UNIVERSIDADE FEDERAL FLUMINENSE

VENICIUS GONÇALVES DA ROCHA JUNIOR

Explorando o Desempenho de Algoritmos de Aprendizado de Máquina Aplicados à Classificação de Problemas de Rede a Partir de Descrições Textuais

NITERÓI

2023

UNIVERSIDADE FEDERAL FLUMINENSE

VENICIUS GONÇALVES DA ROCHA JUNIOR

Explorando o Desempenho de Algoritmos de Aprendizado de Máquina Aplicados à Classificação de Problemas de Rede a Partir de Descrições Textuais

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação

Orientador:
Prof. Antônio Augusto de Aragão Rocha, D.Sc.

NITERÓI

2023

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

R672e Rocha Junior, Venicius Gonçalves da
Explorando o Desempenho de Algoritmos de Aprendizado de Máquina Aplicados à Classificação de Problemas de Rede a Partir de Descrições Textuais / Venicius Gonçalves da Rocha Junior. - 2023.
132 f.: il.

Orientador: Antônio Augusto de Aragão Rocha.
Dissertação (mestrado)-Universidade Federal Fluminense, Instituto de Computação, Niterói, 2023.

1. Processamento de Linguagem Natural. 2. Aprendizado de Máquina. 3. Classificação Textual. 4. Pré-processamento textual personalizado. 5. Produção intelectual. I. Rocha, Antônio Augusto de Aragão, orientador. II. Universidade Federal Fluminense. Instituto de Computação.III. Título.

CDD - XXX

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

VENICIUS GONÇALVES DA ROCHA JUNIOR

EXPLORANDO O DESEMPENHO DE ALGORITMOS DE APRENDIZADO DE MÁQUINA APLICADOS À CLASSIFICAÇÃO DE PROBLEMAS DE REDE A PARTIR DE DESCRIÇÕES TEXTUAIS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação

Aprovada em Julho de 2023.

BANCA EXAMINADORA

Prof. D.Sc. Antônio Augusto de Aragão Rocha - Orientador, UFF

Prof. D.Sc. Aline Marins Paes Carvalho, UFF

Prof. D.Sc. Giovanni Ventorim Comarela, UFES

Niterói

2023

Lembre-se que as pessoas podem tirar tudo de você, menos o seu conhecimento.

(Albert Einstein)

Agradecimentos

Em primeiro lugar, agradeço a Deus por ter me ajudado a superar cada dificuldade encontrada e por ter me dado a capacidade de concluir essa dissertação. Gostaria de agradecer também a minha mãe, a minha esposa e a minha família por todo apoio e por todo suporte para que eu pudesse desenvolver esse trabalho.

Gostaria de agradecer também o professor Antônio Augusto de Aragão Rocha que me forneceu toda orientação, todas as ferramentas e materiais necessários para que esse trabalho fosse desenvolvido.

Gostaria de agradecer também a professora Flávia Cristina Bernardini por todo conhecimento passado na disciplina Aprendizado de Máquina que constituiu a base para o desenvolvimento dessa pesquisa.

Gostaria de agradecer também a professora Adita Kulkarni que atua na State University of New York - Brockport por todo conhecimento compartilhado durante o desenvolvimento dessa pesquisa.

Resumo

Com o avanço da tecnologia, muitas organizações utilizam *Trouble Ticket Systems* (TTS) para registrar e gerenciar problemas, facilitando o processo de designação para a equipe técnica adequada. Porém, em organizações, que recebem um grande número de reclamações, a tarefa de alocar ou classificar um problema torna-se um desafio. Nesse contexto, essa pesquisa teve como objetivo elaborar uma solução customizada para classificação automática de reclamações de clientes relacionadas ao serviço de banda larga fixa para uma grande empresa brasileira de telecomunicações, visando viabilizar o aprimoramento de soluções que auxiliem na tarefa de alocar problemas para equipes técnicas com precisão elevada. A metodologia foi composta pela prospecção de soluções que utilizam algoritmos de Aprendizado de Máquina Supervisionado aplicados em conjunto com técnicas de Processamento de Linguagem Natural. Os testes realizados ajudaram a propor uma técnica de pré-processamento textual customizada para classificar uma reclamação em classes de problemas abordados nessa pesquisa. Os resultados obtidos demonstram que a técnica customizada proposta gerou um aumento de acurácia de 84,55% para 88,80%, quando comparada às técnicas comumente utilizadas em pré-processamento textual. Os resultados também demonstram que o algoritmo *Stacking Classifier do Scikit-learn* alcançou o melhor desempenho entre todos os modelos com uma acurácia de 92,33%, considerando toda a base de dados fornecida e 88,80%, considerando uma base de dados para um período de 30 dias. Essa pesquisa pôde contribuir com a melhoria do processo de resolução de reclamações da empresa que disponibilizou os dados para essa pesquisa.

Palavras-chave: Processamento de Linguagem Natural, Aprendizado de Máquina, Classificação Textual, Pré-processamento textual personalizado.

Abstract

With the advancement of technology, many organizations utilize Trouble Ticket Systems (TTS) to record and manage issues, making the process of assigning the appropriate technical staff easier. However, in organizations, which receive a large number of complaints, the task of allocating or classifying a problem becomes a challenge. In this context, this research aimed to develop a customized solution for automatic classification of customer complaints related to fixed broadband service for a large Brazilian telecommunications company, aiming to enable the improvement of solutions that assist in the task of allocating problems to technical teams with high accuracy. The methodology was composed by the prospection of solutions that use Supervised Machine Learning algorithms applied in conjunction with Natural Language Processing techniques. The tests performed helped to propose a customized textual preprocessing technique to classify a complaint into classes of problems addressed in this research. The results obtained demonstrate that the proposed customized technique generated an increase in accuracy from 84.55% to 88.80%, when compared to the techniques commonly used in textual preprocessing. The results also demonstrate that the Stacking Classifier algorithm of Scikit-learn achieved the best performance among all models with an accuracy of 92.33%, considering the entire database provided and 88.80%, considering a database for a period of 30 days. This research could contribute to the improvement of the complaint resolution process of the company that made the data available for this research.

Keywords: Natural Language Processing, Machine Learning, Text Classification, Custom Text Preprocessing.

Lista de Figuras

2.1 Arquitetura do ITIL v1 7

2.2 Arquitetura do ITIL v3 8

2.3 Cubo do Cobit 9

2.4 Níveis de conhecimento linguístico 13

2.5 Exemplo - *Count Vectorizer* 22

2.6 O hiperplano ótimo - SVM. Fonte: [166] 28

2.7 Diagrama de fluxo da técnica *Staking* 32

2.8 Classificação Binária versus Classificação Multi-classe 33

2.9 Um contra um - One vs one 34

2.10 Um contra todos - One vs Rest 34

4.1 Base de dados 50

4.2 Nuvem de palavras com as 200 mais frequentes 53

4.3 Registros com poucos ruídos 53

4.4 Registros com muitos ruídos 53

4.5 Texto bruto e sem modificações 54

4.6 Histograma com as 10 palavras mais frequentes 54

4.7 Remoção e correção de palavras de forma empírica ou manual 56

4.8 Tokenização antes da correção ortográfica 57

4.9 Textos antes da limpeza 57

4.10 Textos após a limpeza 58

4.11 Base de dados com nomes próprios 60

4.12 *Tokens* dos nomes próprios 60

4.13 Tentativa de correção ortogáfica com a biblioteca *pyspellchecker* 61

4.14 182 palavras geradas acrescentando uma letra 63

4.15 243 palavras geradas acrescentando e removendo uma letra 64

4.16 424 palavras geradas acrescentando, removendo e substituindo uma letra . 65

4.17 430 palavras geradas acrescentando, removendo, substituindo e invertendo uma letra 66

4.18 Exemplo de Stemização 69

4.19 Exemplo de Lematização 69

4.20 Fluxograma da empresa 72

4.21 Fluxograma da metodologia 72

5.1 Máquina para execução dos testes - CPU 73

5.2 Máquina para execução dos testes - Memória 74

5.3 Máquina para execução dos testes - Top 74

Lista de Tabelas

2.1 Matriz de Confusão 36

4.1 Motivo3 e seus números de registros 51

4.2 Motivo3 — escolhidos para testes 51

4.3 Problemas e seus números de registros 52

4.4 Quantidade de registros por estado. 52

4.5 *Stop words* removidas e *stop words* não removidas 59

4.6 Exemplo de substituição de palavras 70

5.1 Resultados do teste 1 74

5.2 Resultados do teste 2 75

5.3 Resultados do teste 3 75

5.4 Resultados do teste 4 76

5.5 Resultados do teste 5 76

5.6 Resultados do teste 6 77

5.7 Resultados do teste 7 77

5.8 Resultados do teste 8 78

5.9 Resultados do teste 9 78

5.10 Resultados do teste 10 79

5.11 Resultados do teste 11 79

5.12 Resultados do teste 12 80

5.13 Resultados do teste 13 80

5.14 Resultados do teste 14 81

5.15 Resultados do teste 15 81

5.16 Resultados do teste 16 82
5.17 Resultados do teste 17 82
5.18 Resultados do teste 18 83
5.19 Resultados do teste 19 83
5.20 Resultados do teste 20 84
5.21 Resultados do teste 21 84
5.22 Resultados do teste 22 85
5.23 Resultados do teste 23 85
5.24 Resultados do teste 24 86
5.25 Resultados do teste 25 86
5.26 Resultados do teste 26 87
5.27 Resultados do teste 27 87
5.28 Resultados do teste 28 87
5.29 Resultados do teste 29 87
5.30 Resultados do teste 30 88
5.31 Resultados do teste 31 88
5.32 Resultados do teste 32 88

Lista de Abreviaturas e Siglas

ANATEL	:	Agência Nacional de Telecomunicações
AM	:	Aprendizado de Máquina
Cobit	:	*Control Objectives for Information and Related Technology*
CRM	:	*Customer Relationship Management*
DBI	:	*Davies-Bouldin Index*
DT	:	*Decision Trees*
DF	:	*Document Frequency*
Dunn	:	*Dunn Index*
ERP	:	*Enterprise Resource Planning*
Exin	:	*Examination Institute for Information Science*
FM	:	*F-Measure*
GPR	:	*Gaussian Process Regression*
IR	:	*Information Retrieval*
ISEB	:	*Information Systems Examinations Board*
ITIL	:	*Information Technology Infrastructure Library*
IA	:	Inteligência Artificial
ISO	:	*International Organization for Standardization*
IDF	:	*Inverse Document Frequency*
LDA	:	*Latent Dirichlet Allocation*
MAE	:	*Mean Ebsolute Error*
MSE	:	*Mean Square Error*
MD	:	Mineração de Dados
NB	:	*Naïve Bayes*
NLTK	:	*Natural Language Toolkit*
OGC	:	*Office of Government Commerce*
PLN	:	Processamento de Linguagem Natural
QoE	:	*Quality of Experience* / Qualidade de Experiência
RF	:	*Random Forest*
RNN	:	*Recurrent Neural Networks*

RL	:	Regressão Linear
RL	:	Regressão Logística
RE	:	*Regular Expressions*
RMSE	:	*Root Mean Square Erro*
SI	:	Sistemas de Informação
SVM	:	*Support Vector Machine*
TI	:	Tecnologia da Informação
TF	:	*Term Frequency*
TF-IDF	:	*Term Frequency - Inverse Document Frequency*
TS	:	*Ticket System*
TTS	:	*Trouble Ticket Systems*
VC	:	*Voting Classifier*

Sumário

1 Introdução **1**

1.1 Contextualização do problema 1

1.2 Composição do problema 2

1.3 Solução proposta 3

2 Fundamentação teórica **5**

2.1 Sistemas de Gestão 5

2.1.1 Introdução aos Sistemas de Gestão 5

2.1.2 ITIL 6

2.1.3 COBIT 7

2.1.4 Service Desk 8

2.1.5 Sistemas de controle de chamados 10

2.2 Processamento de Linguagem Natural 11

2.2.1 Introdução ao PLN 12

2.2.2 Conceitos básicos 13

2.2.3 Pré-processamento dos dados 15

2.2.3.1 Remoção de pontuação e caracteres especiais 16

2.2.3.2 Remoção de URLs e *e-mails* 17

2.2.3.3 Normalização 17

2.2.3.4 *Tokenization* 18

2.2.3.5 Remoção de *stop words* 18

2.2.3.6 *Stemming* 19

2.2.3.7 *Lemmatization* 20
2.2.4 Vetorização ou Codificação 21
2.2.4.1 *Bag of Words* 21
2.2.4.2 *TF-IDF* 21
2.2.4.3 *Count Vectorizer* 22
2.2.4.4 *Word Embeddings* 22
2.2.5 Balanceamento de classes 22
2.3 Algoritmos de classificação 24
2.3.1 Árvores de decisão 24
2.3.2 *Naïve Bayes* 24
2.3.3 *Random Forest* 25
2.3.4 Máquinas de Vetor Suporte 26
2.3.5 Regressão Logística 28
2.4 Algoritmos de Regressão 29
2.4.1 Regressão Linear 29
2.4.2 Regressão por Processo Gaussiano 30
2.5 Técnicas de Combinação de Algoritmos 30
2.5.1 Algoritmo de Votação 31
2.5.2 *Stacking* 32
2.5.3 Classificação Multi-classe 33
2.5.3.1 Um contra Um 33
2.5.3.2 Um contra Todos 34
2.6 Métricas 35

3 Trabalhos relacionados 42

3.1 Fluxos e roteamento de chamados pelas equipes técnicas 42
3.2 Classificação de chamados 45

3.3 Classificação de problemas de rede 46

3.4 Qualidade de Experiência 46

4 Metodologia 48

4.1 A base de dados 48

4.2 Pré-processamento dos dados 53

4.2.1 Remoção de URLs e e-mails 54

4.2.2 Remoção de pontuação e caracteres especiais 55

4.2.3 Normalização 55

4.2.4 Remoção e correção de palavras de forma empírica ou manual . . . 55

4.2.5 *Tokenization* 56

4.2.6 Remoção de *Stop Words* 57

4.2.7 Remoção de nomes próprios 59

4.2.8 Método para correção ortográfica 61

4.2.9 *Stemming* 68

4.2.10 *Lemmatization* 69

4.3 Detalhamento da metodologia experimental 70

5 Testes e Análises 73

5.1 Descrição dos testes realizados 73

5.2 Análise dos resultados 93

6 Conclusão 97

Referências 100

Capítulo 1

Introdução

Este capítulo apresentará temas que contextualizam o problema abordado e fornecem uma base teórica para que o leitor possa entender as atividades desenvolvidas. A Seção 1.1 apresenta uma contextualização do problema que esse trabalho aborda. A Seção 1.2 traz para o leitor uma visão mais detalhada do ambiente computacional relacionado ao tema dessa pesquisa. A Seção 1.3 descreve como essa pesquisa irá propor uma solução para o problema descrito nas Seções 1.1 e 1.2.

1.1 Contextualização do problema

Muitas instituições, empresas e até órgãos públicos utilizam sistemas de atendimento (*Trouble Ticket Systems - TTS*) [104, 23, 6] para gerenciar o grande número de problemas, reclamações ou solicitações. Na área de telecomunicações, esses sistemas desempenham um papel essencial apoiando o gerenciamento de reclamações [53] internas e externas a uma organização. A maior parte dessas reclamações estão relacionadas à qualidade da experiência (*Quality of Experience - QoE*) [31, 195] e expressam o grau de satisfação do cliente sobre o serviço recebido. Com o atual avanço das mídias sociais, a insatisfação de um cliente rapidamente é disseminada pela Internet [36], obrigando as empresas a solucionarem seus problemas da forma mais rápida possível, pois, de acordo com [35], a quantidade de reclamações sobre uma empresa tem impacto negativo no seu valor de mercado, fator que diminui sua popularidade e seu número de clientes.

Segundo o relatório semestral de reclamações da Agência Nacional de Telecomunicações (ANATEL) [5], foram registradas 393.910 reclamações no 1º semestre de 2020, 370.165 no 2º semestre de 2020 e 314.235 no 1º semestre de 2021, considerando apenas o setor de banda larga fixa e considerando que essas são apenas as reclamações registradas

na ANATEL [5], sem considerar todas as reclamações recebidas diretamente pelas empresas de telecomunicações. Nesse contexto, de acordo com [178, 127, 192, 53], a análise eficiente, automatizada e rápida de reclamações se torna um desafio para as empresas de telecomunicações, pois essa análise irá definir os recursos necessários para a resolução dos problemas.

1.2 Composição do problema

Uma infraestrutura de rede de uma empresa de grande porte é composta por uma infinidade de enlaces ligando dois ou mais pontos. Essas conexões são compostas, principalmente, por enlaces de fibra óptica, por enlaces de rádio, por enlaces de par trançado tradicional, por enlaces via satélite, além de novas tecnologias não muito difundidas até a época desse trabalho. Uma vez que dois ou mais pontos estão conectados, formando um enlace, chega o momento de estabelecer a comunicação com auxílio de diversos equipamentos utilizados para transmissão, encaminhamento e gerenciamento de dados, tais como: roteadores, *firewalls*, servidores, *switchs*, *HUBs*, conversores, etc.

Os equipamentos de uma infraestrutura de TI geralmente são instalados em salas que possuem sistemas para controlar temperatura, umidade, bem como, sistemas para alertar e combater possíveis incêndios. Além disso, salas para essa finalidade também possuem sistemas para controle, tratamento e monitoramento das fontes de energia elétrica. Além de todos os ativos e meios mencionados, para que a comunicação aconteça de forma satisfatória para o usuário, são necessários diversos tipos de configurações e procedimentos, envolvendo equipes de diversas áreas de conhecimento.

Nesse contexto, para que essas infraestruturas funcionem com alta disponibilidade e com elevado nível de resiliência, são utilizados sistemas de monitoramento ativo e passivo. Nos sistemas de monitoramento ativo, os chamados são criados de forma automatizada a partir de dados de monitoramento, não sendo necessário que alguém registre uma reclamação. Contrariamente, nos sistemas de monitoramento passivo, os chamados são criados por atendentes ou por integrantes de equipes técnicas, responsáveis por inserir informações que possam descrever as características do problema observado.

Geralmente, um sistema de monitoramento ativo produz uma grande quantidade de chamados com informações padronizadas, seguindo sempre uma mesma formatação, facilitando o processamento de forma automatizada por um computador. No entanto, os chamados gerados por sistemas passivos ainda representam os maiores desafios [21], pois

contêm descrição subjetiva, sem padronização ou formatação definida, com erros de ortografia, erros gramaticais e erros de sintaxe, fatores que dificultam o processamento por computador.

Nesse cenário, com diversos tipos de meios de transmissão de dados, com diversos tipos de equipamentos, com diversos sistemas de monitoramento, com chamados sendo criados de forma automatizada [50] e de forma manual, uma grande quantidade de chamados é gerada e armazenada. Como cada chamado está relacionado a uma área específica de conhecimento, para que um chamado seja resolvido, são necessárias equipes com diversas especializações [192], ou seja, com capacidade de resolver tipos específicos de problemas. Tradicionalmente, a alocação de um determinado problema para uma determinada equipe é feita de forma manual por meio de profissionais com conhecimentos multidisciplinares, capazes de identificar a causa principal de um problema e direcioná-lo para a equipe com capacidade técnica de resolvê-lo.

Embora o método manual de alocação de problema seja mais confiável e preciso, devido à inspeção direta por um profissional capacitado, os métodos de alocação de problemas de forma automatizada se apresentam como uma alternativa para redução de custos e aumento da eficiência no processo de alocação de chamados. Embora haja muitas pesquisas relacionadas a esse tema, muitos trabalhos apresentam resultados pouco satisfatórios, ainda não suficientes para implementação em um ambiente real, sugerindo continuação das pesquisas visando o aprimoramento das técnicas existentes. Dessa forma, a melhoria das técnicas automatizadas de alocação de problemas se faz necessária a fim de otimizar o processo de resolução de um chamado e elevar a confiabilidade de todo o processo.

1.3 Solução proposta

Nos últimos anos, diversas técnicas que interpretam a língua humana, em sua forma denominada natural, foram desenvolvidas e aprimoradas. Essas técnicas são capazes de interpretar dados textuais e produzir atributos com informações que os caracterizam, por meio de métodos estatísticos e por meio da relação entre as palavras de um texto. Dessa forma, os atributos gerados podem ser utilizados como variáveis em modelos preditivos, possibilitando automatização de procedimentos que tradicionalmente precisam de um ser humano para leitura e interpretação.

O Processamento de Linguagem Natural (PLN) é uma técnica computacional que analisa textos ou discursos usando um conjunto de teorias e tecnologias, incluindo linguís-

tica (ou seja, o estudo científico da forma da linguagem, significado e contexto) e métodos estatísticos que inferem regras e padrões a partir de dados [125]. Dessa forma, é possível categorizar um texto como uma estrutura formada por elementos hierarquicamente discriminados, com uma organização fixa e com uma terminologia padronizada para cada elemento, para que o texto seja facilmente consultado e manipulado [108] por processos computacionais.

Nesse contexto, esse trabalho apresenta uma prospecção de soluções que utilizam algoritmos de Aprendizado de Máquina Supervisionado aplicados em conjunto com técnicas de PLN para classificar descrições textuais subjetivas de problemas de rede de banda larga. Essa pesquisa teve como objetivo elaborar uma solução customizada para classificação automática de reclamações de clientes relacionadas ao serviço de banda larga fixa para uma grande empresa brasileira de telecomunicações, visando viabilizar o aprimoramento de soluções que auxiliem na tarefa de alocar problemas para equipes técnicas com precisão elevada.

Capítulo 2

Fundamentação teórica

Com a finalidade de nivelar conhecimento e de disponibilizar um guia teórico sobre as áreas do conhecimento necessárias para reprodução das atividades desenvolvidas nesse trabalho, esse capítulo apresenta uma introdução sobre as teorias e técnicas que guiaram todo o processo de pesquisa desenvolvido.

2.1 Sistemas de Gestão

A Seção 2.1 apresenta ao leitor uma introdução à área de conhecimento relacionada ao problema para o qual esse trabalho testa e compara diversas soluções e propõe a mais adequada. Os sistemas de controle de chamados surgem dentro do contexto dos Sistemas de Gestão.

2.1.1 Introdução aos Sistemas de Gestão

Com meios de comunicação global de alta velocidade e com todas as facilidades para disseminar e obter informações pela grande rede, o consumidor contemporâneo passou a exercer grande influência no sucesso ou fracasso de uma organização. Como a insatisfação de um cliente pode atingir milhares de pessoas em um curto intervalo de tempo, a rápida resolução de problemas vem se tornando uma qualidade essencial ao se escolher um produto ou serviço. Dessa forma, os meios e as atividades necessárias para controlar e gerir problemas de forma cada vez mais eficiente se tornaram desafios e geraram demandas por soluções tecnológicas.

No mundo tecnológico contemporâneo, como a velocidade de obtenção de informações pelos consumidores aumentou de forma considerável, as organizações foram estimuladas

a desenvolver soluções que pudessem auxiliar o processo de resolução de problemas de forma cada vez mais rápida e eficiente. Nesse contexto, os sistemas que pudessem auxiliar, facilitar e aumentar a eficiência da gestão começaram a ser desenvolvidos e estão sendo aperfeiçoados continuamente até o presente momento.

Na medida em que o desenvolvimento dos Sistemas de Informação (SI) foi se aprimorando, a utilização dessas tecnologias se tornou um fator indispensável para uma organização conseguir perdurar em um ambiente empresarial cada vez mais exigente, competitivo e otimizado como mostra [9]. Nesse cenário, as ferramentas de *Enterprise Resource Planning* (ERP) e *Customer Relationship Management* (CRM) surgiram e vêm sendo aprimoradas continuamente a fim de acompanhar e atender as novas demandas geradas pelas organizações [101]. Além disso, foram surgindo diversas padronizações com a finalidade de orientar como cada organização deveria agir em relação a processos envolvendo Tecnologia da Informação (TI), como, por exemplo: *Information Technology Infrastructure Library* (ITIL) e *Control Objectives for Information and Related Technology* (Cobit). É nesse contexto, que as ferramentas de gestão de chamados estão inseridas.

2.1.2 ITIL

O *Information Technology Infrastructure Library* (ITIL) é uma padronização de referência para gerenciamento de processos de Tecnologia da Informação (TI). Criado pela secretaria de comércio do governo da Inglaterra (*Office of Government Commerce* - OGC), o ITIL foi concebido por meio de pesquisas envolvendo consultores, especialistas e acadêmicos com a finalidade de criar um modelo de boas práticas para a gestão da processos da área de TI [186].

A metodologia descrita na biblioteca ITIL é a mais aceita no mundo e objetiva descrever os processos necessários para gerenciar serviços de TI de forma eficiente e eficaz, facilitando a garantia dos níveis de serviço acordados [61]. Para garantir que seus conceitos estão sendo aplicados de forma correta, o ITIL conta com institutos responsáveis por certificações. Os principais institutos certificadores são o *Examination Institute for Information Science* (Exin) e o *Information Systems Examinations Board* (ISEB). Além desses institutos, diversas organizações oferecem treinamentos, preparações e auxílios para implementação dos conceitos preconizados no ITIL.

A implementação dos processos preconizados pelo ITIL tem a finalidade de aumentar a disponibilidade, a satisfação dos usuários e clientes e a economia de recursos financeiros gastos com serviços de TI, diminuindo o retrabalho e otimizando o gerenciamento de re-

cursos [158]. Para que isso seja possível, em sua primeira versão, a arquitetura do ITIL era composta por cinco grandes áreas: perspectiva do negócio, aplicações de gerenciamento, entrega de serviços, suporte a serviços e gerenciamento de infraestrutura como ilustrado na Figura 2.1.

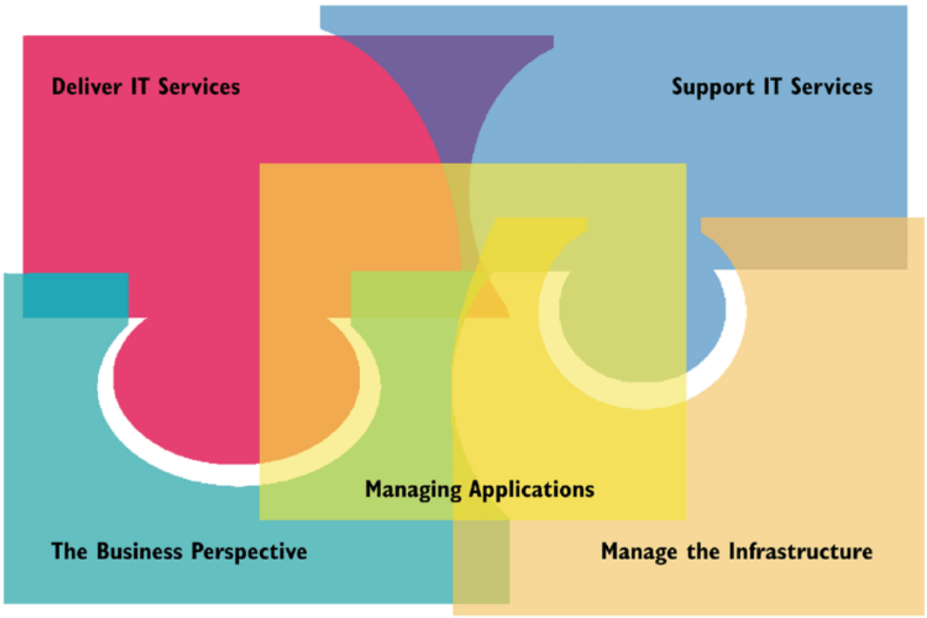

Figura 2.1: Arquitetura do ITIL v1

Em 2001, foi publicada a versão 2 do ITIL com melhorias nos módulos de Suporte de Serviços e Entrega de Serviços, visando modelos mais concisos e mais adequados às realidades das organizações, se tornando em 2002 a biblioteca de boas práticas mais utilizada para gerenciamento de serviços de TI.

Em 2007, foi publicada a versão 3 do ITIL com novas abordagens focadas no ciclo de vida de serviços de TI funcionando de forma integrada com as atividades de negócio. Nessa versão, foram incluídos módulos sobre Estratégia de Serviços, Design de Serviços, Transição de Serviços, Operações de Serviços e Melhoria Contínua, como ilustrado na Figura 2.2.

2.1.3 COBIT

O *Control Objectives for Information and Related Technology* (Cobit) foi criado com a finalidade de focar no negócio por meio do controle dos processos de uma organização. Teve seu início em 1996, focando em processos para controle e análise de Sistemas de Informação [4]. Em 1998, foi publicada a segunda versão do Cobit com melhorias e

Figura 2.2: Arquitetura do ITIL v3

novos módulos fornecendo guia prático para implementação e execução. Em 2000, foi publicada a terceira versão do Cobit com a adição de um módulo com recomendações para gerenciamento de ambientes de TI visando aumentar os níveis de governança. Em 2005 e 2007, foram lançadas as versões 4 e 4.1 do Cobit focando no controle para garantir a segurança da informação, a alta disponibilidade dos ativos de TI, incorporando práticas de governança de TI.

Em 2012, o COBIT 5 foi publicado fornecendo para as organizações padrões integrados com os recursos da Biblioteca de Infraestrutura da Tecnologia da Informação (ITIL) e seguindo os padrões da Organização Internacional de Padronização (ISO). A versão mais atual é a publicada em 2019, focando em uma governança de TI mais flexível e colaborativa de acordo com os últimos avanços tecnológicos. O Cobit é dividido em quatro domínios e em 34 processos [57], como representado na Figura 2.3.

2.1.4 Service Desk

De acordo com [120], um escritório de serviços (*Service Desk*) deve ser único para um determinado domínio de uma organização, visando a centralização da gestão de chamados e evitando desperdícios com duplicidade de informação. A versão 3 do ITIL estabelece

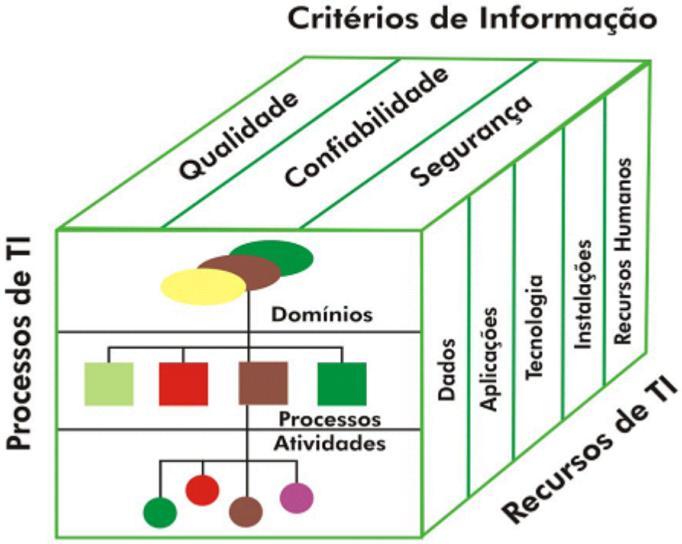

Figura 2.3: Cubo do Cobit

que essa é a área de uma organização responsável por apoiar áreas como: gerência de incidentes, gerência de mudanças, gerência de configuração, etc. De acordo com ITIL, *Service Desk* é definido como uma função composta por ferramentas que documentam as requisições e inspecionam todo o processamento. No contexto desse trabalho, as análises e propostas estão relacionadas à gestão de problemas e incidentes.

