas nesta Seção foram realizadas a partir da execução dos mesmos *scripts* usados para o treinamento dos Algoritmos de Classificação: para a arquitetura que usa o modelo BERT para fazer a classificação, o *script* escrito em *Python* que chama funções da biblioteca *HuggingFace Transformers*[10]; para a arquitetura que usa o modelo DIETClassifier para fazer a classificação, o *script* de linha de comando *rasa train*, incluso com o *software* RASA[11].

Os exemplos coletados a partir da base de dados da GoBots já estavam sem sinais de pontuação, uma vez que essa etapa de pré-processamento já foi feita quando esses exemplos foram tratados pelo *software* da empresa.

3.5.1 BERT

O Tokenizador do modelo BERTimbau Base funciona dividindo a sentença fornecida como entrada em elementos de um vetor, dividindo palavras específicas em múltiplos *tokens* (*Stemming*), associando um número de identificação único a cada *token*, e acrescentando *tokens* especiais que indicam o início e o fim da sentença. Esse Tokenizador não aplica muitos dos passos da etapa de pré-processamento de dados vistos na Seção 2.4.4, pois possui as seguintes caracerísticas:

- *Case sensitivity*: “esse” e “Esse” são tratados como *tokens* diferentes, por exemplo, o que indica a não aplicação de *lowercasing*;
- Manutenção de *stopwords*: palavras como “os” e “de” são consideradas *tokens* válidos, por exemplo;
- Manutenção da pontuação e/ou números: o ponto de interrogação, por exemplo, é tratado como um *token* individual;
- Não aplicação de *Lemmatization*: “bom” e “melhor” são *tokens* diferentes, por exemplo.

[8] <https://huggingface.co/neuralmind/bert-base-portuguese-cased>
[9] <https://huggingface.co/bert-base-multilingual-cased>
[10] <https://github.com/huggingface/transformers/>
[11] <https://rasa.com/>

As Figuras 17 e 18 retratam como funciona o processo de Tokenização a partir do Tokenizador do modelo BERTimbau Base relatado. Na Figura 17, *tokenizer* é o Tokenizador do modelo BERTimbau Base, que foi baixado em linhas de código anteriores. Ao receber uma sentença como entrada, ele retorna um dicionário com as chaves "input_ids", "token_type_ids" e "attention_mask" (Figura 18). O valor da chave "input_ids" é uma lista de números de identificação únicos, onde cada número representa um *token* e está mapeado internamente à uma representação vetorial daquele *token*. Esse dicionário é o que é efetivamente fornecido como entrada ao algoritmo de classificação em treinamento. Se a função *convert_ids_to_tokens()* é chamada, é possível ver quais *tokens* a lista de números de identificação únicos da chave "input_ids" representa (Figura 18).

```
1  encoded_text = tokenizer("Esse item é feito de silicone?")
2  tokens = tokenizer.convert_ids_to_tokens(encoded_text["input_ids"])
3
4  print(encoded_text)
5  print(tokens)
```

Figura 17 – Código em *Python* que faz uso da biblioteca *HuggingFace Transformers* para aplicar um Tokenizador do modelo BERTimbau Base à uma pergunta de um cliente real.

```
{'input_ids': [101, 3758, 18685, 253, 2160, 125, 7296, 10132,
22279, 136, 102], 'token_type_ids': [0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0], 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}

['[CLS]', 'Esse', 'item', 'é', 'feito', 'de', 'sil', '##icon',
'##e', '?', '[SEP]']
```

Figura 18 – Saída do código *Python* da Figura 17, mostrando como a pergunta de um cliente real foi pré-processada pelo Tokenizador do modelo BERTimbau Base.

3.5.2 DIETClassifier

Como descrito anteriormente, o uso de componentes extras em conjunto com a arquitetura de Transformador *DIETClassifier* diferencia a sua etapa de pré-processamento dos dados em comparação com a etapa de pré-processamento dos dados usada com o BERT. Os componentes extras estão enunciados a seguir, ordenados do primeiro a ser usado ao último:

- uma camada de *WhitespaceTokenizer*;

O *WhitespaceTokenizer* é um Tokenizador que separa as palavras nos espaços em branco. Além disso, sinais de pontuação no início ou no fim de palavras são removidos. Acentos são mantidos.

A Figura 19 retrata como o Tokenizador *WhitespaceTokenizer* funciona. Uma lista de *tokens* foi gerada, e cada *token* representa uma palavra da sentença "Esse item é feito de silicone?". O acento agudo em "é" foi mantido, mas o ponto de interrogação ao final da sentença foi removido, pois ele está ao final de uma palavra. Um *token* especial, chamado de "__CLS__", foi introduzido para indicar o fim da sentença.

- uma camada de *LanguageModelFeaturizer* (constituída pelo Tokenizador do modelo BERTimbau Base ou pelo Tokenizador do modelo BERT Multilingual Base, a depender do experimento);

  O *LanguageModelFeaturizer* funciona da mesma forma que o Tokenizador descrito na Subseção 3.5.1.

- duas camadas de *CountVectorsFeaturizer*.

  O *CountVectorsFeaturizer* é um componente responsável pela transformação dos *tokens* em representações vetoriais. Ele cria representações vetoriais no formato *bag-of-words* da entrada recebida. Na arquitetura de Transformador que usa o modelo DIETClassifier para fazer a classificação, o componente *CountVectorsFeaturizer* foi usado duas vezes: na primeira vez, representações *bag-of-words* foram feitas a nível de palavras; na segunda vez, representações *bag-of-words* foram feitas a nível de sequências de 1 a 5 caracteres.

  A Figura 20 retrata como o *CountVectorsFeaturizer* funciona quando aplicado após uma camada de *WhitespaceTokenizer* e uma camada de *LanguageModelFeaturizer*.

  Na parte central da Figura 20, são mostrados os vetores gerados após a aplicação da primeira camada de *CountVectorsFeaturizer*, sendo o primeiro vetor um vetor de *tokens* e o segundo vetor um vetor que indica a quantidade de vezes que cada *token* aconteceu em cada sentença. Os vetores gerados apresentam grande diferença em relação à lista de *tokens* da Figura 19. Todos os *tokens* ficaram com letras minúsculas. A palavra "silicone" foi dividida em múltiplos *tokens*, por causa do *LanguageModelFeaturizer*. 'cls' e 'sep' são os *tokens* especiais "__CLS__" e "__SEP__", usados para indicar o fim de cada sentença e o início de uma sentença nova, respectivamente.

  Na parte inferior da Figura 20, são mostrados os vetores gerados após a aplicação da segunda camada de *CountVectorsFeaturizer*. Os *tokens* apresentados são constituídos de todas as sequências de 5 caracteres possíveis de serem criadas a partir das sentenças fornecidas como entrada. Os símbolos # representam os *tokens* criados a partir da divisão de uma palavra em múltiplos *tokens* ("silicone" se tornou "sil", "##icon", "##e"). Todos os *tokens* ficaram com letras minúsculas. '[cls]' e '[sep]'

são os mesmos *tokens* especiais explicados anteriormente. Os vetores constituídos de 1 e 0 na parte inferior indicam quantas vezes cada sequência de 5 caracteres acontece em cada sentença fornecida como entrada.

```
"Esse item é feito de silicone?"
```

```
['Esse', 'item', 'é', 'feito', 'de', 'silicone', '__CLS__']
```

Figura 19 – Exemplo de aplicação do Tokenizador *WhitespaceTokenizer*.

```
["Esse item é feito de silicone?",
 "Esse item é feito de plástico?"]
```

(a) Sentenças de exemplo fornecidas como entrada aos tokenizadores.

```
['cls' 'de' 'esse' 'feito' 'icon' 'item' 'plástico' 'sep' 'sil']
[[1 1 1 1 1 1 0 1 1]
[1 1 1 1 0 1 1 1 0]]
```

(b) Representações *bag-of-words* a nível de palavras geradas a partir da aplicação dos tokenizadores sobre as sentenças de exemplo fornecidas como entrada.

```
['##e[s' '##ico' '#e[se' '#icon' '[cls]' '[sep]' ']esse' 'cls]e'
 'co[se' 'con##' 'deplá' 'desil' 'e[sep' 'eitem' 'eitod' 'eméfe'
 'eplás' 'esil#' 'essei' 'feito' 'ico[s' 'icon#' 'il##i' 'itemé'
 'itode' 'l##ic' 'ls]es' 'lásti' 'méfei' 'n##e[' 'o[sep' 'odepl'
 'odesi' 'on##e' 'plást' 's]ess' 'seite' 'sil##' 'sseit' 'stico'
 'teméf' 'tico[' 'todep' 'todes' 'ástic' 'éfeit']
[[1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 0
0 1 1 0 1 1 1 1 0 1 0 0 1 0 1]
[0 0 0 0 1 1 1 1 1 0 1 0 0 1 1 1 1 0 1 1 1 0 0 1 1 0 1 1 1 0 1
1 0 0 1 1 1 0 1 1 1 1 1 0 1 1]]
```

(c) Representações *bag-of-words* a nível de sequências de 5 caracteres geradas a partir da aplicação dos tokenizadores sobre as sentenças de exemplo fornecidas como entrada.

Figura 20 – Resultado após aplicação dos Tokenizadores *WhitespaceTokenizer*, *LanguageModelFeaturizer* e duas camadas de *CountVectorsFeaturizer* em sequência sobre duas sentenças de exemplo.

3.6 Treinamento dos Algoritmos de Classificação

Esta seção descreve o método de treinamento dos dois Algoritmos de Classificação usados: BERT e DIETClassifier. Ambos os Algoritmos de Classificação foram treinados por meio de *fine-tuning*, conceito explicado na Subseção 2.4.6.1.

As seções de treinamento foram feitas através da plataforma *Google Colaboratory*, em virtude da disponibilidade gratuita de placas de vídeo dedicadas de alto poder computacional. O acesso a esse recurso tornou possível que cada treinamento levasse no máximo 2 horas de duração.

3.6.1 BERT

Ao invés de usar o Transformador BERT original, apresentado em Devlin et al. (2019), optou-se por usar o BERTimbau Base, apresentado em Souza, Nogueira e Lotufo (2020). Essa escolha foi feita com base nos resultados de experimentos anteriores de outros projetos da empresa GoBots, nos quais chegou-se à conclusão de que modelos treinados com *tokens* em Português apresentam melhores resultados para aplicações em Português, como é o caso deste trabalho.

Os modelos BERTimbau são definidos pelos seus criadores como "modelos BERT para Português Brasileiro, treinados usando dados do brWaC, um grande *corpus* de páginas Web" (SOUZA; NOGUEIRA; LOTUFO, 2020). À época, os modelos BERTimbau avançaram o estado-da-arte das abordagens BERT multilíngua e monolíngua. O termo Base se refere ao tamanho do modelo menor, de 12 camadas e 110 milhões de parâmetros. Ambos os modelos foram disponibilizados pelos autores para *download* gratuito em bibliotecas *open-source*.

Todas as etapas que usaram o Algoritmo de Classificação BERTimbau Base, do pré-processamento dos dados à Avaliação dos Classificadores, foram feitas por meio de chamadas às funções da biblioteca *HuggingFace Transformers*, escrita na linguagem *Python*. Por meio dela, é possível definir os hiperparâmetros de treino e efetivamente treinar o Algoritmo de Classificação. Para isso, basta que a base de dados esteja em um formato compatível, como JSON ou CSV.

A biblioteca *HuggingFace Transformers* permite a alteração dos hiperparâmetros de treino, tais como *batch_size* (a quantidade de exemplos vistos a cada passo de treinamento) e *learning_rate* (o fator pelo qual os pesos do Algoritmo de Classificação em treinamento são multiplicados durante o processo). Por isso, o BERTimbau Base foi treinado em 2 configurações:

- na primeira, identificada como HF1, com quantidade de exemplos a cada passo = 8 exemplos.

- na segunda, identificada como HF2, com quantidade de exemplos a cada passo = 1 exemplo.

Ao alterar o número de exemplos que o Algoritmo de Classificação em treinamento vê a cada passo, aumenta-se o número de passos totais para que ele veja todos os exemplos. Assim, ao ver menos exemplos por vez, o Algoritmo é otimizado mais vezes. Foram feitos 5 treinamentos em cada configuração para buscar validade estatística (ou seja, para desprezar efeitos do acaso nas medidas de avaliação que aquele Algoritmo de Classificação atingiu).

Duas rodadas de experimentos foram feitas com o BERTimbau Base: na primeira, as duas configurações HF1 e HF2 foram treinadas na base de dados rotulada em 40 classes. Na segunda rodada de experimentos, foi escolhida a configuração com o melhor desempenho nas medidas de avaliação da primeira rodada para ser treinada na base de dados rotulada em 28 classes.

3.6.2 DIETClassifier

O outro algoritmo de classificação que foi treinado foi o Transformador DIETClassifier, apresentado em Bunk et al. (2020). Um dos motivos pelo qual ele foi escolhido é por causa da modularidade da sua arquitetura, que permite que sejam usados Tokenizadores de variados métodos de pré-treino, assim como outros componentes, em conjunto com ele. Outro motivo para a escolha é que os autores alegam que o melhor modelo DIETClassifier supera um modelo BERT que passou por *fine-tuning* e ainda é seis vezes mais rápido de se treinar (BUNK et al., 2020).

Todas as etapas que usaram o Algoritmo de Classificação DIETClassifier, do pré-processamento dos dados à Avaliação dos Classificadores, foram feitas por meio do *software* de linha de comando RASA *Open Source*. A instalação[12] se dá pelo Gerenciador de Pacotes do *Python*, o PIP. Depois disso, é preciso criar um projeto com o comando *rasa init* e procurar os arquivos recém-criados *data/nlu.yml* e *config.yml*. No primeiro arquivo são inseridos todos os exemplos da base de dados. No segundo arquivo são inseridas as especificações de todos os componentes das etapas de pré-processamento dos dados e de Treinamento dos Algoritmos de Classificação. Após o preenchimento dos dois arquivos, é possível começar o treinamento do DIETClassifier com o comando *rasa train nlu*.

O *software* RASA *Open Source* permite a adição e personalização dos componentes do Transformador que será treinado, tais como o *FallbackClassifier*, que retorna uma classe especial indicando que não foi possível prever com assertividade a qual classe aquele exemplo pertence caso a pontuação *confidence* não tenha atingido um valor mínimo de 0.7. Além disso, existe o parâmetro *constrain_similarities*, que pode ser aplicado ao Algoritmo

[12] <https://rasa.com/docs/rasa/installation/installing-rasa-open-source>

de Classificação para que seja executada uma função extra para diminuir a similaridade entre a sentença fornecida como entrada e sentenças de outras classes. O DIETClassifier foi treinado em 3 configurações:

- na primeira, identificada como RASA1, o *LanguageModelFeaturizer* usado foi o BERT Multilingual Base, o componente CountVectorsFeaturizer foi configurado para criar representações *bag-of-words* de sequências de 1 a 4 caracteres, o parâmetro *constrain_similarities* foi definido como verdadeiro e o componente *FallbackClassifier* foi utilizado;
- na segunda, identificada como RASA2, o *LanguageModelFeaturizer* usado foi o BERTimbau Base, o componente CountVectorsFeaturizer foi configurado para criar representações *bag-of-words* de sequências de 3 a 5 caracteres, o parâmetro *constrain_similarities* foi definido como falso e o componente *FallbackClassifier* não foi utilizado;
- na terceira, identificada como RASA3, o *LanguageModelFeaturizer* usado foi o BERTimbau Base, o componente CountVectorsFeaturizer foi configurado para criar representações *bag-of-words* de sequências de 3 a 5 caracteres, o parâmetro *constrain_similarities* foi definido como verdadeiro e o componente *FallbackClassifier* foi utilizado.

Duas rodadas de experimentos foram feitas com o DIETClassifier: na primeira, as três configurações RASA1, RASA2 e RASA3 foram treinadas na base de dados rotulada em 40 classes. Na segunda rodada de experimentos, foi escolhida a configuração com o melhor desempenho nas medidas de avaliação da primeira rodada para ser treinada na base de dados rotulada em 28 classes. Foram feitos 5 treinamentos em cada configuração para buscar validade estatística.

3.7 Medidas e Método de Avaliação dos Classificadores

Os testes (e consequentemente, a geração de métricas) foram executados pela biblioteca *HuggingFace Transformers* (no caso do BERTimbau Base) ou através de um comando do *software RASA* (no caso do DIETClassifier). Na biblioteca *HuggingFace Transformers*, basta que seja usado o parâmetro *compute_metrics=* e seja fornecida a função responsável por calcular as métricas. Além das métricas geradas ao final, a biblioteca *HuggingFace Transformers* calcula métricas preliminares toda vez que todos os exemplos de Treinamento são vistos. Contudo, as métricas preliminares foram descartadas, pois não são importantes para os objetivos deste estudo. No *software RASA Open Source*, a geração de métricas é feita por padrão, sem a adição de novos parâmetros.

Não foi realizada Validação Cruzada nos experimentos com o BERTimbau como Algoritmo de Classificação. A estratégia empregada foi a separação dos dados de Treino e Teste. Os exemplos de Teste, que compõem 20% da base de dados total, foram usados para os testes, e não influenciaram nos pesos do algoritmo de classificação em treinamento. Os exemplos de Treino, que compõem 80% da base de dados total, foram usados para o treinamento. Os experimentos foram feitos desta forma por causa da alta demanda de recursos computacionais para realizar Validação Cruzada em treinamentos de modelos BERT, que aumentaria em 5 vezes o tempo necessário para cada um dos 5 treinamentos para ambas as configurações HF1 e HF2. Além disso, a biblioteca *HuggingFace Transformers* não apresenta suporte nativo a Validação Cruzada, o que dificulta a implementação.

Foi realizada Validação Cruzada nos experimentos com o DIETClassifier, pelo fato deste Algoritmo de Classificação não demandar tantos recursos computacionais para isso quanto o BERTimbau Base. A base de dados definida no arquivo *data/nlu.yml* foi dividida em 5 partes: a cada treinamento, 4 partes eram definidas como exemplos de Treino, e 1 parte era definida como exemplos de Teste, em um processo iterativo. Após todas as partes terem sido definidas como exemplos de Teste, o Treinamento foi finalizado. As métricas de avaliação daquele Treinamento como um todo eram tomadas a partir da média das métricas de avaliação de cada etapa de Validação Cruzada.

É importante salientar que o uso de dois métodos diferentes de avaliação faz com que as métricas encontradas nos experimentos com o algoritmo de classificação BERTimbau não possam ser comparadas com as métricas encontradas nos experimentos com o algoritmo de classificação DIETClassifier. O principal motivo para não se realizar Validação Cruzada na avaliação dos modelos BERTimbau é o fato de que essa técnica aumenta em cinco vezes o número de treinamentos, o que faria que o tempo total gasto com treinamento e avaliação dos modelos BERTimbau passasse de 30 horas para 150 horas e prejudicaria a conclusão dos experimentos.

Além das métricas Acurácia, Precisão, Revocação e F1-Score, as Matrizes de Confusão, como são um bom recurso para comparar o desempenho dos modelos classe a classe e para enxergar pontos fortes e fracos, também foram geradas. A biblioteca *HuggingFace Transformers* não cria Matrizes de Confusão por padrão. Por isso, quando treinando o Transformador que tem como Algoritmo de Classificação o BERTimbau Base, elas foram produzidas com chamadas aos elementos confusion_matrix() e ConfusionMatrixDisplay() da biblioteca *sklearn.metrics*[13]. Essas chamadas foram adicionadas à função compute_metrics(). No *software RASA Open Source*, as Matrizes de Confusão são geradas sem adição de código extra.

As métricas F1, Precisão e Revocação foram ponderadas, ou seja, os valores das métricas dependeram também do número de exemplos daquela classe na base de dados.

[13] <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html>

Essa escolha pela ponderação foi feita porque a base de dados é desbalanceada, com algumas classes tendo mais exemplos do que outras. A ponderação é feita fazendo um somatório da multiplicação entre uma métrica de uma classe pela razão entre o número de exemplos daquela classe e o número de exemplos total, como mostra a Equação 3.1.

$$\text{Métrica\_ponderada} = \sum \text{Métrica}_k \cdot \frac{\text{Quant\_exemplos}_k}{\text{Quant\_exemplos}_{\text{total}}} \tag{3.1}$$

3.8 Considerações Finais

Para o desenvolvimento deste estudo, foram aplicadas diferentes abordagens de Classificação de Texto a um problema prático consistido em definir a qual atributo uma determinada pergunta de um usuário real do Mercado Livre se refere.

Na primeira etapa, 39 atributos importantes foram selecionados para serem as classes para as quais as arquiteturas de Transformadores seriam treinadas. Na segunda etapa, dados não-rotulados foram coletados de diferentes fontes. Na terceira etapa, esses dados foram rotulados com o objetivo de criar uma base de dados rotulada. Essa base de dados consiste de 1419 exemplos. Uma segunda rotulação de exemplos com 28 atributos foi feita para que fossem possíveis novos experimentos.

Em seguida, foram definidas formas diferentes de se realizar o pré-processamento dos dados. Para a arquitetura de Transformador que faz uso do classificador BERTimbau Base, apenas o Tokenizador desse classificador foi utilizado. No entanto, para a arquitetura de Transformador que faz uso do classificador DIETClassifier, outros componentes como o *WhitespaceTokenizer* e o *CountVectorsFeaturizer* também foram testados em conjunto com o Tokenizador do BERT Multilingual Base e o Tokenizador do BERTimbau Base.

Finalmente, foi feito o treinamento: 2 configurações diferentes de Transformadores com BERTimbau Base e 3 configurações diferentes de Transformadores com DIETClassifier foram treinadas, 5 vezes cada para que os resultados tenham validade estatística. As configurações com as melhores métricas passaram por uma nova rodada de experimentos sendo treinadas na base de dados com 28 classes. O cálculo de métricas como Acurácia, Precisão, Revocação e F1-Score, assim como a elaboração de Matrizes de Confusão, foi adicionado ao código que faz o treinamento para que fosse possível comparar os modelos treinados.

4 Resultados

Este capítulo tem como objetivo descrever e analisar os resultados obtidos pelas etapas descritas no capítulo anterior, desde a definição das classes até as métricas e métodos de avaliação dos classificadores treinados.

A Seção 4.1 discorre sobre os atributos escolhidos para serem classes e os motivos para que sua escolha tenha sido feita.

A Seção 4.2 apresenta a representatividade de cada fonte de coleta de dados no conjunto de todos os exemplos rotulados, enquanto a Seção 4.3 mostra a distribuição de exemplos em cada classe da base de dados rotulada, antes e depois da execução do procedimento de aglutinação de classes.

A Seção 4.4 mostra a divisão da base de dados em subconjuntos de Treino e Teste, apresenta os resultados da avaliação de todos os modelos, considerando tanto a base de dados com 40 classes quanto a base de dados com 28 classes, traça comparações entre os modelos e sumariza as classes em que os modelos apresentaram o melhor e o pior desempenho.

A Seção 4.5 visa elucidar os motivos por trás do desempenho ruim de todos os modelos em uma classe específica.

4.1 Definição das Classes

Como visto na Seção 3.2, quatro critérios foram aplicados nos atributos que aparecem no maior número de subcategorias do Mercado Livre (mostrados na Figura 15) para definir os 39 atributos (ou classes) com maior interesse comercial.

A Figura 21 mostra, por meio de uma Nuvem de Palavras, a representatividade dos 39 atributos de maior interesse comercial escolhidos em termos do número de subcategorias do Mercado Livre em que esses atributos podem ser usados. Percebe-se pela Figura 21 e pela análise do JSON, extraído a partir das requisições à API do Mercado Livre, que **BRAND** e **MODEL** são os atributos mais genéricos que se encaixam nos critérios aplicados. Ambos os atributos podem ser usados em 4892 subcategorias de produtos.

Por outro lado, os atributos menos genéricos que foram definidos como classes, pois se encaixam nos critérios aplicados, são **CHEST_CIRCUMFERENCE** e **COMPATIBLE_MODELS**. Ambos os atributos podem ser usados em 87 subcategorias de produtos. **CHEST_CIRCUMFERENCE**, traduzido como “circunferência do busto” em Português, se aplica apenas a itens de Vestuário, setor em que a empresa GoBots

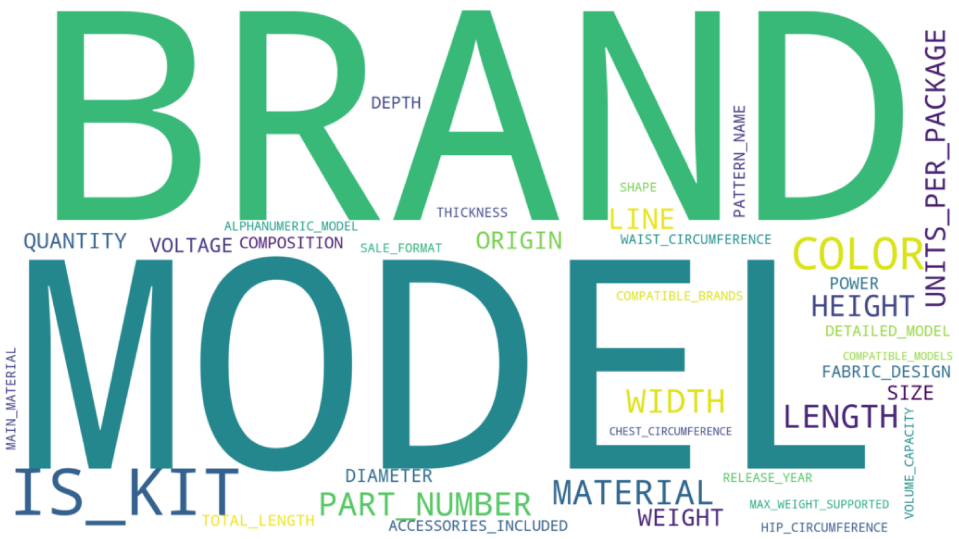

Figura 21 – Nuvem de Palavras com os 39 atributos escolhidos para serem classes. O tamanho das palavras reflete o número de subcategorias do Mercado Livre em que esses atributos podem ser usados.

ainda não possui clientes, e por isso representa um bom experimento para avaliar a possibilidade de atração de clientes do setor. **COMPATIBLE_MODELS**, de forma similar, é usado para descrever apenas a compatibilidade de peças e acessórios, como motores e válvulas, com modelos de objetos maiores, como veículos e eletrodomésticos. Portanto, o baixo número de subcategorias onde esses atributos podem ser usados se justifica pela especificidade de suas aplicações.

4.2 Coleta de Dados

A Tabela 5 apresenta uma estimativa de qual porcentagem cada fonte de coleta de dados representa em relação a base de dados criada. A maior parte das perguntas, estimada em 70%, foi coletada a partir do banco de dados da GoBots, que já existia anteriormente. Os outros 30% foram coletados por meio de coleta manual no mercadolivre.com.br ou através de criação manual, em um esforço para garantir que cada classe possuísse pelo menos 15 exemplos. A coleta de dados através de criação manual foi de apenas 10%, com o objetivo de fazer com que os exemplos vistos fossem, em sua grande maioria, representações da forma de perguntar de clientes reais. Em virtude do alto número de exemplos, não é possível saber com exatidão quantos exemplos foram coletados de cada fonte.