As principais funções de um *Service Desk* são: receber chamados, registrar e rastrear incidentes e reclamações, manter histórico de ações para resolução, avaliar e encaminhar requisições para equipes especializadas, gerenciar o ciclo de vida de uma requisição, comunicar a clientes possíveis mudanças, coordenar diferentes equipes de suporte, oferecer informações gerenciais e recomendações de melhoria, identificação proativa de problemas, identificar necessidades de treinamentos específicos de acordo com as demandas, etc.

Segundo ITIL publicou em 2005, em sua quarta versão, os seguintes benefícios podem ser alcançados com a implantação de um *Service Desk*: melhoria do serviço, percepção e satisfação do cliente, centralização de informações por meio de um ponto único de contato, melhoria do gerenciamento de requisições do cliente, melhoria da comunicação e do trabalho em grupo, redução de impactos negativos, melhoria do controle sobre a infraestrutura, otimização do uso dos recursos de TI, aumento da produtividade, melhoria das informações gerenciais para apoio à decisão, etc.

2.1.5 Sistemas de controle de chamados

Conhecidos em inglês como *Ticket System* (TS) ou *Trouble Ticket System* (TTS), esses sistemas são compostos por programas, sensores e, em alguns casos por atuadores, responsáveis por registrar e gerenciar solicitações de usuários ou clientes de uma organização, bem como, em algumas implementações, são capazes de registrar chamados de forma automatizada a partir de dados de monitoramento e realizar determinadas ações pré-programadas de contato com equipes técnicas [69].

Como definido no ITIL, cada chamado (*Ticket*) representa uma solicitação gerada por uma demanda de um cliente, de um usuário, ou uma demanda gerada pelo próprio sistema automaticamente. De forma generalizada, os sistemas de controle de chamados mantêm um histórico de cada chamado, catalogando dados das ações e possíveis soluções já aplicadas. Dessa forma, com um sistema desse tipo em funcionamento por um certo período de tempo, é possível obter dados sobre o ciclo de vida de diversos tipos de chamados e analisar as possíveis soluções já aplicadas em tentativas de resolução.

Como sugerido e padronizado pelo ITIL nos módulos de suporte de serviços, um sistema de chamados deve ser responsável por registrar todo o contato com clientes e usuários de uma organização. A fim de gerir todos os registros feitos de forma eficiente, o ITIL também estabelece que o atendimento deve ser organizado em níveis com funções bem definidas e segmentadas, com níveis de conhecimento técnico adequados.

O primeiro nível definido é responsável pelo primeiro contato com o usuário a fim de registrar os relatos sobre o problema ou solicitação. Nesse nível, caso o atendente identifique que o problema é recorrente e que já tem solução bem definida no sistema, é sua responsabilidade fornecer a solução ao usuário, bem como, encerrar o chamado. Esse nível também deve ser o responsável por contactar o autor requisitante para verificar e encerrar chamados que receberam soluções de outros níveis.

O segundo nível definido é responsável por fornecer o suporte que o primeiro nível não tinha capacitação técnica para fornecer. A equipe de segundo nível deve possuir formação em TI com diversas especializações, e ser capaz de acessar os ativos e equipamentos de uma organização para realizar testes e modificações necessárias.

O terceiro nível sugerido pelo ITIL deve ser o responsável pelos problemas que o segundo nível não foi capaz de resolver. Esse nível deve receber incidentes com elevado nível de complexidade, que, geralmente, impactam grande número de usuários. A equipe técnica do terceiro nível deve ser composta por indivíduos com capacitação elevada, com

conhecimentos amplos sobre o *design*, projeto e implementação dos serviços ofertados por uma organização.

No contexto desse trabalho, as atividades foram desenvolvidas com a finalidade de identificar e propor soluções que possam melhorar a gestão de problemas de um sistema de controle de chamados, aprimorando a distribuição de problemas de rede para equipes técnicas especializadas, aumentando a eficiência de todo o processo e reduzindo custos.

2.2 Processamento de Linguagem Natural

Nessa seção, os principais conceitos sobre o Processamento de Linguagem Natural (PLN) serão apresentados para que o leitor tenha à disposição a base teórica necessária para a reprodução desse trabalho. Na Subseção 2.2.1, o tema será abordado de forma geral, introduzindo os principais conceitos que serão abordados nas próximas subseções. Na Subseção 2.2.2, serão definidos e detalhados os principais conceitos de PLN que foram necessários para as atividades desenvolvidas nesse trabalho. Na Subseção 2.2.3, serão apresentados e detalhados os conceitos de pré-processamento de dados mais utilizados na literatura, indicando os que foram usados nesse trabalho.

Na Subseção 2.2.3.1, serão apresentadas as principais técnicas e referências relacionadas à remoção de pontuação e caracteres especiais. Na Subseção 2.2.3.2, as principais técnicas disponíveis na literatura para remoção de URLs e e-mails serão apresentadas, apontando as que foram empregadas nesse trabalho. Na Subseção 2.2.3.3, serão apresentadas as principais técnicas de normalização empregadas para uniformizar os textos analizados nesse trabalho. Na Subseção 2.2.3.4, serão apresentados os conceitos de tokenização utilizados de acordo com a necessidade de cada teste.

Na Subseção 2.2.3.5, as técnicas utilizadas para a remoção de *stop words* serão apresentadas, algumas seguindo as padronizações amplamente utilizadas na literatura e outras técnicas personalizadas que geraram aumento de performance nos testes realizados nesse trabalho. Na Subseção 2.2.3.6, as técnicas de *Stemming* utilizadas serão apresentadas. Na Subseção 2.2.3.7, a técnica utilizada de *Lemmatization* será apresentada. Na Subseção 2.2.4, serão apresentadas as principais técnicas aplicadas na codificação de dados para o PLN, bem como, serão indicadas as que foram utilizadas nas pesquisas desenvolvidas. Na Subseção 2.2.5, serão abordadas as principais justificativas para utilização de balanceamento dos dados utilizados em testes com algoritmos de Aprendizado de Máquina (AM).

2.2.1 Introdução ao PLN

O primeiro registro acadêmico de algo relacionado ao PLN ocorreu em 15 de julho de 1949 e foi escrito pelo matemático *Warren Weaver*, quando atuava como pesquisador da *Rockefeller Foundation*. *Weaver* escreveu um memorando denominado *"Translation"*, reproduzido em [115], apresentando um conjunto de propostas sobre tradução entre idiomas utilizando máquinas e o enviou para mais de 200 colegas pesquisadores. Todas as sugestões e ideias vieram dos seus estudos sobre teoria da informação e sobre o sucesso da quebra de dados criptografados durante a 2ª Guerra Mundial. Desse momento em diante, muitas universidades americanas começaram a desenvolver pesquisas na área de tradução com máquina.

Alguns anos após a primeira referência ao assunto, em 1954, como resultado de uma cooperação entre a IBM e a *Georgetown University*, como descrito em [77], foi realizado o que ficou conhecido como o primeiro experimento de tradução com máquina. Nesse experimento, um computador IBM 701 conseguiu traduzir 60 frases do idioma russo para o inglês de forma automática. A partir dessa época, novas técnicas envolvendo análises textuais por máquinas foram e estão sendo desenvolvidas até hoje.

Conforme apresentado por [108], o PLN é definido como um conjunto de técnicas computacionais para analisar e representar textos produzidos de forma natural pelo ser humano, de forma escrita ou falada, em qualquer idioma, estilo ou gênero, considerando um ou mais níveis de análise linguística, com o objetivo de se aproximar do processamento de linguagem que um ser humano realiza de forma natural. Na definição apresentada, também há a condição de que o texto analisado não tenha sido construído com o propósito da análise, pois a ideia é produzir conhecimento a partir de textos produzidos pela comunicação natural do ser humano. Para que o PLN atinja os objetivo definidos, diversas teorias e tecnologias são utilizadas, bem como, novas abordagens são continuamente desenvolvidas até o momento de produção desse trabalho.

Alguns autores consideram que o PLN é uma das partes que compõem o conceito mais abrangente de Inteligência Artificial (IA). Essa consideração faz sentido, pois o objetivo das técnicas de PLN é se aproximar da capacidade humana de compreender um texto, ou seja, conseguir desenvolver em uma máquina a parte da inteligência humana responsável pela compreensão textual. Adicionalmente, o PLN também está presente em outras áreas de pesquisa, tais como: Recuperação de Informação (*Information Retrieval* - IR), Perguntas e Respostas (*Question-Answering*), Extração de Informação (*Information Extraction*), Mineração de Textos (*Text Mining*), Geração Automática de Metadados (*Automatic Me-*

tadata Generation), Recuperação de Informação entre Idiomas (*Cross-Language Retrieval*) e Resumo de Documentos (*Document Summarization*).

2.2.2 Conceitos básicos

Como apresentado por [108], para compreender o que acontece no PLN humano, é preciso analisar os diversos níveis de conhecimentos linguísticos. Segundo a pesquisadora, pesquisas na área de psicolinguística sugerem que o PLN humano é dinâmico e resultado da troca contínua e simultânea de informações entre os diversos níveis de conhecimentos linguísticos. Dessa forma, quanto mais níveis de conhecimentos linguísticos um sistema utilizar de forma simultânea, mais eficiente esse sistema será. Nesse contexto, os principais níveis de conhecimento linguístico seriam, como ilustrado na Figura 2.4: fonologia, morfologia, lexicologia, análise sintática, análise semântica, análise de discurso e pragmática.

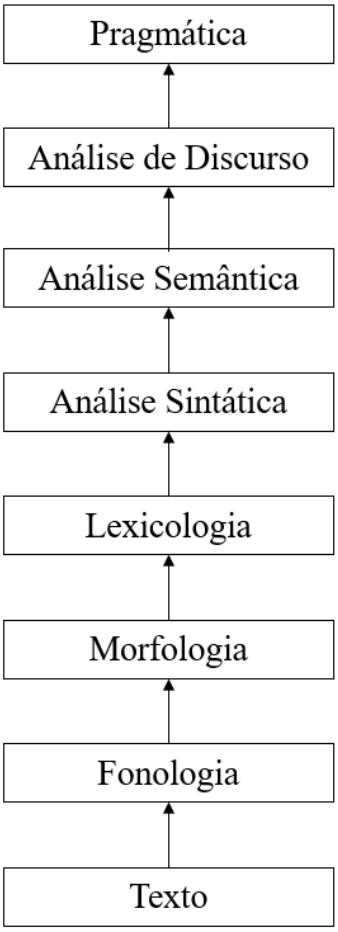

Figura 2.4: Níveis de conhecimento linguístico

A fonologia é responsável pelas técnicas relacionadas à interpretação dos sons das

palavras. Nesse nível de conhecimento, são utilizados três tipos de regras para análise fonológica: regras fonéticas, que analisam sons das palavras de forma quase que isolada, regras fonêmicas, que analisam variações de pronúncia quando as palavras são faladas em conjunto, regras prosódicas, que analisam a variação de entonação ao longo de uma frase [114].

A morfologia é responsável pelas técnicas de análise com foco no tipo de classificação de uma palavra, geralmente, as análises são feitas com base na separação das partes que compõem uma palavra: prefixo, radical e sufixo, denominados morfemas. Como o significado de cada morfema não se altera, os seres humanos conseguem decompor uma palavra desconhecida em seus morfemas constituintes para compreender o seu significado. Nesse contexto, um sistema de PLN que analisa a parte morfológica, tenta identificar o significado de cada morfema para tentar estabelecer relações de significado [141].

A lexicologia é responsável por interpretar o semântica de uma palavra em um contexto individual, com base em sua constituição mórfica, em variações flexionais e com base em sua classificação semântica em reação a outros vocábulos. Sistemas que utilizam análises léxicas geralmente utilizam técnicas de atribuição de rótulos (*tags*) específicos para cada palavra, considerando o rótulo mais provável com base no contexto em que a palavra ocorre para palavras que podem assumir mais de uma posição em um discurso [154].

A análise sintática é responsável por analisar as palavras quando utilizadas em conjunto com outras palavras em uma frase, tentando identificar a estrutura gramatical da frase. Para que esse tipo de análise seja possível, é necessário que se tenha a definição de uma gramática e de um analisador. O resultado dessa análise é uma representação da frase com as relações de dependência estrutural entre as palavras. Existem várias gramáticas que podem ser utilizadas e que, por sua vez, têm influência na escolha de um analisador [32]. Esse tipo de análise pode não mostrar o significado real de uma frase na maioria das línguas, pois a ordem e a dependência entre as palavras contribuem para construção do significado. Por exemplo, as frases "o cachorro perseguiu o gato" e "o gato perseguiu o cachorro" possuem estruturas semelhantes em termos de sintaxe, mas transmitem significados bastante diferentes.

A análise semântica é responsável por analisar o texto em relação ao sentido, porém, essa análise é complementada pelas outras apresentadas para que se possa encontrar o verdadeiro significado de um texto, concentrando-se nas relações entre os significados das palavras. Este nível de análise pode resolver o problema da ambiguidade semântica, no qual, as palavras podem assumir múltiplos sentidos, de forma semelhante ao problema

da ambiguidade sintática, no qual, as palavras podem ocupar diversas posições em uma frase. A análise semântica permite definir somente um sentido para palavras polissêmicas [7]. Diversos métodos podem ser implementados para retirar a ambiguidade de um texto, alguns utilizam a frequência de cada sentido de palavra em um determinado corpus, alguns casos precisam ser tratados de forma pontual analisando o contexto em que ocorre, e outros utilizam conhecimento pragmático determinado pelo tipo de documento cujos textos estão sendo analisados.

A análise de discurso tem o objetivo de analisar amostras de textos maiores que uma frase. O reconhecimento da estrutura do discurso determina as funções das frases no texto, que, por sua vez, contribui para a representação textual com significado [32].

A pragmática é o nível de conhecimento linguístico que leva em consideração o propósito do uso da linguagem em determinadas situações, analisando o contexto de utilização de um texto. Esse nível de conhecimento busca explicar como significados adicionais podem ser obtidos de textos sem estarem escritos de forma explicita. Para tornar essa tarefa possível, a pragmática considera o conhecimento geral sobre o mundo e os acontecimentos relacionados ao texto, incluindo o entendimento de intenções, planos e objetivos [106].

2.2.3 Pré-processamento dos dados

Embora a maior parte dos trabalhos de pesquisa utilizarem pré-processamento [46] de texto para obter melhores resultados nos testes feitos e reduzir a quantidade de informações com ruído, ainda existem propostas que não utilizam as técnicas que serão apresentadas a seguir, como é o caso de [99, 28].

Há também trabalhos que fazem uma comparação entre os resultados obtidos utilizando ou não o pré-processamento de textos, como é o caso de [142, 157, 172] e trabalhos que fazem uma análise comparativa das diferentes técnicas utilizadas em pré-processamento, como é o caso de [44, 90, 156].

As tarefas mais comuns de pré-processamento de textos são realizadas de forma simples, pois a linguagem de programação utilizada nesse trabalho, bem como, algumas ferramentas desenvolvidas para análise de dados [189] sem necessidade de uma linguagem de programação, possuem bibliotecas, funções e funcionalidades prontas para esses tipos de manipulação de textos. No entanto, algumas tarefas de pré-processamento personalizado necessitam de programação, como no caso do corretor ortográfico personalizado que será descrito na Seção 4.2.8.

2.2.3.1 Remoção de pontuação e caracteres especiais

De forma geral, em trabalhos envolvendo o PLN, os caracteres de pontuação e caracteres especiais não se configuram como atributos que trazem informações importantes para as técnicas de classificação [2, 96], como é o caso desse trabalho. Os algoritmos utilizados no PLN são projetados e testados com textos seguindo as regras e padrões estabelecidos por uma gramática e têm suas performances pioradas quando encontram textos não padronizados [87, 91]. No entanto, alguns trabalhos específicos mantêm esses caracteres e os utilizam com algum objetivo [132, 87, 91, 180].

Todas as partes de palavras ou textos, cuja remoção melhora o desempenho de técnicas de PLN são definidas como ruído [2], no entanto, não há um método bem definido para identificar o que é ruído e o que não é. Em relação a esse assunto, as técnicas ainda são aplicadas de forma empírica, pois, para um determinado texto, um caractere especial pode trazer informações úteis, mas, para outro texto, o mesmo caractere pode não acrescentar informação que possa ser interpretada por algoritmos. O trabalho de [174] faz uma análise dos tipos de ruídos encontrados em mensagens de texto de celular, *e-mails* e plataformas de bate-papo.

Textos informais são encontrados principalmente em plataformas de mídias sociais e ao longo de toda a internet [47], pois quem está redigindo o texto não tem necessidade ou compromisso de respeitar as regras gramaticais. Esses textos com conteúdo não padronizado vêm sendo estudados por diversos trabalhos [134, 191, 70, 41, 102, 129] e têm gerado diversas oportunidades de pesquisa sobre a área [87].

Nesse contexto, o trabalho de [2] tenta definir e apresentar uma análise do que seria ruído prejudicial e do que seria ruído útil. Nos testes feitos por [2], são feitas comparações removendo todos os ruídos, removendo alguns e mantendo outros e removendo alguns, mantendo alguns e transformando alguns. Segundo os autores, o teste removendo todos apresenta desempenho inferior quando comparado com os outros testes. O teste removendo alguns caracteres e mantendo outros apresentou desempenho intermediário, melhor que o teste que removeu todos. O teste que apresentou melhor performance foi o que removia alguns caracteres, mantinha alguns e transformava certos caracteres e *emojis* em palavras ou frases, confirmando que a estratégia de remoção de caracteres pode ter diferentes impactos e deve ser feita de acordo com cada modelo criado.

2.2.3.2 Remoção de URLs e *e-mails*

No contexto deste trabalho, URLs e *e-mails* aparecem quando algum usuário registra alguma reclamação relacionada a um problema para acessar determinada página na internet ou algum problema em que o atendente julgue necessário registrar o *e-mail* do usuário no texto da reclamação. Para ambos os casos, a URL ou o *e-mail* não acrescenta informação útil para o objetivo desse trabalho. Dessa forma, como realizado em outros trabalhos relacionados [24, 18, 81] ao PLN, as URLs e *e-mails* foram removidas dos textos da base de dados utilizada nos testes realizados.

Embora no contexto desse trabalho, as URLs e *e-mail* foram removidos dos textos, há casos em que esses termos agregam informações relevantes para determinadas pesquisas, como é o caso de identificação de URLs geradas por robôs de *spam* [73], identificação de URLs usadas em ataques de *phishing* [17, 161], identificação de tendências da área de PLN pela análise de informações sobre artigos científicos [143], com necessidade de analisar URLs, detecção de notícias falsas [201], dentre outros trabalhos.

Há também casos em que a remoção da URL não é indicada, como [58] que, em vez de remover URLs, realiza a substituição de URLs por palavras identificadoras. O trabalho de [2] mostrou que os testes com substituição de caracteres especiais e símbolos por palavras identificadoras obtêm melhor performance do que os testes que realizam apenas a remoção. De forma semelhante, o trabalho de [42] mostra que a substituição de URLs, *hashtags*, e *e-mails* por palavras identificadoras gera aumento de performance, quando a técnica é comparada com a remoção simples.

2.2.3.3 Normalização

Como etapa principal do pré-processamento de um texto para aplicação de técnicas de PLN, e para que no processo de vetorização não haja duas palavras semanticamente idênticas sendo tratadas como palavras diferentes, apenas pelo fato de uma conter letras maiúsculas e outra letras minúsculas, opta-se por deixar todas as palavras com letras minúsculas ou todas as palavras com letras maiúsculas como observado em diversos trabalhos de pesquisa [137, 155, 44, 99].

Em termos práticos, essa tarefa é realizada de forma simples, pois a linguagem de programação utilizada nesse trabalho fornece funções prontas para esse tipo de manipulação de textos.

2.2.3.4 *Tokenization*

A técnica denominada *Tokenization* [131] é responsável por receber uma frase ou um texto e dividi-lo em um conjunto de palavras, encontrando o limite de início e de término de cada palavra [79], ou em alguns casos, dependendo do objetivo do trabalho, em um conjunto de frases, também encontrando os limites de início e de término de cada frase.

O termo Tokenização é bem conhecido por quem trabalha com processamento e análise textual envolvendo processos computacionais. Para analisar um texto de linguagem natural com processos computacionais, diversos mecanismos são definidos para eliminar ambiguidades léxicas e estruturais, pois as linguagens naturais são informais e geralmente ignoram regras definidas de sintaxe e semântica.

Mesmo não sendo o caso desse trabalho, e como o objetivo desse capítulo é passar para o leitor uma visão geral sobre os conhecimentos da área, cabe destacar o que se segue. Existe uma diferença fundamental na tokenização de linguagens separadas por espaço e linguagens cujas palavras não são separadas por espaço. Em línguas nas quais as palavras não são separadas por espaço, como, por exemplo, a chinesa e a tailandesa, as palavras são escritas em sequência sem indicação de final. A tokenização de textos nesses idiomas necessita de informação léxica e morfológica adicional.

Dessa forma, com o texto separado como se fosse um vetor composto por palavras denominadas *tokens*, é possível determinar quantas vezes uma palavra ocorre, ou seja, sua frequência, também é possível implementar a etapa de remoção de palavras vazias, além de tornar possível a aplicação de outras técnicas utilizadas no processamento textual. Na seção sobre vetorização ou codificação serão apresentadas algumas técnicas adicionais que se relacionam com essa técnica, tais como *Bag of Words*, TF-IDF e *Word Embeddings*.

2.2.3.5 Remoção de *stop words*

No contexto de PLN, *stop word* ou palavra vazia seria uma palavra que pode ser removida de um texto, pois não acrescentam informações semânticas ou não identificam algo na língua, servem apenas para indicar quantidade, substituir outras palavras ou fazer as conexões entre diversas palavras que formam uma frase ou texto [163]. De forma simplificada, já temos conhecimento intuitivo sobre palavras vazias, pois é algo que é usado em consultas em um buscador qualquer na internet. Quando alguém faz uma pesquisa, o mais comum é escolher o que considera como palavras-chave para fazer a pergunta certa para o buscador. Dessa forma, a tarefa de excluir artigos, conjunções e preposições é feita

de forma intuitiva.

Na língua portuguesa, essas palavras são artigos, numerais, pronomes, conjunções, alguns verbos, bem como, outras palavras que podem ser removidas de forma particular em um determinado trabalho. A remoção dessas palavras para fins de utilização de algoritmos de previsão não causa grandes impactos na identificação de textos [55].

Uma forma mais robusta para realizar a tarefa de remover palavras de um texto é considerar a frequência de ocorrência de palavras em um texto ou conjunto de textos. Os termos que aparecem em maior número podem se tornar candidatos para a lista de palavras que devem ser removidas [170]. Em relação às técnicas de identificação e remoção de palavras vazias, não há uma definição completa e geral na literatura. Portanto, para cada trabalho, recomenda-se que a lista de palavras que devem ser removidas seja elaborada e revisada manualmente, pois palavras que podem ser removidas em um determinado trabalho, podem influenciar resultados em outro trabalho diferente [155]. Há também casos em que as palavras podem ser muito frequentes e, mesmo assim, apresentar contribuições significativas. Geralmente, isso ocorre quando os textos analisados são sobre um assunto específico.

A etapa após a normalização do texto é a vetorização. Essa etapa será discutida em detalhes posteriormente na seção vetorização deste capítulo, mas a etapa principal na vetorização é construir um vocabulário ou dicionário de todos os *tokens*. O tamanho deste vocabulário pode ser reduzido removendo as palavras vazias, reduzindo o número de cálculos que precisam ser executados durante as etapas de treinamento e de teste. Assim, esse tipo de limpeza textual pode trazer benefícios em termos de velocidade de execução e espaço para armazenamento. Algumas técnicas mais recentes na área de PLN trabalham com listas de palavras vazias cada vez menores, devido ao desenvolvimento de técnicas de codificação e métodos de cálculo mais eficientes.

Diversas ferramentas de PLN fornecem listas de palavras vazias para serem removidas do texto. No caso desse trabalho, foi utilizada a biblioteca *stopwords* do pacote *Natural Language Toolkit* (NLTK) [193] para *Python* em combinação com método empírico para encontrar caracteres e palavras para serem removidas.

2.2.3.6 *Stemming*

Essa técnica, que recebe a denominação *Stemming* ou Stemização, é responsável pela remoção de prefixos e sufixos das palavras de um texto, pois, para o PLN, essas partes

que compõem algumas palavras não têm utilidade e não acrescentam informação útil que possa ser analisada por algoritmos computacionais[171]. Essas técnicas ainda estão pouco desenvolvidas para o idioma português e em alguns casos geram vocábulos que não existem, pois são removidas partes que são semelhantes a um prefixo ou sufixo, mas que na verdade não são nem um, nem outro.

Outro problema comum na aplicação dessa técnica é a obtenção de dois vocábulos iguais, mas cujas palavras originais tinham significados totalmente diferentes. Esses problemas gerados por essa técnica podem impactar o resultado de um modelo, portanto, devem ser observados e analisados de forma cuidadosa. Essa técnica é mais indicada para construção de mecanismos de busca e de indexação de palavras [181].

2.2.3.7 *Lemmatization*

A lematização é uma técnica que também tenta simplificar uma palavra, reduzindo-a à sua forma mais básica. Diferentemente da Stemização, na lematização, o resultado é sempre uma palavra [150]. Por exemplo, a palavra "carreiras" pela Stemização é reduzida ao vocábulo "carr", que não carrega muito significado semântico e pode ter seu significado confundido com o de outras palavras. Diferentemente da técnica anterior, utilizando a Lematização, a palavra "carreiras" é reduzida e transformada na palavra "carreira", mantendo significado e dificilmente poderá ser confundida com outra palavra da língua.

A lematização está diretamente ligada à análise léxica, relacionando variantes morfológicas aos seus lemas, que nada mais são do que as formas mais simples das palavras, ou a forma em que as palavras se encontram no dicionário em sua forma básica de origem. Para essa tarefa, é necessário utilizar um dicionário de lemas em conjunto com informações semânticas e sintáticas [79]. A lematização é utilizada de diferentes formas, que dependem do tipo de tarefa a ser realizada.

A técnica de lematização pode ainda ser estendida para tentar associar palavras que possuem alguma relação semântica, como por exemplo estabelecer relação entre a palavra engarrafamento e a palavra trânsito, que de maneira informal é utilizada para se referir a engarrafamento. Essa técnica é indicada para ser usada em conjunto com outras, tais como, técnicas que utilizam vetorização, *TF-IDF* e modelagem por tópicos como *Latent Dirichlet Allocation* (LDA).

2.2.4 Vetorização ou Codificação

2.2.4.1 *Bag of Words*

É uma das técnicas mais utilizadas para organizar informações de forma estruturada, que possa ser processada por um computador. Os computadores têm mais facilidade de lidar com números, dessa forma, cria-se uma estrutura que associa informações numéricas a textos. A técnica *Bag of Words* permite representar um texto com base na ocorrência de cada palavra, desconsiderando a ordem e a posição em que as palavras aparecem no texto. De fato, como não há manutenção da ordem, nem da posição, as palavras ficam armazenadas em uma estrutura de forma desordenada. Portanto, criou-se o termo *Bag of Words*, pois as palavras ficam como se fossem colocadas dentro de um saco, sem ordem e sem posição [83].

2.2.4.2 *TF-IDF*

TF-IDF significa *Term Frequency - Inverse Document Frequency* ou Frequência do Termo - Inverso da Frequência no Documento, traduzindo para o português. Trata-se de medidas estatísticas que demonstram o quão importante uma palavra é em um texto, assemelhando-se muito com a técnica de *Bag of Words*, mas com algumas diferenças. *Term Frequency* mede a frequência com que um termo aparece em um documento. O termo *Inverse Document Frequency* representa o quão importante um termo é em relação a todo o conjunto de palavras, de acordo com a quantidade de vezes que o termo aparece. Essas duas métricas são usadas para definir a importância da palavra, ou seja, quanto mais frequente a palavra é, mais importante ela será no texto [33].

$$TF - IDF = TF * IDF \tag{2.1}$$

$$TF = \frac{Numero\ de\ vezes\ que\ um\ termo\ aparece\ em\ um\ documento}{Numero\ total\ de\ termos\ no\ documento} \tag{2.2}$$

$$IDF = \log\left(\frac{Numero\ de\ documentos\ no\ corpus}{Numero\ de\ documentos\ no\ corpus\ contendo\ o\ termo + 1}\right) \tag{2.3}$$

2.2.4.3 *Count Vectorizer*

As máquinas não conseguem entender caracteres e palavras. Portanto, para que se consiga trabalhar com dados de textuais sendo interpretados e utilizados por algoritmos computacionais, há necessidade de representá-los em algum formato numérico. O *Count vectorizer* é uma técnica que converte texto em dados numéricos [124]. Para realizar isso, uma matriz esparsa é criada como representado na Figura 2.5:

Texto: Cliente informa que está com problema para acessar internet. Cliente está com lentidão.
Palavras únicas: Cliente Informa que está com problema para acessar internet lentidão
Representação: 2 1 1 2 2 1 1 1 1 1

Figura 2.5: Exemplo - *Count Vectorizer*

Tem-se 10 palavras únicas no texto e, portanto, 10 colunas diferentes, cada uma representando uma palavra única na matriz. Como as palavras 'Cliente', 'está' e 'com' aparecem duas vezes no texto, na matriz que representa o texto, essas palavras receberão a contagem 2 e as demais 1. O *Count vectorizer* facilita o uso direto de dados de texto em modelos de AM que trabalham com classificação de textos.

2.2.4.4 *Word Embeddings*

Traduzindo para o português podemos chamar de incorporação de palavras. Cada palavra é representada por um vetor com valores numéricos. Mantendo o contexto linguístico das palavras, essa técnica foca na semelhança radical e na relação de sentido existente entre os termos, para criar a representação numérica e armazenar em um vetor. Cabe ressaltar que os valores mudam conforme as características da palavras, sendo possível separar palavras no masculino e palavras no feminino [194].