Tabela 5 – Fontes para Coleta de Dados e estimativa de sua representatividade na base de dados

Fonte dos Dados	
Banco de Dados da GoBots	70%
Coleta manual no mercadolivre.com.br	20%
Criação manual	10%
Total	1419

4.3 Criação da Base de Dados Rotulada

Como visto na Seção 3.4, a criação da base de dados rotulada ocorreu em duas rodadas. Na primeira rodada, as perguntas foram rotuladas em relação aos atributos a que elas se referem, considerando 40 atributos descritores de produtos do Mercado Livre. Na segunda rodada, atributos similares foram aglutinados em um único atributo, com o objetivo de minimizar a ocorrência de situações em que uma pergunta poderia ser classificada em mais de um atributo ao mesmo tempo. Isso proporcionou com que uma nova rotulação, baseada em 28 classes ao invés de 40, pudesse ser feita.

A Tabela 6 apresenta a quantidade de exemplos rotulados para cada classe considerando a configuração inicialmente proposta na primeira rodada de experimentos, constituída de 40 classes:

Tabela 6 – Quantidade de exemplos rotulados para cada classe na configuração com 40 classes

Classe	**Exemplos**	**Classe**	**Exemplos**
BRAND	**105**	MODEL	**32**
IS_KIT	**18**	COLOR	**94**
WIDTH	**32**	LENGTH	**29**
PART_NUMBER	**21**	MATERIAL	**58**
HEIGHT	**41**	LINE	**23**
UNITS_PER_PACKAGE	**22**	ORIGIN	**25**
RELEASE_YEAR	**27**	WEIGHT	**22**
QUANTITY	**23**	VOLTAGE	**60**
SIZE	**80**	DIAMETER	**26**
DEPTH	**21**	FABRIC_DESIGN	**16**
POWER	**33**	COMPOSITION	**18**
PATTERN_NAME	**16**	TOTAL_LENGTH	**23**
DETAILED_MODEL	**72**	ACCESSORIES_INCLUDED	**55**
MAIN_MATERIAL	**16**	THICKNESS	**29**
MATERIALS	**35**	WAIST_CIRCUMFERENCE	**16**
VOLUME_CAPACITY	**23**	HIP_CIRCUMFERENCE	**15**
ALPHANUMERIC_MODEL	**23**	COMPATIBLE_BRANDS	**20**
SHAPE	**20**	SALE_FORMAT	**34**
MAX_WEIGHT_SUPPORTED	**32**	CHEST_CIRCUMFERENCE	**15**
COMPATIBLE_MODELS	**48**	out_of_scope	**101**
Total de Exemplos			**1419**

É possível perceber o desbalanceamento entre o número de exemplos de cada classe. Esse desbalanceamento ocorreu, sobretudo, por conta da facilidade em se encontrar perguntas relacionadas a atributos como marca e cor do produto (perguntas classificadas como **BRAND** e **COLOR**) em detrimento da dificuldade de se encontrar perguntas relacionadas a atributos como a estampa de uma peça de roupa e os modelos compatíveis

(perguntas classificadas como **FABRIC_DESIGN** e **COMPATIBLE_MODELS**). Essa dificuldade se justifica pelo menor número de subcategorias de produtos que podem receber esses atributos, como abordado na Seção 4.1. A Figura 22 retrata, por meio de um gráfico de barras, o número de exemplos rotulados em cada classe.

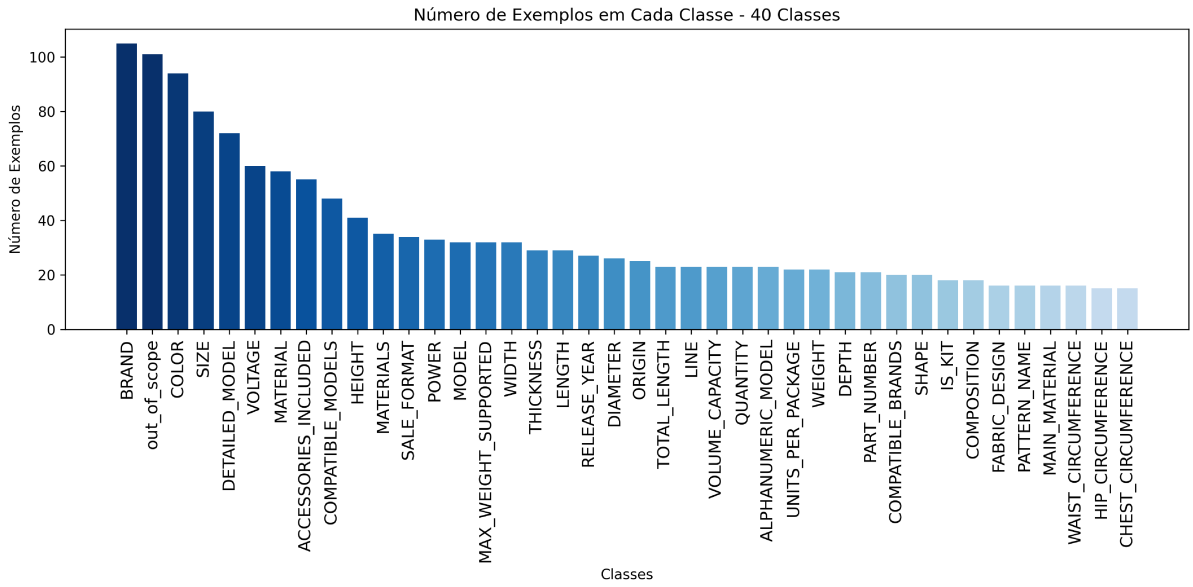

Figura 22 – Gráfico de Barras retratando o número total de exemplos rotulados para cada uma das 40 classes, da classe com mais exemplos para a com menos exemplos.

Para a segunda rodada de criação da base de dados rotulada, foi criada uma nova base de dados, com 28 classes. A quantidade de exemplos de cada classe está descrita na Tabela 7 e as aglutinações de classes feitas foram ilustradas na Figura 16.

Tabela 7 – Quantidade de exemplos rotulados para cada classe na configuração com 28 classes

Classe	**Exemplos**	**Classe**	**Exemplos**
BRAND	**105**	MODEL	**127**
COLOR	**94**	PART_NUMBER	**21**
MATERIAL	**127**	LINE	**23**
ORIGIN	**25**	RELEASE_YEAR	**27**
WEIGHT	**22**	QUANTITY	**97**
VOLTAGE	**60**	SIZE	**205**
DIAMETER	**26**	DEPTH	**21**
FABRIC_DESIGN	**16**	POWER	**33**
PATTERN_NAME	**16**	ACCESSORIES_INCLUDED	**55**
THICKNESS	**29**	WAIST_CIRCUMFERENCE	**16**
VOLUME_CAPACITY	**23**	HIP_CIRCUMFERENCE	**15**
COMPATIBLE_BRANDS	**20**	SHAPE	**20**
MAX_WEIGHT_SUPPORTED	**32**	CHEST_CIRCUMFERENCE	**15**
COMPATIBLE_MODELS	**48**	out_of_scope	**101**
Total de Exemplos			**1419**

A partir da Tabela 7, é possível perceber que o problema de desbalanceamento de classes se acentuou ainda mais por causa da aglutinação. Isso pode ocasionar um efeito negativo nas métricas de avaliação e será analisado nas seções posteriores. Contudo, o objetivo principal de minimizar a ocorrência de situações em que uma pergunta poderia

ser classificada em mais de um atributo ao mesmo tempo foi atingido. Além disso, nenhum exemplo foi removido da base de dados. A Figura 23 retrata, por meio de um gráfico de barras, o número de exemplos rotulados em cada classe nessa configuração com 28 classes.

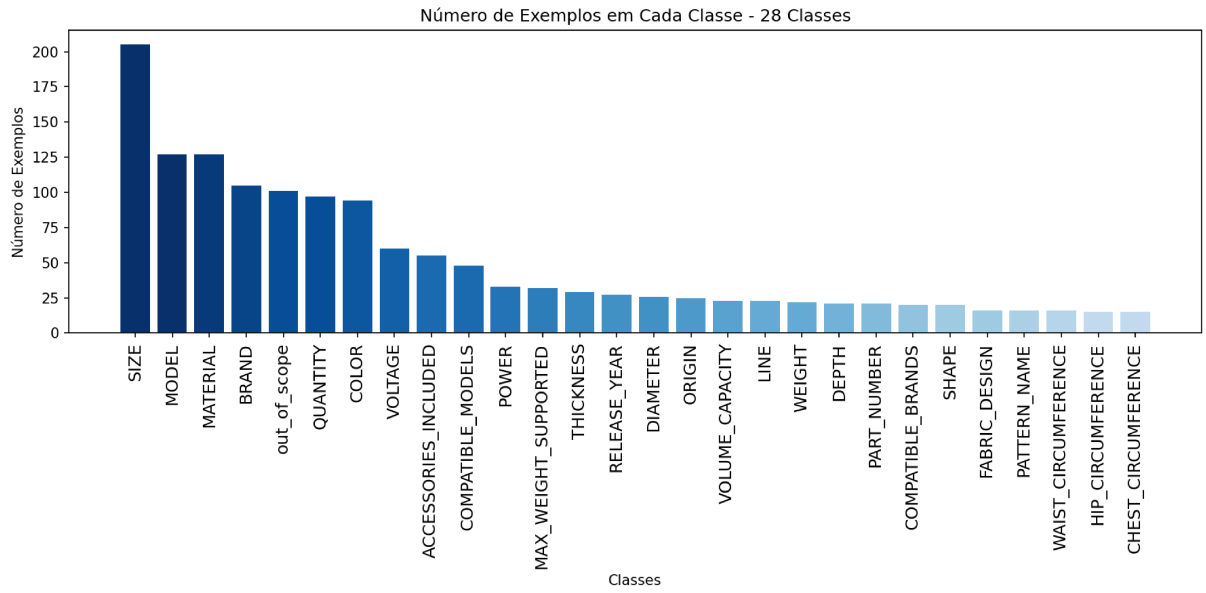

Figura 23 – Gráfico de Barras retratando o número total de exemplos rotulados para cada uma das 28 classes, da classe com mais exemplos para a com menos exemplos.

Analisando a Figura 23, é possível perceber que **SIZE** (tamanho) passou a ser a classe com o maior número de exemplos. **BRAND** (marca), que na primeira rodada da criação da base de dados rotulada era a classe com o maior número de exemplos, passou a ser apenas a 4ª maior classe. Outras classes resultantes de aglutinação, **MODEL** (modelo), **MATERIAL** (material), e **QUANTITY** (quantidade), passaram a ser a 2ª, 3ª, e 6ª maior classe, respectivamente. Pelo fato de possuírem um maior número de exemplos quando considerando a base de dados com 28 classes, espera-se que as medidas de avaliação dos modelos classificadores nessas classes aglutinadas sejam melhoradas.

4.4 Treinamento e Avaliação dos Classificadores

A seguir, será apresentada uma discussão quanto ao desempenho individual de cada um dos algoritmos de classificação treinados, em cada uma das configurações apresentadas. Em um primeiro momento, serão apresentados os resultados dos experimentos envolvendo a base de dados com 40 classes. Em seguida, serão apresentados os resultados dos experimentos envolvendo a base de dados com classes aglutinadas, de 28 classes.

4.4.1 Configuração do Treinamento e Método de Avaliação

A Tabela 8 mostra o número de exemplos de Treino e Teste em cada classe na primeira rodada de experimentos, de 40 classes, enquanto a Tabela 9 mostra o número

de exemplos de Treino e Teste em cada classe na segunda rodada de experimentos, de 28 classes.

Tabela 8 – Quantidade de exemplos de Treino e Teste para cada classe na configuração com 40 classes

Classe	**Exemplos de Treino**	**Exemplos de Teste**	**Total de Exemplos**
BRAND	84	21	**105**
MODEL	26	6	**32**
IS_KIT	14	4	**18**
COLOR	75	19	**94**
WIDTH	25	7	**32**
LENGTH	23	6	**29**
PART_NUMBER	16	5	**21**
MATERIAL	46	12	**58**
HEIGHT	32	9	**41**
LINE	19	4	**23**
UNITS_PER_PACKAGE	18	4	**22**
ORIGIN	20	5	**25**
RELEASE_YEAR	22	5	**27**
WEIGHT	17	5	**22**
QUANTITY	18	5	**23**
VOLTAGE	48	12	**60**
SIZE	64	16	**80**
DIAMETER	21	5	**26**
DEPTH	17	4	**21**
FABRIC_DESIGN	12	4	**16**
POWER	26	7	**33**
COMPOSITION	15	3	**18**
PATTERN_NAME	13	3	**16**
TOTAL_LENGTH	19	4	**23**
DETAILED_MODEL	58	14	**72**
ACCESSORIES_INCLUDED	44	11	**55**
MAIN_MATERIAL	13	3	**16**
THICKNESS	23	6	**29**
MATERIALS	28	7	**35**
WAIST_CIRCUMFERENCE	13	3	**16**
VOLUME_CAPACITY	18	5	**23**
HIP_CIRCUMFERENCE	12	3	**15**
ALPHANUMERIC_MODEL	18	5	**23**
COMPATIBLE_BRANDS	16	4	**20**
SHAPE	16	4	**20**
SALE_FORMAT	28	6	**34**
MAX_WEIGHT_SUPPORTED	26	6	**32**
CHEST_CIRCUMFERENCE	12	3	**15**
COMPATIBLE_MODELS	39	9	**48**
out_of_scope	81	20	**101**
Total	**1135**	**284**	**1419**

Tabela 9 – Quantidade de exemplos de Treino e Teste para cada classe na configuração com 28 classes

Classe	**Exemplos de Treino**	**Exemplos de Teste**	**Total de Exemplos**
BRAND	84	21	**105**
MODEL	102	25	**127**
COLOR	75	19	**94**
PART_NUMBER	16	5	**21**
MATERIAL	102	25	**127**
LINE	19	4	**23**
ORIGIN	20	5	**25**
RELEASE_YEAR	22	5	**27**
WEIGHT	17	5	**22**
QUANTITY	78	19	**97**
VOLTAGE	48	12	**60**
SIZE	163	42	**205**
DIAMETER	21	5	**26**
DEPTH	17	4	**21**
FABRIC_DESIGN	12	4	**16**
POWER	26	7	**33**
PATTERN_NAME	13	3	**16**
ACCESSORIES_INCLUDED	44	11	**55**
THICKNESS	23	6	**29**
WAIST_CIRCUMFERENCE	13	3	**16**
VOLUME_CAPACITY	18	5	**23**
HIP_CIRCUMFERENCE	12	3	**15**
COMPATIBLE_BRANDS	16	4	**20**
SHAPE	16	4	**20**
MAX_WEIGHT_SUPPORTED	26	6	**32**
CHEST_CIRCUMFERENCE	12	3	**15**
COMPATIBLE_MODELS	39	9	**48**
out_of_scope	81	20	**101**
Total	**1135**	**284**	**1419**

4.4.2 Experimentos com 40 Classes

As Tabelas 10 e 11 apresentam os resultados da primeira rodada de experimentos, que usou a base de dados com 40 classes. A Tabela 10 mostra o número de classes que atingiram o valor máximo de Revocação e Precisão, assim como o número de classes que atingiram o valor mínimo dessas métricas. A Tabela 11 traz um sumário das métricas de avaliação de cada modelo ponderadas pelo número de exemplos em cada classe. Os modelos estão identificados com os nomes definidos nas Seções 3.6.1 e 3.6.2. As Matrizes de Confusão estão dispostas no Apêndice A.

É possível perceber pela Tabela 10 que o modelo no qual mais classes tiveram todos os seus exemplos corretamente previstos foi o modelo HF2. Ao mesmo tempo, esse modelo se destacou negativamente pelo fato de não ter previsto corretamente nenhum exemplo de 6 classes, o que indica que o modelo confunde essas classes com outras classes que apresentam exemplos similares.

O modelo que apresentou nos testes o maior número de classes com Precisão igual a 1 foi o modelo HF1. Esse resultado indica que é possível ter grande confiança nesse modelo quando ele prevê que uma pergunta é de determinada classe, ao menos para as classes que apresentaram o valor máximo de Precisão.

Ao prever corretamente ao menos 1 exemplo de Teste para todas as classes, o

modelo RASA1 se mostrou um modelo generalista, ou seja, com habilidade de aprender características de todas as classes. Os outros modelos que possuem como algoritmo de classificação o modelo DIETClassifier apresentaram resultados próximos.

Tabela 10 – Número de classes que atingiram o valor máximo e o valor mínimo das métricas de avaliação (base de dados com 40 classes)

ID	**Revocação = 1**	**Precisão = 1**	**Revocação = 0 e Precisão = 0**
HF1	10	**16**	4
HF2	**16**	13	6
RASA1	10	12	**0**
RASA2	9	14	1
RASA3	12	12	1

A Tabela 11 mostra que o modelo HF2 obteve os melhores resultados gerais, apesar do grande número de classes em que nenhum exemplo foi corretamente previsto. Os resultados mostram que esse modelo se mostrou um modelo especialista em algumas classes. Por esse motivo, ele foi o modelo BERT escolhido para os experimentos com a base de dados aglutinada. O modelo RASA3 se destacou em relação à Precisão e por isso foi o modelo DIETClassifier escolhido.

Tabela 11 – Métricas de avaliação ponderadas pelo número de exemplos de cada classe (base de dados com 40 classes)

ID	**Encoder**	**Classificador**	**Acurácia**	**F1-Score**	**Precisão**	**Revocação**
HF1	BERTimbau	BERTimbau	0,702	0,669	0,686	0,702
HF2	BERTimbau	BERTimbau	**0,749**	**0,720**	0,720	**0,749**
RASA1	BERT-Multi	DIET	0,680	0,668	0,705	0,680
RASA2	BERTimbau	DIET	0,687	0,673	0,700	0,687
RASA3	BERTimbau	DIET	0,718	0,711	**0,731**	0,718

A Tabela 12 sumariza as classes que obtiveram os melhores desempenhos na medida Revocação. Todas as classes mostradas obtiveram desempenho máximo, ou seja, todos os seus exemplos de Teste foram corretamente classificados. Uma característica em comum entre essas classes é que seus exemplos são compostos por palavras ou termos frequentemente repetidos, o que facilita a predição.

Tabela 12 – Classes com os maiores valores de Revocação (base de dados com 40 classes)

Posição	**Classe**	**HF1**	**HF2**	**RASA1**	**RASA2**	**RASA3**	**Média**
1°	DEPTH	1,000	1,000	1,000	1,000	1,000	1,000
1°	HIP_CIRCUMFERENCE	1,000	1,000	1,000	1,000	1,000	1,000
1°	THICKNESS	1,000	1,000	1,000	1,000	1,000	1,000
1°	VOLTAGE	1,000	1,000	1,000	1,000	1,000	1,000
1°	VOLUME_CAPACITY	1,000	1,000	1,000	1,000	1,000	1,000
1°	WAIST_CIRCUNFERENCE	1,000	1,000	1,000	1,000	1,000	1,000

A Tabela 13 sumariza as classes que obtiveram os piores desempenhos na medida Revocação, ou seja, as classes em que percentualmente menos exemplos de Teste foram

classificados corretamente. É possível perceber que 4 das 5 classes apresentadas são compostas de exemplos que poderiam ser classificados em mais de uma classe ao mesmo tempo na ausência de informações complementares, situação que o procedimento de aglutinação de classes tenta resolver. **COMPATIBLE_BRANDS**, no entanto, representa um problema maior: ela é uma classe que foi confundida com **COMPATIBLE_MODELS** em 4 dos 5 modelos, e um modelo necessitaria de conhecimento de mundo para poder classificá-la corretamente.

Tabela 13 – Classes com os piores valores de Revocação (base de dados com 40 classes)

Posição	**Classe**	**HF1**	**HF2**	**RASA1**	**RASA2**	**RASA3**	**Média**
40°	COMPATIBLE_BRANDS	0,0000	0,0000	0,5000	0,0000	0,0000	0,1000
39°	ALPHANUMERIC_MODEL	0,0000	0,0000	0,2000	0,2000	0,6000	0,2000
38°	UNITS_PER_PACKAGE	0,0000	0,0000	0,2500	0,5000	0,5000	0,2500
37°	MATERIALS	0,1429	0,0000	0,5714	0,4286	0,5714	0,3429
36°	MODEL	0,1667	0,3333	0,6667	0,1667	0,5000	0,3667

4.4.3 Experimentos com 28 Classes

As Tabelas 14 e 15 apresentam o desempenho dos modelos treinados na base de dados com 28 classes e aglutinações de classes propostas na Seção 3.4. Os modelos treinados nessa etapa são iguais aos modelos treinados na etapa anterior. As Matrizes de Confusão estão dispostas no Apêndice A.

O resultado mostrado na Tabela 14 indica que nesse cenário o modelo HF2 performou melhor que o modelo RASA3 em todos os aspectos. A diminuição no número de classes em que os modelos atingiram o valor máximo das métricas de avaliação é naturalmente esperada pela aplicação da aglutinação de classes. O modelo RASA3 apresentou dificuldade em aprender quando classificar um exemplo como **MODEL**: os exemplos dessa classe foram classificados como pertencentes a 10 classes diferentes.

Tabela 14 – Número de classes que atingiram o valor máximo e o valor mínimo das métricas de avaliação (base de dados com 28 classes)

ID	**Revocação = 1**	**Precisão = 1**	**Revocação = 0 e Precisão = 0**
HF2	**11**	**11**	**1**
RASA3	**11**	7	**1**

A Tabela 15 confirma o bom desempenho do modelo HF2 quando comparado ao modelo RASA3. Além disso, ela demonstra que a aplicação da aglutinação de classes foi benéfica para os modelos: houve uma melhoria de 14,8% em Acurácia, de 18,6% em F1-Score, de 19,7% em Precisão e de 14,8% em Revocação no desempenho do modelo HF2 na base de dados com 28 classes em comparação ao desempenho do modelo HF2 na base de dados com 40 classes. O modelo RASA3 também foi influenciado positivamente pelo procedimento, porém com menos intensidade.

Tabela 15 – Métricas de avaliação ponderadas pelo número de exemplos de cada classe (base de dados com 28 classes)

ID	Encoder	Classificador	Acurácia	F1-Score	Precisão	Revocação
HF2	BERTimbau	BERTimbau	**0,860**	**0,854**	**0,862**	**0,860**
RASA3	BERTimbau	DIET	0,763	0,755	0,775	0,763

A Tabela 16 sumariza as classes da base de dados com 28 classes que obtiveram os melhores desempenhos na medida Revocação, ou seja, as classes em que percentualmente mais exemplos de Teste foram classificados corretamente. Mais uma vez, há predomínio de classes consideradas de fácil predição, com exceção de **ORIGIN**.

Tabela 16 – Classes com os maiores valores de Revocação (base de dados com 28 classes)

Posição	Classe	HF2	RASA3	Média
1°	CHEST_CIRCUMFERENCE	1,000	1,000	1,000
1°	DEPTH	1,000	1,000	1,000
1°	DIAMETER	1,000	1,000	1,000
1°	HIP_CIRCUMFERENCE	1,000	1,000	1,000
1°	ORIGIN	1,000	1,000	1,000
1°	WAIST_CIRCUNFERENCE	1,000	1,000	1,000
1°	WEIGHT	1,000	1,000	1,000

A Tabela 17 sumariza as classes que obtiveram os piores desempenhos na medida Revocação. No geral, os modelos apresentaram valores mais altos para essa métrica do que os valores vistos na Tabela 13, o que confirma a eficácia da aglutinação de classes. A classe **MODEL** foi a única entre as classes aglutinadas que continuou na lista das 5 classes com os piores resultados.

A classe **COMPATIBLE_BRANDS**, que havia sido a classe com os piores resultados na etapa anterior, continuou a ser a classe em que os modelos mais apresentam dificuldade de predição. A média de pontuação em Revocação era de 0,1000 no experimento anterior. No experimento com 28 classes, essa média caiu para 0,0000. Isso indica que nenhum dos dois modelos que tiveram os melhores desempenhos foi capaz de classificar corretamente 1 pergunta como **COMPATIBLE_BRANDS**.

De forma geral, não é possível apontar um modelo mais generalista e um modelo mais especialista entre os modelos treinados na base de dados de 28 classes. Ambos os modelos não conseguiram prever corretamente os exemplos de uma única classe. Sendo assim, como as medidas de avaliação do modelo HF2 descritas na Tabela 15 apontam um desempenho geral muito superior desse modelo, ele é o melhor modelo treinado considerando as duas rodadas de experimentos.

Tabela 17 – Classes com os piores valores de Revocação (base de dados com 28 classes)

Posição	**Classe**	**HF2**	**RASA3**	**Média**
28°	COMPATIBLE_BRANDS	0,000	0,000	0,000
27°	MAX_WEIGHT_SUPPORTED	0,500	0,500	0,500
27°	SHAPE	0,250	0,750	0,500
25°	MODEL	0,720	0,320	0,520
24°	COMPATIBLE_MODELS	0,667	0,556	0,611

4.5 Comparativo entre Classes Frequentemente Confundidas

As Figuras 24 e 25 apresentam as nuvens de palavras geradas a partir dos exemplos de Treino e de Teste das classes **COMPATIBLE_BRANDS** e **COMPATIBLE_MODELS**, respectivamente.

É possível perceber que, apesar dessas duas classes serem frequentemente confundidas, as palavras que compõem seus exemplos se repetem pouco entre si. A maior exceção está relacionada à palavra **compatível**, que aparece 9 vezes nos exemplos da classe **COMPATIBLE_BRANDS** e 16 vezes nos exemplos da classe **COMPATIBLE_MODELS**.

É possível concluir que um dos motivos para a dificuldade apresentada pelos modelos em classificar corretamente os exemplos da classe **COMPATIBLE_BRANDS** é a falta de conhecimento de mundo para entender quais palavras representam marcas e quais palavras representam modelos. Como estatisticamente a palvra **compatível** aparece mais vezes na classe **COMPATIBLE_MODELS**, os modelos esperam que qualquer exemplo que apresente essa palavra também pertença a essa classe. Uma possível solução para esse problema é fazer uso de grandes estruturas de dados que salvem nomes de marcas e de modelos, como o grafo de conhecimento interno do Mercado Livre. Uma solução alternativa que poderia ser testada é o uso de outros modelos de Aprendizado Profundo que possuam um maior número de parâmetros.

Figura 24 – Nuvem de palavras gerada a partir dos exemplos de Treino e Teste da classe **COMPATIBLE_BRANDS**, após remoção de *stopwords*.