Apesar dos *Embeddings* serem comumente feitos com palavras, eles podem ser feitos à nível de caracteres ou frases. Diversas técnicas podem ser usadas para incorporação, como, por exemplo, *Word2Vec*. *Word2Vec* é um algoritmo desenvolvido pelo *Google* para realizar o *Word Embeddings*.

2.2.5 Balanceamento de classes

Uma base de dados não balanceada é uma base em que o número de dados pertencentes a uma classe, isto é, dados rotulados como parte de um mesmo conjunto com características semelhantes, é significativamente maior ou menor do que o número de dados pertencentes

a outras classes. A maioria dos algoritmos de classificação utilizados no PLN e em AM de forma geral, não estão preparados para receber bases de dados com essas características e tendem a gerar resultados enviesados, favorecendo as classes com maior quantidade de dados.

Nesse contexto, um modelo com um valor de acurácia elevado pode não ser um modelo mais adequado, pois os resultados podem ter sido obtidos com bases de dados não balanceadas. Os algoritmos de classificação mais comuns, como, por exemplo, Árvores de Decisão e Regressão Logística, tendem a apresentar desempenho melhor para classes majoritárias com bases de dados desbalanceadas, quando comparados aos resultados obtidos com bases de dados balanceadas. Isso gera resultados enviesados favorecendo as classes majoritárias, enquanto as classes minoritárias são tratadas como ruído e frequentemente ignoradas. O resultado é uma taxa de erros de classificação mais alta para classes minoritárias em comparação com as classes majoritárias. Para evitar que isso ocorra e dependendo do objetivo de cada projeto, realiza-se o balanceamento do conjunto de dados de treinamento visando obter resultados melhores tanto para as classes majoritárias quanto para as classes minoritárias. Além disso, é importante destacar que não se deve balancear os dados de teste, pois eles representam os dados reais que serão analisados quando o modelo for colocado em produção.

Cabe ressaltar que a acurácia geral de um modelo pode apresentar valores menores quando são utilizados dados balanceados, pois a taxa de acerto da classe majoritária diminui. No entanto, a taxa de acerto da classe minoritária geralmente aumenta, apresentando um valor de *Recall* maior. A escolha de um modelo que tenha desempenho melhor para classes majoritárias ou minoritárias depende do objetivo de cada estudo de caso de análise de dados.

Além disso, a Acurácia não é tão indicada para avaliar o desempenho de um modelo treinado com dados não balanceados, pois tende a apresentar um valor elevado como consequência do número maior de registros da classe majoritária. Para esses casos, as métricas Acurácia, *Recall*, *Precision* e *F-score* devem ser analisadas em conjunto, pois conseguem expressar resultados para as classes majoritárias e minoritárias.

Há na literatura algumas soluções para lidar com bases de dados com classes não balanceadas, como, por exemplo: *Undersampling*, *Oversampling*, *SMOTE* e *Data Augmentation*. A técnica *Undersamling* se baseia em reduzir a quantidade de dados das classes com maior número de elementos por processos de amostragem [138]. A técnica *Oversampling* se baseia em aumentar o número de dados da classe com menor número de

elementos por replicação aleatória [122]. A técnica *SMOTE* se baseia e gerar amostras sintéticas a partir de cada amostra da classe com menor número de elementos, introduzindo exemplos sintéticos com base em algum tipo de medição com valores mais próximos dos vizinhos do elemento amostrado [122]. Para dados textuais, a técnica *Data Augmentation* pode ser implementada por meio da tokenização e mudança da ordem das palavras de uma frases. Outra alternativa, seria gerar novas frases por meio da substituição de adjetivos, verbos e outras palavras por sinônimos para gerar texto diferente com significado semelhante [8].

2.3 Algoritmos de classificação

2.3.1 Árvores de decisão

Técnicas de árvores de decisão têm sido amplamente utilizadas para construir modelos de classificação, destacando-se por sua simplicidade e facilidade de entendimento [94]. Há aplicação de árvores de decisão em medicina[88], análise financeira [119, 199], biologia [59], astronomia [89] e diversas áreas da tecnologia da informação como mostrado nos trabalhos de [64, 86].

A estrutura de uma árvore de decisão se assemelha com um fluxograma, no qual, cada nó indica um atributo e cada ramificação representa um resultado de alguma operação realizada. As árvores de decisão podem ser binárias ou não binárias e é possível manipular dados multidimensionais [64]. Para utilização em algoritmos de classificação, são feitos testes com os nós internos até que se chegue em uma folha contento um rótulo, ou seja, o resultado da classificação [94]. Em alguns casos, técnicas de poda são utilizadas para remover da árvore os nós desnecessários por meio de métodos estatísticos para melhorar o resultado da classificação. Um algoritmo de previsão bem conhecido que utiliza arvore de decisão é o J48 proposto por *Ross Quinlan* em 1993 implementado na linguagem C [88]. É baseado na técnica de dividir para conquistar, onde um problema complexo é dividido em subproblemas mais simples [190, 107].

2.3.2 *Naïve Bayes*

Assim como os classificadores baseados em árvores de decisão, o Classificador *Naive Bayes* (NB) é um dos mais utilizados por apresentar um bom desempenho tanto com dados numéricos quanto com dados categóricos [29]. Também é aplicado em diversas áreas, como

medicina [185], categorização de linguagem [49], análise de sentimento [144], filtragem de *span* [51], etc. Com base no Teorema de *Bayes*, este classificador utiliza estatística e cálculos de probabilidade para realizar a classificação.

O algoritmo *Naïve Bayes* calcula a probabilidade de um elemento qualquer pertencer a cada uma das classes. Dessa forma, seja X uma instância com n atributos e C_i o valor do atributo i. O algoritmo em questão calcula a probabilidade a posteriori $P(C_i|X)$ de X ser da classe C_i , considerando os valores dos atributos de X para cada classe C_i. X será da classe C_i, se e somente se $P(C_i|X)$ for maior do $P(C_j|X)$ para qualquer outra classe C_j. Ou seja, X é da classe C_i , se e somente se $P(C_i|X) > P(C_j|X)$, para todo $1 \leq j \leq m$, $j \neq i$.

A probabilidade a posteriori $P(C_i|X)$ é calculada a partir do Teorema de *Bayes*, de acordo com a Equação 2.4:

$$P(C_i|X) = \frac{P(X|C_i) \times P(C_i)}{P(X)} \tag{2.4}$$

Como $P(X)$ é constante para todas as classes, basta maximizar o numerador da equação 2.4, calculando $P(X|C_i)$ e $P(C_i)$. A probabilidade a priori da classe C_i é representada por $P(c_i)$ e é estimada pela equação 2.5, onde $|C_i, T_r|$ é o número de instâncias da classe C_i em T_r e $|T_r|$ é o número de instâncias de T_r.

$$P(C_i) = \frac{|C_i, T_r|}{|Tr|} \tag{2.5}$$

A implementação do *Naïve Bayes* considera que os atributos são condicionalmente independentes dado o valor da classe [29, 64]. Portanto, $P(X|C_i)$ pode ser calculado conforme a Equação 2.6.

$$P(X|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times ... \times P(x_n|C_i) \tag{2.6}$$

2.3.3 *Random Forest*

O algoritmo *Random Forest* (RF) ou florestas aleatórias é um classificador que usa o método de árvore de decisão do tipo *ensemble* [13, 64]. Um método *ensemble* cria um conjunto de classificadores e combina as classes indicadas por cada um dos classificadores para classificar uma nova instância.

Os métodos *ensemble* mais conhecidos são: (i) *bagging* (*bootstrap aggregation*) que gera classificadores base diferentes modificando aleatoriamente os subconjuntos de treinamento para obter a média dos resultados gerados por cada modelo e assim reduzir a variância, (ii) *boosting* que cria diferentes classificadores base sequencialmente de acordo com o comportamento do classificador anterior para as instâncias do conjunto de treinamento visando reduzir o viés e não a variância e (iii) *stacking* que funciona a partir da criação de um meta-modelo. Utilizam-se "modelos fracos" em uma parte do dataset de treino e um modelo mais robusto na outra parte do dataset, como redes neurais. No entanto, as predições dos modelos fracos são usadas como pesos no modelo robusto. O algoritmo *Random Forest* constrói um conjunto de árvores de decisão como o *bagging*, mas com árvores menos profundas, e as combina para classificar novas instâncias por meio de método de votação [64, 139, 103].

O RF representa uma alteração do *bagging* clássico e oferece algumas vantagens [66], tais como:

- Pode ser usado tanto para classificação quanto para regressão;
- Trabalha com diferentes tipos de dados, tais como categóricos, numéricos e binários;
- A sua acurácia é comparável à do algoritmo *AdaBoost*, entretanto mais robusto a erros e *outliers*;
- Evita problema de *overfitting*;
- São eficientes em bancos de dados muito grandes.

2.3.4 Máquinas de Vetor Suporte

Os conceitos iniciais de *Support Vector Machine* (SVM) foram propostos em 1992 por *Vladimir Vapnik* [11]. A técnica foi formalizada com todas as demonstrações matemáticas no livro de 1998 [182] e descrita e detalhada em seu outro livro de 1999 [183]. Outras referências sobre o algoritmo podem ser encontradas em [39, 72, 20]. O algoritmo é utilizado em diversas áreas como investigação de câncer [75, 76], classificação de imagens [26], previsão de propriedades de reservas de petróleo [140], reconhecimento de caracteres escritos à mão [63], etc. Há também aplicações específicas para bases de dados não balanceadas como mostrado no trabalho de [78].

Esse método para classificação de dados lineares e não-lineares [64], surgiu das pesquisas da área de estatística e ultimamente vem sendo amplamente aplicado por pesquisa-

dores da área de Inteligência Artificial, mais especificamente na área de AM. O algoritmo possui as características de fornecer uma boa precisão, boa capacidade de generalização, boa robustez diante de objetos de dimensões elevadas, como imagens, boa capacidade de lidar com dados com ruídos e *outliers*. Esse método pode ser usado tanto para previsão de dados numéricos (regressão) quanto para classificação.

Com dados linearmente separáveis, o SVM funciona como um hiperplano que separa um conjunto de exemplos positivos de um conjunto de exemplos negativos com uma margem. A margem é definida pela distância do hiperplano para o exemplo mais próximo entre os exemplos definidos como positivos e como negativos [149]. Na Figura 2.6, pode-se observar uma ilustração de um hiperplano ótimo, com a máxima margem de separação. Esse hiperplano fornece a maior separação entre as classes e proporciona maior acurácia na generalização. A Figura 2.6 mostra um exemplo para dados linearmente separáveis.

O processo de classificação de um SVM pode ser resumido dessa forma:

- Dados duas classes C1 e C2 e um conjunto de pontos que pertencem a essas classes;
- O SVM determina um hiperplano que separa os pontos com o objetivo de que o maior número de pontos de uma mesma classe fique do mesmo lado, enquanto maximiza a distância de cada classe a esse hiperplano.
- O hiperplano criado pelo SVM é determinado por um subconjunto de pontos das duas classes chamados de vetores de suporte.
- Após os processos iniciais de treinamento, o algoritmo pode classificar novos elementos em relação ao hiperplano divisor, determinando a qual classe esse novo elemento pertence.

Quando os dados não são linearmente separáveis, realiza-se um mapeamento não linear para transformar os elementos de treinamento, até que esses elementos sejam linearmente separáveis por um hiperplano [64, 43, 117]. Nesses casos, uma função de *kernel* é utilizada para a não-linearização do SVM, contribuindo para a eficiência do algoritmo, possibilitando a construção de hiperplanos simples em um espaço de alta dimensão, simplificando o processo do ponto de vista computacional [117]. As principais funções *kernels* são:

- O *kernel Polynomial* [64, 117];
- O *Pearson VII function-based universal kernel* (Puk) [179, 43];
- O *kernel Radial Basis Function* (RBF) [64, 117].

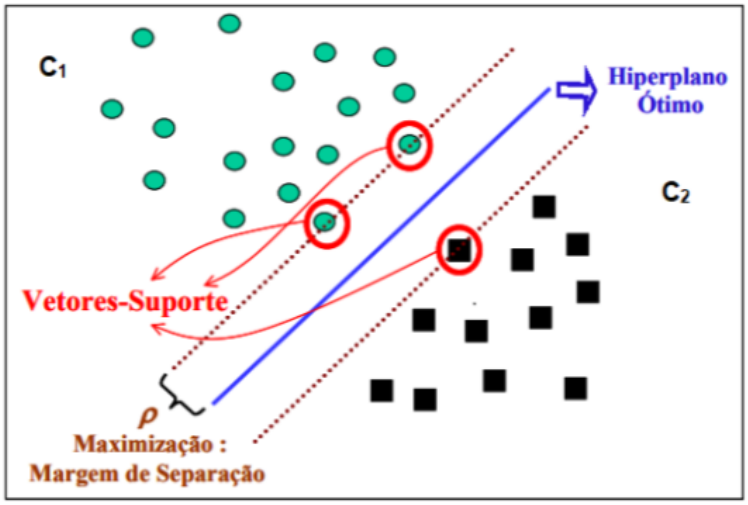

Figura 2.6: O hiperplano ótimo - SVM. Fonte: [166]

2.3.5 Regressão Logística

Mesmo tendo a palavra regressão em seu nome, a Regressão Logística (RL) é um algoritmo de Aprendizagem de Máquina (AM) Supervisionado utilizado para classificação [34]. A nomenclatura desse algoritmo ficou dessa forma, porque ele e construído a partir da aplicação de uma transformação, denominada função logística ou *sigmoide*, sobre a Regressão Linear (RL) [167]. Possui aplicação em diversas áreas tais como análise de imagens de tomografia [160], análises de canais hidrográficos subterrâneos [136], comparação de performance entre a Regressão Logística (RL) e o Random Forest (RF) com bases de dados heterogêneas [93], análise das avaliações de clientes de serviços de entregas de comida [128], além de outras áreas de pesquisa.

Utilizar o método de Regressão Linear (RL) diretamente em problemas de classificação se torna inviável, pois o objetivo da previsão é identificar rótulos e não valores quantitativos. Para lidar com esse impasse, seria interessante ter uma função responsável por uma transformação, permitindo um mapeamento entre valores numéricos e rótulos [116]. Essa função em questão é denominada função logística ou sigmoide [60] consegue fazer uma transformação que possibilita a solução do problema.

$$g(z) = \frac{1}{1 + e^{-z}} \tag{2.7}$$

- Na equação 2.7, $g(z)$ representa a probabilidade de uma dada instância pertencer a classe analisada.

- Na equação 2.7, z é um número real dado pela combinação linear dos atributos utilizados na predição, derivado da regressão linear.

Dessa forma, a classificação só é possível se for definido um limiar de decisão. Registros com probabilidade acima desse limiar são considerados da classe 1 e registros com probabilidade abaixo desse limiar são considerados da classe 0. Após essas definições, o funcionamento do algoritmo é semelhante a uma regressão [34].

2.4 Algoritmos de Regressão

As técnicas envolvendo regressão foram desenvolvidas por diversos altores ao longo do tempo, como é possível observar em [175, 16, 56, 14, 85, 173]. A regressão é um método matemático utilizado para inferir relações entre uma variável dependente e variáveis independentes com o objetivo de gerar previsões de valor [175]. As técnicas de regressão já foram aplicadas em diversas áreas de pesquisa, tais como medicina [118, 98], finanças envolvendo criptomoedas [151], finanças [37, 15], biologia [19], análise de imagens em astronomia [10], análises envolvendo telecomunicações [123], entre outro áreas de pesquisa.

O processo de regressão aplicado à análise de dados é composto por etapas de aprendizado e de predição. Na análise feita utilizando regressão, o atributo que se deseja prever possui um valor numérico em vez de um valor categórico ou um rótulo. O modelo matemático é composto por uma função gerada com base nos valores conhecidos dos dados de treinamento. O modelo utiliza essa função matemática gerada para realizar a previsão de uma nova instância do atributo dependente [27]. O objetivo desta seção é fornecer uma visão geral e resumida de como cada algoritmo funciona para que o leitor tenha base para entender e reproduzir o trabalho desenvolvido.

2.4.1 Regressão Linear

O termo "Regressão" foi proposto em 1885 [100, 95] pelo matemático e estatístico *Francis Galton*, em suas pesquisas sobre estudo das medidas e da matemática dos corpos humanos. Na época, *Galton* de dedicava a estudar o desenvolvimento de pais e filhos e observou que filhos de pais com altura baixa em relação à média tendem a ser mais altos que seus pais, e filhos de pais com estatura alta em relação à média tendem a ser mais baixos que seus pais, ou seja, as alturas dos seres humanos em geral tendem a regredir à média [100].

Regressão Linear (RL) é um tipo de algoritmo supervisionado que modela a variável

a ser prevista y, denominada dependente, como uma função linear do conjunto de dados x, denominados atributos independentes, usando o seguinte Equação 2.8:

$$y = wx + b \tag{2.8}$$

Na equação 2.8, w é o coeficiente de regressão e b é o componente estocástico (muitas vezes referido ao ruído), que representa possíveis erros e desvios [65]. Mais detalhes sobre a implementação do algoritmo podem ser encontrados em [126, 74].

2.4.2 Regressão por Processo Gaussiano

O algoritmo *Gaussian Process Regression* (GPR) está baseado em uma abordagem bayesiana que usa uma coleção de variáveis aleatórias com distribuição Gaussiana para inferir a distribuição de probabilidade de uma função de regressão. O Algoritmo infere a probabilidade distribuição sobre todas as funções aplicáveis que se ajustam aos dados, identificando os relacionamentos entre os dados por inferência Bayesiana. Este algoritmo usa a abordagem de aprendizado denominada preguiçosa e uma função *kernel* para identificar padrões e realizar previsões.

Esse algoritmo é aplicado em diversas áreas tais como física atômica e molecular [45], estudos de geração de energia renovável [169], estudos de como o COVID-19 se espalhou pelos Estados Unidos [184], estudos sobre previsão da capacidade de recarga de baterias [111, 105], estudos relacionados a previsão da velocidade do vento [22], estudos de localização em ambientes internos com base em dados de redes *WI-FI* [176], previsão de temperatura de supercondutores [197], e muitos outros estudos e pesquisas.

Como o detalhamento matemático desse algoritmo é um pouco mais complexo, caso o leitor tenha interesse em aprofundar seus conhecimentos, as seguintes referências são recomendadas [165, 45].

2.5 Técnicas de Combinação de Algoritmos

A performance de um modelo de Aprendizado de Máquina (AM) geralmente é medida utilizando-se alguma técnica de medição, como, por exemplo, uma das técnicas apresentadas na seção sobre métricas desse trabalho. No entanto, há na literatura algumas técnicas que tentam melhorar a performance obtida pelo uso de um único algoritmo aplicado de forma simples e tradicional. Nesse contexto, existem várias técnicas cujo objetivo é me-

lhorar o desempenho de modelos que utilizam algoritmos de forma isolada, por meio do uso combinado de um ou mais algoritmos, seguindo determinada lógica de aplicação.

Essa técnica de combinar diversos algoritmos tem mostrado de acordo com diversos trabalhos na literatura [159, 97, 48, 121] que pode melhorar o desempenho de modelos preditivos, combinando vários modelos básicos para produzir um modelo mais robusto. Nessa seção, serão apresentadas as principais técnicas que utilizam a combinação de algoritmos que foram utilizadas nos testes feitos nesse trabalho. Dentre essas técnicas, se destacam *Voting*, *Staching*, *OneVsOne* e *OneVsRest*.

2.5.1 Algoritmo de Votação

Um Voting Classifier (VC) ou classificador por votação é um algoritmo de AM utiliza dois ou mais modelos na etapa de treinamento e descobre um resultado com base na comparação das descobertas fornecidas por cada algoritmo de forma isolada. Os critérios de votação podem ser de dois tipos:

- *Hard Voting*: o resultado é calculado com base na classe resultado mais frequente. Nesse tipo de votação, a classe prevista como resultado é a classe com maior número de votos, ou seja, a classe que mais apareceu como resultado por cada um dos classificadores. Por exemplo, suponha que três classificadores geraram os três resultados (C1, C1, C2), logo, a maioria previu C1. Portanto, C1 será o resultado da previsão por votação.
- *Soft Voting*: o resultado é calculado com base na maior probabilidade da classe resultado prevista. Nesse tipo de votação, o resultado final é obtido pela maior média entre as probabilidades obtidas pelos resultados de cada algoritmo individualmente. Por exemplo, suponha que a probabilidade obtida por três algoritmos indicando classe C1 seja (0,30, 0,47, 0,53) e indicando classe C2 seja (0,20, 0,32, 0,40). Logo, a média da classe C1 será 0,4333 e a da classe C2 será 0,3067. Portanto, o resultado será classe C1, pois obteve a maior probabilidade média de cada classificador.

Essas duas formas de se utilizar o algoritmo de votação são válidas e pode produzir resultados diferentes de acordo com cada base de dados. Portanto, somente é possível determinar qual irá obter um resultado melhor por meio de testes.

O *Scikit Learn* oferecem a implementação do *Voting Classifier* em algumas linhas de código na linguagem *Python*. Na tabela 5.1, é possível observar um pequeno exemplo

comparando os resultados obtidos por algoritmos de forma individual e utilizando o Algoritmo de Votação. Cabe ressaltar, que nem sempre o desempenho do Algoritmo de Votação será superior ao de um algoritmo aplicado de forma isolada [110].

2.5.2 *Stacking*

Nessa seção, será apresentada uma breve descrição da técnica com as principais referências para que o leitor tenha o conhecimento teórico e condições de reproduzir o trabalho realizado, consultando as referências que falam sobre a parte teórica e sobre a parte prática. Nesse trabalho, utilizou-se a linguagem de programação *Python* e a implementação *Staking* do pacote do *Scikit Learn* [62, 68, 148, 147].

A técnica denominada *Staking* em inglês, cuja tradução para o português seria empilhamento, é composta pela combinação de diversos algoritmos organizados de acordo com o que será descrito a seguir e ilustrado pela Figura 2.7. Esse algoritmo, que utiliza em sua composição outros algoritmos, é classificado como sendo do tipo ensemble em inglês, palavra cuja tradução para a língua portuguesa seria conjunto. Nesse contexto, esse algoritmo utiliza os resultados obtidos por dois ou mais algoritmos de AM [71] como entrada para um outro algoritmo, responsável pela tarefa de encontrar uma combinação de resultados que forneça a melhor performance. Como mostrado por [112, 198], essa técnica de combinar diversos algoritmos geralmente gera aumento de performance nas previsões, quando comparada à aplicação de modelos de forma individual.

Nessa tarefa de tentar melhorar a performance de um modelo a partir de vários modelos de AM, atribuir pesos maiores para os algoritmos com melhor desempenho individual pode não resultar em um desempenho global melhor. É nesse contexto que a técnica de *Staking* se destaca, pois o algoritmo, denominado meta-algoritmo, utilizado para analisar os resultados dos outros tenta obter a melhor combinação de pesos, visando aumentar o desempenho global como exemplificado na Figura 2.7.

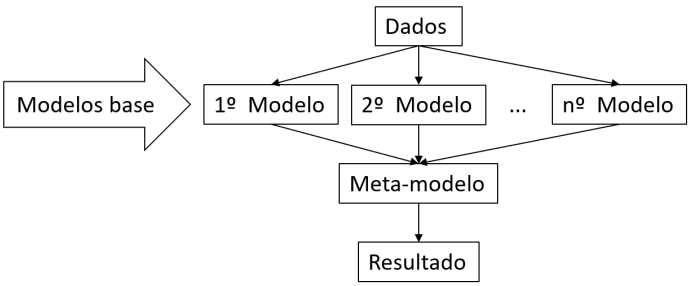

Figura 2.7: Diagrama de fluxo da técnica *Staking*

Geralmente, o meta-modelo é um modelo simples, por exemplo, Regressão Linear (RL) para tarefas de regressão e Regressão Logística (RL) para tarefas de classificação, como no trabalho de [112], pois não há na literatura ainda uma definição de qual seria o melhor modelo para ser usado como meta-modelo. No entanto, há trabalhos que mostram a eficácia do uso de outros algoritmos como meta-modelo, como no trabalho de [3], que utiliza o *Random Forest*. De acordo com [147], o *Scikit Learn* utiliza por padrão a Regressão Logística (RL) como meta-modelo em sua implementação, bem como exemplos podem ser encontrados em [147].

2.5.3 Classificação Multi-classe

A tarefa de classificação engloba praticamente dois tipos de classificação: a Classificação Binária que envolve apenas duas classes e a Classificação Multi-classe que envolve mais de duas classes, como ilustrado na Figura 2.8. Geralmente, os algoritmos obtêm melhores resultados em Classificações Binárias, dessa forma, foram desenvolvidas duas principais técnicas para melhorar a performance em Classificações Multi-classe, denominadas: Um contra um (*One vs. One*) e Um contra todos (*One vs. Rest*).

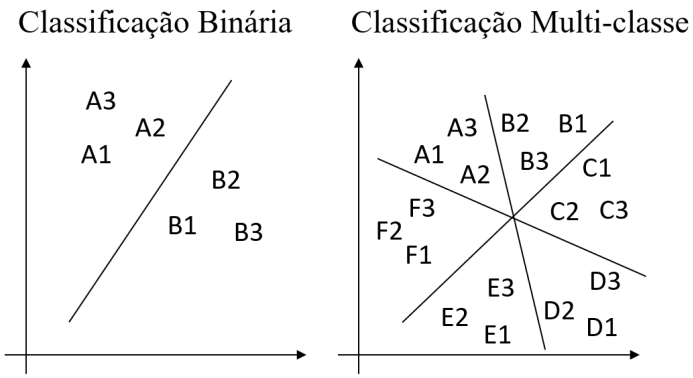

Figura 2.8: Classificação Binária versus Classificação Multi-classe

2.5.3.1 Um contra Um

Em uma Classificação Multi-classe, uma das abordagens utilizadas é a denominada Um Contra Um ou *One-vs-One* em inglês. Essa técnica consiste em fazer uma classificação binária para cada par de classe que se possa escolher dentre todas as classes, como ilustrado na Figura 2.9. Dessa forma, para n classes, serão realizadas n*(n-1)/2 Classificações Binárias, para que a partir da combinação dos resultados se possa obter um resultado

global. Nesse trabalho, foi utilizada a implementação do *Scikit Learn* [145], onde podem ser encontrados exemplos de implementação.

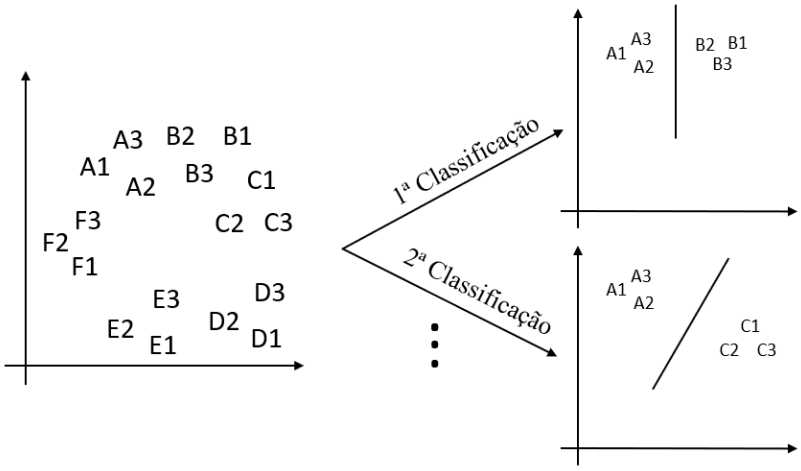

Figura 2.9: Um contra um - One vs one

2.5.3.2 Um contra Todos

Outra abordagem utilizada na Classificação Multi-classe é a técnica Um Contra Todos ou *One-vs-Rest* em inglês. Essa técnica consiste em realizar n classificações binárias para uma base de dados com n classes, sempre considerando em cada uma das n classificações binárias a primeira classe sendo uma escolhida dentre as n e a segunda classe composta por todos os elementos de todas as outras classes, como ilustrado na Figura 2.10. Nesse trabalho, foi utilizada a implementação do *Scikit Learn* [146], onde podem ser encontrados exemplos de implementação.

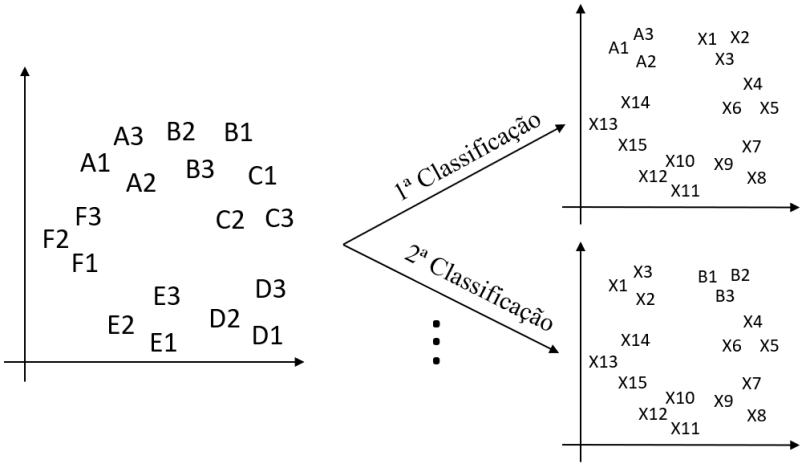

Figura 2.10: Um contra todos - One vs Rest

2.6 Métricas

Nas áreas de AM, Mineração de Dados (MD) e Inteligência Artificial (IA), no momento em que técnicas ou modelos são criados ou aplicados, surge a necessidade de um método para avaliação [25], com a finalidade de se descobrir a assertividade que a técnica fornece. Dessa forma, a avaliação da capacidade preditiva de um determinado modelo ganha importância [135], pois ela poderá definir se o modelo possui resultados que possibilitam seu uso ou não. Há também casos em que diversos modelos são utilizados ou combinados e o método de avaliação é importante para que se consiga comparar as diversas técnicas e escolher o conjunto de técnicas mais adequado para uso.

Para a maior parte dos pesquisadores que utilizam modelos de AM, esses modelos parecem caixas pretas complexas [67], pois sua lógica e funcionamento interno, na maior parte das vezes, estão ocultos para o usuário e até mesmo alguns especialistas não conseguem entender completamente a lógica por trás de suas previsões. Dessa forma, a avaliação das decisões previstas pelos algoritmos, aumenta a demanda pela capacidade de questionar, entender e garantir a confiabilidade desses modelos [25].