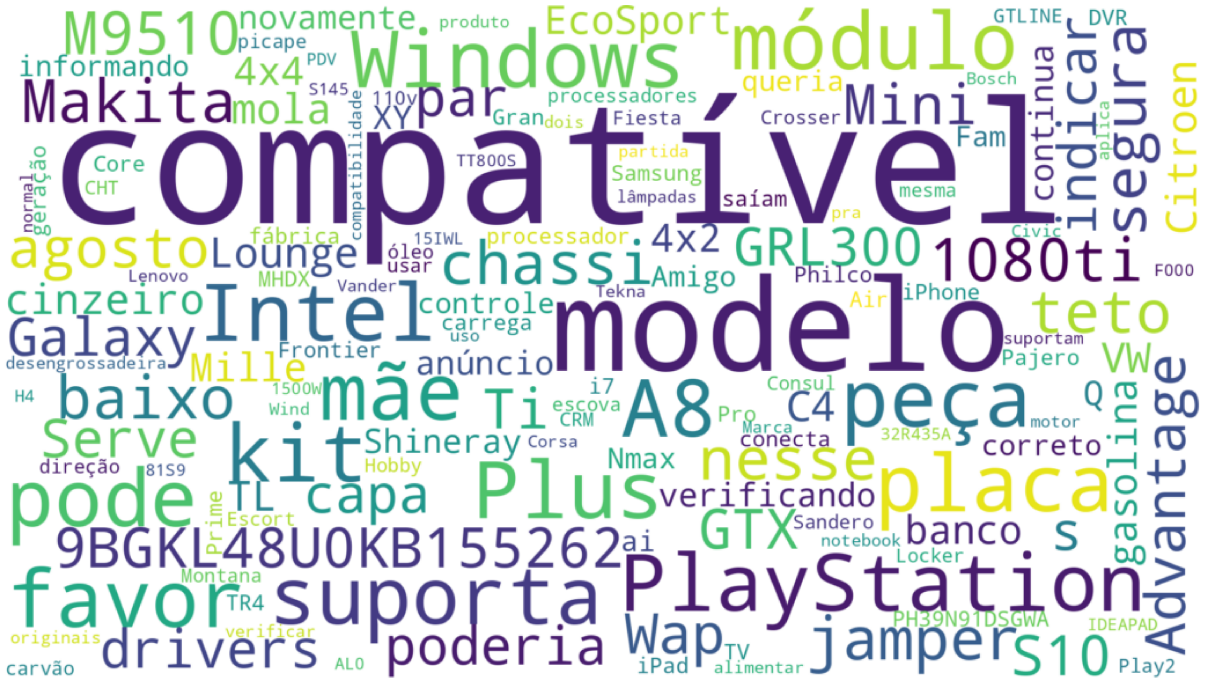

Figura 25 – Nuvem de palavras gerada a partir dos exemplos de Treino e Teste da classe **COMPATIBLE_MODELS**, após remoção de *stopwords*.

4.6 Considerações Finais

Neste capítulo, uma visão geral dos resultados obtidos em cada etapa do trabalho foi apresentada. Foi feita uma análise da representatividade de cada atributo definido como classe em relação ao número de subcategorias de produtos do Mercado Livre em que esse atributo pode ser utilizado. Além disso, foi realizada uma estimativa da quantidade de dados coletados de cada fonte. Uma análise em relação ao número de exemplos rotulados de cada classe e as condições que levaram ao desbalanceamento entre as classes também foi feita.

Após testes com cinco configurações de modelos classificadores e duas configurações de base de dados, foi possível escolher o modelo HF2 como o melhor algoritmo de classificação. O modelo mostrou ser especialista em algumas classes na configuração de base de dados original, que se aproxima mais do problema real. Em seguida, o modelo foi aplicado sobre a base de dados aglutinada e melhorou substancialmente suas métricas de avaliação, o que indica a eficiência desse procedimento na recategorização de perguntas que poderiam pertencer a mais de uma classe. O modelo RASA3 também apresentou um resultado positivo: foi o segundo melhor modelo de acordo com as métricas de avaliação e se mostrou mais generalista.

A partir dos resultados obtidos, foi possível concluir que o método apresentado é eficaz em classificar corretamente variados tipos de perguntas. No entanto, algumas classes são frequentemente confundidas em todos os modelos testados por falta de conhecimento de mundo. Por isso, avaliações de outras metodologias deveriam ser feitas caso haja interesse em classificar essas classes de forma satisfatória.

5 Conclusão

Este trabalho apresentou uma metodologia para a resolução de dúvidas de clientes reais sobre produtos do comércio eletrônico usando inteligência artificial, desde a etapa de definição das classes até a etapa de medidas e método de avaliação dos classificadores treinados. As disciplinas Algoritmos e Programação de Computadores e Inteligência Artificial, do curso de Graduação em Engenharia Mecatrônica da Universidade Federal de Uberlândia, se mostraram importantes para o entendimento dos conceitos teóricos aplicados na realização deste trabalho.

O Mercado Livre foi a plataforma de comércio eletrônico escolhida, em virtude da grande quantidade de perguntas feitas na plataforma disponível na base de dados da empresa GoBots.

Na prática, a metodologia consiste em tratar a situação como um problema de classificação de texto multi-classe, onde cada classe representa um atributo de produto. A resolução das dúvidas em si é feita após a predição da classe e consulta à API do Mercado Livre pelo nome do atributo inferido. No entanto, o trabalho apresentou um foco maior na etapa anterior à resolução das dúvidas, ou seja, no procedimento de treinamento dos modelos classificadores responsáveis por prever a qual atributo uma pergunta se refere.

Para atingir o objetivo, foram realizadas as seguintes etapas da Classificação de Texto Multi-classe: definição das classes, coleta de dados, criação da base de dados rotulada, pré-processamento dos dados, treinamento dos algoritmos de classificação e medidas e método de avaliação dos classificadores. A abordagem de classificação de texto aplicada na base de dados rotulada foi o uso de dois modelos transformadores, BERT e DIET-Classifier, avaliados quanto a diferentes configurações de hiperparâmetros e etapas de pré-processamento.

Os experimentos foram feitos em duas rodadas. Na primeira rodada, a base de dados se aproxima mais do problema real, pois consiste de 40 classes existentes no Mercado Livre. Nessa rodada, percebeu-se que a arquitetura de transformador que usa o modelo brasileiro BERTimbau Base tanto no pré-processamento quanto na classificação de texto apresentou um desempenho excelente em algumas classes específicas, porém abaixo da média em outras. Ao mesmo tempo, a arquitetura de transformador que usa BERTimbau Base no pré-processamento e DIETClassifier na classificação de texto apresentou um desempenho razoável em todas as classes.

Na segunda rodada de experimentos, a base de dados foi aglutinada para 28 classes com o objetivo de minimizar as situações em que uma pergunta poderia ser classificada em mais de uma classe. Nessa situação, a arquitetura de transformador que usa BERTim-

bau Base tanto no pré-processamento quanto na classificação de texto se mostrou muito superior.

Considerando a primeira rodada de experimentos, uma boa opção para a resolução de perguntas de clientes reais é aplicar o melhor modelo DIETClassifier e o melhor modelo BERT em paralelo. O primeiro, mais generalista, iria ponderar igualmente a possibilidade da pergunta pertencer a cada uma das classes. O segundo, mais especialista, serviria para verificar a previsão feita pelo primeiro, ao apresentar uma maior certeza quanto a classe a qual a pergunta pertence. Uma forma de se fazer isso frequentemente abordada na literatura é a criação de um mecanismo de votação, no qual os dois modelos classificam a pergunta fornecida ao mesmo tempo e tomam uma decisão em conjunto dependendo da pontuação retornada por cada modelo ao fazer a classificação.

Algumas classes foram destaques negativos por conta da dificuldade dos modelos em aprenderem a identificá-las, em todas as configurações avaliadas. Entre esses destaques estão as classes que descrevem as marcas compatíveis com determinado produto e os modelos compatíveis com determinado produto.

5.1 Principais Contribuições

- Elaboração e divulgação de uma metodologia de resolução de dúvidas de clientes reais em plataformas de comércio eletrônico;
- Indicação de duas arquiteturas de transformadores que apresentam bom desempenho na tarefa de classificação de perguntas quanto ao atributo de produto ao qual elas se referem;
- Divulgação das medidas de avaliação atingidas pelas arquiteturas de transformadores utilizadas, que servem como motivação para que trabalhos futuros busquem melhores resultados;
- Criação de uma base de dados rotulada privada, composta por 1419 exemplos de perguntas rotuladas a respeito do atributo de produto ao qual elas se referem.

5.2 Trabalhos Futuros

Neste trabalho, foram treinados modelos classificadores de texto de alto desempenho na identificação de múltiplas classes. No entanto, a predição de algumas classes, notadamente as relacionadas com compatibilidade de produtos com marcas e modelos, apresentou resultados negativos.

Para resolver esse problema, outras metodologias podem ser testadas. Entre elas, o uso de grandes estruturas de dados, como os grafos de conhecimento, que armazenam nomes de marcas e nomes de modelos. Uma outra possibilidade é o uso de modelos de aprendizado de máquina de dimensões maiores, como os modelos de linguagem de grande porte, que naturalmente guardam consigo noções de nomes de marcas e nomes de modelos por conta do seu alto número de parâmetros. Além disso, pode ser avaliada a possibilidade de que esses modelos apresentem uma maior facilidade na identificação de como os exemplos de cada classe são estruturados, pelo fato de terem sido treinados em um número muito maior de exemplos.

Uma alternativa para o uso de modelos de aprendizado de máquina de dimensões maiores que também pode ser abordada em trabalhos futuros é o uso de técnicas de *ensemble*, ou seja, fazer a classificação de uma mesma pergunta em modelos diferentes e usar um algoritmo de votação para determinar a resposta correta. O algoritmo de votação pode ser, por exemplo, considerar a classe prevista pela maioria dos modelos como a classe correta.

Outros trabalhos podem apresentar resultados diferentes ao fazer uso de outras formas de pré-processamento. As pessoas frequentemente fazem uso de gírias ou grafias diferentes da norma culta da Língua Portuguesa ao fazer perguntas nos sites de *e-commerce*, e os tokenizadores utilizados neste trabalho não são totalmente eficientes no tratamento dessas situações.

Referências

AGGARWAL, C. C.; ZHAI, C. A survey of text classification algorithms. In: **Mining text data**. [S.l.]: Springer, 2012. p. 163–222. Citado 4 vezes nas páginas 19, 24, 28 e 36.

ALAMMAR, J. **Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)**. 2018. [Acesso em 16 Nov. 2022]. Disponível em: <https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>. Citado 2 vezes nas páginas 5 e 30.

BAGATINI, F. Z.; LAIMER, C. G. O contexto do e-commerce no brasil: Análise do desempenho do varejo online no período de 2003 a 2018. In: **CLAV 2019**. [S.l.: s.n.], 2019. Citado na página 17.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. **Diário Oficial [da] República Federativa do Brasil**, Brasília, DF, 2018. Disponível em: <https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm>. Citado na página 44.

BREIMAN, L. Bagging predictors. **Machine learning**, Springer, v. 24, n. 2, p. 123–140, 1996. Citado na página 28.

BROMBIM, E. **Como reduzir custos com atendimento**. 2021. [Acesso em 14 Nov. 2022]. Disponível em: <https://www.ecommercebrasil.com.br/artigos/como-reduzir-custos-com-atendimento>. Citado na página 18.

BUNK, T.; VARSHNEYA, D.; VLASOV, V.; NICHOL, A. Diet: Lightweight language understanding for dialogue systems. **arXiv preprint arXiv:2004.09936**, 2020. Citado 2 vezes nas páginas 32 e 53.

Callcenter.inf.br. **IA como norma**. 2019. [Acesso em 14 Nov. 2022]. Disponível em: <https://www.callcenter.inf.br/estatisticas/69280/ia-como-norma/ler.aspx>. Citado na página 19.

CASSIDY, J. **Dot.con: How America Lost Its Mind and Money in the Internet Era**. [S.l.]: Harper Perennial, 2003. ISBN 0060008814. Citado na página 11.

CHO, K.; MERRIËNBOER, B. V.; GULCEHRE, C.; BAHDANAU, D.; BOUGARES, F.; SCHWENK, H.; BENGIO, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. **arXiv preprint arXiv:1406.1078**, 2014. Citado na página 30.

CORRÊA, I. T. et al. Análise dos sentimentos expressos na rede social twitter em relação aos filmes indicados ao oscar 2017. Universidade Federal de Uberlândia, 2017. Citado na página 20.

CORTES, E. G. **Quando, Onde, Quem, O que ou Por que? Um Modelo Híbrido de Classificação de Perguntas para Sistemas de Question Answering**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, 2019. Citado na página 36.

COX, D. R.; SNELL, E. J. **Analysis of binary data**. [S.l.]: Routledge, 2018. Citado na página 29.

DENIZ, E.; ERBAY, H.; COşAR, M. Multi-label classification of e-commerce customer reviews via machine learning. **Axioms**, MDPI AG, v. 11, n. 9, p. 436, Aug 2022. ISSN 2075-1680. Disponível em: <http://dx.doi.org/10.3390/axioms11090436>. Citado na página 37.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In: **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**. Minneapolis, Minnesota: Association for Computational Linguistics, 2019. p. 4171–4186. Disponível em: <https://aclanthology.org/N19-1423>. Citado 3 vezes nas páginas 29, 31 e 52.

EBIT; NIELSEN. **Webshoppers 43**. 2021. [Acesso em 17 Out. 2022]. Disponível em: <https://www.mobiletime.com.br/wp-content/uploads/2021/03/Webshoppers_43.pdf>. Citado 3 vezes nas páginas 12, 15 e 17.

ETAIWI, W.; NAYMAT, G. The impact of applying different preprocessing steps on review spam detection. **Procedia computer science**, Elsevier, v. 113, p. 273–279, 2017. Citado 2 vezes nas páginas 25 e 26.

FELITTI, G. **O Mercado Livre se veste de Amazon para combater a própria Amazon**. 2021. <https://manualdousuario.net/podcast/tecnocracia-47/>. Accesso em 26 Dez. 2022. Citado 2 vezes nas páginas 11 e 12.

FUCHS, G.; ACRICHE, Y. Product titles-to-attributes as a text-to-text task. In: **Proceedings of The Fifth Workshop on e-Commerce and NLP (ECNLP 5)**. [S.l.: s.n.], 2022. p. 91–98. Citado na página 37.

Gov.br. **90% dos lares brasileiros já tem acesso à internet no Brasil, aponta pesquisa**. 2022. [Acesso em 31 Out. 2022]. Disponível em: <https://www.gov.br/casacivil/pt-br/assuntos/noticias/2022/setembro/90-dos-lares-brasileiros-ja-tem-acesso-a-internet-no-brasil-aponta-pesquisa>. Citado na página 17.

GRANDINI, M.; BAGLI, E.; VISANI, G. Metrics for multi-class classification: an overview. **arXiv preprint arXiv:2008.05756**, 2020. Citado 4 vezes nas páginas 33, 34, 35 e 36.

HARRIS, Z. S. Distributional structure. **WORD**, Routledge, v. 10, n. 2-3, p. 146–162, 1954. Disponível em: <https://doi.org/10.1080/00437956.1954.11659520>. Citado na página 26.

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 29.

INTERNET Resource Discovery Services by Bytes (Logarithmic Scale). 1994. <http://www.quarterman.com/pictures/1991-1994--mn/SCAN0419.html>. Accesso em 26 Dez. 2022. Citado na página 11.

IYYER, M.; MANJUNATHA, V.; BOYD-GRABER, J.; III, H. D. Deep unordered composition rivals syntactic methods for text classification. In: **Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)**. [S.l.: s.n.], 2015. p. 1681–1691. Citado na página 29.

JIANG, S.; PANG, G.; WU, M.; KUANG, L. An improved k-nearest-neighbor algorithm for text categorization. **Expert Systems with Applications**, Elsevier, v. 39, n. 1, p. 1503–1509, 2012. Citado na página 29.

JONES, K. S. A statistical interpretation of term specificity and its application in retrieval. **Journal of documentation**, MCB UP Ltd, v. 28, n. 1, p. 11–21, 1972. Citado na página 26.

KALCHBRENNER, N.; GREFENSTETTE, E.; BLUNSOM, P. A convolutional neural network for modelling sentences. **arXiv preprint arXiv:1404.2188**, 2014. Citado na página 29.

KHAN, I.; NAQVI, S. K.; ALAM, M.; RIZVI, S. N. A. An efficient framework for real-time tweet classification. **International Journal of Information Technology**, v. 9, n. 2, p. 215–221, Jun 2017. ISSN 2511-2112. Disponível em: <https://doi.org/10.1007/s41870-017-0015-x>. Citado na página 20.

KOWSARI, K.; MEIMANDI, K. J.; HEIDARYSAFA, M.; MENDU, S.; BARNES, L.; BROWN, D. Text classification algorithms: A survey. **Information**, MDPI, v. 10, n. 4, p. 150, 2019. Citado 2 vezes nas páginas 20 e 28.

LIU, P.; QIU, X.; CHEN, X.; WU, S.; HUANG, X.-J. Multi-timescale long short-term memory neural network for modelling sentences and documents. In: **Proceedings of the 2015 conference on empirical methods in natural language processing**. [S.l.: s.n.], 2015. p. 2326–2335. Citado na página 29.

LIU, Y.; OTT, M.; GOYAL, N.; DU, J.; JOSHI, M.; CHEN, D.; LEVY, O.; LEWIS, M.; ZETTLEMOYER, L.; STOYANOV, V. Roberta: A robustly optimized bert pretraining approach. **arXiv preprint arXiv:1907.11692**, 2019. Citado na página 32.

LUO, B.; LAU, R. Y. K.; LI, C.; SI, Y.-W. A critical review of state-of-the-art chatbot designs and applications. **WIREs Data Mining and Knowledge Discovery**, v. 12, n. 1, p. e1434, 2022. Disponível em: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1434>. Citado na página 19.

MANTHA, M. **Introducing DIET: state-of-the-art architecture that outperforms fine-tuning BERT and is 6X faster to train**. 2020. [Acesso em 27 Nov. 2022]. Disponível em: <https://rasa.com/blog/introducing-dual-intent-and-entity-transformer-diet-state-of-the-art-performance-on-a-lightweight-architecture/>. Citado na página 32.

MATSUBARA, E. T.; MARTINS, C. A.; MONARD, M. C. Pretext: Uma ferramenta para pré-processamento de textos utilizando a abordagem bag-of-words. **Techinical Report**, v. 209, n. 4, p. 10–11, 2003. Citado 2 vezes nas páginas 26 e 27.

Mercado Livre. **História do Mercado Livre: nossos primeiros passos, nossa trajetória**. 2020. [Acesso em 30 Out. 2022]. Disponível em: <https://www.mercadolivre.com.br/institucional/estamos/historia-do-mercado-livre>. Citado na página 16.

MercadoLibre, Inc. **Reports: Second Quarter 2022**. 2022. [Acesso em 30 Out. 2022]. Disponível em: <https://investor.mercadolibre.com/static-files/707802c1-7cfb-4cfc-86a1-cc99dea44df7>. Citado 2 vezes nas páginas 12 e 16.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013. Citado na página 36.

MINAEE, S.; KALCHBRENNER, N.; CAMBRIA, E.; NIKZAD, N.; CHENAGHLU, M.; GAO, J. Deep learning–based text classification: A comprehensive review. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 54, n. 3, apr 2021. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3439726>. Citado na página 29.

MITCHELL, R. **Web Scraping with Python: Collecting More Data from the Modern Web**. [S.l.]: O'Reilly Media, 2018. ISBN 9781491985526. Citado na página 24.

Mona Bushnell. **Human or AI? Realistic Chatbots on the Rise**. 2023. [Acesso em 02 Mai. 2023]. Disponível em: <https://www.businessnewsdaily.com/9821-smb-artificial-intelligence-chatbots.html>. Citado na página 19.

MOONEY, R. J.; ROY, L. Content-based book recommending using learning for text categorization. In: **Proceedings of the Fifth ACM Conference on Digital Libraries**. New York, NY, USA: Association for Computing Machinery, 2000. (DL '00), p. 195–204. ISBN 158113231X. Disponível em: <https://doi.org/10.1145/336597.336662>. Citado na página 20.

MORGAN, J. N.; SONQUIST, J. A. Problems in the analysis of survey data, and a proposal. **Journal of the American statistical association**, Taylor & Francis, v. 58, n. 302, p. 415–434, 1963. Citado na página 29.

MUNKHDALAI, T.; YU, H. Neural semantic encoders. In: NIH PUBLIC ACCESS. **Proceedings of the conference. Association for Computational Linguistics. Meeting**. [S.l.], 2017. v. 1, p. 397. Citado na página 29.

NUAI, W. Khern-am; GHASEMKHANI, H.; KANNAN, K. How questions and answers shape online marketplaces: The case of amazon answer. 2017. Citado na página 16.

NUMBER of People Using the Internet. 2023. <https://ourworldindata.org/grapher/number-of-internet-users?time=earliest..2000&country=~OWID_WRL>. Accesso em 03 Mar. 2023. Citado na página 11.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011. Citado 2 vezes nas páginas 35 e 36.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)**. [S.l.: s.n.], 2014. p. 1532–1543. Citado na página 37.

PORTER, M. F. An algorithm for suffix stripping. **Program**, MCB UP Ltd, 1980. Citado na página 29.

Power Reviews. **How Q&A Eliminates Uncertainty and Boosts Ecommerce Sales**. 2022. [Acesso em 31 Out. 2022]. Disponível em: <https://www.powerreviews.com/insights/how-q-and-a-boosts-ecommerce-sales/>. Citado 2 vezes nas páginas 12 e 17.

RILOFF, E. Little words can make a big difference for text classification. In: **Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval**. [S.l.: s.n.], 1995. p. 130–136. Citado na página 20.

ROCCHIO, J. Relevance feedback in information retrieval. **The Smart retrieval system-experiments in automatic document processing**, Prentice Hall, p. 313–323, 1971. Citado na página 28.

ROSA, J. R. C. **Marketplace no Brasil: Desafios, Vantagens e Tendências deste Modelo de Negócio para Empresas Varejistas**. Dissertação (Mestrado) — Faculdade FIA de Administração e Negócios, São Paulo, SP, Março 2019. Citado na página 16.

ROSE, T.; HADDOCK, N.; TUCKER, R. The effects of corpus size and homogeneity on language model quality. Goldsmiths, University of London, 1997. Citado na página 24.

Tf–idf. In: SAMMUT, C.; WEBB, G. I. (Ed.). **Encyclopedia of Machine Learning**. Boston, MA: Springer US, 2010. p. 986–987. ISBN 978-0-387-30164-8. Disponível em: <https://doi.org/10.1007/978-0-387-30164-8_832>. Citado na página 27.

SANT'ANNA, D. T.; CAUS, R. O.; RAMOS, L. dos S.; HOCHGREB, V.; REIS, J. C. dos. Generating knowledge graphs from unstructured texts: Experiences in the e-commerce field for question answering. In: **ASLD@ ISWC**. [S.l.: s.n.], 2020. p. 56–71. Citado 3 vezes nas páginas 13, 18 e 22.

SCHAPIRE, R. E. The strength of weak learnability. **Machine learning**, Springer, v. 5, n. 2, p. 197–227, 1990. Citado na página 28.

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 28.

SINGH, M. E-services and their role in b2c e-commerce. **Managing Service Quality: An International Journal**, v. 12, n. 6, p. 434–446, 2002. Citado 2 vezes nas páginas 16 e 17.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. Bertimbau: pretrained bert models for brazilian portuguese. In: SPRINGER. **Intelligent Systems: 9th Brazilian Conference, BRACIS 2020, Rio Grande, Brazil, October 20–23, 2020, Proceedings, Part I 9**. [S.l.], 2020. p. 403–417. Citado na página 52.

STONE, B. **A loja de tudo**. Editora Intrinseca, 2013. ISBN 9788580574906. Disponível em: <https://books.google.com.br/books?id=2dktAwAAQBAJ>. Citado na página 11.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. **Advances in neural information processing systems**, v. 27, 2014. Citado na página 30.

The Data Detective. **The 80/20 Split Intuition and an Alternative Split Method**. 2020. [Acesso em 14 Nov. 2022]. Disponível em: <https://towardsdatascience.com/finally-why-we-use-an-80-20-split-for-training-and-test-data-plus-an-alternative-method-oh-yes-edc77e96295d>. Citado na página 25.

THOMAS, M. **AI in Retail and E-Commerce: 17 Examples to Know**. 2022. <https://builtin.com/artificial-intelligence/ai-retail-ecommerce-tech>. Accesso em 13 Mar. 2023. Citado na página 12.

Tidio. **The Future of Chatbots: 80+ Chatbot Statistics for 2023**. 2023. [Acesso em 02 Mai. 2023]. Disponível em: <https://www.tidio.com/blog/chatbot-statistics/>. Citado na página 19.

TUNSTALL, L.; WERRA, L. von; WOLF, T. **Natural Language Processing with Transformers: Building Language Applications with Hugging Face**. [S.l.]: O'Reilly Media, 2022. ISBN 9781098103248. Citado 6 vezes nas páginas 5, 25, 28, 29, 30 e 32.

VAJJALA, S.; MAJUMDER, B.; GUPTA, A.; SURANA, H. **Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems**. [S.l.]: O'Reilly Media, 2020. ISBN 9781492054023. Citado 3 vezes nas páginas 23, 25 e 26.

VANDERPLAS, J. **Python Data Science Handbook: Essential Tools for Working with Data**. [S.l.]: O'Reilly Media, Incorporated, 2016. ISBN 9781491912058. Citado na página 25.

VAPNIK, V.; CHERVONENKIS, A. Y. A class of algorithms for pattern recognition learning. **Avtomat. i Telemekh**, v. 25, n. 6, p. 937–945, 1964. Citado na página 29.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. **Advances in neural information processing systems**, v. 30, 2017. Citado 2 vezes nas páginas 29 e 32.

WANG, Q.; YANG, L.; KANAGAL, B.; SANGHAI, S.; SIVAKUMAR, D.; SHU, B.; YU, Z.; ELSAS, J. Learning to extract attribute value from product via question answering: A multi-task approach. In: **Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. [S.l.: s.n.], 2020. p. 47–55. Citado 2 vezes nas páginas 21 e 23.

WOLF, T.; DEBUT, L.; SANH, V.; CHAUMOND, J.; DELANGUE, C.; MOI, A.; CISTAC, P.; RAULT, T.; LOUF, R.; FUNTOWICZ, M.; DAVISON, J.; SHLEIFER, S.; PLATEN, P. von; MA, C.; JERNITE, Y.; PLU, J.; XU, C.; SCAO, T. L.; GUGGER, S.; DRAME, M.; LHOEST, Q.; RUSH, A. Transformers: State-of-the-art natural language processing. In: **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations**.

Online: Association for Computational Linguistics, 2020. p. 38–45. Disponível em: <https://aclanthology.org/2020.emnlp-demos.6>. Citado na página 28.

WU, Y.; SCHUSTER, M.; CHEN, Z.; LE, Q. V.; NOROUZI, M.; MACHEREY, W.; KRIKUN, M.; CAO, Y.; GAO, Q.; MACHEREY, K. 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 28.

YAN, Z.; DUAN, N.; CHEN, P.; ZHOU, M.; ZHOU, J.; LI, Z. Building task-oriented dialogue systems for online shopping. **Proceedings of the AAAI Conference on Artificial Intelligence**, v. 31, n. 1, Fev. 2017. Disponível em: <https://ojs.aaai.org/index.php/AAAI/article/view/11182>. Citado na página 38.

ZHU, Y.; KIROS, R.; ZEMEL, R.; SALAKHUTDINOV, R.; URTASUN, R.; TORRALBA, A.; FIDLER, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In: **Proceedings of the IEEE international conference on computer vision**. [S.l.: s.n.], 2015. p. 19–27. Citado na página 31.

ZUCCHI, L. E. A.; REIS, J. C. dos. **Sistema automatizado de questão e respostas em e-commerce baseado em similaridade de sentenças multilíngues**. Dissertação (Bacharelado) — Universidade Estadual de Campinas, Campinas, SP, Julho 2021. Citado 2 vezes nas páginas 12 e 18.