Esse problema de o pesquisador conseguir entender e interpretar os algoritmos de modelos utilizados em AM e Mineração de Dados (MD) já foi reconhecido pela comunidade de pesquisa, que se concentrou no desenvolvimento modelos interpretáveis e métodos de explicação ao longo dos últimos anos. No entanto, o surgimento desses métodos mostra que não há consenso sobre como avaliar a qualidade da explicação sobre esses métodos [25, 200]. Dessa forma, nessa seção, será apresentada uma breve descrição das métricas utilizadas nesse trabalho para avaliar a qualidade dos modelos e técnicas testadas.

Aqui estão seis termos importantes que são usados para calcular várias métricas:

- Tuplas Positivas (P): refere-se às instâncias da classe de interesse.
- Tuplas Negativas (N): refere-se às instâncias que não pertencem à classe de interesse.
- Verdadeiro Positivo (TP): refere-se a instâncias positivas que foram classificadas corretamente como positivas.
- Verdadeiro Negativo (TN): refere-se a instâncias negativas que foram classificadas corretamente como negativas.
- Falso Positivo (PF): refere-se às instâncias negativas que foram classificadas como positivas.

- Falso Negativo (FN): refere-se a instâncias positivas que foram classificadas como negativas.

Geralmente esses termos aparecem em uma matriz denominada matriz de confusão e é apresentada de uma forma que proporciona fácil compreensão para avaliar a predição de um modelo ou técnica de classificação. A diagonal principal da matriz (TP e TN) mostra onde o classificador está acertando. A diagonal secundária (FN e FP) mostra onde o classificador está errando. Para que um modelo seja considerado como um de bom desempenho, os valores das diagonais secundárias devem se aproximar de zero o máximo possível. A Tabela 2.1 mostra a matriz de confusão descrita acima.

Tabela 2.1: Matriz de Confusão

		Previsto		
		Classe 1	Classe 2	Total
Atual	Classe 1	TP	FN	P
	Classe 2	FP	TN	N

Após cada termo ser bem definido, as seguintes métricas podem ser calculadas:

- Acurácia: representa a porcentagem de instâncias de teste que foram classificadas corretamente [113, 54], como mostrado na equação 2.9.

$$Acuracia = \frac{TP + TN}{P + N} \tag{2.9}$$

- Precisão: corresponde ao número de instâncias classificadas corretamente como positivas sobre o número de instâncias classificadas como positivas mais o número de instâncias classificadas como falso positivas [130, 82], como mostrado na equação 2.10.

$$Precisao = \frac{TP}{TP + FP} \tag{2.10}$$

- *Recall*: corresponde ao número de instâncias classificadas corretamente como positivas sobre o número de instâncias classificadas como positivas mais o número de instâncias classificadas como falso negativas [82, 38], como mostrado na equação 2.11.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{2.11}$$

- *F-Measure* (FM): Um classificador pode ter uma alta precisão, indicando que todas as instâncias se classificam como pertencente à classe positiva. No entanto, a precisão não indica quantas instâncias foram classificadas incorretamente pelo classificador. Adicionalmente, um classificador pode ter um alto *Recall*, indicando que todas as instâncias da classe positiva foram classificadas como pertencentes à classe positiva. Dessa forma, o *Recall* não indica quantas instâncias da classe negativa foram incorretamente classificados como pertencentes à classe positiva. Essas duas métricas se relacionam de forma inversamente proporcional. Para avaliar um classificador, uma forma alternativa é o uso do *F-Measure* [153, 66], métrica que combina precisão e *Recall* usando uma média harmônica, como mostrado na equação 2.12.

$$F - Measure = \frac{2 \times precisao \times recall}{precisao + recall} \tag{2.12}$$

As métricas apresentadas até agora são utilizadas para análises de classificação. Modelos que utilizam regressão possuem métricas mais indicadas desenvolvidas na área de estatística.

- O coeficiente de correlação linear [164] é uma dessas métricas, muitas vezes indicado pelas letras R e ρ. Esse coeficiente calcula a correlação entre dois atributos numéricos A e B, como mostrado na equação 2.13.

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i - \overline{A})(b_i - \overline{B})}{n\sigma_A\sigma_B} \tag{2.13}$$

Na fórmula, n é o número de instâncias, a_i e b_i são os respectivos valores dos atributos A e B na instância i, $\overline{A}$ e $\overline{B}$ correspondem aos valores médios de A e B, σ_A e σ_B são os respectivos desvios padrão de A e B.

Geralmente, em problemas envolvendo regressão com valores numéricos, os valores previstos nem sempre correspondem aos valores reais dos dados originais. Saber a diferença entre valores previstos e reais é útil para refinar as previsões futuras, tornando-as mais precisas [40].

- O *Mean Ebsolute Error* (MAE) pode ser calculado usando a equação 2.14. Informações mais detalhadas podem ser encontradas em [92, 12, 187]. Na fórmula, X_{pi} corresponde ao valor previsto para o atributo X da instância i, X_{ai} corresponde ao valor real do atributo X da instância i e n corresponde ao número de instâncias.

$$MAE = \frac{\sum_{i=1}^{n}(|X_{pi} - X_{ai}|)}{n} \tag{2.14}$$

- O *Mean Square Error* (MSE) pode ser calculado usando a equação 2.15. Informações mais detalhadas podem ser encontradas em [30, 133]. Na fórmula, X_{ai} corresponde ao valor real do atributo X da instância i, X_{pi} corresponde ao valor previsto para o atributo X da instância i e n corresponde ao número de instâncias.

$$MSE = \frac{\sum_{i=1}^{n}(X_{ai} - X_{pi})^2}{n} \tag{2.15}$$

- O *Root Mean Square Erro* (RMSE) pode ser calculado usando a equação 2.16. Informações mais detalhadas podem ser encontradas em [30, 187]. Na fórmula, o RMSE é obtido pela raiz quadrada do MSE calculado com a equação 2.15.

$$RMSE = \sqrt{MSE} \tag{2.16}$$

Alguns algoritmos de classificação usam medições em suas operações. Tais medidas podem servir como critérios de seleção, corte ou parada, por exemplo. Algumas das medidas utilizadas neste trabalho e que são empregados na implementação do algoritmo de discretização são: entropia, ganho de informação, taxa de ganho e índice de *Gini*.

- O ganho de informação é uma das métricas utilizadas na seleção de atributos. Com esta medida, é possível selecionar o melhor atributo A que minimize as informações necessárias para classificar as instâncias de uma partição D resultante, levando à redução de dados indesejáveis em uma partição. Em outras palavras, reduzir o número de instâncias de diferentes classes dentro de uma partição para obter um particionamento que contenha apenas instâncias de uma única classe.

- A entropia mede o grau de "impureza" da partição resultante da seleção de um atributo. Seja C o atributo de classe contendo m valores distintos, C_i seja um dos

m valores da classe e C_i, D seja o conjunto de instâncias da classe C_i em D. Seja também $|C_i, D|$ e $|D|$ ser, respectivamente, o número de instâncias em C_i, D e em D. As informações esperadas necessárias para classificar uma instância em uma partição $D(Info(D))$, também conhecida como entropia de D, é calculado pela equação 2.17.

$$info(D) = -\sum_{i=1}^{m} p_i \log_2 (p_i) \tag{2.17}$$

Na fórmula de cálculo da entropia, p_i é a probabilidade diferente de zero de que uma instância na partição D pertença à classe C_i e é estimada por $|C_i; D||D|$. $Info(D)$ então representa a quantidade média de informação necessária para identificar a classe de uma instância na partição D.

Espera-se também que um atributo categórico A tenha v valores distintos, $a_1, a_2, ..., a_v$. Quando se usa A para dividir D em v partições $D_1, D_2, ..., D_v$, onde D_j contém as instâncias de D que possuem o valor a_j de A, é desejável obter partições puras, gerando classificação de suas instâncias. No entanto, é provável que as partições não serão puras.

Para calcular quanta informação ainda é necessária para obter uma classificação exata, aplica-se a equação 2.18. Nesta equação, $|D_j|/|D|$ corresponde ao peso do j^{th} partição e $Info_A(D)$ é a informação esperada necessária para classificar uma instância de D, com base no particionamento por A.

$$info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j) \tag{2.18}$$

O ganho de informação é definido pela diferença entre a informação original necessária ($Info(D)$) e a nova quantidade de informação necessária obtida após o particionamento usado pelo atributo selecionado A ($Info_A(D)$). Ou seja, o ganho de informação indica como muita informação seria obtida pelo particionamento através do atributo A. O atributo com o maior ganho de informação é escolhido como o atributo de divisão. Seu cálculo é realizado usando a equação 2.19.

$$Gain(A) = Info(D) - Info_a(D) \tag{2.19}$$

A medida de ganho de informação é tendenciosa. Prefere selecionar atributos com grande número de valores distintos. Por exemplo, em um conjunto de dados com um ID

de atributo que armazena um identificador único, uma divisão no ID resultaria em muitas partições (até o número de valores em ID). As informações necessárias para classificar instâncias com base neste particionamento é zero, porque cada partição é pura. Portanto, as informações obtidas pela divisão no atributo ID é máxima, embora esse particionamento seja inútil para classificação.

Assim, a medida de razão de ganho é uma extensão do ganho de informação visando superar seu viés. A taxa de ganho aplica uma normalização ao ganho de informações usando uma divisão do valor da informação, definido pela equação 2.20.

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2 (\frac{|D_j|}{|D|}) \tag{2.20}$$

O valor $SplitInfo_A(D)$ representa a informação potencial gerada pela divisão D em v partições, correspondendo aos v valores do atributo A. A taxa de ganho é então calculada conforme mostrado na equação 2.21. O atributo com a taxa de ganho máxima é selecionado como o atributo de divisão.

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)} \tag{2.21}$$

O índice de *Gini* também usado na discretização e mede a impureza de D conforme mostrado na equação 2.22.

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2 \tag{2.22}$$

Na equação 2.22, pi é a probabilidade diferente de zero de que uma instância em D pertença à classe C_i e é calculado por $|C_i, D|/|D|$. A soma é calculada sobre os m valores de classe de C.

O índice *Gini* considera uma divisão binária para cada atributo. Seja A um atributo com v valores distintos, $a_1, a_2, ..., a_v$. Para determinar a melhor divisão binária em A, todos os possíveis subconjuntos formados com os valores conhecidos de A são examinados. Cada subconjunto S_A é um teste binário para o atributo A da forma $A \in S_A$. Dada uma instância, este teste é satisfeito se o valor de A para a instância está entre os valores listados em S_A. Se A tem v valores possíveis, então existem $2^v - 2$ subconjuntos possíveis, porque o subconjunto vazio e o subconjunto contendo todos os valores são excluídos, pois não representam uma divisão.

Em uma divisão binária, a soma ponderada de cada impureza de partição resultante é calculada. Se uma divisão binária em A particiona D em D_1 e D_2, o índice de *Gini* de D é calculado pela equação 2.23.

$$Gini_A(D) = \frac{|D_1|}{|D|}Gini(D_1) + \frac{|D_2|}{|D|}Gini(D_2) \tag{2.23}$$

Para cada atributo, cada uma das divisões binárias possíveis é considerada. O subconjunto que dá o índice *Gini* mínimo para um atributo é selecionado como seu subconjunto de divisão. A redução na impureza incorrida por uma divisão binária no atributo A é calculada pela equação 2.24.

$$\Delta Gini(A) = Gini(D) - Gini_A(D) \tag{2.24}$$

O atributo que maximiza a redução da impureza ou que possui o menor índice de *Gini* é selecionado como o atributo de divisão.

Capítulo 3

Trabalhos relacionados

Na medida em que as redes de grande porte se tornam cada vez maiores e mais complexas, a resolução de incidentes de rede se torna uma tarefa importante e, consequentemente, consome boa parte do tempo dos profissionais técnicos de uma empresa. A fim de contribuir para a melhoria dessa situação, diversos trabalhos tentam extrair conhecimento de bases de dados de sistemas de chamados. Para identificar as pesquisas desenvolvidas relacionadas a esse trabalho, foi realizada uma revisão da literatura existente. Nessa seção, serão destacados os principais trabalhos que abordaram PLN e classificação de chamados em sistemas de atendimento.

3.1 Fluxos e roteamento de chamados pelas equipes técnicas

Nesse contexto, [168, 127, 109, 196, 188] propõem métodos responsáveis por extrair automaticamente fluxos de trabalho, focando em identificar e sugerir qual caminho o chamado deve seguir pelas equipes técnicas até sua resolução.

Por meio da mineração de dados históricos de chamados, [168] criam um modelo de rede social que representa a relação funcional entre as diversas equipes técnicas de uma empresa, com a finalidade de sugerir qual caminho um problema deveria seguir passando por essas equipes. Esses pesquisadores testaram seu modelo, chamado de *EasyTicket*, com uma base de dados obtida da IBM com 1.4 milhões de problemas reais catalogados durante o ano de 2006, divididos em 553 categorias de problemas. Os resultados mostraram que, para um conjunto de cinco tipos de problemas escolhidos para teste, o modelo pode melhorar a eficiência do processo de alocação de problemas entre equipes técnicas em 35%. Esse trabalho propõe um modelo baseado em Cadeias de *Markov* para capturar o

relacionamento entre as diversas equipes técnicas e foi desenvolvido utilizando a linguagem de programação Java com um banco de dados DB2.

Outro trabalho relacionado a sistemas de chamados, porém utilizando técnicas diferentes, foi proposto por [127] e foi denominado *TroubleMiner*. Esses pesquisadores propõem um mecanismo com base em clusterizacão de problemas, produzindo uma hierarquia de conjuntos de problemas com rótulos novos específicos, visando melhorar e facilitar a alocação de chamados feita por um especialista em rede. Os pesquisadores trabalham somente como as palavras contidas nas duas primeiras frases da descrição do problema, utilizam *Porter's Stemmer* para reduzir as palavras aos seus radicais, removem as palavras que aparecem com menos frequência e removem as palavras que não agregam significado ao texto por meio de verificação manual das palavras que devem ser removidas. Os pesquisadores também testaram *Term Frequency* (TF), *Document Frequency* (DF), *Inverse Document Frequency* (IDF) e TF-IDF como métodos de atribuição de pesos, e escolheram DF, método que apresentou melhor resultado. Os pesquisadores testam a eficácia do modelo proposto aplicando-o a duas bases de dados, formadas por dados armazenados entre os anos de 2005 e 2008 e obtidas de diferentes centros de operações de rede: *Abilene* e *Switchlan*. Esses centros estão localizados nos Estados Unidos e Suíça e gerenciam redes responsáveis por conectar instituições de pesquisa e universidades.

Os pesquisadores do trabalho [109] propõem dois *frameworks* para clusterizacão de alertas e incidentes de rede, tendo como base dados textuais contidos em chamados. Um primeiro *framework* foi desenvolvido para os dados com textos semiestruturados e um segundo *framework* foi desenvolvido para os dados com textos não estruturados. Dessa forma, com o objetivo de eliminar a classificação manual, o *framework* proposto recebe textos semiestruturados e não estruturados gerados por componentes de infraestrutura, tais como *storages*, servidores e ativos de rede, e os classifica em conjuntos com características semelhantes usando diferentes abordagens de teoria dos grafos. Na etapa de pré-processamento, os pesquisadores utilizaram normalização de radicais, remoção de *Stop Words* e *Bag of Words*. Na etapa de clusterização, os pesquisadores utilizam uma matriz de distância aos pares como métrica para a formação dos clusteres. Em uma última etapa, os pesquisadores propõem um tipo de visualização gráfica que é mais indicada para o problema em questão.

O trabalho de [196] propõe um método de classificação multi-rótulo de chamados com base em uma hierarquia definida para os problemas com a finalidade de facilitar o diagnóstico de problemas e de automatizar todo o processo. A base de dados dos

testes feitos nessa proposta é composta por chamados gerados de forma automatizada por um sistema de monitoramento de um provedor de serviços. Essa base de dados é composta por 23.000 instâncias, das quais 20.000 foram usadas para treinamento e 3000 foram usadas para teste. Os rótulos de cada instância foram obtidos por meio das informações catalogadas durante o processo de resolução do chamado. Há nessa base 98 rótulos diferentes, porém, cada problema está associado a 3 rótulos em média. Os atributos de cada chamado são provenientes de uma curta mensagem de texto que descreve o problema. Inicialmente, os pesquisadores aplicam técnicas de PLN para preprocessar os textos. Em uma segunda etapa, os pesquisadores utilizam *TF-IDF* com as 900 palavras mais frequentes para compor os atributos de cada instância de problema. Após essas etapas, o classificador binário *SVM* foi aplicado de forma paralela para cada um dos 98 rótulos. Esse trabalho comparou os resultados obtidos usando o algoritmo proposto *GLabel* e os resultados obtidos usando CSSA e HIROM, dois algoritmos frequentemente usados nesse tipo de problema. O algoritmo proposto obteve melhor performance.

O método proposto por [188] mostra como extrair o fluxo de trabalho de um chamado de forma automatizada a partir de textos livres, sem formatação ou padrão previamente definido. O fluxo de trabalho é extraído em forma de grafo com as ações que um técnico deveria realizar para tentar solucionar o incidente analisado. Segundo os autores, a extração de fluxo de trabalho em forma de grafo facilita a identificação das ações necessárias para solucionar um problema, pois um operador humano pode visualizar os possíveis passos de forma intuitiva. Os pesquisadores também desenvolveram um algoritmo para lidar com determinadas ações que aparecem com descrições diferentes. O algoritmo *Naive Bayes* foi utilizado para estimar os rótulos atribuídos a cada sentença. Os problemas da base de dados usada no experimento foram escritos em japonês e, como as palavras em japonês não são separadas por espaço, considerou-se que cada 2 caracteres representavam uma unidade ou palavra. Embora esse projeto de pesquisa tenha utilizado uma base de dados pequena, com 41 chamados, conseguiu atingir acurácias de 87% em testes de previsão de rótulos.

Seguindo uma abordagem um pouco diferente, porém visando objetivos semelhantes, [152] apresentam *NetSieve*. Essa ferramenta tem o objetivo de inferir as causas dos problemas, os procedimentos realizados para identificação, bem como, as ações necessárias para resolução, buscando realizar essas atividades de forma automatizada em sistemas de chamados de problemas de rede. Com uma base de dados com 10.000 chamados obtida de um provedor, para atingir os objetivos descritos, a ferramenta combina o PLN com técnicas de representação do conhecimento, modelagem de ontologias e aprendizado in-

cremental. Uma das bases de teste foi composta por 155 chamados sobre substituição e reparo de 2 tipos de equipamentos, com os dados desses chamados validados pelos fabricantes. Uma segunda base de teste foi composta por 696 chamados rotulados por um especialista. Os testes foram implementados usando *Cython* [1], ferramenta que traduz o código em *Python* para C/C++, diminuindo o tempo de execução de 5 a 20 vezes. Nos testes realizados, a ferramenta atinge em torno de 96% de acurácia para previsão de problemas e 94% para previsão de ações.

Segundo [192], sistemas de chamados contém uma enorme quantidade de informação, com dados sobre sintomas de problemas e ações adotadas para resolução. Nesse contexto, o encaminhamento de chamados para o grupo especializado correto ainda é um grande desafio para um sistema de gerenciamento de chamados. Como solução para essa questão, os pesquisadores propõem três modelos para roteamento de chamados, com base na mineração de combinações de descrições de problemas e possíveis sequências de roteamento, elevando a eficiência de todo o processo e reduzindo a intervenção humana. O experimento foi realizado com uma base de dados de monitoramento de mais de 200 servidores e ativos de rede, composta por 40000 instâncias de chamados, armazenados em um período de 8 meses, com 95 tipos diferentes de incidentes, com distribuição desbalanceada. Para esse período do experimento, 150 grupos de especialistas estavam envolvidos no tratamento dos incidentes. 10% dos dados foram utilizados para teste, enquanto o restante para treinamento. Os métodos propostos apresentam acurácias entre 95% e 97%.

3.2 Classificação de chamados

Com o objetivo de recomendar a melhor forma de resolução de chamado, [177] utilizam técnicas de aprendizado supervisionado para prever qual a resolução mais adequada para um determinado chamado. Foram feitos testes utilizando três bases de dados obtidas por meio de três contas de usuários administradores responsáveis por resolução de chamados. A primeira base de dados foi composta por 50377 problemas, a segunda por 6121 e a terceira por 4066, para períodos de 55, 29 e 48 dias respectivamente. Esse trabalho propõe a adição de uma função probabilística à função de similaridade do algoritmo *KNN* e compara 4 tipos de testes diferentes, todos com base no *KNN*. 90% dos dados foram utilizados para treinamento dos modelos e o restante para teste. O algoritmo proposto pelos pesquisadores mostrou um pequeno aumento de acurácia, porém, a diferença entre a maior acurácia e a menor atingida por todos os 4 testes não ultrapassou 10%.

[80] combinam técnicas de modelagem com tópicos, clusterizacão e técnicas de recuperação da informação para automatizar a análise de chamados de uma central de suporte de tecnologia da informação com foco nos problemas dos usuários. Essa abordagem foca na construção de um conceito que represente uma frase e, com base nesse conceito, a clusterização de documentos é realizada. Os testes foram realizados com dois conjuntos de problemas cada um com 20000 instâncias e relacionados a caixas de *e-mail* e a portais de aplicação. O método proposto apresentou resultados melhores que os resultados obtidos com *Latent Dirichlet Allocation* (LDA)[3] e *Lingo* [2], quando analisados de acordo com *Dunn Index* (Dunn) [4] e com *Davies-Bouldin Index* (DBI) [5].

3.3 Classificação de problemas de rede

[178] também propõem modelo preditivo que tenta melhorar o processo de alocação de chamados e reduzir o tempo necessário para resolução. Esse trabalho foi realizado com uma base de dados com chamados composta por três tipos de atributos relacionados ao equipamento, ou a elemento afetado, relacionados à descrição do problema e atributos contendo informações dinâmicas que refletem cada momento da falha. Algumas técnicas de mineração de textos, tais como redução ao radical da palavra, modelos entidade relacional e recontagem de frequência, foram empregadas, além disso, foi aplicada uma técnica de redução de atributos denominada *wrappers* [21]. Nesse contexto, esse trabalho aplicou diversos algoritmos de AM tais como: *Decision Table*, *Tree C4.5*, *Bayes AODE*, *Bayes Net*, *Naive Bayes*, *C4.5*, *Hyper Pipes* e *Decision Stump*. Os pesquisadores obtiveram 93% de acurácia para previsão dos problemas que precisariam de manutenção presencial feita por um técnico, 97% de acurácia para previsão dos níveis técnicos para os quais um problema pode ser escalado, 92% de acurácia para previsão do novo grau de severidade que um chamado recebe após ser escalado entre equipes técnicas e 92% de acurácia para previsão do tempo passado até que um chamado seja escalado para uma equipe.

3.4 Qualidade de Experiência

Com a expansão do acesso à internet, o conceito de Qualidade de Experiência (QoE) surgiu como um indicador essencial para avaliar a qualidade percebida pelos usuários das redes sem fio e das redes de celulares. Dessa forma, esse indicador é composto por métricas responsáveis por quantificar e qualificar as expectativas dos usuários e fatores que influenciam aplicações e as redes de forma geral. O modelo exato para medição da

qualidade da experiência de um usuário ainda não foi definido, no entanto, há diversos modelos propostos [7], [8], [9], [10], [11], [12], [13], [14] inclusive, alguns tentam estabelecer uma relação entre QoE e Qualidade de Serviço (QoS).

Nesse contexto, [195] propõe uma abordagem para classificar a Qualidade da Experiência (QoE) de usuário em relação a um serviço de *streaming* de vídeo, utilizando uma variação do algoritmo *SVM*. Com aplicação Multi-classe e incremental (Multiclass-iSVM), esse algoritmo é utilizado em conjunto com a técnica conhecida como um contra um (*One-against-One*) [16]. Os pesquisadores também avaliam a efetividade da técnica da área de AM denominada Aprendizado Incremental, também avaliada em trabalhos como [15], quando aplicada a bases de dados dinâmicas e de grande escala. Como métrica, esse trabalho utiliza a acurácia e a complexidade computacional.

A base de dados dessa pesquisa foi construída com a participação de 45 pessoas, com idades entre 20 e 38 anos. Esses participantes avaliaram 24 vídeos em diferentes condições de reprodução. Nos testes realizados, o modelo proposto consegue atingir acurácia de 89% utilizando abordagem incremental e começando com 10% de todos os dados no início do processo. Os resultados do modelo proposto foram comparados com resultados obtidos por *Support Vector Machine* (SVM) [17], *Decision Trees* (DT) [18], *Random Forests* (RF) [19], *Recurrent Neural Networks* (RNN) [20], apresentando acurácias melhores.

Capítulo 4

Metodologia

Esse capitulo se inicia com a descrição da base de dados utilizada nos testes feitos nesse trabalho. Em seguida, há uma descrição de todas as etapas de pré-processamento de dados, iniciando pelas técnicas mais tradicionais como remoção de URLs e *e-mails*, remoção de pontuação e caracteres especiais, normalização, remoção e correção de palavras de forma empírica ou manual, *Tokenization*, remoção de *stop words*, *Stemming*, *Lemmatization*, com destaque para remoção de nomes próprios e um método customizado para correção ortográfica. Por fim, uma descrição detalhada da metodologia é apresentada.

4.1 A base de dados

Como mostrado nas Seções 1.1 e 1.2, muitas instituições, empresas e órgãos públicos utilizam sistemas de atendimento ou monitoramento ativo ou passivo para gerenciar problemas, reclamações ou solicitações de clientes e usuários, visando manter alta disponibilidade, com elevado nível de resiliência. A rápida resolução de chamados por uma empresa também tem influência direta sobre sua reputação, uma vez que, atualmente, as informações são disseminadas de forma fácil por meio da internet.

Nos sistemas de monitoramento ativo, os chamados são criados de forma automatizada e em grande quantidade, com informações padronizadas, seguindo sempre uma mesma formatação, a partir de dados de monitoramento, não sendo necessário que alguém registre uma reclamação. Essa padronização das informações dos chamados gerados facilita o processamento de forma automatizada por um computador, mesmo que esse tipo de sistema gere um número maior de chamados.

Nos sistemas de monitoramento passivo, os chamados são criados por atendentes

ou por integrantes de equipes técnicas, responsáveis por inserir informações que possam descrever as características do problema observado, como é o caso da base de dados utilizada nesse trabalho. Assim que um chamado é registrado, se inicia um ciclo de atendimento, no qual, o problema reportado é analisado pelas diversas equipes técnicas da empresa até a sua resolução.

Como cada chamado está relacionado a uma área específica de conhecimento, para que um chamado seja resolvido, são necessárias equipes técnicas com diversas especializações [192], ou seja, com capacidade de resolver tipos específicos de problemas. Tradicionalmente, a alocação de um determinado problema para uma determinada equipe é feita de forma manual por meio de profissionais com conhecimentos multidisciplinares, capazes de identificar a causa principal de um problema e direcioná-lo para a equipe com capacidade técnica de resolvê-lo.

Embora o método manual de alocação de problema seja mais confiável e preciso, devido à inspeção direta por um profissional capacitado, os métodos de alocação de problemas de forma automatizada se apresentam como uma alternativa para otimização do tempo de resolução de um chamado, bem como, para redução de custos com determinados profissionais que podem ser empregados em outras funções. Embora haja muitas pesquisas relacionadas a esse tema, muitos trabalhos apresentam resultados pouco satisfatórios, ainda não suficientes para implementação em um ambiente real, sugerindo continuação das pesquisas visando o aprimoramento das técnicas existentes.

É nesse contexto que a base de dados desse trabalho foi criada. Esses dados foram fornecidos por uma grande empresa de telecomunicações que opera no Brasil com mais de 69 milhões de usuários, responsável por 26% desse segmento do mercado nacional, oferecendo serviços de telefonia móvel com mais de 62 milhões de usuários, telefonia fixa com mais de 700 mil usuários, internet móvel com mais de 50 milhões de usuários, banda larga fixa com mais de 700 mil usuários, dentre outros serviços de telecomunicações.

A base de dados utilizada é composta por relatos de um sistema de *Call Center*, incluindo relatos de problemas informados por clientes e relatos feitos por equipes de diversos níveis de conhecimento técnico. Essa base possui 33 tipos de dados com informações sobre medições realizadas por equipamentos, informações sobre localização, tipos de problemas encontrados, descrição dos problemas encontrados, etc. Nesse trabalho, foram utilizadas as colunas "PROBLEMA" e "MOTIVO3", que contém as diversas categorias de problemas, e a coluna "RELATOCLIENTE", que possui a descrição textual do problema do cliente relatado por um atendente, como mostrado na Figura 4.1

PROBLEMA	RELATOCLIENTE
Queda de ligação/muda	sem sucesso
Status de Visita Técnica (Status ...	ott sem agendamento enviado msg pra a mesma agendar
Modem	cliente informa seu modem não liga LEDs apagados realizado procedimento sem sucesso.solici...
Baixa Velocidade	cliente não tem como realizar testes no momento e nem fazer a mudança de senha no mome...
Transferência mult skill	CLIENTE SOLICITA INFORMAÇÕES SOBRE NOVO ENDEREÇO;
Transferência mult skill	NOME:[redacted]TITULAR:[redacted]CPF:[redacted]PROTOCOL...
Modem sem sincronismo	CLEINTE PEDE REPARO TECNICO POIS NÃO TEM CONEXÃO COM A INTERNET SEU MODEM N...
Modem sem sincronismo	TITULAR:[redacted]SEM LINHA:[redacted]...
Completamento ? Qualquer nume...	está sem sinal, imformado a aguardar o atedniemtno
Parâmetros Ruins	Cliente relata que esta sem internet, apos os testes, não voltou ao normal.
Modem sem sincronismo	SEM CONEXÃO,MODEM SEM SINCRONISMO,NÃO TEM MASSIVA OU BLOQUEIO ATI
Queda / Intermitência	relata quedas de conexão e não foi identificada nenhuma oscilação via gpon chrome, orientad...
Modem sincronizado e autenticado	CLIENTE INFORMA QUE O SINAL FICA CONECTADO SEM INTERNET / @ PISCANDO SEM PARAR

Figura 4.1: Base de dados

Foram criados dois conjuntos de testes. Para o primeiro conjunto, foi utilizada uma base de dados que continha o texto do relato do cliente e o campo "PROBLEMA" com um tipo de classificação do problema. Para o segundo conjunto de testes, foi utilizada uma base de dados que continha o texto do relato do cliente e o campo "MOTIVO3" com um outro tipo de classificação do problema. Essa segunda base de dados não continha o campo "PROBLEMA", por isso, para classificação do problema de rede, o campo "MOTIVO3" foi escolhido.

A coluna "PROBLEMA" contém as diversas classificações dos problemas de telecomunicações, tais como problemas de telefonia, de configuração de roteador WiFi, de internet banda larga fixa e móvel, bem como, problemas de natureza administrativa, tais como problemas de mudança de endereço, de inadimplência, de alteração cadastral, de 2º via de conta, etc. Essa primeira base possui 194382 registros e cada um desses está associado a uma das 56 classificações de problemas, como mostrado na Tabela 4.3 de forma resumida, que mostra cada tipo de problema e seu respectivo número de registros.

A coluna 'MOTIVO3' também contém as diversas classificações dos problemas de telecomunicações, semelhantes às classificações encontradas na coluna "PROBLEMA", como mostrado na Tabela 4.1. Essa segunda base possui 371260 registros e para os testes realizados nesse trabalho foram escolhidos os seguintes tipos de problemas: Sem Sincronismo, Exige Técnico e Parâmetros Ruins, como mostrado na Tabela 4.2. Para os testes realizados nessa pesquisa, os dados foram pré-processados e divididos de acordo com o objetivo de cada teste realizado.

Tabela 4.1: Motivo3 e seus números de registros

Motivo3	**Registros**
Sem Sincronismo	246798
Exige Técnico	64983
Parâmetros Ruins	59479
Queimado	4227
Acessórios	3549
Porta/Par Trocado	3365
Modem recondicionado	1620
Quedas / Intermitência	1588
CTT Não Efetua Chamada	807
CTT Não recebe chamada	580

Tabela 4.2: Motivo3 — escolhidos para testes

Motivo3	**Registros**
Sem Sincronismo	246798
Exige Técnico	64983
Parâmetros Ruins	59479

A coluna "RELATOCLIENTE"é composta por textos escritos pelos atendentes de acordo com o relato de cada cliente que entra em contato para registrar uma reclamação ou solicitação. Os textos contidos nesses registros não são padronizados, são escritos por atendentes com níveis de conhecimento variados, são escritos de forma coloquial na maior parte das ocorrências, possuem estilos diversos, são compostos por algumas palavras que não fazem parte do idioma português, como ilustrado na Figura 4.2 e, em muitos casos, contêm erros gramaticais e erros de digitação.

Esses textos são conhecidos como língua natural humana, pois é a linguagem com que os atendentes estão acostumados a se comunicarem, com estilo se aproximando da linguagem falada, com expressões coloquiais, com gírias, não seguindo todas as regras preconizadas pela gramática de uma determinada língua. Dessa forma, torna-se um desafio fazer um algoritmo computacional processar e analisar essas informações não padronizadas. Para lidar com esse tipo de dificuldade, que reduz a performance de técnicas de classificação, foram realizadas diversas etapas de pré-processamento de dados que serão descritas na Seção 4.2. Como essa pesquisa foi desenvolvida com uma base com dados reais, para cada remoção ou alteração nos textos originais, foi realizada uma verificação para saber se a modificação trouxe ou não aumento de performance.

Tabela 4.3: Problemas e seus números de registros

Problemas	**Registros**
Modem sem sincronismo	47137
Status de Visita Técnica (Status OTT)	20697
Massiva	14499
Queda de ligação/muda	9860
Modem sincronizado e autenticado	9839
Informações sobre o Produto	9436
Completamento ? Qualquer numero (Não Recebe, Não Efetua ou Tom de Ocupado)	9384
Baixa Velocidade	8595
Parâmetros Ruins	8490
Transferência mult skill	8053
Wi-Fi - Performance	7791
Queda / Intermitência	7132
Exige Técnico	7050
Transferência retenção	4002
Cliente redução - inadimplente	3268
Modem sem autenticação / Falha na autenticação	2729
Wi-Fi - Configuração	2503
Modem	2411
Segunda via de conta - Email	2133
Ligação Muda	2100
Configuração de Dispositivos	1609
Cliente suspenso - inadimplente	1595

Tabela 4.4: Quantidade de registros por estado.

Estados	**Número de registros**
SP	93738
RJ	47413
None	16207
GO	12851
BA	8248
TE (PI)	4931
AM	4750
PE	3951
MG	1335
DF	958

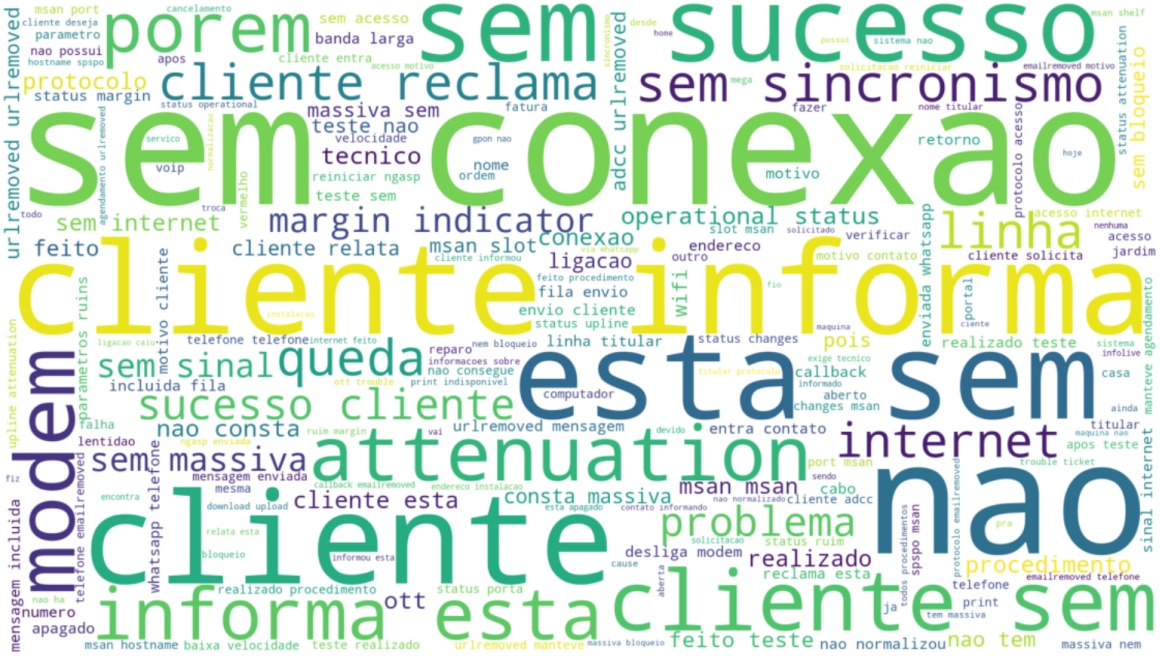

Figura 4.2: Nuvem de palavras com as 200 mais frequentes

PROBLEMA	RELATOCLIENTE
Modem sem sincronismo	Cliente [redacted]) informa que esta sem sinal de internet.Sem massiva (Nenhum registro foi encontrado) e sem bloqueio por fatura. Ledes apagados em modem. Modem sem sincronismo.Testes sem sucesso.
Transferência mult skill	cliente com suspensao total.favor verificar sobre desbloqueio.nao se trata de problema tecnico
Baixa Velocidade	cliente informa que a internet esta lenta realizado teste com sucesso.obs.: cliente não tem computador teste feito via WIFI.
Informações sobre o Produto	Cliente estava ligando do fixo do modem pedi que encerrasse para que eu retornasse no celular.
Modem sincronizado e autenticado	CLIENTE RECLAMA DA SUA INETRENET SEM NAVEGAÇÃO SEM MASSIVA SEM BLOQUEIO TESTE REALIZADO SEM SUCESSO
Queda / Intermitência	cliente entrou em contato informando que está sem sinal em sua internet, feito testes em sistema sem sucesso.
Wi-Fi - Configuração	Cliente não consegue conectar com o celular no 2g, aparece "acesso negado" Foi realizado os procedimentos para ver se foi. Foi feito o reset do modem.

Figura 4.3: Registros com poucos ruídos

PROBLEMA	RELATOCLIENTE
Modem sem sincronismo	Titular [redacted] CPF [redacted] Endereço [redacted] RIO DE JANEIRO RJ [redacted] Protocolo [redacted] Callback [redacted] Acesso [redacted] Email [redacted] Motivo cliente reclama que esta sem acesso a internet / sem massiva sem bloqueio / sistemas indicam sem sincronismo MSAN:\tRJRJO_MSAN0461\tSLOT:\t4\tPORTA:\t46Status Porta:\tNao sincronizada\tVelocidade AAA (UP / DOWN):\t30.00 / 60.00\tVelocidade Gerência (UP / DOWN):\t30.00 / 60.00\tVelocidade Sincronismo (UP / DOWN):\t0.00 / 0.00\tABR (UP / DOWN):\t7.30 / 20.88\tKL0:\t12.2Vlan(s) Configurada(s):\t1-2 - 50 - 100 - 150
Modem sem sincronismo	Cliente sem conexão. Operational Status DOWN Line Attenuation / UP - Attenuation Up Status - Line Attenuation / DOWN - Attenuation Down Status - Noise Margin Indicator / UP - Noise Margin Up Status - Noise Margin Indicator / DOWN - Noise Margin Down Status - Operational Status Changes - MSAN / Hostname SPSPO_MSAN2796 MSAN / Slot 2 MSAN / Port 0 MSAN / ONT Id 31 MSAN / Shelf 0 Ont RX Power - Ont Olt RX Power - Ont Last Up Time 2020-12-28 11:29:23 Ont Last Down Time 2020-12-28 11:31:13 Ont Last Down Cause losi
Queda de ligação/muda	Em linha: [redacted] Endereço [redacted] SAO PAULO SP [redacted] Motivo do contato: cliente sem conexão, cciu a ligação no final do atendimento, calback com sucesso, para dar informações do reparo
Parâmetros Ruins	nome [redacted] titular [redacted] cpf [redacted] telefone [redacted] email [redacted] end [redacted] SAO PAULO SP [redacted] acesso [redacted] motivo PARAMETROS RUINS Operational Status\tUPLine Attenuation / UP\t07.0Attenuation Up Status\tBOMLine Attenuation / DOWN\t05.3Attenuation Down Status\tBOMNoise Margin Indicator / UP\t08.8Noise Margin Up Status\tRUIMNoise Margin Indicator / DOWN\t20.0Noise Margin Down Status\tBOMOperational Status Changes\t0MSAN / Hostname\tSPSPO_MSAN2072MSAN / Slot\t3MSAN / Port\t35MSAN / ONT Id\t-MSAN / Shelf\t1Ont RX Power\t-Ont Olt RX Power\t-Ont Last Up Time\t-Ont Last Down Time\t-Ont Last Down Cause\t-protocolo 20201310704270

Figura 4.4: Registros com muitos ruídos

4.2 Pré-processamento dos dados

Alguns textos de relatos de clientes dessa base de dados apresentam muitos ruídos, como mostrado nas Figuras 4.5 e 4.4, porém, outros relatos apresentam poucos ruídos como

mostrado na Figura 4.3. Os ruídos consistem em palavras, sinais e caracteres especiais que não contribuem com valor semântico relevante para a resolução do problema que se pretende abordar, como detalhado teoricamente na Seção 2.2.3. Além dessas remoções, foram realizados testes removendo algumas palavras mais frequentes e menos frequentes, pois, como observado na Figura 4.6, algumas palavras aparecem com uma frequência muito maior do que outras, tornando os textos desbalanceados em termos de frequência de palavras. Essa etapa também foi realizada com verificações para identificar possíveis melhoras ou pioras nos resultados.

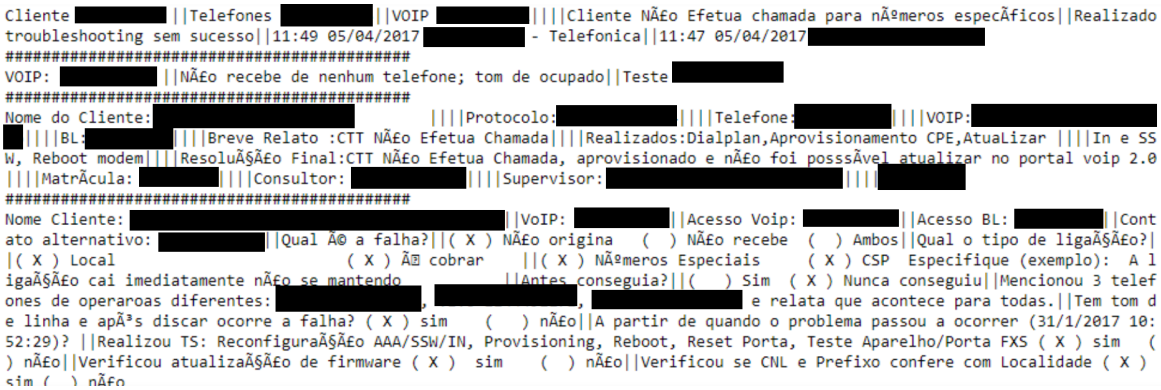

Figura 4.5: Texto bruto e sem modificações

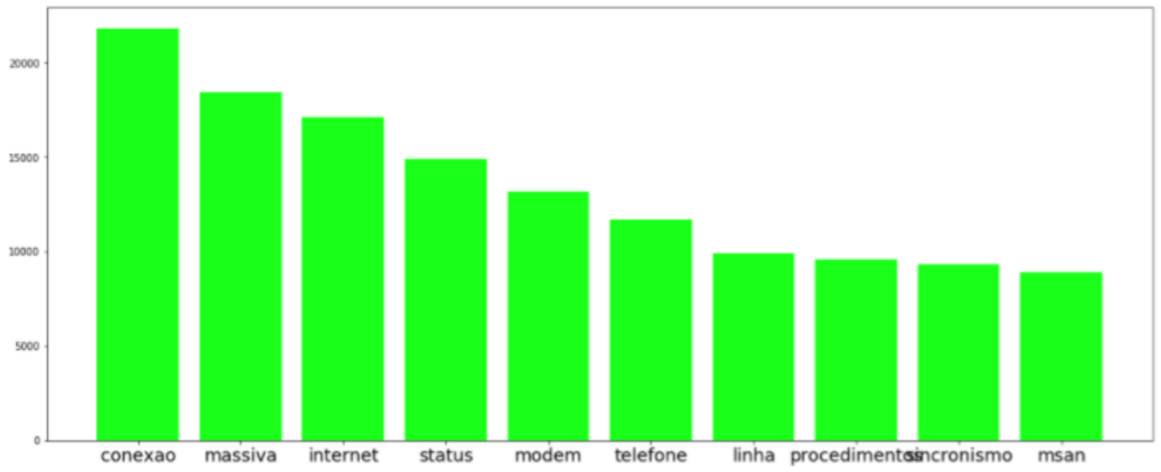

Figura 4.6: Histograma com as 10 palavras mais frequentes

4.2.1 Remoção de URLs e e-mails

Como descrito na Seção 2.2.3.2, URLs e e-mails aparecem quando o usuário decide deixar um contato de e-mail diretamente em sua solicitação, ou, quando há reclamação envolvendo acesso a alguma página da internet. Nas Figuras 4.9 e 4.10, pode-se observar uma

comparação entre o texto antes e após a remoção de URLs. Essas remoções foram realizadas por meios de expressões regulares utilizando a biblioteca *Regular Expressions* (*RE*) para linguagem *Python*. Com a finalidade de manter a informação de que no texto havia um e-mail ou uma URL, as URLs e os e-mails foram substituídos pelas palavras *urlremoved* e *emailremoved*, como se pode observar na Figura 4.10, pois esses dados podem ter impacto nos resultados dos testes com algoritmos de classificação. Na Seção 2.2.3.2, é possível encontrar o referencial teórico, bem como a indicação de exemplos de como realizar essas tarefas.

4.2.2 Remoção de pontuação e caracteres especiais

Nessa fase, primeiramente foi realizada a remoção de sinais, pontuações, acentuações, números e caracteres especiais, com o auxílio de expressões regulares e funções nativas da linguagem *Python* para encontrar e substituir caracteres, utilizando a biblioteca *Regular Expressions* (*re*) para linguagem *Python*. Nas Figuras 4.9 e 4.10, pode-se observar uma comparação entre o texto antes e após a remoção. Na Seção 2.2.3.1, é possível encontrar o referencial teórico, bem como a indicação de exemplos de como realizar essas tarefas.

4.2.3 Normalização

Como descrito na Seção 2.2.3.3, para aplicação de técnicas de PLN, opta-se por deixar todas as palavras com letras minúsculas ou maiúsculas. No caso dessa pesquisa, optou-se por deixar todas as letras minúsculas, como mostrado nas Figuras 4.9 e 4.10, mostrando exemplos de textos antes e após essa modificação. Essa abordagem evita que no processo de vetorização haja duas palavras semanticamente idênticas sendo tratadas como palavras diferentes, apenas pelo fato de uma conter letras maiúsculas e outra letras minúsculas. Na Seção 2.2.3.3, é possível encontrar o referencial teórico, bem como a indicação de exemplos de como realizar essas tarefas.

4.2.4 Remoção e correção de palavras de forma empírica ou manual

Além da remoção das palavras vazias contidas por padrão na biblioteca NLTK, cuja descrição encontra-se na Seção 4.2.6, primeiramente, optou-se por remover e realizar algumas correções de palavras de forma empírica, com a finalidade de tentar reduzir confusões e ruídos nos textos, visando melhorar o desempenho dos modelos de AM testados. Na

```
" eebde "-->"espaço";" aaa "-->"espaço";" cciu "-->"espaço";"ott"-->" ott ";" cs "-->"espaço";" ta "-->" esta ";" ezs "-->"espa
ço";"a§"-->"c";"£"-->"";"       "-->"espaço";"a"-->"i";"a□"-->"a";"a□"-->"c";"a□"-->"u";"a□"-->"a";" x "-->"espaço";"_"-->"espa
ço";" id "-->"espaço";"a³"-->"o";"µ"-->"o";"a@"-->"e";"na mero"-->"numero ";" ex "-->"espaço";"["-->"espaço";"]"-->"espaço";" f
"-->"espaço";"a¡"-->"a";"□"-->"espaço";"a□"-->"i";" fxs "-->"espaço";" ssw "-->"espaço";" oi "-->"espaço";" ok "-->"espaço";" i
n "-->"espaço";" tel "-->"espaço";"@"-->"espaço";" rj "-->"espaço";" bl "-->"espaço";" vp "-->"espaço";"cpf"-->"espaço";" br "-
->"espaço";" e "-->"espaço";" a "-->"espaço";" de "-->"espaço";" na "-->"espaço";" o "-->"espaço";" da "-->"";" cpf "-->"espaç
o";" r "-->"espaço";" os "-->"espaço";" as "-->"espaço";" se "-->"espaço";" zl "-->"espaço";" kl "-->"espaço";" zs "-->"espaç
o";" ctt "-->"espaço";"pa□a"-->"espaço";" s "-->"espaço";" ts "-->"espaço";" csp "-->"espaço";" tmkt "-->"espaço";" ltda "-->"e
spaço";" cnl "-->"espaço";" gera ncia "-->" gerencia ";"a´"-->"o";"xxxx"-->"espaço";" ont "-->"espaço";" ade "-->"espaço";" cop
"-->"espaço";" hora "-->"espaço";" data "-->"espaço";" ente "-->"espaço";"a□"-->"o";"*"-->"espaço";"a□"-->"a";" yahoo "-->"espa
ço";" mail "-->" email ";" gmail "-->"espaço";" hotmail "-->"espaço";" email "-->"espaço";"a□"-->"e";" mb "-->"espaço";" y "--
>"espaço";" sn "-->"espaço";" cp "-->"";" n "-->"espaço";" vlan "-->"espaço";" rx"-->"espaço";" tx"-->"espaço";" olt "-->"espaç
o";"power"-->"espaço";" last "-->"espaço";" time "-->"espaço";" losi "-->"espaço";" orem "-->" porem ";"nascido"-->"espaço";" s
p "-->"espaço";" nd "-->"espaço";"rio janeiro rj"-->"espaço";" crt "-->"espaço";"stuatusporta"-->" status porta ";"ruimnoise"--
>" ruim noise ";"bomline"-->" bom line ";"ruimoperational"-->" ruim operational ";"print´s"-->" print ";"ligacao"-->" ligacao
";"solicitacao"-->" solicitacao ";" cato "-->" contato ";" wi fi "-->" wifi ";" sobr "-->" sobre ";" abr "-->"espaço";" rjrjo
"-->"espaço";" encrado "-->" encerrado ";" ledes "-->" leds ";" nonecall "-->" none  call ";" brsenha "-->" senha ";" cobnexta
"-->" conecta ";" encra "-->" encerra ";" orenetada "-->" orientada ";" cnpj "-->"espaço";" doc "-->"espaço";" estr "-->" esta
";" testesaaa "-->" testes ";" testes "-->" teste ";" obs "-->"espaço";" reclamasua "-->" reclama sua ";" inetrenet "-->" inter
net ";" none "-->"espaço";" call back "-->" callback ";"call back "-->" callback ";"tecnico"-->" tecnico ";" reset "-->" renici
ar ";"cliente"-->" cliente ";"reparo"-->" reparo ";"endereco"-->" endereco ";"bucket"-->" bucket ";"motivo"-->" motivo ";"sincr
onismo"-->" sincronismo ";"cidade"-->" cidade ";"comentario"-->" comentario ";"tecnica"-->" tecnica ";"informacao"-->" informac
ao ";"informacoes"-->" informacoes ";"sucesso"-->" sucesso ";"titular"-->" titular ";"procedimentos"-->" procedimentos ";"brasi
l"-->" brasil ";"expoediente"-->"expediente";"failure"-->" failure ";"peripheral"-->" peripheral ";"desliga"-->" desliga ";" am
esma "-->" a mesma ";"solicitante"-->" solicitante ";" bomnoise "-->" bom noise ";"errorcode"-->" error code ";"reclamando"-->"
reclamando ";"priorizada"-->" priorizada ";"protocolo"-->" protocolo ";"acesso"-->" acesso ";"whatsapp"-->" whatsapp ";"watsap
p"-->" whatsapp ";"reclamademora"-->" reclama demora ";"pendentenonebanda"-->" pendente none banda ";"prazoid"-->" prazo id
";"ledinternet"-->" leds internet ";"retorneisr"-->" retornei sr ";"sinalsua"-->" sinal sua ";" seje "-->" seja ";" noneteste
"-->" none teste ";"velo cidade"-->" velocidade ";" dow "-->" down ";" resetada "-->" reiniciar ";"nonefalei"-->" none falei
";"wpp"-->" whatsapp ";" reinit "-->" reiniciar ";" qiue "-->" que ";" goistaria "-->" gostaria ";"wattsap"-->" whatsapp ";" te
levendasem "-->" televenda sem ";"massiva"-->" massiva ";" modeme "-->" modem e ";" luzinternet "-->" luz internet ";" apagadat
eve "-->" apagada teve ";" adccmensagem "-->" adcc mensagem ";" testeping "-->" teste ping ";" msdownload "-->" ms download ";"
desdepartemanha "-->" desde parte manha ";" realizadosmodem "-->" realizados modem ";" fito "-->" feito ";" pin "-->" ping ";"a
dcc"-->" adcc ";" uplo "-->" upload ";"none"-->" none ";" emailremovedco "-->" emailremoved co ";" luafibra "-->" sua fibra ";"
statusnotworkingont "-->" status not working ";" not "-->" nao ";" working "-->" funcionando ";" printrmagsdr "-->" print ";" g
etbrassessioninfo "-->" no ";" sabr "-->" sobre ";" backend error sessions found matching criteria "-->" backend error ";" soli
citante "-->" solicita ";" resolvidongaspoperational "-->" resolvido ngasp operational ";" resete "-->" reiniciar ";" prints "-
->" print ";" pere "-->"espaço";" mota "-->"espaço";" dz "-->" desligado reiniciar ";" stom "-->" sem tom ";" seabra "-->"espaç
o";" cancelamentobanda "-->" cancelamento banda ";" cancelamentoassinatura "-->" cancelamento assinatura ";" cancelarassinatura
por "-->" cancelar assinatura ";" cancelaresta "-->" cancelar esta ";" canc "-->" cancelamento ";" office "-->"espaço";" downla
od "-->" download ";" conf "-->" confirma ";" upinfolive "-->" infolive ";" mg "-->" mega ";" megas "-->" mega ";" ms "-->" mil
iseconds ";" fiber "-->" fibra ";" inf "-->" informa ";" but "-->" botao ";" the "-->"espaço";" of "-->"espaço";" is "-->" esta
";" stable "-->" estavel ";" operating "-->" operando ";" administrative "-->" administrativo ";" dying "-->" queda ";" odem "-
->" modem ";" resetar "-->" reiniciar ";" tando "-->" esta ";" bomoperational "-->" bom operational ";" bo "-->" bom ";" bomsta
tus "-->" bom status ";" recebimentomensagemmensagem "-->" recebimento mensagem mensagem ";" caiena "-->"espaço";" logarminicrm
"-->" logar minicrm ";" logar "-->" login ";" info "-->" informa ";" nt "-->"espaço";" df "-->"espaço";" aprovisionamentonome
"-->" aprovisionamento nome ";" evdsloperational "-->" evdsl operational ";" portela "-->"espaço";
```

Figura 4.7: Remoção e correção de palavras de forma empírica ou manual

Figura 4.7, pode-se observar alguns caracteres ou conjunto de caracteres especiais, que não foram removidos com expressões regulares, como descrito na Seção 4.2.2, bem como, palavras, nomes próprios, palavras aglutinadas sem espaço, palavras no idioma inglês, ou seja, todos os vocábulos que não foram removidos ou corrigidos de forma automatizada como descrito nas Seções 4.2.7 e 4.2.8. Para cada modificação feita nas palavras dos textos, foram realizados testes que podem ser observados na Seção 5.1.

4.2.5 *Tokenization*

Como explicado na Seção 2.2.3.4, essa técnica é responsável por receber uma frase ou um texto e dividi-lo em um conjunto de palavras, gerando um vetor contendo cada palavra do texto para cada texto de relato do cliente. Na Figura 4.8, pode-se observar um exemplo prático desse processo útil para as etapas seguintes de remoção de *Stop Words*, remoção de nomes próprios, correção ortográfica, *Stemming* e *Lemmatization*, nas Seções 4.2.6, 4.2.7,

```
["titular", "endereco", "sobre", "protocolo", "callback", "acesso", "emailremoved", "cliente", "reclama", "esta",   ...  "entr
a", "contato", "informa", "sem", "sinal", "internet", "efetuado", "teste", "porem", "sem",   ...  "sem", "acesso", "wifinormali
zou", "apos", "procedimentos", "cliente", "informa", "esta", "com", "baixa",   ...  "cliente", "informa", "nao", "resolver", "p
roblema", "desligou", "protocolo", "nao", "verbalizado", "ligacao",   ...  "conexao", "esta", "apresentando", "lentidoes", "dis
se", "realizou", "teste", "computador", "cabeada", "mega",   ...  "aberta", "outros", "caso", "cliente", "encontra", "com", "pr
oblemas", "conexao", "aproveitamos", "atualizar",   ...  "cliente", "solicita", "alteracao", "cadastro", "mailsem", "conexao",
"totalmente", "cliente", "informa", "esta",   ...  "computador", "cliente", "informa", "modem", "esta", "reiniciando", "constan
temente", "fazendo", "banda", "larga",   ...  "esta", "com", "problema", "ja", "faz", "tempo", "ate", "agora", "nao", "resolvid
o",   ...  "resolvidorealizamos", "teste", "conforme", "instrucoesarvore", "somente", "conecta", "alguns", "dispositivos", "cli
ente", "sem",   ...  "whatsapp", "cliente", "reclama", "esta", "sem", "internet", "sistema", "consta", "sem", "sincronismo",
...  "solicitou", "diversas", "vezes", "reparo", "tecnico", "ninguem", "comparece", "consta", "sistema", "ordem",   ...  "com",
"quedas", "intermitente", "ngasp", "sem", "massiva", "internet", "desliga", "com", "esta",   ...  "recebendo", "sem", "porem",
"nao", "possui", "cabo", "conectado", "modem", "cliente", "informa",   ...  "cliente", "reclama", "nao", "consegue", "conecta
r", "internet", "feito", "teste", "sem", "sincronismo",   ...  "nao", "funciona", "operational", "status", "upline", "attenuati
on", "up", "attenuation", "up", "status",   ...  ]
```

Figura 4.8: Tokenização antes da correção ortográfica

4.2.8, 4.2.9 e 4.2.10, respectivamente. Há algoritmos de vetorização utilizados no PLN que realizam esse processo de vetorização como etapa intermediária, antes de transformarem e associarem números às palavras. As aplicações desses algoritmos foram explicadas de forma teórica na Seção 2.2.4.