Apêndices

APÊNDICE A – Matrizes de Confusão

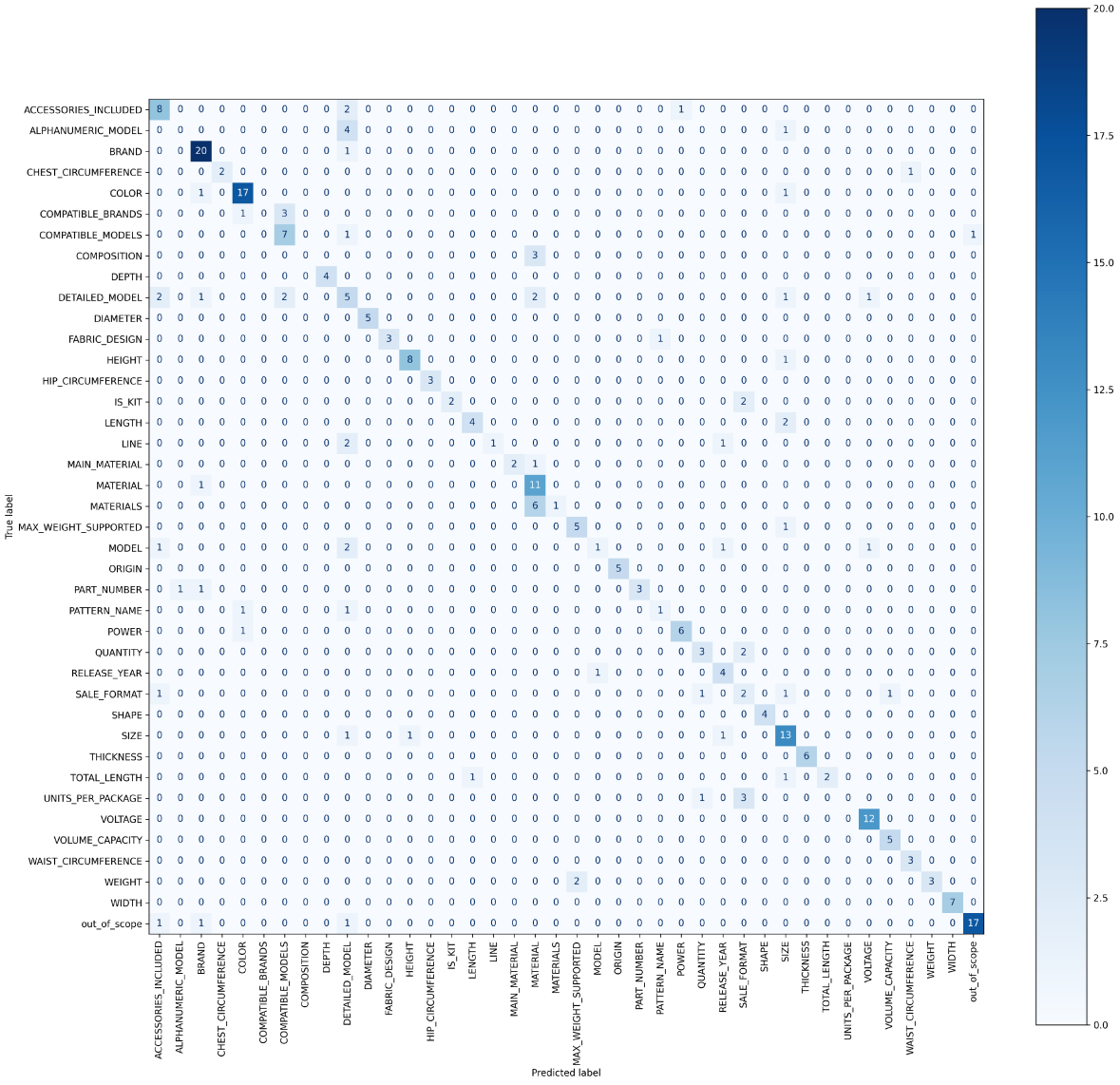

Figura 26 – Matriz de Confusão gerada após teste do modelo HF1, na configuração com 40 classes.

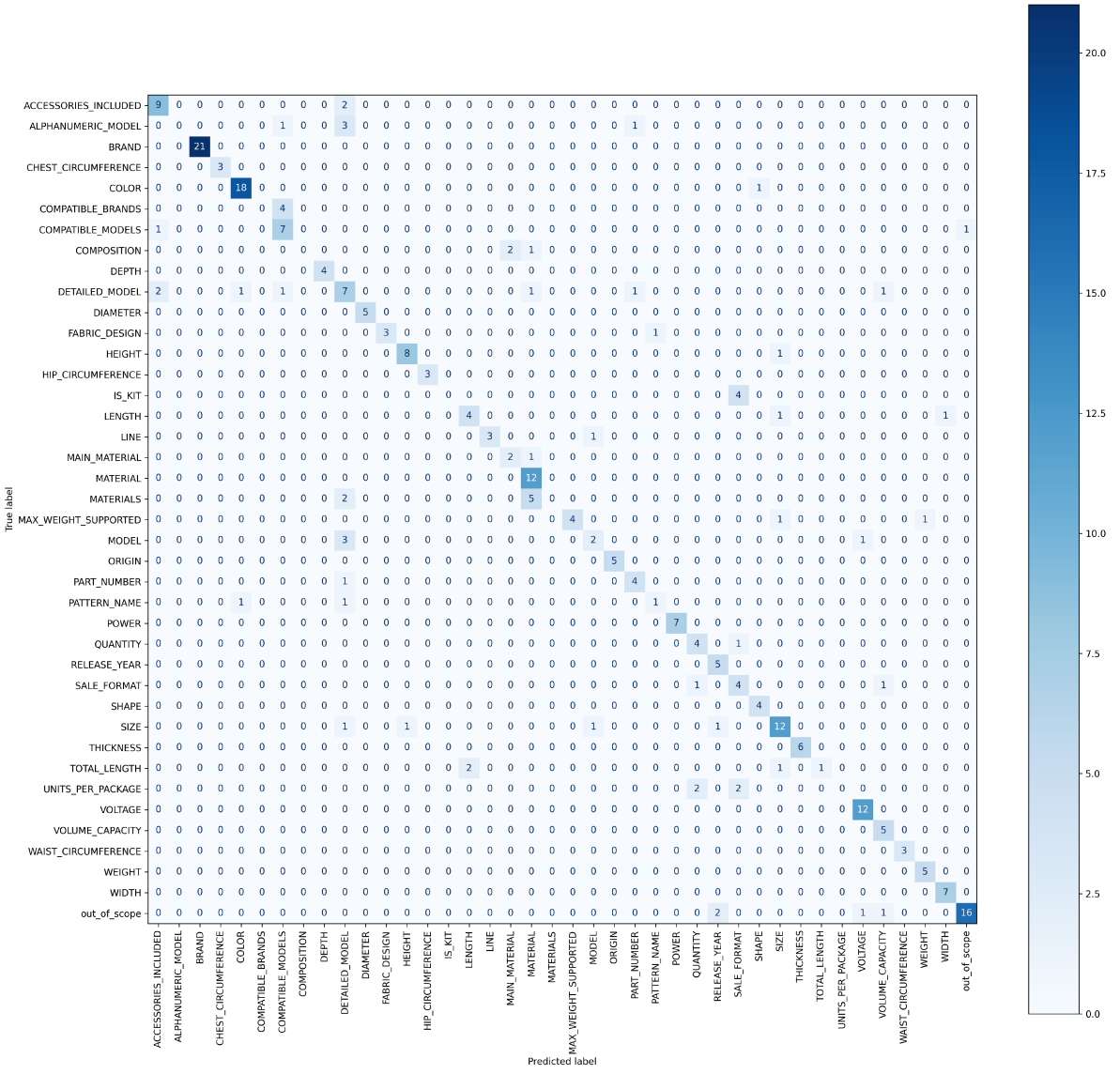

Figura 27 – Matriz de Confusão gerada após teste do modelo HF2, na configuração com 40 classes.

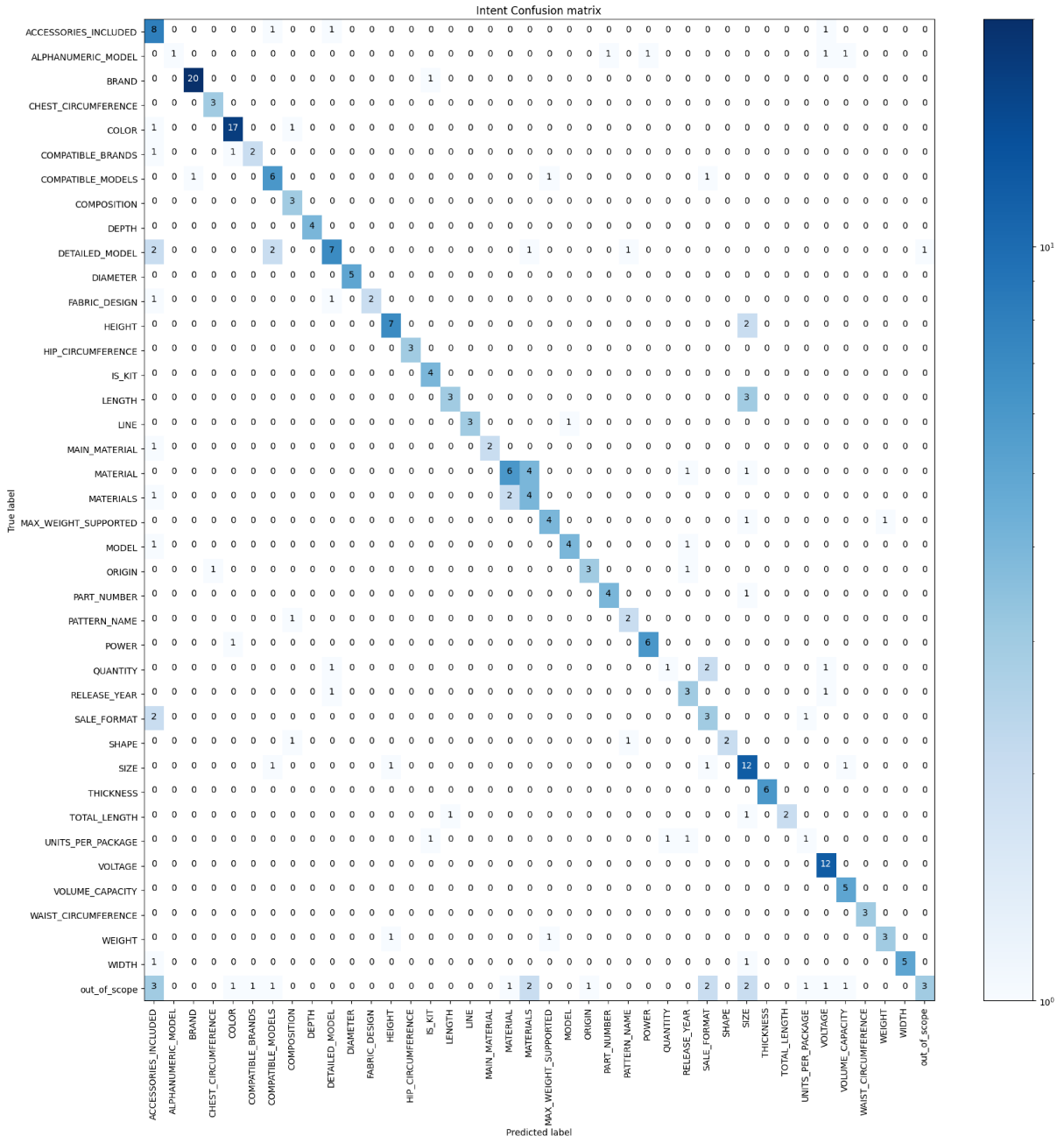

Figura 28 – Matriz de Confusão gerada após teste do modelo RASA1, na configuração com 40 classes.