4.2.6 Remoção de *Stop Words*

Nessa etapa, foi feita uma análise para remoção das palavras vazias, ou *stop words*, tais como: conjunções, artigos, pronomes e alguns verbos. De acordo com a descrição teórica na Seção 2.2.3.5, a remoção dessas palavras em técnicas de PLN é uma prática comum e geralmente aumenta o desempenho dos modelos. Para a remoção de *stop words*, foi utilizada a biblioteca *Natural Language Toolkit* (NTLK) na linguagem *Python* como base, com o parâmetro portuguese, para a língua portuguesa. A Figura 4.9 mostra um exemplo de um conjunto de textos de mais de um chamado antes da remoção de palavras vazias para exemplificar o processo. Como pode-se observar, há muito ruído, caracteres especiais, pontuação, dados pessoais, nomes próprios, além de erros de digitação, que são os mais difíceis de se corrigir.

```
'Titular [redacted] CPF [redacted] Endereço [redacted]
Protocolo [redacted] Callback [redacted] Acesso [redacted] Email [redacted] Motivo cliente reclama que esta sem a
cesso a internet / sem massiva sem bloqueio / sistemas indicam sem sincronismo MSAN:\tRJRJO_MSAN0461\tSLOT:\t4\tPORTA:\t46Sta
tus Porta:\tNao sincronizada\tVelocidade AAA (UP / DOWN):\t30.00 / 60.00\tVelocidade Gerência (UP / DOWN):\t30.00 / 60.00\tVe
locidade Sincronismo (UP / DOWN):\t0.00 / 0.00\tABR (UP / DOWN):\t7.30 / 20.88\tKL0:\t12.2Vlan(s) Configurada(s):\t1-2 - 50 -
100 - 150COM SUCESSO [redacted] Cliente ([redacted] - gerente) informa que esta sem sinal de internet.Sem massiva (Nen
hum registro foi encontrado) e sem bloqueio por fatura. Ledes apagados em modem. Modem sem sincronismo.Testes sem sucesso. cl
iente com suspensao total.favor verificar sobre desbloqueio.nao se trata de problema tecnicoNoneCall back com sucesso para [redacted]
em linha. [redacted] protocolo [redacted] contato; [redacted] [redacted] senha nao cobnext
a no celular ligara depois pois a conexao se encontra normal e somnete no celular dela. nao tem massiva orenetada a reniciar
o celualr e tentar novamnete Solicitante: [redacted] ... Protocolo: [redacted] Acesso: [redacted]
Validação URA: Não [redacted] Estr.: [redacted] cliente com baixa velocidade, mas sem computador para testesAAA atua
lizado com sucessocliente informa que a internet esta lenta realizado teste com sucesso.obs.: cliente não tem computador test
e feito via WIFI.Cliente estava ligando do fixo do modem pedi que encerrasse para que eu retornasse no celular. CLIENTE RECLA
MA DA SUA INETRENET SEM NAVEGAÇÃO SEM MASSIVA SEM BLOQUEIO TESTE REALIZADO SEM SUCESSO cliente entrou em contato informando q
ue está sem sinal em sua internet, feito testes em sistema sem sucesso.Cliente não consegue conectar com o celular no 2g, apa
rece "acesso negado" Foi realizado os procedimentos para ver se foi. Foi feito o reset do modem.CLIENTE COM QUEDAS REALIZEI O
S TESTE E ENCAMINHEISolicitação de Reset NGASP enviada com sucesso.CLIENTE GPON / SEM NGASPCLIENTE RECLAMA QUE SUA INTERNET E
STA SEM CONEXÃO, POREM POSSUI FALHA DE SINCRONISMO.Cliente sem conexão. Operational Status DOWN Line Attenuation / UP -  Atte
```

Figura 4.9: Textos antes da limpeza

```
' titular endereco sobre protocolo callback acesso emailremoved cliente reclama esta sem acesso internet sem massiva sem bloq
ueio sistemas indicam sem sincronismo msan msan slot porta status porta nao sincronizada velocidade up down velocidade gerenc
ia up down velocidade sincronismo up down up down configurada com sucesso cliente gerente informa esta sem sinal internet sem
massiva nenhum registro encontrado sem bloqueio fatura leds apagados modem modem sem sincronismo teste sem sucesso cliente co
m suspensao total favor verificar sobre desbloqueio nao trata problema tecnico callback com sucesso linha protocolo contato e
mailremoved nao conecta celular ligara pois conexao encontra normal somente celular nao tem massiva orientada reiniciar celul
ar tentar novamente solicita protocolo acesso validacao ura nao esta cliente com baixa velocidade sem computador teste atuali
zado com sucesso cliente informa internet esta lenta realizado teste com sucesso cliente nao tem computador teste feito via w
ifi cliente ligando fixo modem pedi encerrasse retornasse celular cliente reclama internet sem navegacao sem massiva sem bloq
ueio teste realizado sem sucesso cliente entrou contato informando esta sem sinal internet feito teste sistema sem sucesso cl
iente nao consegue conectar com celular aparece acesso negado realizado procedimentos ver feito reiniciar modem cliente com q
uedas realizei teste encaminhei solicitacao reiniciar ngasp enviada com sucesso cliente gpon sem ngasp cliente reclama intern
et esta sem conexao porem possui falha sincronismo cliente sem conexao operational status down attenuation up attenuation up
status attenuation down attenuation down status margin indicator up margin up status margin indicator down margin down status
operational status changes msan hostname spspo msan msan slot msan port msan msan shelf up down down cause cliente sem conexa
o nao consta massiva realizado teste nao normalizado maquina sem print cliente informa esta sem conexao com internet informa
sempre verifica ordem informa esta com tecnico porem nunca aparece apos verificar porem cliente esta agendada hoje informei c
liente onde informa vai aguardar comparecimento tecnico local ott trouble ticket gpon nao iniciado linha titular protocolo te
lefone callback endereco [redacted] emailremoved ligacao cliente informa esta sem conexao realizado teste sem sucesso cliente nao
```

Figura 4.10: Textos após a limpeza

Na Figura 4.10, é possível observar uma ilustração do texto após a remoção das palavras vazias, bem como, outras modificações descritas nessa Seção 4.2. Na Tabela 4.5, estão listadas as palavras vazias para língua portuguesa disponíveis na biblioteca NLTK. Essas palavras foram removidas dos textos de descrição dos problemas, porém optou-se por não remover algumas palavras que, no contexto desse trabalho, agregam significado e resultam em modificações dos resultados nos testes de classificação realizados.

Tabela 4.5: *Stop words* removidas e *stop words* não removidas

***Stop words* removidas**	**não removidas**
a, à, ao, aos, aquela, aquelas, aquele, aqueles, aquilo, as, às, até, com, como, da, das, de, dela, delas, dele, deles, depois, do, dos, e, é, ela, elas, ele, eles, em, entre, era, eram, éramos, essa, essas, esse, esses, esta, está, estamos, estão, estar, estas, estava, estavam, estávamos, este, esteja, estejam, estejamos, estes, esteve, estive, estivemos, estiver, estivera, estiveram, estivéramos, estiverem, estivermos, estivesse, estivessem, estivéssemos, estou, eu, foi, fomos, for, fora, foram, fôramos, forem, formos, fosse, fossem, fôssemos, fui, há, haja, hajam, hajamos, hão, havemos, haver, hei, houve, houvemos, houver, houvera, houverá, houveram, houvéramos, houverão, houverei, houverem, houveremos, houveria, houveriam, houveríamos, houvermos, houvesse, houvessem, houvéssemos, isso, isto, já, lhe, lhes, mais, mas, me, mesmo, meu, meus, minha, minhas, muito, na, não, nas, nem, no, nos, nós, nossa, nossas, nosso, nossos, num, numa, o, os, ou, para, pela, pelas, pelo, pelos, por, qual, quando, que, quem, são, se, seja, sejam, sejamos, sem, ser, será, serão, serei, seremos, seria, seriam, seríamos, seu, seus, só, somos, sou, sua, suas, também, te, tem, tém, temos, tenha, tenham, tenhamos, tenho, terá, terão, terei, teremos, teria, teriam, teríamos, teu, teus, teve, tinha, tinham, tínhamos, tive, tivemos, tiver, tivera, tiveram, tivéramos, tiverem, tivermos, tivesse, tivessem, tivéssemos, tu, tua, tuas, um, uma, você, vocês, vos.	com, fora, há, haja, não, nem, sem, tem, tém, esta, está.

4.2.7 Remoção de nomes próprios

Na Figura 4.11, pode-se observar a base de dados inicial contendo nomes próprios obtida em [84]. Para tornar possível a utilização dos nomes contidos nessa base para a remoção desejada nesse trabalho, realizou-se um pré-processamento dos dados nela contidos, para adequá-los à forma de remoção de palavras utilizada nesse trabalho. Dessa forma, após esse pré-processamento, os dados ficaram no formato desejado, todos em forma de *tokens*, como se pode observar na Figura 4.12. Essa lista de tokens de nomes próprios ficou com

650681 itens. Com essa lista de nomes próprios pronta, a remoção dos nomes próprios foi realizada substituindo-se cada nome próprio contido em um chamado de cliente por um espaço. No final de todo o processamento dos dados, os espaços em excesso foram removidos, como se pode observar no exemplo da Figura 4.10. Os nomes próprios que foram realmente removidos dos textos não foram apresentados nesse trabalho para manter o sigilo sobre os dados pessoais de clientes da empresa que forneceu a base de dados para essa pesquisa.

	alternative_names	classification	first_name	group_name
0	AILINE|ALEINE|ALIINE|ALINE|ALINER|ALINHE|ALINN...	F	AALINE	ALINE
1	ARAAO|ARAO	M	AARAO	ARAO
2	AHARON|AROM|ARON|ARYON|HARON	M	AARON	ARON
3	ADA|ADAH|ADAR|ADHA|HADA	F	ABA	ADA
4	NaN	M	ABADE	ABADE
...	...	...	...	...
100782	MACILEIA	F	MACLEIA	MACILEIA
100783	GELINE|GILEINE|GLEINE|GLEINER|GLEYNE|JAELINE|J...	F	GIULINE	JALINE
100784	DEMILTOM|DEMILTON	M	DEMAILTON	DEMILTON
100785	ALIVIA|ELIVIA|EULIVIA|HOLIVIA|LEIVIA|LIIVIA|LI...	F	ILIVIA	LIVIA
100786	FABSON	M	FADSON	FABSON

100787 rows × 4 columns

Figura 4.11: Base de dados com nomes próprios

```
["ailine", "aleine", "aliine", "aline", "aliner", "alinhe", "alinne", "alyne", "alynne", "ayline",   ...  "elizangla", "elizanj
ela", "elysangela", "helisangela", "helizangela", "ilisangela", "ilizangela", "lisangela", "lizangela", "luisangela",   ...  "c
ali", "keli", "celian", "kalian", "kelian", "kilian", "calian", "kilian", "celiana", "ciliana",   ...  "darc", "darcieli", "dar
cilei", "darciley", "darclei", "darcley", "darceli", "darcilei", "darcilia", "darcelia",   ...  "nan", "edineire", "edineire",
"edinis", "ediniz", "edineis", "edineis", "nan", "edineiva", "edineiva",   ...  "emilayne", "emiliane", "emilyane", "emylayne",
"emyliane", "hemilaine", "hemilayne", "hemiliane", "emylaine", "emiliane",   ...  "genielsom", "genielson", "genilsom", "genlso
n", "jenailson", "jeneilson", "jenelson", "jenielson", "jenilsom", "jenilson",   ...  "luidi", "luidy", "luydi", "luydy", "heli
di", "luidi", "alibia", "alidia", "elibia", "elidia",   ...  "irinelza", "nan", "irineo", "irineo", "irinez", "irines", "irine
s", "nan", "irinete", "irinete",   ...  "jhonnata", "jonata", "jhonatam", "jhonatan", "jhonatann", "jhonatham", "jhonathan", "j
honnathan", "johnatam", "johnatan",   ...  "karleane", "carleane", "carleani", "karleani", "karleany", "carleani", "carieli",
"carlei", "carley", "carli",   ...  "eleize", "eleizer", "elise", "eliser", "elize", "elizer", "eluise", "eluize", "elyse", "eu
lise",   ...  "mauir", "may", "mayr", "mia", "miya", "mai", "mair", "maiaa", "maiah", "maiar",   ...  "naualy", "neli", "nahua
n", "nauam", "nauan", "nauan", "nahuana", "nauanna", "nauhana", "nauana",   ...  "raylenne", "reilene", "rilene", "roilene", "r
olene", "railene", "railene", "raileni", "raileny", "raileni",   ...  "salma", "salmar", "selma", "selmar", "sielma", "silma",
"silmar", "soelma", "solma", "solmar",   ...  "tainara", "tainnara", "taynara", "taynnara", "thainara", "thainnara", "thaynnar
a", "thaynara", "tainara", "taine",   ...  "weruska", "werusca", "weruska", "weruska", "nan", "werventon", "werventon", "werver
som", "werverson", "werverson",   ...  "genifer", "geniffe", "geniffer", "gennife", "gennifer", "genniffer", "gennyfer", "genyf
er", "genyffer", "ghenifer",   ...  "salia", "selair", "selia", "sielia", "silai", "silair", "silia", "soelia", "solair", "soli
a",   ...  "ismaile", "ismailer", "ismayle", "ismile", "ismale", "ismaele", "alenaldo", "alenildo", "elenaldo", "elenildo",
...  "wellighton", "welligthon", "welligtom", "welligton", "wellygton", "willigton", "wuelligton", "wlligton", "welligton", "ac
heley",   ...  ]
```

Figura 4.12: *Tokens* dos nomes próprios

```
endereco --> endereço;callback --> None;emailremoved --> None;msan --> man;msan --> man;gerencia --> gerência;configurada --> c
onfigurado;leds --> leis;suspensao --> suspensão;tecnico --> técnico;callback --> None;emailremoved --> None;ligara --> ligar;c
onexao --> conexão;somnete --> somente;reniciar --> reiniciar;celualr --> celular;novamnete --> novamente;validacao --> validaç
ão;ura --> uma;wifi --> vivi;encerrasse --> enterrasse;retornasse --> tornasse;navegacao --> navegação;reniciar --> reiniciar;e
ncaminhei --> encaminhe;solicitacao --> solicitação;reniciar --> reiniciar;ngasp --> nas;gpon --> upon;ngasp --> nas;conexao --
> conexão;conexao --> conexão;operational --> operacional;attenuation --> None;attenuation --> None;attenuation --> None;attenu
ation --> None;margin --> martin;indicator --> indicador;margin --> martin;margin --> martin;indicator --> indicador;margin -->
martin;operational --> operacional;changes --> chances;msan --> man;hostname --> None;spspo --> susto;msan --> man;msan --> ma
n;msan --> man;port --> por;msan --> man;msan --> man;shelf --> selo;conexao --> conexão;normalizado --> normalidade;print -->
point;conexao --> conexão;tecnico --> técnico;comparecimento --> compartimento;tecnico --> técnico;ott --> out;gpon --> upon;ca
llback --> None;endereco --> endereço;barao --> barco;emailremoved --> None;ligacao --> ligaçao;conexao --> conexão;tecnico -->
técnico;whatsapp --> None;solicitacao --> solicitação;reniciar --> reiniciar;ngasp --> nas;tecnica --> técnica;tecnica --> técn
ica;presencial --> presenciar;endereco --> endereço;ritapolis --> None;tolstoi --> None;conexao --> conexão;ligacao --> ligaça
o;calback --> None;informacoes --> informações;conexao --> conexão;numeracao --> numeração;conexao --> conexão;sisma --> sismo;
ligacao --> ligaçao;voip --> vip;excecao --> excepto;provisioning --> None;habiliado --> habilitado;opcao --> opção;regman -->
regras;conexao --> conexão;tecnico --> técnico;print --> point;conexao --> conexão;ott --> out;bucket --> bunker;reciorede -->
recorde;normalizada --> normalidade;leds --> leis;vdsl --> vos;ligacao --> ligaçao;indisponivel --> indisponível;supervisao -->
supervisão;conexao --> conexão;based --> base;print --> point;endereco --> endereço;solicitacao --> solicitação;print --> poin
t;ott --> out;bucket --> bunker;alp --> ali;arroyo --> arroto;tercnico --> terceiro;conexao --> conexão;ocilando --> citando;me
smp --> mesmo;ersta --> esta;emailremoved --> None;parametros --> parâmetros;operational --> operacional;upline --> uplink;atte
nuation --> None;attenuation --> None;attenuation --> None;attenuation --> None;margin --> martin;indicator --> indicador;margi
n --> martin;margin --> martin;indicator --> indicador;margin --> martin;operational --> operacional;changes --> chances;msan -
-> man;hostname --> None;spspo --> susto;msan --> man;msan --> man;msan --> man;port --> por;msan --> man;msan --> man;shelf --
> selo;ligacao --> ligaçao;intermitencia --> None;efetuados --> efectuados;nergia --> energia;agira --> agora;reniciar --> rein
iciar;ngasp --> nas;conexao --> conexão;msan --> man;reniciar --> reiniciar;ligacao --> ligaçao;caiuabr --> None;delt --> dele;
moden --> modem;reiniciando --> reiniciado;print --> point;whatsapp --> None;whatsapp --> None;nomalizou --> localizou;conexao
--> conexão;emailremoved --> None;endereco --> endereço;reniciar --> reiniciar;normalizou --> normalizar;normalizou --> normali
zar;informacao --> informaçao;reducao --> educado;solicitase --> solicitar;tecnico --> técnico;conexao --> conexão;print --> po
int;urlremoved --> None;backend --> None;error --> erro;beatrizsilva --> None;costamatricula --> None;beloem --> belo;callback
--> None;callback --> None;emailremoved --> None;whatsapp --> None;solicitacao --> solicitação;tecnica --> técnica;wifi --> viv
i;aparendo --> aprendo;conexao --> conexão;endereco --> endereço;ritapolis --> None;tolstoi --> None;conexao --> conexão;consta
tado --> contratado;whatsapp --> None;print --> point;
```

Figura 4.13: Tentativa de correção ortogáfica com a biblioteca *pyspellchecker*

4.2.8 Método para correção ortográfica

Em uma primeira abordagem, tentou-se fazer a correção ortográfica com as seguintes bibliotecas disponíveis para a linguagem de programação *Python*: *pyspellchecker*, *autocorrect* e *textblob*. A correção ortográfica utilizando essas bibliotecas não apresentou bons resultados, modificando diversas palavras importantes nos textos para identificação da classificação correta de um determinado relato de problema de cliente. A biblioteca *textblob* desenvolvida pelo *Google* não oferece serviços de tradução gratuitos, dessa forma, teria um custo para utilizá-la nesse trabalho. As bibliotecas *autocorrect* e *pyspellchecker* geram resultados semelhantes, com muitos erros e correções que prejudicam o desempenho dos algoritmos utilizados nesse trabalho, como pode-se observar na Figura 4.13.

Utilizando a biblioteca *pyspellchecker*, por exemplo, são geradas as seguintes correções indesejadas ou inadequadas para esse trabalho: *callback* corrigido para *None*, *emailremoved* para *None*, *msan* para *man*, *leds* para leis, *ura* para uma, *wifi* para vivi, encerrasse para enterrasse, retornasse para tornasse, *ngasp* para nas, *gpon* para *upon*, *attenuation* para *None*, *margin* para *martin*, *changes* para *chances*, *hostname* para *None*, *spspo* para susto, *port* para por, *shelf* para selo, *print* para *point*, comparecimento para compartimento, *ott* para *out*, barao para barco, *whatsapp* para *None*, *voip* para *vip*, excecao para *excepto*, *provisioning* para *None*, *regman* para regras, *bucket* para *bunker*, reciorede para recorde, *vdsl* para vos, *alp* para ali, arroyo para arroto, tercnico para terceiro, ocilando

para citando, *upline* para *uplink*, *hostname* para *None*, *port* para por, *shelf* para selo, intermitencia para *None*, reducao para educado, aparendo para aprendo, constatado para contratado, entre outras.

Nesse contexto, foi observado que essas bibliotecas fazem as correções com base em um corpus próprio, no qual, cada palavra aparece com uma frequência, que, pelos resultados das correções, é diferente da frequência da palavra no corpus composto pelas palavras da base de dados dessa pesquisa. Dessa forma, para tentar melhorar a base de dados com a correção de erros ortográficos, optou-se por uma correção com base em substituição de palavras de acordo com suas frequências calculadas para a base de dados utilizada, ou seja, criando um corpus próprio.

Primeiramente, observa-se que nos corretores dessas bibliotecas mencionadas, há a substituição ou correção de palavras que já estão corretas, que não precisariam de correção, tanto para o idioma português, quanto para o idioma inglês, tais como: leds, wifi, encerrasse, retornasse, comparecimento, barao, entre outras. Essa palavras já existem em seus idiomas e não precisariam de correção, portanto será necessário um método para verificar se uma palavra existe ou não no seu idioma. Para a língua portuguesa, a lista fornecida em [52] foi utilizada e contém 245365 palavras. Para a língua inglesa, a lista fornecida em [1] foi utilizada e contém 58109 palavras. Dessa forma, por meio de uma consulta simples a essas listas, foi possível verificar se a palavra já existe ou não no idioma. Essas listas não são completas, mas, para os objetivos desse trabalho, geraram correções ortográficas mais precisas em conjunto com as próximas técnicas que serão descritas.

Em uma próxima etapa, será preciso incluir as informações sobre as palavras mais comuns da base de dados estudada, pois na hora de escolher determinada palavra como correta, a palavra mais frequentes dessa base têm maior chance de ser a correta. Dessa forma, diversos textos de relatos de clientes foram concatenados e transformados em uma lista de *tokens*, formando um corpus que representa a base de dados. Após esse processo, criou-se procedimentos para corrigir pequenos erros ortográficos, tais como: palavra faltando uma letra, palavra com uma letra a mais, palavra com uma letra trocada, palavra com uma letra invertida e palavra com duas letras erradas para mais, para menos ou duas trocadas. Os códigos para corrigir esses pequenos erros ortográficos foram criados com base no curso online "corretor ortográfico"disponível na Alura [162]. Embora esses cinco tipos de erros sejam simples, a implementação de métodos para essas correções gerou uma correção ortográfica mais adequada do que a obtida pelas bibliotecas.

Para o primeiro tipo de erro ortográfico, é possível utilizar a palavra "conexão"como

```
aconeao, bconeao, cconeao, econeao, dconeao, fconeao, gconeao, hconeao, iconeao, jconeao, kconeao, lconeao, mconeao, nconeao, o
coneao, pconeao, qconeao, rconeao, sconeao, tconeao, uconeao, vconeao, wconeao, xconeao, yconeao, zconeao, caoneao, cboneao, cc
oneao, ceoneao, cdoneao, cfoneao, cgoneao, choneao, cioneao, cjoneao, ckoneao, cloneao, cmoneao, cnoneao, cooneao, cponeao, cqo
neao, croneao, csoneao, ctoneao, cuoneao, cvoneao, cwoneao, cxoneao, cyoneao, czoneao, coaneao, cobneao, cocneao, coeneao, codn
eao, cofneao, cogneao, cohneao, coineao, cojneao, cokneao, colneao, comneao, conneao, cooneao, copneao, coqneao, cornea0, cosne
ao, cotneao, couneao, covneao, cowneao, coxneao, coyneao, cozneao, conaeao, conbeao, conceao, coneeao, condeao, confeao, congea
o, conheao, conieao, conjeao, conkeao, conleao, conmeao, conneao, conoeao, conpeao, conqeao, conreao, conseao, conteao, conuea
o, conveao, conweao, conxeao, conyeao, conzeao, coneaao, conebao, conecao, coneeao, conedao, conefao, conegao, conehao, coneia
o, conejao, conekao, conelao, conemao, conenao, coneoao, conepao, coneqao, conerao, conesao, conetao, coneuao, conevao, conewa
o, conexao, coneyao, conezao, coneaao, coneabo, coneaco, coneaeo, coneado, coneafo, coneago, coneaho, coneaio, coneajo, coneak
o, conealo, coneamo, coneano, coneaoo, coneapo, coneaqo, conearo, coneaso, coneato, coneauo, coneavo, coneawo, coneaxo, coneay
o, coneazo, coneaoa, coneaob, coneaoc, coneaoe, coneaod, coneaof, coneaog, coneaoh, coneaoi, coneaoj, coneaok, coneaol, coneao
m, coneaon, coneaoo, coneaop, coneaoq, coneaor, coneaos, coneaot, coneaou, coneaov, coneaow, coneaox, coneaoy, coneaoz,
```

Figura 4.14: 182 palavras geradas acrescentando uma letra

exemplo. Suponha que essa palavra apareça no texto como "coneao", faltando a letra "x". Uma forma de achar a palavra correta para substituir essa errada é gerar todas as possíveis palavras acrescentando uma letra, como na Figura 4.14, e, dentre essas palavras, escolher a correta com base na palavra mais frequente que aparece no corpus criado, como descrito abaixo.

- Seja palavra_errada = "coneao";
- Seja lista_frequencia_corpus = [[palavra1, frequencia1], [palavra2, frequencia2], ...] ;
- Gera-se todas as palavras inserindo uma letra na palavra errada;
- Palavras_geradas = [aconeao, bconeao, cconeao, econeao, dconeao, ... coneaoz];
- Calcula-se o número de palavras no corpus: numero_palavras_corpus;
- Para cada palavra em palavras_geradas consulta-se a frequência da palavra;
- Calcula-se a probabilidade para cada palavra em palavras_geradas;
- Probabilidade = frequência/numero_palavras_corpus;
- A palavra corrigira será a que tiver maior probabilidade.

Para o segundo tipo de erro ortográfico, é possível também utilizar a palavra "conexão"como exemplo. Suponha que essa palavra apareça no texto como "connexao", com uma letra "n"a mais. Uma forma de achar a palavra correta para substituir essa errada é gerar todas as possíveis palavras retirando uma letra. Nesse momento, como já foi criado um procedimento para gerar todas as possíveis palavras acrescentando uma letra, basta incluir no método uma parte para gerar também as possíveis palavras retirando uma letra. O resultado para o exemplo é exibido na Figura 4.15, e, dentre essas palavras, basta

aconnexao, bconnexao, cconnexao, econnexao, dconnexao, fconnexao, gconnexao, hconnexao, iconnexao, jconnexao, kconnexao, lconne
xao, mconnexao, nconnexao, oconnexao, pconnexao, qconnexao, rconnexao, sconnexao, tconnexao, uconnexao, vconnexao, wconnexao, x
connexao, yconnexao, zconnexao, caonnexao, cbonnexao, cconnexao, ceonnexao, cdonnexao, cfonnexao, cgonnexao, chonnexao, cionnex
ao, cjonnexao, ckonnexao, clonnexao, cmonnexao, cnonnexao, coonnexao, cponnexao, cqonnexao, cronnexao, csonnexao, ctonnexao, cu
onnexao, cvonnexao, cwonnexao, cxonnexao, cyonnexao, czonnexao, coannexao, cobnnexao, cocnnexao, coennexao, codnnexao, cofnnexa
o, cognnexao, cohnnexao, coinnexao, cojnnexao, coknnexao, colnnexao, comnnexao, connnexao, coonnexao, copnnexao, coqnnexao, cor
nnexao, cosnnexao, cotnnexao, counnexao, covnnexao, cownnexao, coxnnexao, coynnexao, coznnexao, conanexao, conbnexao, concnexa
o, conenexao, condnexao, confnexao, congnexao, conhnexao, coninexao, conjnexao, conknexao, conlnexao, conmnexao, connnexao, con
onexao, conpnexao, conqnexao, conrnexao, consnexao, contnexao, conunexao, convnexao, conwnexao, conxnexao, conynexao, conznexa
o, connaexao, connbexao, conncexao, conneexao, conndexao, connfexao, conngexao, connhexao, conniexao, connjexao, connkexao, con
nlexao, connmexao, connnexao, connoexao, connpexao, connqexao, connrexao, connsexao, conntexao, connuexao, connvexao, connwexa
o, connxexao, connyexao, connzexao, conneaxao, connebxao, connecxao, conneexao, connedxao, connefxao, connegxao, connehxao, con
neixao, connejxao, connekxao, connelxao, connemxao, connenxao, conneoxao, connepxao, conneqxao, connerxao, connesxao, connetxa
o, conneuxao, connevxao, connewxao, connexxao, conneyxao, connezxao, connexaao, connexbao, connexcao, connexeao, connexdao, con
nexfao, connexgao, connexhao, connexiao, connexjao, connexkao, connexlao, connexmao, connexnao, connexoao, connexpao, connexqa
o, connexrao, connexsao, connextao, connexuao, connexvao, connexwao, connexxao, connexyao, connexzao, connexaao, connexabo, con
nexaco, connexaeo, connexado, connexafo, connexago, connexaho, connexaio, connexajo, connexako, connexalo, connexamo, connexan
o, connexaoo, connexapo, connexaqo, connexaro, connexaso, connexato, connexauo, connexavo, connexawo, connexaxo, connexayo, con
nexazo, connexaoa, connexaob, connexaoc, connexaoe, connexaod, connexaof, connexaog, connexaoh, connexaoi, connexaoj, connexao
k, connexaol, connexaom, connexaon, connexaoo, connexaop, connexaoq, connexaor, connexaos, connexaot, connexaou, connexaov, con
nexaow, connexaox, connexaoy, connexaoz, onnexao, cnnexao, conexao, conexao, connxao, conneao, connexo, connexa, connexao,

Figura 4.15: 243 palavras geradas acrescentando e removendo uma letra

escolher a correta com base na palavra mais frequente que aparece no corpus criado, como descrito abaixo.

- Seja palavra_errada = "connexao";
- Seja lista_frequencia_corpus = [[palavra1, frequencia1], [palavra2, frequencia2], ...] ;
- Gera-se todas as palavras inserindo uma letra na palavra errada;
- Gera-se todas as palavras retirando uma letra da palavra errada;
- Palavras_geradas = [aconnexao, bconnexao, cconnexao, econnexao, ... connexao];
- Calcula-se o número de palavras no corpus: numero_palavras_corpus;
- Para cada palavra em palavras_geradas consulta-se a frequência da palavra;
- Calcula-se a probabilidade para cada palavra em palavras_geradas;
- Probabilidade = frequência/numero_palavras_corpus;
- A palavra corrigira será a que tiver maior probabilidade.

Nesse momento, o método já é capaz de corrigir palavras com uma letra a mais e palavras com uma letra a menos. Agora, é preciso resolver o problema da correção de palavras com uma letra trocada. Para isso, também é possível utilizar a palavra "conexão"como exemplo. Suponha que essa palavra apareça no texto como "covexao", com a letra "n"trocada pela letra "v". Uma forma de achar a palavra correta para

```
acovexao, bcovexao, ccovexao, ecovexao, dcovexao, fcovexao, gcovexao, hcovexao, icovexao, jcovexao, kcovexao, lcovexao, mcovexa
o, ncovexao, ocovexao, pcovexao, qcovexao, rcovexao, scovexao, tcovexao, ucovexao, vcovexao, wcovexao, xcovexao, ycovexao, zcov
exao, caovexao, cbovexao, ccovexao, ceovexao, cdovexao, cfovexao, cgovexao, chovexao, ciovexao, cjovexao, ckovexao, clovexao, c
movexao, cnovexao, coovexao, cpovexao, cqovexao, crovexao, csovexao, ctovexao, cuovexao, cvovexao, cwovexao, cxovexao, cyovexa
o, czovexao, coavexao, cobvexao, cocvexao, coevexao, codvexao, cofvexao, cogvexao, cohvexao, coivexao, cojvexao, cokvexao, colv
exao, comvexao, convexao, coovexao, copvexao, coqvexao, corvexao, cosvexao, cotvexao, couvexao, covvexao, cowvexao, coxvexao, c
oyvexao, cozvexao, covaexao, covbexao, covcexao, coveexao, covdexao, covfexao, covgexao, covhexao, coviexao, covjexao, covkexa
o, covlexao, covmexao, covnexao, covoexao, covpexao, covqexao, covrexao, covsexao, covtexao, covuexao, covvexao, covwexao, covx
exao, covyexao, covzexao, coveaxao, covebxao, covecxao, coveexao, covedxao, covefxao, covegxao, covehxao, coveixao, covejxao, c
ovekxao, covelxao, covemxao, covenxao, coveoxao, covepxao, coveqxao, coverxao, covesxao, covetxao, coveuxao, covevxao, covewxa
o, covexxao, coveyxao, covezxao, covexaao, covexbao, covexcao, covexeao, covexdao, covexfao, covexgao, covexhao, covexiao, cove
xjao, covexkao, covexlao, covexmao, covexnao, covexoao, covexpao, covexqao, covexrao, covexsao, covextao, covexuao, covexvao, c
ovexwao, covexxao, covexyao, covexzao, covexaao, covexabo, covexaco, covexaeo, covexado, covexafo, covexago, covexaho, covexai
o, covexajo, covexako, covexalo, covexamo, covexano, covexaoo, covexapo, covexaqo, covexaro, covexaso, covexato, covexauo, cove
xavo, covexawo, covexaxo, covexayo, covexazo, covexaoa, covexaob, covexaoc, covexaoe, covexaod, covexaof, covexaog, covexaoh, c
ovexaoi, covexaoj, covexaok, covexaol, covexaom, covexaon, covexaoo, covexaop, covexaoq, covexaor, covexaos, covexaot, covexao
u, covexaov, covexaow, covexaox, covexaoy, covexaoz, ovexao, cvexao, coexao, covxao, coveao, covexo, covexa, covexao, aovexao,
bovexao, covexao, dovexao, eovexao, fovexao, govexao, hovexao, iovexao, jovexao, kovexao, lovexao, movexao, novexao, oovexao, p
ovexao, qovexao, rovexao, sovexao, tovexao, uovexao, vovexao, wovexao, xovexao, yovexao, zovexao, cavexao, cbvexao, ccvexao, cd
vexao, cevexao, cfvexao, cgvexao, chvexao, civexao, cjvexao, ckvexao, clvexao, cmvexao, cnvexao, covexao, cpvexao, cqvexao, crv
exao, csvexao, ctvexao, cuvexao, cvvexao, cwvexao, cxvexao, cyvexao, czvexao, coaexao, cobexao, cocexao, codexao, coeexao, cofe
xao, cogexao, cohexao, coiexao, cojexao, cokexao, colexao, comexao, conexao, cooexao, copexao, coqexao, corexao, cosexao, cotex
ao, couexao, covexao, cowexao, coxexao, coyexao, cozexao, covaxao, covbxao, covcxao, covdxao, covexao, covfxao, covgxao, covhxa
o, covixao, covjxao, covkxao, covlxao, covmxao, covnxao, covoxao, covpxao, covqxao, covrxao, covsxao, covtxao, covuxao, covvxa
o, covwxao, covxxao, covyxao, covzxao, coveaao, covebao, covecao, covedao, coveeao, covefao, covegao, covehao, coveiao, coveja
o, covekao, covelao, covemao, covenao, coveoao, covepao, coveqao, coverao, covesao, covetao, coveuao, covevao, covewao, covexa
o, coveyao, covezao, covexao, covexbo, covexco, covexdo, covexeo, covexfo, covexgo, covexho, covexio, covexjo, covexko, covexl
o, covexmo, covexno, covexoo, covexpo, covexqo, covexro, covexso, covexto, covexuo, covexvo, covexwo, covexxo, covexyo, covexz
o, covexaa, covexab, covexac, covexad, covexae, covexaf, covexag, covexah, covexai, covexaj, covexak, covexal, covexam, covexa
n, covexao, covexap, covexaq, covexar, covexas, covexat, covexau, covexav, covexaw, covexax, covexay, covexaz, covexaoa, covexa
ob, covexaoc, covexaod, covexaoe, covexaof, covexaog, covexaoh, covexaoi, covexaoj, covexaok, covexaol, covexaom, covexaon, cov
exaoo, covexaop, covexaoq, covexaor, covexaos, covexaot, covexaou, covexaov, covexaow, covexaox, covexaoy, covexaoz,
```

Figura 4.16: 424 palavras geradas acrescentando, removendo e substituindo uma letra

substituir essa errada é gerar todas as possíveis palavras substituindo uma letra da palavra errada. Nesse momento, como já foi criado um procedimento para lidar com os dois problemas anteriores, basta incluir no método uma parte para gerar também as possíveis palavras substituindo uma letra da palavra errada por outra. O resultado para o exemplo é exibido na Figura 4.16 e, dentre essas palavras, basta escolher a correta com base na palavra mais frequente que aparece no corpus criado, como descrito abaixo.

- Seja palavra_errada = "connexao";
- Seja lista_frequencia_corpus = [[palavra1, frequencia1], [palavra2, frequencia2], ...] ;
- Gera-se todas as palavras inserindo uma letra na palavra errada;
- Gera-se todas as palavras retirando uma letra da palavra errada;
- Gera-se todas as palavras substituindo uma letra da palavra errada;
- Palavras_geradas = [aconnexao, bconnexao, cconnexao, econnexao, ... connexao];
- Calcula-se o número de palavras no corpus: numero_palavras_corpus;
- Para cada palavra em palavras_geradas consulta-se a frequência da palavra;
- Calcula-se a probabilidade para cada palavra em palavras_geradas;

- Probabilidade = frequência/numero_palavras_corpus;
- A palavra corrigira será a que tiver maior probabilidade.

```
acoenxao, bcoenxao, ccoenxao, ecoenxao, dcoenxao, fcoenxao, gcoenxao, hcoenxao, icoenxao, jcoenxao, kcoenxao, lcoenxao, mcoenxa
o, ncoenxao, ocoenxao, pcoenxao, qcoenxao, rcoenxao, scoenxao, tcoenxao, ucoenxao, vcoenxao, wcoenxao, xcoenxao, ycoenxao, zcoe
nxao, caoenxao, cboenxao, ccoenxao, ceoenxao, cdoenxao, cfoenxao, cgoenxao, choenxao, cioenxao, cjoenxao, ckoenxao, cloenxao, c
moenxao, cnoenxao, cooenxao, cpoenxao, cqoenxao, croenxao, csoenxao, ctoenxao, cuoenxao, cvoenxao, cwoenxao, cxoenxao, cyoenxa
o, czoenxao, coaenxao, cobenxao, cocenxao, coeenxao, codenxao, cofenxao, cogenxao, cohenxao, coienxao, cojenxao, cokenxao, cole
nxao, comenxao, conenxao, cooenxao, copenxao, coqenxao, corenxao, cosenxao, cotenxao, couenxao, covenxao, cowenxao, coxenxao, c
oyenxao, cozenxao, coeanxao, coebnxao, coecnxao, coeenxao, coednxao, coefnxao, coegnxao, coehnxao, coeinxao, coejnxao, coeknxa
o, coelnxao, coemnxao, coennxao, coeonxao, coepnxao, coeqnxao, coernxao, coesnxao, coetnxao, coeunxao, coevnxao, coewnxao, coex
nxao, coeynxao, coeznxao, coenaxao, coenbxao, coencxao, coenexao, coendxao, coenfxao, coengxao, coenhxao, coenixao, coenjxao, c
oenkxao, coenlxao, coenmxao, coennxao, coenoxao, coenpxao, coenqxao, coenrxao, coensxao, coentxao, coenuxao, coenvxao, coenwxa
o, coenxxao, coenyxao, coenzxao, coenxaao, coenxbao, coenxcao, coenxeao, coenxdao, coenxfao, coenxgao, coenxhao, coenxiao, coen
xjao, coenxkao, coenxlao, coenxmao, coenxnao, coenxoao, coenxpao, coenxqao, coenxrao, coenxsao, coenxtao, coenxuao, coenxvao, c
oenxwao, coenxxao, coenxyao, coenxzao, coenxaao, coenxabo, coenxaco, coenxaeo, coenxado, coenxafo, coenxago, coenxaho, coenxai
o, coenxajo, coenxako, coenxalo, coenxamo, coenxano, coenxaoo, coenxapo, coenxaqo, coenxaro, coenxaso, coenxato, coenxauo, coen
xavo, coenxawo, coenxaxo, coenxayo, coenxazo, coenxaoa, coenxaob, coenxaoc, coenxaoe, coenxaod, coenxaof, coenxaog, coenxaoh, c
oenxaoi, coenxaoj, coenxaok, coenxaol, coenxaom, coenxaon, coenxaoo, coenxaop, coenxaoq, coenxaor, coenxaos, coenxaot, coenxao
u, coenxaov, coenxaow, coenxaox, coenxaoy, coenxaoz, oenxao, cenxao, conxao, coexao, coenao, coenxo, coenxa, coenxao, aoenxao,
boenxao, coenxao, doenxao, eoenxao, foenxao, goenxao, hoenxao, ioenxao, joenxao, koenxao, loenxao, moenxao, noenxao, ooenxao, p
oenxao, qoenxao, roenxao, soenxao, toenxao, uoenxao, voenxao, woenxao, xoenxao, yoenxao, zoenxao, caenxao, cbenxao, ccenxao, cd
enxao, ceenxao, cfenxao, cgenxao, chenxao, cienxao, cjenxao, ckenxao, clenxao, cmenxao, cnenxao, coenxao, cpenxao, cqenxao, cre
nxao, csenxao, ctenxao, cuenxao, cvenxao, cwenxao, cxenxao, cyenxao, czenxao, coanxao, cobnxao, cocnxao, codnxao, coenxao, cofn
xao, cognxao, cohnxao, coinxao, cojnxao, coknxao, colnxao, comnxao, connxao, coonxao, copnxao, coqnxao, cornxao, cosnxao, cotnx
ao, counxao, covnxao, cownxao, coxnxao, coynxao, coznxao, coeaxao, coebxao, coecxao, coedxao, coeexao, coefxao, coegxao, coehxa
o, coeixao, coejxao, coekxao, coelxao, coemxao, coenxao, coeoxao, coepxao, coeqxao, coerxao, coesxao, coetxao, coeuxao, coevxa
o, coewxao, coexxao, coeyxao, coezxao, coenaao, coenbao, coencao, coendao, coeneao, coenfao, coengao, coenhao, coeniao, coenja
o, coenkao, coenlao, coenmao, coennao, coenoao, coenpao, coenqao, coenrao, coensao, coentao, coenuao, coenvao, coenwao, coenxa
o, coenyao, coenzao, coenxao, coenxbo, coenxco, coenxdo, coenxeo, coenxfo, coenxgo, coenxho, coenxio, coenxjo, coenxko, coenxl
o, coenxmo, coenxno, coenxoo, coenxpo, coenxqo, coenxro, coenxso, coenxto, coenxuo, coenxvo, coenxwo, coenxxo, coenxyo, coenxz
o, coenxaa, coenxab, coenxac, coenxad, coenxae, coenxaf, coenxag, coenxah, coenxai, coenxaj, coenxak, coenxal, coenxam, coenxa
n, coenxao, coenxap, coenxaq, coenxar, coenxas, coenxat, coenxau, coenxav, coenxaw, coenxax, coenxay, coenxaz, coenxaoa, coenxa
ob, coenxaoc, coenxaod, coenxaoe, coenxaof, coenxaog, coenxaoh, coenxaoi, coenxaoj, coenxaok, coenxaol, coenxaom, coenxaon, coe
nxaoo, coenxaop, coenxaoq, coenxaor, coenxaos, coenxaot, coenxaou, coenxaov, coenxaow, coenxaox, coenxaoy, coenxaoz, ocenxao, c
eonxao, conexao, coexnao, coenaxo, coenxoa,
```

Figura 4.17: 430 palavras geradas acrescentando, removendo, substituindo e invertendo uma letra

Para o problema de palavra com uma letra invertida, pode-se usar também a palavra "conexão"como exemplo. Suponha que essa palavra apareça no texto como "coenxao". o processo é semelhante bastando incluir no método anterior uma parte para gerar todas as palavras invertendo a posição de duas letras, como exibido na Figura 4.17 e, ao final, escolher a correta com base na palavra mais frequente que aparece no corpus criado, como descrito abaixo.

- Seja palavra_errada = "connexao";
- Seja lista_frequencia_corpus = [[palavra1, frequencia1], [palavra2, frequencia2], ...] ;
- Gera-se todas as palavras inserindo uma letra na palavra errada;
- Gera-se todas as palavras retirando uma letra da palavra errada;
- Gera-se todas as palavras substituindo uma letra da palavra errada;
- Gera-se todas as palavras invertendo a posição de duas letras;

- Palavras_geradas = [aconnexao, bconnexao, cconnexao, econnexao, ... connexao];
- Calcula-se o número de palavras no corpus: numero_palavras_corpus;
- Para cada palavra em palavras_geradas consulta-se a frequência da palavra;
- Calcula-se a probabilidade para cada palavra em palavras_geradas;
- Probabilidade = frequência/numero_palavras_corpus;
- A palavra corrigira será a que tiver maior probabilidade.

Para o ultimo tipo de problema, uma alternativa para gerar todas as palavras possíveis é utilizar os 4 métodos de geração de palavras anteriores e, em seguida, para cada palavra gerada, utilizar novamente os 4 métodos de geração de palavras. Essa abordagem gera um conjunto de palavras muito grande e calcular a probabilidade para todas essas palavras aumenta o tempo de execução. Tomando como exemplo também a palavra conexão. Suponha que ela apareça como "ccopnexao", ou seja, com duas letras erradas. Ao realizar o processo de geração de palavras duas vezes, tem-se uma lista com 304402 palavras. Como muitas dessas palavras não existem, em vez de calcular a probabilidade para todas essas palavras, optou-se por calcular a probabilidade apenas para palavras dessa lista que estejam no corpus criado a partir da base de dados. Dessa forma, o método final fica assim:

- Seja palavra_errada = "ccopnexao";
- Seja corpus = [palavra1, palavra2, ...];
- Seja lista_frequencia_corpus = [[palavra1, frequencia1], [palavra2, frequencia2], ...] ;
- Gera-se todas as palavras inserindo uma letra na palavra errada;
- Gera-se todas as palavras retirando uma letra da palavra errada;
- Gera-se todas as palavras substituindo uma letra da palavra errada;
- Gera-se todas as palavras invertendo a posição de duas letras;
- Tem-se a primeira_lista_de_palavras;
- Para cada palavra em primeira_lista_de_palavras realiza-se novamente:

- Gera-se todas as palavras inserindo uma letra na palavra errada;
- Gera-se todas as palavras retirando uma letra da palavra errada;
- Gera-se todas as palavras substituindo uma letra da palavra errada;
- Gera-se todas as palavras invertendo a posição de duas letras;
- Tem-se a segunda_lista_de_palavras;
- Para cada palavra em segunda_lista_de_palavras, elimina-se as que não estão no corpus, gerando a terceira lista terceira_lista_de_palavras bem menor que a segunda;
- Calcula-se o número de palavras no corpus: numero_palavras_corpus;
- Para cada palavra em terceira_lista_de_palavras consulta-se a frequência da palavra;
- Calcula-se a probabilidade para cada palavra em palavras_geradas;
- Probabilidade = frequência/numero_palavras_corpus;
- A palavra corrigira será a que tiver maior probabilidade.

4.2.9 *Stemming*

Conforme descrição teórica na Seção 2.2.3.6, essa técnica é responsável pela remoção de prefixos e sufixos das palavras de um texto. Dessa forma, com a aplicação dessa técnica, vocábulos com significados semânticos semelhantes, que antes da aplicação eram entendidos como sendo diferentes por algoritmos, passam a ser entendidos como vocábulos iguais. Nesse trabalho, utilizou-se a implementação em Python do Scikit Learn. A Figura 4.18 ilustra a aplicação dessa técnica.

Na Seção 5.1, os testes realizados são apresentados comparando-se a performance com e sem a aplicação da técnica. Cabe ressaltar que, como a técnica ainda precisa ser melhorada para o idioma português, algumas partes de palavras são removidas, sendo entendidas como prefixo ou sufixo, quando não deveriam ser removidas. Como os testes feitos nesse trabalho estão utilizando uma base de dados reais, que não foi planejada e estruturada para aplicação de certos tipos específicos de técnicas de PLN, as remoções de prefixos e sufixos de forma indevida resultaram em uma diminuição de performance nos testes realizados.

```
'titular', 'nomeremoved', 'nomeremoved', 'nomeremoved', 'nomeremoved', 'enderec
o', 'nomeremoved', 'nomeremoved', 'sobre', 'nomeremoved', 'nomeremoved', 'nomer
emoved', 'protocolo', 'callback', 'acesso', 'emailremoved', 'cliente', 'reclam
a', 'esta', 'sem', 'acesso', 'internet', 'sem', 'massiva', 'sem', 'bloqueio',
'sistemas', 'indicam', 'sem', 'sincronismo', 'msan', 'msan', 'slot', 'porta',
'status', 'porta', 'nao', 'sincronizada', 'velocidade', 'up', 'down', 'velocida
de', 'gerencia', 'up', 'down', 'velocidade', 'sincronismo', 'up', 'down', 'up',
'down', 'configurada',

'titul', 'nomeremoved', 'nomeremoved', 'nomeremoved', 'nomeremoved', 'enderec',
'nomeremoved', 'nomeremoved', 'sobr', 'nomeremoved', 'nomeremoved', 'nomeremove
d', 'protocol', 'callback', 'acess', 'emailremoved', 'client', 'reclam', 'est',
'sem', 'acess', 'internet', 'sem', 'mass', 'sem', 'bloquei', 'sistem', 'indic',
'sem', 'sincron', 'msan', 'msan', 'slot', 'port', 'statu', 'port', 'nao', 'sinc
ron', 'veloc', 'up', 'down', 'veloc', 'gerenc', 'up', 'down', 'veloc', 'sincro
n', 'up', 'down', 'up', 'down', 'configur',
```

Figura 4.18: Exemplo de Stemização

4.2.10 *Lemmatization*

Essa técnica, que é responsável por simplificar uma palavra, reduzindo-a à sua forma mais básica, conforme descrição teórica na Seção 2.2.3.7, foi aplicada aos textos utilizados nesse trabalho com tentativa de melhoria de performance nos testes. A Figura 4.19 mostra uma ilustração da técnica com textos reais da base de dados, exibindo os textos antes e depois da aplicação. Como é possível observar na Figura 4.19, a Lematização produz como resultado uma palavra inteira da língua portuguesa, diferentemente da técnica de Stemização, que pode gerar um vocábulo que não é uma palavra. As performances dos testes realizados utilizando essa técnica podem ser observados na Seção 5.1.

```
'titular', 'nomeremoved', 'nomeremoved', 'nomeremoved', 'nomeremoved', 'enderec
o', 'nomeremoved', 'nomeremoved', 'sobre', 'nomeremoved', 'nomeremoved', 'nomer
emoved', 'protocolo', 'callback', 'acesso', 'emailremoved', 'cliente', 'reclam
a', 'esta', 'sem', 'acesso', 'internet', 'sem', 'massiva', 'sem', 'bloqueio',
'sistemas', 'indicam', 'sem', 'sincronismo', 'msan', 'msan', 'slot', 'porta',
'status', 'porta', 'nao', 'sincronizada', 'velocidade', 'up', 'down', 'velocida
de', 'gerencia', 'up', 'down', 'velocidade', 'sincronismo', 'up', 'down', 'up',
'down', 'configurada',

'titular', 'nomeremoved', 'nomeremoved', 'nomeremoved', 'nomeremoved', 'enderec
o', 'nomeremoved', 'nomeremoved', 'sobre', 'nomeremoved', 'nomeremoved', 'nomer
emoved', 'protocolo', 'callbackr', 'acesso', 'emailremoved', 'cliente', 'reclam
ar', 'este', 'sem', 'acesso', 'Internet', 'sem', 'massiva', 'sem', 'bloqueio',
'sistema', 'indicar', 'sem', 'sincronismo', 'msan', 'msan', 'slot', 'porta', 's
tatus', 'porto', 'nao', 'sincronizar', 'velocidade', 'up', 'down', 'velocidad
e', 'gerenciar', 'up', 'down', 'velocidade', 'sincronismo', 'up', 'down', 'up',
'down', 'configurar',
```

Figura 4.19: Exemplo de Lematização

Tabela 4.6: Exemplo de substituição de palavras

Palavra original	**Palavra substituída**
rio janeiro rj	espaço
stuatusporta	status porta
ruimnoise	ruim noise
ruimoperational	ruim operational
nonecall	none call
brsenha	senha
reclamasua	reclama sua
call back	callback
solicitante	solicitante
bomnoise	bom noise
errorcode	error code
reclamademora	reclama demora
pendentenonebanda	pendente none banda

4.3 Detalhamento da metodologia experimental

As reclamações dos clientes são anotadas pelos atendentes da empresa e são compostas por descrição textual informal em língua portuguesa. Essas descrições são armazenadas em um banco de dados para que, posteriormente, um técnico as analise e as classifique em um dos tipos de problemas existentes, como ilustrado na Figura 4.20. Essas descrições nem sempre estão gramaticalmente corretas, podem conter erros de ortografia, erros de sintaxe, palavras abreviadas e outras informações irrelevantes para a descrição do problema. Assim, após diversos testes descritos nesse trabalho, foi desenvolvida uma metodologia composta por uma técnica de pré-processamento personalizada gerando um modelo que apresentou melhor resultado nos testes, conforme ilustrado na Figura 4.21.

Uma reclamação pode conter URLs e e-mails sempre que um usuário decidir deixar seus dados de contato diretamente em sua solicitação, ou quando houver uma reclamação envolvendo o acesso a alguma página da web. Todas as URLs e e-mails foram substituídos pelos sinalizadores '*urlremoved*' e '*emailremoved*', respectivamente. Esses sinalizadores que indicam a presença ou não de URLs e e-mails impactam nos resultados dos testes com algoritmos de classificação. Também foram removidas pontuações e caracteres especiais. A biblioteca *Regular Expressions* (RE) em *Python* foi utilizada para essa remoção.

Em seguida, as palavras foram normalizadas, transformando todas as letras em minúsculas. As palavras aglutinadas foram separadas em duas, como mostrado na Tabela 4.6, e os principais jargões da empresa foram corrigidos e removidos, dependendo de seu

valor semântico.

Em uma próxima etapa, as reclamações foram tokenizadas e tiveram as *stop words* removidas usando a biblioteca *Natural Language Toolkit* (NLTK) em *Python* para a língua portuguesa, com exceção das seguintes palavras: com, fora, há, haja, não, nem, sem, tem, têm, esta e está. Essas palavras não foram removidas porque denotam informações importantes sobre a reclamação, por exemplo, se há internet ou não, coloquialmente.

Algumas reclamações incluem os nomes dos clientes, quando o cliente deixa suas informações pessoais no relato com um atendente. Todos esses nomes foram removidos, por meio de um banco de dados contendo nomes próprios de [84]. Esta lista é composta por 650.681 nomes. Como as reclamações são digitadas, elas contêm erros de ortografia. Esses erros foram corrigidos. Para fazer isso, inicialmente foram utilizadas as seguintes bibliotecas em *Python*: *pyspellchecker, autocorrect e textblob*. A correção ortográfica usando essas bibliotecas não apresentou bons resultados, modificando várias palavras importantes em textos como, por exemplo: *callback para None, msan para man, wifi para vivi, margin para martin, print para point, ott para out, voip para vip*, etc... Como o uso dessas bibliotecas não gerou bons resultados, outra abordagem para corrigir esses erros foi utilizada. Uma lista de palavras corretas em português foi obtida em [52] e uma lista de palavras corretas em inglês em [1]. Essas listas contêm 245.365 e 58.109 palavras, respectivamente.

A partir dessas listas e com base em um corpus construído a partir dos relatos dos clientes, foi utilizado um método baseado nas frequências desse corpus para corrigir alguns tipos básicos de erros ortográficos, tais como: falta de letra em uma palavra, palavra com letra a mais, palavra com letra trocada, palavra com letra invertida e palavra com duas letras erradas para mais, para menos ou duas trocas. Mais detalhes sobre este método podem ser encontrados em [162]. Finalmente, os dados foram vetorizados usando a biblioteca *TfidfVectorizer* ou *CountVectorizer* do *Scikit Learn*. Todo esse processo é ilustrado na Figura 4.21.

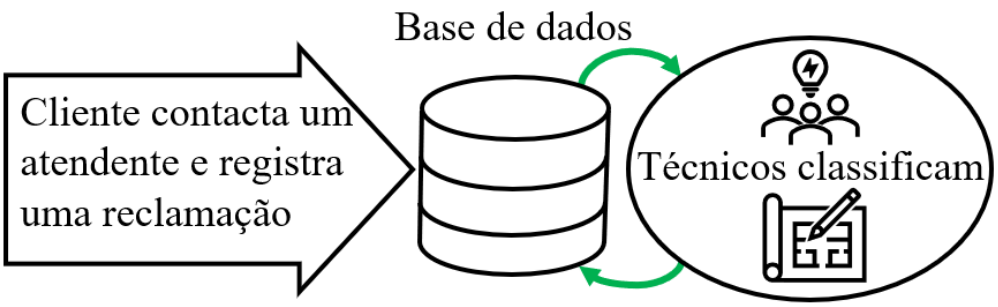

Figura 4.20: Fluxograma da empresa

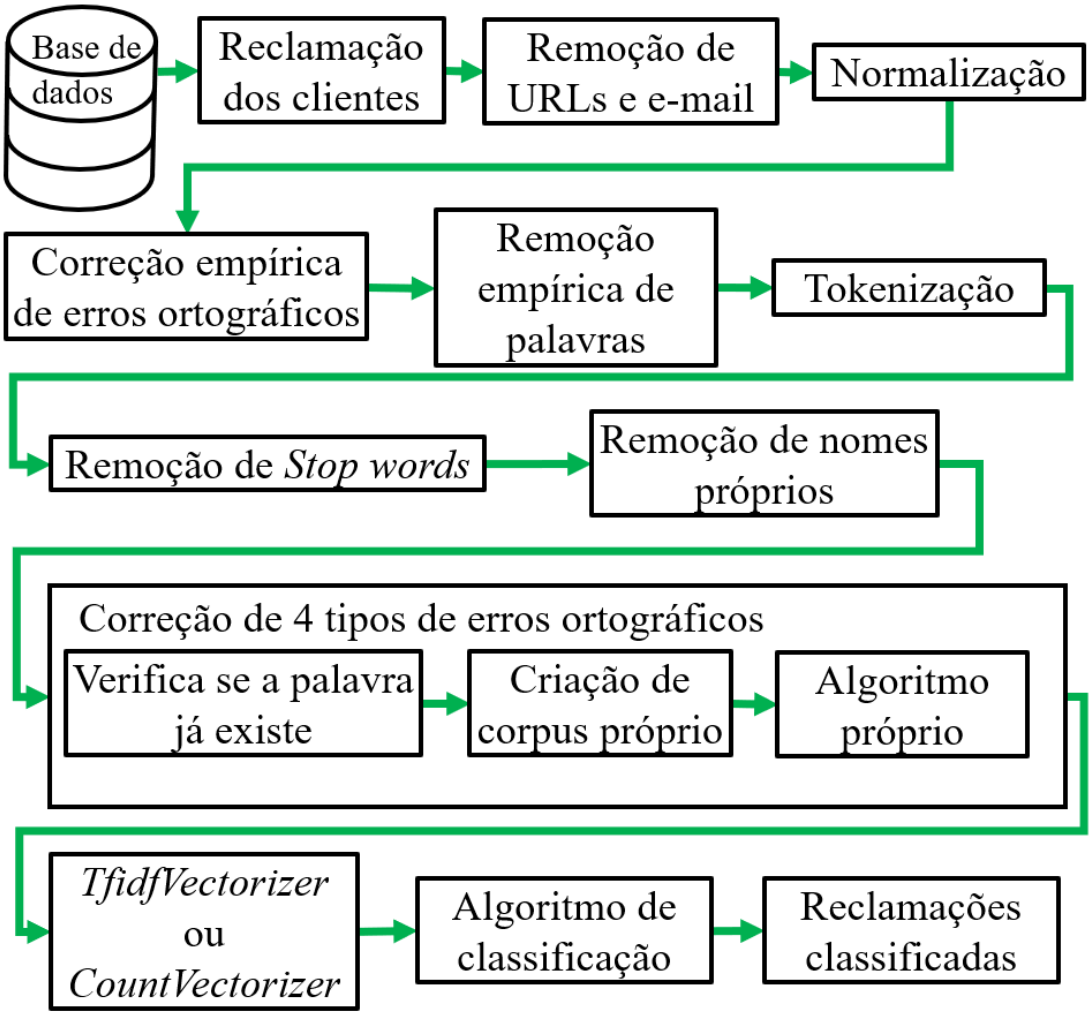

Figura 4.21: Fluxograma da metodologia

Capítulo 5

Testes e Análises

5.1 Descrição dos testes realizados

Os testes cujos resultados estão exibidos nas Tabelas com numeração de 5.1 a 5.32 foram executados em uma máquina virtual com 70 *Cores* Intel (R) Xeon (R) Gold 6230 com 2.10GHz de velocidade, como mostrado na figura 5.1. Essa máquina virtual estava com 264GB de memória RAM como mostrado na figura 5.2. O Sistema Operacional (SO) utilizado foi o Ubuntu Server 22.04.1 LTS. Os algoritmos que tinham a opção de processamento paralelo foram executados utilizando esse recurso para agilizar os testes, chegando a 100% de processamento dos 70 *Cores* para esses algoritmos, como é possível observar na figura 5.3.