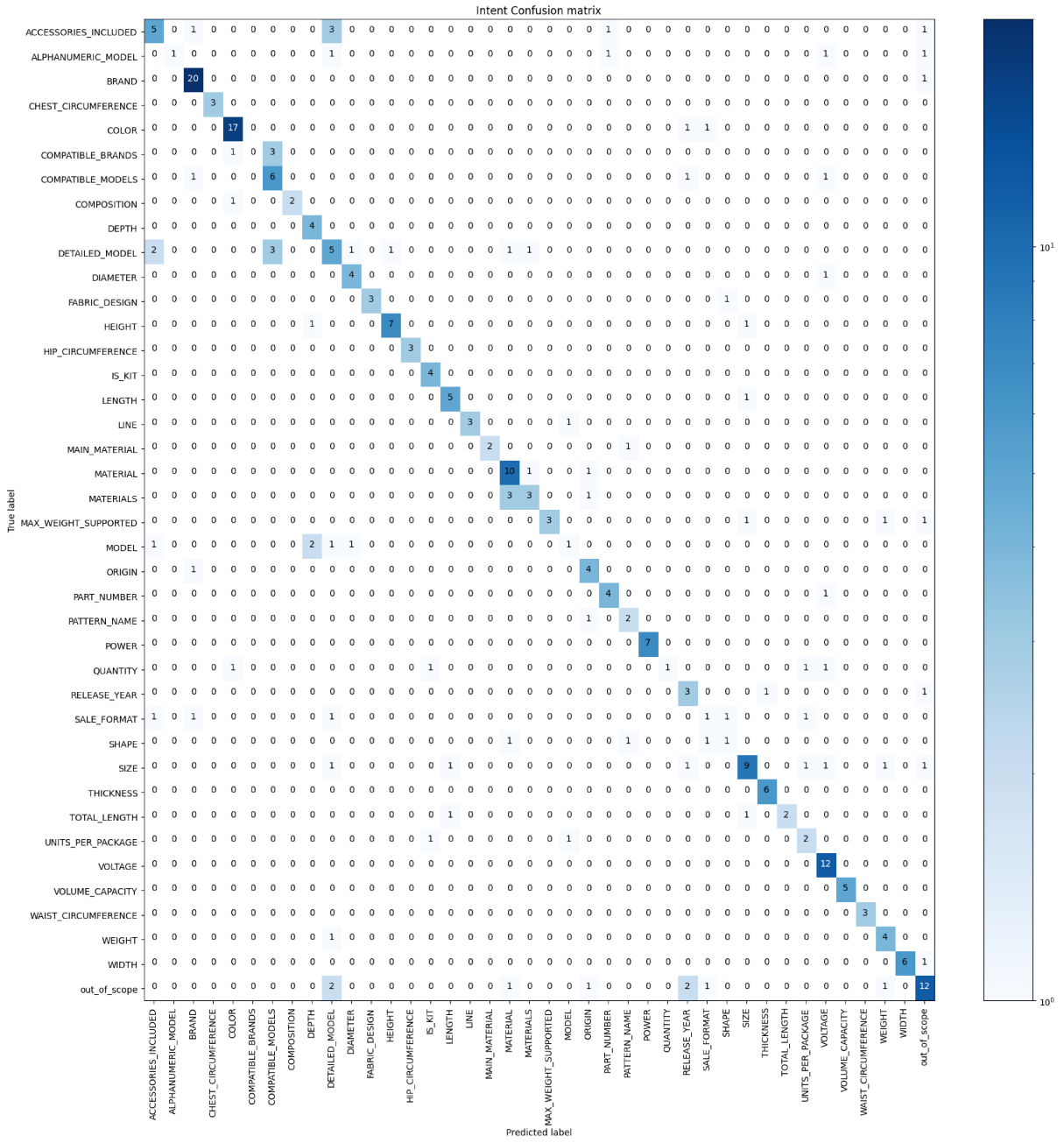

Figura 29 – Matriz de Confusão gerada após teste do modelo RASA2, na configuração com 40 classes.

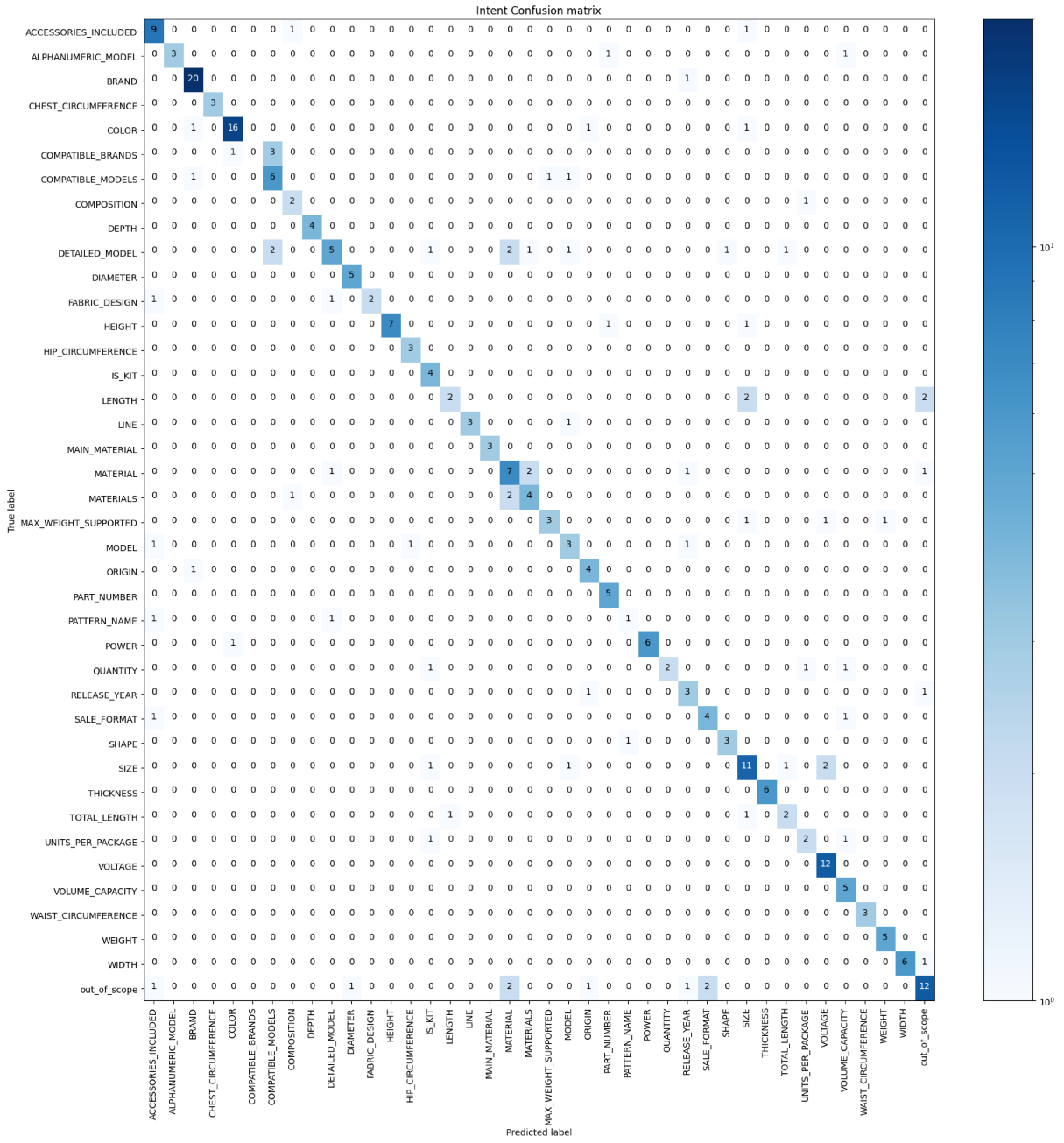

Figura 30 – Matriz de Confusão gerada após teste do modelo RASA3, na configuração com 40 classes.

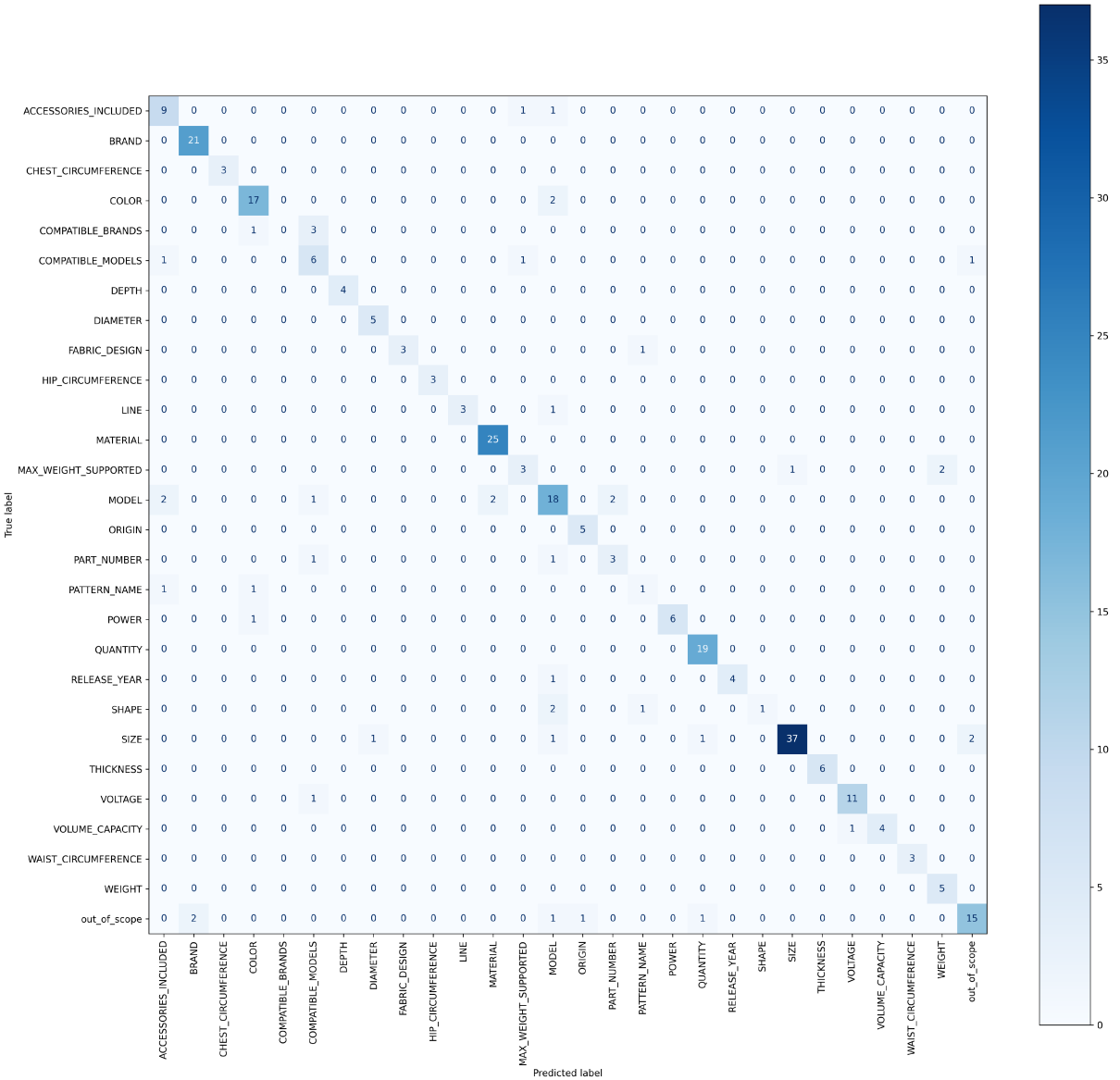

Figura 31 – Matriz de Confusão gerada após teste do modelo HF2, na configuração com 28 classes.

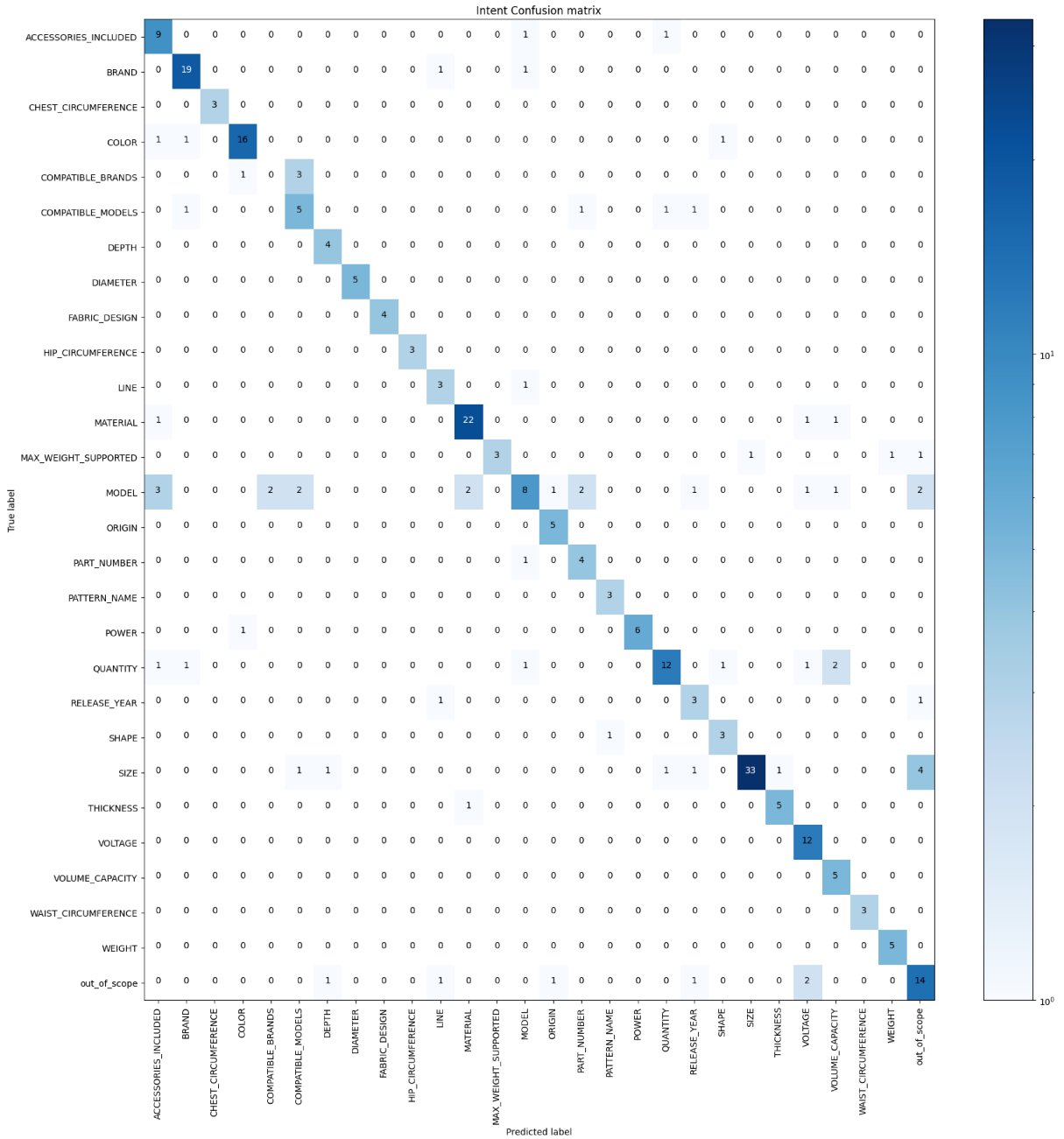

Figura 32 – Matriz de Confusão gerada após teste do modelo RASA3, na configuração com 28 classes.