Foram utilizados os seguintes algoritmos: *Multi-layer Perceptron Classifier* (MPC), *AdaBoost Classifier* (ABC), *Decision Tree Classifier* (DTC), *Gaussian Naive Bayes* (GNB), *Bernoulli Naive Bayes classifier* (BNB), *Multinomial Naive Bayes classifier* (MNB), *Random Forest Classifier* (RFC), *Extra Trees Classifier* (ETC), *Logistic Regression* (LR), *Support Vector Classifier* (SVC), *Voting Classifier 1* (VC1), *Voting Classifier 2* (VC2), *Custom Stacking Classifier 1* (SC1), *Custom Stacking Classifier 2* (SC2), *Custom Stacking Classifier 3* (SC3), *Stacking Classifier Scikit-learn* (SCS), *eXtreme Gradient Boosting Classifier* (XGBC), além da aplicação em conjunto das técnicas *One verus One* (OVO) e

```
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         43 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  70
  On-line CPU(s) list:   0-69
Vendor ID:               GenuineIntel
  Model name:            Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz
```

Figura 5.1: Máquina para execução dos testes - CPU

```
dt@testes-dt:/srm$ free
               total        used        free      shared  buff/cache   available
Mem:       264115420     1052760   260589384        1620     2473276   261452480
Swap:        4194300           0     4194300
```

Figura 5.2: Máquina para execução dos testes - Memória

```
Tasks: 1034 total,  78 running, 880 sleeping,  76 stopped,   0 zombie
%Cpu(s): 98,6 us,  1,4 sy,  0,0 ni,  0,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem : 257925,2 total, 141150,5 free, 104475,1 used,  12299,5 buff/cache
MiB Swap:   4096,0 total,   4096,0 free,      0,0 used. 142382,0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 208492 dt        20   0 5590988 160968  46384 S 581,3   0,1   0:21.19 python3
 208486 dt        20   0 5591156 159204  45928 S 418,1   0,1   0:15.35 python3
 208496 dt        20   0 5590844 152040  46416 S 363,9   0,1   0:14.78 python3
```

Figura 5.3: Máquina para execução dos testes - Top

Tabela 5.1: Resultados do teste 1

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
MPC	127,56	0,8205	0,8228	0,8205	0,8211
ABC	4,76	0,7136	0,7413	0,7136	0,7213
DTC	4,85	0,7719	0,7779	0,7719	0,7738
GNB	19,66	0,5130	0,6093	0,5130	0,5238
BNB	8,72	0,7197	0,7708	0,7197	0,7303
MNB	2,93	0,7675	0,7769	0,7675	0,7696
RFC	2,13	0,8291	0,8324	0,8291	0,8301
ETC	4,07	0,8358	0,8388	0,8358	0,8367
LR	10,57	0,8280	0,8330	0,8280	0,8292
SVC	66,25	0,8433	0,8489	0,8433	0,8448
VC1	442,53	0,8450	0,8481	0,8450	0,8459
VC2	472,68	0,8433	0,8467	0,8433	0,8442
SC1	1166,77	0,8433	0,8475	0,8433	0,8444
SC2	992,33	0,8447	0,8484	0,8447	0,8457
SC3	677,56	0,8475	0,8511	0,8475	0,8482
SCS	258,18	0,8447	0,8480	0,8447	0,8457
XGBC	87,42	0,8380	0,8453	0,8380	0,8401

Tabela 5.2: Resultados do teste 2

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
MPC	92,14	0,8066	0,8108	0,8066	0,8082
ABC	1,66	0,7291	0,7615	0,7291	0,7377
DTC	1,72	0,7925	0,7997	0,7925	0,7951
GNB	7,07	0,5222	0,5723	0,5222	0,5148
BNB	2,82	0,7552	0,7857	0,7552	0,7614
MNB	1,01	0,7666	0,7756	0,7666	0,7672
RFC	1,68	0,8330	0,8378	0,8330	0,8345
ETC	2,46	0,8372	0,8401	0,8372	0,8382
LR	5,43	0,8177	0,8233	0,8177	0,8194
SVC	34,98	0,8319	0,8415	0,8319	0,8347
VC1	283,38	0,8441	0,8491	0,8441	0,8457
VC2	268,11	0,8455	0,8505	0,8455	0,8471
SC1	702,52	0,8436	0,8504	0,8436	0,8455
SC2	590,99	0,8447	0,8508	0,8447	0,8465
SC3	401,11	0,8436	0,8491	0,8436	0,8451
SCS	163,94	0,8450	0,8492	0,8450	0,8464
XGBC	151,88	0,8322	0,8432	0,8322	0,8354

Tabela 5.3: Resultados do teste 3

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
MPC	165,73	0,8207	0,8272	0,8207	0,8228
ABC	4,00	0,7280	0,7585	0,7280	0,7363
DTC	6,64	0,8030	0,8120	0,8030	0,8060
GNB	24,49	0,4891	0,5204	0,4891	0,4699
BNB	10,49	0,7648	0,7817	0,7648	0,7681
MNB	3,32	0,7745	0,7754	0,7745	0,7728
RFC	2,16	0,8410	0,8509	0,8410	0,8439
ETC	4,85	0,8405	0,8478	0,8405	0,8428
LR	9,72	0,8254	0,8321	0,8254	0,8274
SVC	132,02	0,8415	0,8520	0,8415	0,8445
VC1	1293,31	0,8478	0,8548	0,8478	0,8499
VC2	1248,42	0,8465	0,8538	0,8465	0,8486
SC1	2905,23	0,8461	0,8542	0,8461	0,8485
SC2	2686,43	0,8484	0,8566	0,8484	0,8507
SC3	1559,86	0,8507	0,8586	0,8507	0,8529
SCS	674,09	0,8478	0,8547	0,8478	0,8500
XGBC	6,84	0,8390	0,8531	0,8390	0,8430

Tabela 5.4: Resultados do teste 4

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
MPC	86,91	0,7575	0,7602	0,7575	0,7551
ABC	1,97	0,6286	0,6833	0,6286	0,6414
DTC	2,34	0,7358	0,7401	0,7358	0,7330
GNB	5,83	0,5116	0,5424	0,5116	0,4945
BNB	2,46	0,6977	0,7089	0,6977	0,6918
MNB	0,76	0,7119	0,7383	0,7119	0,7172
RFC	0,70	0,7791	0,7854	0,7791	0,7778
ETC	1,55	0,7783	0,7843	0,7783	0,7766
LR	3,93	0,7697	0,7777	0,7697	0,7685
SVC	25,66	0,7827	0,7896	0,7827	0,7821
VC1	245,72	0,7897	0,7953	0,7897	0,7885
VC2	247,82	0,7913	0,7979	0,7913	0,7902
SC1	625,70	0,7891	0,7962	0,7891	0,7880
SC2	522,38	0,7911	0,7975	0,7911	0,7900
SC3	365,01	0,7911	0,7987	0,7911	0,7901
SCS	161,81	0,7941	0,7990	0,7941	0,7931
XGBC	188,21	0,7786	0,7880	0,7786	0,7774

Tabela 5.5: Resultados do teste 5

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
MPC	24,93	0,8005	0,8057	0,8005	0,8021
ABC	1,40	0,7147	0,7569	0,7147	0,7254
DTC	1,02	0,7691	0,7769	0,7691	0,7718
GNB	0,52	0,6158	0,6539	0,6158	0,5828
BNB	0,21	0,7505	0,7703	0,7505	0,7533
MNB	0,05	0,7475	0,7543	0,7475	0,7488
RFC	0,43	0,8183	0,8240	0,8183	0,8201
ETC	0,69	0,8211	0,8269	0,8211	0,8229
LR	2,32	0,7986	0,8081	0,7986	0,8010
SVC	17,95	0,8152	0,8259	0,8152	0,8180
VC1	165,04	0,8266	0,8339	0,8266	0,8287
VC2	179,84	0,8261	0,8335	0,8261	0,8281
SC1	394,45	0,8266	0,8348	0,8266	0,8287
SC2	363,49	0,8255	0,8337	0,8255	0,8277
SC3	224,87	0,8277	0,8343	0,8277	0,8296
SCS	117,50	0,8272	0,8338	0,8272	0,8292
XGBC	140,22	0,8208	0,8336	0,8208	0,8243

Tabela 5.6: Resultados do teste 6

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
MPC	42,34	0,8075	0,8159	0,8075	0,8099
ABC	3,56	0,7369	0,7659	0,7369	0,7449
DTC	5,23	0,7929	0,8013	0,7929	0,7957
GNB	1,93	0,5514	0,6441	0,5514	0,5015
BNB	0,79	0,7529	0,7706	0,7529	0,7561
MNB	0,22	0,7550	0,7570	0,7550	0,7538
RFC	0,95	0,8313	0,8426	0,8313	0,8345
ETC	1,72	0,8338	0,8427	0,8338	0,8364
LR	6,05	0,8055	0,8142	0,8055	0,8081
SVC	96,97	0,8333	0,8460	0,8333	0,8368
VC1	782,80	0,8385	0,8480	0,8385	0,8412
VC2	759,28	0,8355	0,8453	0,8355	0,8383
SC1	1701,90	0,8355	0,8450	0,8355	0,8382
SC2	1640,81	0,8385	0,8488	0,8385	0,8413
SC3	906,52	0,8380	0,8480	0,8380	0,8407
SCS	386,28	0,8397	0,8479	0,8397	0,8422
XGBC	88,38	0,8277	0,8427	0,8277	0,8317

Tabela 5.7: Resultados do teste 7

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
MPC	28,33	0,7958	0,8016	0,7958	0,7979
ABC	1,55	0,7194	0,7506	0,7194	0,7266
DTC	1,29	0,7775	0,7840	0,7775	0,7797
GNB	1,76	0,5797	0,6072	0,5797	0,5686
BNB	0,71	0,7547	0,7754	0,7547	0,7573
MNB	0,18	0,7619	0,7685	0,7619	0,7632
RFC	0,50	0,8227	0,8296	0,8227	0,8248
ETC	0,90	0,8266	0,8323	0,8266	0,8282
LR	2,66	0,8166	0,8249	0,8166	0,8189
SVC	23,64	0,8280	0,8384	0,8280	0,8310
VC1	216,46	0,8322	0,8389	0,8322	0,8341
VC2	216,74	0,8322	0,8392	0,8322	0,8343
SC1	494,76	0,8330	0,8404	0,8330	0,8349
SC2	458,07	0,8330	0,8405	0,8330	0,8351
SC3	271,42	0,8352	0,8415	0,8352	0,8369
SCS	120,12	0,8363	0,8424	0,8363	0,8382
XGBC	84,91	0,8277	0,8406	0,8277	0,8311

Tabela 5.8: Resultados do teste 8

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
MPC	52,34	0,8257	0,8331	0,8257	0,8280
ABC	3,60	0,7351	0,7669	0,7351	0,7443
DTC	5,25	0,7977	0,8049	0,7977	0,8001
GNB	4,91	0,5907	0,6134	0,5907	0,5782
BNB	1,92	0,7648	0,7811	0,7648	0,7680
MNB	0,52	0,7707	0,7709	0,7707	0,7693
RFC	1,04	0,8382	0,8495	0,8382	0,8414
ETC	2,22	0,8421	0,8499	0,8421	0,8445
LR	5,68	0,8238	0,8304	0,8238	0,8257
SVC	111,13	0,8416	0,8526	0,8416	0,8448
VC1	1028,21	0,8490	0,8561	0,8490	0,8511
VC2	1043,82	0,8469	0,8543	0,8469	0,8490
SC1	2260,70	0,8477	0,8556	0,8477	0,8500
SC2	2186,15	0,8489	0,8568	0,8489	0,8512
SC3	1215,83	0,8484	0,8571	0,8484	0,8509
SCS	545,94	0,8483	0,8552	0,8483	0,8504
XGBC	108,57	0,8353	0,8482	0,8353	0,8390

Tabela 5.9: Resultados do teste 9

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
MPC	52.85	0.8435	0.8444	0.8435	0.8436
ABC	3.90	0.7484	0.7647	0.7484	0.7525
DTC	5.30	0.8237	0.8239	0.8237	0.8236
GNB	5.60	0.6118	0.6264	0.6118	0.5981
BNB	2.15	0.7850	0.7913	0.7850	0.7857
MNB	0.55	0.7973	0.7984	0.7973	0.7972
RFC	1.11	0.8620	0.8629	0.8620	0.8622
ETC	2.30	0.8667	0.8674	0.8667	0.8668
LR	5.70	0.8441	0.8462	0.8441	0.8447
SVC	112.11	0.8660	0.8691	0.8660	0.8669
VC1	1043.47	0.8717	0.8732	0.8717	0.8721
VC2	1065.00	0.8683	0.8698	0.8683	0.8687
SC1	2300.11	0.8677	0.8692	0.8677	0.8679
SC2	2223.87	0.8704	0.8719	0.8704	0.8707
SC3	1238.71	0.8712	0.8724	0.8712	0.8713
SCS	551.83	0.8712	0.8723	0.8712	0.8716
XGBC	101.73	0.8644	0.8682	0.8644	0.8654

Tabela 5.10: Resultados do teste 10

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
MPC	60,43	0,8473	0,8479	0,8473	0,8474
ABC	3,85	0,7529	0,7757	0,7529	0,7589
DTC	5,64	0,8287	0,8293	0,8287	0,8288
GNB	8,61	0,5840	0,5986	0,5840	0,5714
BNB	3,52	0,7879	0,7924	0,7879	0,7880
MNB	1,04	0,8000	0,8011	0,8000	0,8000
RFC	1,43	0,8705	0,8710	0,8705	0,8705
ETC	2,70	0,8718	0,8725	0,8718	0,8719
LR	6,13	0,8479	0,8496	0,8479	0,8483
SVC	118,20	0,8721	0,8753	0,8721	0,8729
VC1	1117,00	0,8775	0,8785	0,8775	0,8777
VC2	1118,23	0,8739	0,8749	0,8739	0,8741
SC1	2446,83	0,8745	0,8758	0,8745	0,8747
SC2	2358,39	0,8767	0,8778	0,8767	0,8770
SC3	1306,64	0,8767	0,8775	0,8767	0,8767
SCS	607,47	0,8776	0,8787	0,8776	0,8779
XGBC	92,45	0,8634	0,8666	0,8634	0,8643

Tabela 5.11: Resultados do teste 11

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
MPC	100,30	0,8207	0,8272	0,8207	0,8228
ABC	4,10	0,7280	0,7585	0,7280	0,7363
DTC	6,67	0,8030	0,8120	0,8030	0,8060
GNB	25,82	0,4891	0,5204	0,4891	0,4699
BNB	10,23	0,7648	0,7817	0,7648	0,7681
MNB	3,30	0,7745	0,7754	0,7745	0,7728
RFC	2,05	0,8410	0,8509	0,8410	0,8439
ETC	4,92	0,8405	0,8478	0,8405	0,8428
LR	9,60	0,8254	0,8321	0,8254	0,8274
SVC	132,72	0,8415	0,8520	0,8415	0,8445
VC1	1288,96	0,8479	0,8549	0,8479	0,8500
VC2	1304,79	0,8465	0,8538	0,8465	0,8486
SC1	2893,53	0,8463	0,8543	0,8463	0,8486
SC2	2738,77	0,8485	0,8567	0,8485	0,8508
SC3	1551,49	0,8507	0,8586	0,8507	0,8529
SCS	667,50	0,8478	0,8547	0,8478	0,8500
XGBC	93,29	0,8390	0,8531	0,8390	0,8430

Tabela 5.12: Resultados do teste 12

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
MPC	103,25	0,8542	0,8546	0.8542	0,8543
ABC	1,94	0,7414	0,7534	0.7414	0,7455
DTC	5,10	0,8316	0,8328	0.8316	0,8318
GNB	25,27	0,4571	0,5147	0.4571	0,4363
BNB	35,33	0,7816	0,7833	0.7816	0,7806
MNB	88,99	0,7793	0,7849	0.7793	0,7788
RFC	2,36	0,8749	0,8753	0.8749	0,8749
ETC	5,03	0,8756	0,8762	0.8756	0,8757
LR	19,49	0,8503	0,8519	0.8503	0,8504
SVC	123,14	0,8561	0,8613	0.8561	0,8575
VC1	1119,05	0,8731	0,8743	0.8731	0,8733
VC2	1114,96	0,8718	0,8731	0.8718	0,8720
SC1	2643,97	0,8742	0,8759	0.8742	0,8744
SC2	2394,86	0,8733	0,8752	0.8733	0,8737
SC3	1353,87	0,8799	0,8813	0.8799	0,8801
SCS	663,06	0,8819	0,8826	0.8819	0,8821
XGBC	5,36	0,8635	0,8664	0.8635	0,8643

Tabela 5.13: Resultados do teste 13

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
MPC	115,25	0,8542	0,8546	0,8542	0,8543
ABC	2,14	0,7414	0,7534	0,7414	0,7455
DTC	5,14	0,8316	0,8328	0,8316	0,8318
GNB	28,96	0,4571	0,5147	0,4571	0,4363
BNB	41,02	0,7816	0,7833	0,7816	0,7806
MNB	94,51	0,7793	0,7849	0,7793	0,7788
RFC	2,32	0,8749	0,8753	0,8749	0,8749
ETC	4,91	0,8756	0,8762	0,8756	0,8757
LR	19,01	0,8503	0,8519	0,8503	0,8504
SVC	116,97	0,8561	0,8613	0,8561	0,8575
VC1	1109,83	0,8729	0,8742	0,8729	0,8732
VC2	1079,42	0,8720	0,8732	0,8720	0,8721
SC1	2619,53	0,8742	0,8759	0,8742	0,8744
SC2	2349,57	0,8733	0,8752	0,8733	0,8737
SC3	1324,03	0,8799	0,8813	0,8799	0,8801
SCS	652,71	0,8819	0,8826	0,8819	0,8821
XGBC	94,63	0,8635	0,8664	0,8635	0,8643

Tabela 5.14: Resultados do teste 14

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVO_MPC	29,65	0,8428	0,8443	0,8428	0,8432
OVO_ABC	5,75	0,8175	0,8275	0,8175	0,8197
OVO_DTC	2,50	0,8329	0,8342	0,8329	0,8334
OVO_GNB	42,01	0,5893	0,6029	0,5893	0,5769
OVO_BNB	23,14	0,7879	0,7924	0,7879	0,7880
OVO_MNB	5,44	0,8000	0,8011	0,8000	0,8000
OVO_RFC	3,13	0,8701	0,8710	0,8701	0,8704
OVO_ETC	8,97	0,8700	0,8710	0,8700	0,8703
OVO_LR	0,36	0,8449	0,8473	0,8449	0,8456
OVO_SVC	139,47	0,8713	0,8747	0,8713	0,8723
OVO_VC1	1253,01	0,8738	0,8752	0,8738	0,8742
OVO_VC2	1265,63	0,8716	0,8731	0,8716	0,8720
OVO_SC1	2779,12	0,8728	0,8743	0,8728	0,8732
OVO_SC2	2684,41	0,8736	0,8750	0,8736	0,8739
OVO_SC3	1449,38	0,8783	0,8795	0,8783	0,8786
OVO_SCS	210,74	0,8758	0,8769	0,8758	0,8761
OVO_XGBC	9,46	0,8657	0,8693	0,8657	0,8667

Tabela 5.15: Resultados do teste 15

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVR_MPC	567,21	0,8466	0,8457	0,8466	0,8461
OVR_ABC	621,12	0,7990	0,8072	0,7990	0,8010
OVR_DTC	14,91	0,8028	0,8231	0,8028	0,8067
OVR_GNB	15,90	0,3778	0,5782	0,3778	0,3654
OVR_BNB	9,40	0,7963	0,7983	0,7963	0,7959
OVR_MNB	4,12	0,8001	0,8001	0,8001	0,7995
OVR_RFC	3,37	0,8723	0,8723	0,8723	0,8722
OVR_ETC	8,94	0,8732	0,8732	0,8732	0,8731
OVR_LR	0,96	0,8429	0,8443	0,8429	0,8431
OVR_VC1	27,05	0,8728	0,8731	0,8728	0,8727
OVR_VC2	24,52	0,8689	0,8695	0,8689	0,8689
OVR_SC1	1297,55	0,8696	0,8702	0,8696	0,8695
OVR_SC2	64,36	0,8710	0,8715	0,8710	0,8709
OVR_SC3	618,98	0,8782	0,8782	0,8782	0,8781
OVR_SCS	68,61	0,8736	0,8746	0,8736	0,8738
OVR_XGBC	21,52	0,8628	0,8657	0,8628	0,8636

Tabela 5.16: Resultados do teste 16

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVR_MPC	7247,35	0,8660	0,8659	0,8660	0,8657
OVR_ABC	769,06	0,8025	0,8115	0,8025	0,8041
OVR_DTC	5,58	0,8207	0,8360	0,8207	0,8236
OVR_GNB	41,20	0,3972	0,5871	0,3972	0,3852
OVR_BNB	67,00	0,7914	0,7915	0,7914	0,7900
OVR_MNB	56,45	0,7879	0,7929	0,7879	0,7870
OVR_RFC	5,17	0,8803	0,8803	0,8803	0,8801
OVR_ETC	17,40	0,8818	0,8817	0,8818	0,8816
OVR_LR	423,81	0,8520	0,8539	0,8520	0,8520
OVR_VC1	56,94	0,8788	0,8795	0,8788	0,8788
OVR_VC2	60,03	0,8766	0,8777	0,8766	0,8766
OVR_SC1	8750,04	0,8786	0,8791	0,8786	0,8784
OVR_SC2	189,16	0,8763	0,8769	0,8763	0,8760
OVR_SC3	7335,55	0,8837	0,8838	0,8837	0,8835
OVR_SCS	113,01	0,8825	0,8832	0,8825	0,8827
OVR_XGBC	6,77	0,8691	0,8713	0,8691	0,8695

Tabela 5.17: Resultados do teste 17

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
MPC	336,55	0,8514	0,8515	0,8514	0,8512
ABC	2,17	0,7481	0,7570	0,7481	0,7508
DTC	5,21	0,8338	0,8346	0,8338	0,8339
GNB	25,28	0,4554	0,5127	0,4554	0,4339
BNB	138,31	0,7811	0,7830	0,7811	0,7801
MNB	91,30	0,7787	0,7842	0,7787	0,7782
RFC	2,41	0,8718	0,8720	0,8718	0,8717
ETC	5,48	0,8758	0,8761	0,8758	0,8757
LR	16,65	0,8499	0,8514	0,8499	0,8500
SVC	121,21	0,8565	0,8612	0,8565	0,8577
VC1	1141,25	0,8731	0,8743	0,8731	0,8733
VC2	1126,93	0,8720	0,8731	0,8720	0,8721
SC1	3012,82	0,8734	0,8747	0,8734	0,8734
SC2	2530,14	0,8733	0,8748	0,8733	0,8735
SC3	1597,83	0,8780	0,8785	0,8780	0,8778
SCS	670,33	0,8802	0,8811	0,8802	0,8805
XGBC	103,15	0,8661	0,8689	0,8661	0,8668

Tabela 5.18: Resultados do teste 18

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVR_MPC	88,47	0,8675	0,8676	0,8675	0,8674
OVR_ABC	36,45	0,8025	0,8115	0,8025	0,8041
OVR_DTC	6,07	0,8224	0,8376	0,8224	0,8251
OVR_GNB	47,71	0,3972	0,5871	0,3972	0,3852
OVR_BNB	38,39	0,7915	0,7916	0,7915	0,7902
OVR_MNB	51,19	0,7860	0,7907	0,7860	0,7850
OVR_RFC	7,23	0,8785	0,8786	0,8785	0,8783
OVR_ETC	24,03	0,8788	0,8788	0,8788	0,8786
OVR_LR	1,54	0,8522	0,8539	0,8522	0,8522
SVC	226,50	0,8633	0,8634	0,8633	0,8628
OVR_VC1	2252,64	0,8766	0,8778	0,8766	0,8767
OVR_VC2	2553,45	0,8756	0,8767	0,8756	0,8756
OVR_SC1	5333,72	0,8766	0,8775	0,8766	0,8765
OVR_SC2	5078,23	0,8758	0,8766	0,8758	0,8756
OVR_SC3	2905,77	0,8828	0,8828	0,8828	0,8824
OVR_SCS	962,05	0,8823	0,8833	0,8823	0,8825
OVR_XGBC	11,37	0,8691	0,8713	0,8691	0,8695

Tabela 5.19: Resultados do teste 19

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVR_MPC	41,40	0,8519	0,8518	0,8519	0,8513
OVR_ABC	3,38	0,7468	0,7659	0,7468	0,7494
OVR_DTC	4,62	0,8074	0,8121	0,8074	0,8076
OVR_GNB	10,09	0,4713	0,6756	0,4713	0,4603
OVR_BNB	7,59	0,7581	0,7605	0,7581	0,7540
OVR_MNB	8,54	0,7794	0,7811	0,7794	0,7776
OVR_RFC	4,34	0,8500	0,8498	0,8500	0,8492
OVR_ETC	9,96	0,8468	0,8467	0,8468	0,8462
OVR_LR	1,01	0,8515	0,8518	0,8515	0,8511
SVC	200,86	0,8450	0,8444	0,8450	0,8441
OVR_VC1	2265,59	0,8586	0,8596	0,8586	0,8583
OVR_VC2	2647,16	0,8582	0,8591	0,8582	0,8579
OVR_SC1	5204,59	0,8573	0,8577	0,8573	0,8567
OVR_SC2	5125,55	0,8563	0,8571	0,8563	0,8558
OVR_SC3	2908,36	0,8557	0,8557	0,8557	0,8550
OVR_SCS	823,92	0,8585	0,8596	0,8585	0,8582
OVR_XGBC	9,80	0,8382	0,8408	0,8382	0,8382

Tabela 5.20: Resultados do teste 20

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVR_MPC	26,52	0,6756	0,6744	0,6756	0,6734
OVR_ABC	3,65	0,5923	0,5879	0,5923	0,5874
OVR_DTC	2,68	0,6453	0,6890	0,6453	0,6535
OVR_GNB	0,59	0,2652	0,3835	0,2652	0,2009
OVR_BNB	0,53	0,4222	0,4317	0,4222	0,4117
OVR_MNB	0,47	0,5530	0,5710	0,5530	0,5533
OVR_RFC	3,09	0,7609	0,7626	0,7609	0,7610
OVR_ETC	8,78	0,7680	0,7705	0,7680	0,7683
OVR_LR	0,91	0,6025	0,5972	0,6025	0,5938
SVC	97,38	0,6004	0,5951	0,6004	0,5758
OVR_VC1	1841,26	0,7338	0,7338	0,7338	0,7313
OVR_VC2	1999,75	0,7325	0,7314	0,7325	0,7299
OVR_SC1	3985,67	0,7157	0,7133	0,7157	0,7117
OVR_SC2	3942,02	0,7347	0,7330	0,7347	0,7315
OVR_SC3	2138,23	0,7510	0,7492	0,7510	0,7486
OVR_SCS	438,70	0,7678	0,7695	0,7678	0,7680
OVR_XGBC	10,17	0,7502	0,7508	0,7502	0,7503

Tabela 5.21: Resultados do teste 21

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVR_MPC	182,00	0,8635	0,8641	0,8635	0,8637
OVR_ABC	22,12	0,8137	0,8154	0,8137	0,8132
OVR_DTC	30,96	0,7843	0,8103	0,7843	0,7891
OVR_GNB	9,91	0,3430	0,5602	0,3430	0,3137
OVR_BNB	8,10	0,7081	0,7204	0,7081	0,7074
OVR_MNB	8,35	0,7060	0,7333	0,7060	0,7085
OVR_RFC	8,17	0,8758	0,8774	0,8758	0,8763
OVR_ETC	20,34	0,8778	0,8788	0,8778	0,8782
OVR_LR	3,79	0,8440	0,8452	0,8440	0,8439
SVC	1476,11	0,8498	0,8501	0,8498	0,8490
OVR_VC1	15857,71	0,8715	0,8727	0,8715	0,8716
OVR_VC2	18837,84	0,8712	0,8724	0,8712	0,8714
OVR_SC1	36465,45	0,8721	0,8732	0,8721	0,8722
OVR_SC2	36187,95	0,8733	0,8743	0,8733	0,8734
OVR_SC3	20555,44	0,8801	0,8806	0,8801	0,8801
OVR_SCS	6093,69	0,8794	0,8809	0,8794	0,8798
OVR_XGBC	91,03	0,8770	0,8787	0,8770	0,8774

Tabela 5.22: Resultados do teste 22

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVR_MPC	24,69	0,8597	0,8600	0,8597	0,8597
OVR_ABC	27,56	0,8002	0,8016	0,8002	0,7990
OVR_DTC	8,39	0,8042	0,8206	0,8042	0,8067
OVR_GNB	1,69	0,4218	0,6441	0,4218	0,4005
OVR_BNB	1,10	0,7717	0,7713	0,7717	0,7699
OVR_MNB	0,80	0,7729	0,7761	0,7729	0,7724
OVR_RFC	3,69	0,8685	0,8685	0,8685	0,8684
OVR_ETC	7,44	0,8759	0,8759	0,8759	0,8758
OVR_LR	1,31	0,8351	0,8356	0,8351	0,8348
SVC	683,22	0,8720	0,8721	0,8720	0,8717
OVR_VC1	2635,88	0,8734	0,8743	0,8734	0,8735
OVR_VC2	2779,80	0,8715	0,8721	0,8715	0,8715
OVR_SC1	6175,62	0,8713	0,8716	0,8713	0,8711
OVR_SC2	6103,70	0,8731	0,8738	0,8731	0,8731
OVR_SC3	3507,26	0,8765	0,8765	0,8765	0,8763
OVR_SCS	479,22	0,8774	0,8783	0,8774	0,8776
OVR_XGBC	18,93	0,8668	0,8675	0,8668	0,8668

Tabela 5.23: Resultados do teste 23

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVR_MPC	48,73	0,8460	0,8457	0,8460	0,8454
OVR_ABC	165,13	0,7497	0,7667	0,7497	0,7518
OVR_DTC	20,40	0,8018	0,8148	0,8018	0,8034
OVR_GNB	9,80	0,4933	0,6759	0,4933	0,4841
OVR_BNB	6,00	0,7581	0,7605	0,7581	0,7540
OVR_MNB	2,57	0,7924	0,7912	0,7924	0,7910
OVR_RFC	4,51	0,8494	0,8490	0,8494	0,8486
OVR_ETC	10,88	0,8531	0,8526	0,8531	0,8524
OVR_LR	0,97	0,8376	0,8377	0,8376	0,8371
SVC	7833,78	0,8433	0,8446	0,8433	0,8436
OVR_VC1	18835,95	0,8577	0,8578	0,8577	0,8572
OVR_VC2	19633,37	0,8571	0,8573	0,8571	0,8567
OVR_SC1	46572,14	0,8568	0,8566	0,8568	0,8561
OVR_SC2	46313,63	0,8558	0,8559	0,8558	0,8552
OVR_SC3	27551,69	0,8564	0,8560	0,8564	0,8556
OVR_SCS	1217,09	0,8587	0,8597	0,8587	0,8585
OVR_XGBC	52,21	0,8367	0,8375	0,8367	0,8361

Tabela 5.24: Resultados do teste 24

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVR_MPC	29,24	0,6692	0,6675	0,6692	0,6672
OVR_ABC	11,55	0,6300	0,6259	0,6300	0,6273
OVR_DTC	3,62	0,6275	0,6834	0,6275	0,6381
OVR_GNB	0,54	0,3196	0,5090	0,3196	0,2990
OVR_BNB	0,67	0,4222	0,4317	0,4222	0,4117
OVR_MNB	1,44	0,5405	0,5446	0,5405	0,5386
OVR_RFC	5,64	0,7589	0,7603	0,7589	0,7590
OVR_ETC	10,75	0,7698	0,7727	0,7698	0,7700
OVR_LR	1,29	0,6107	0,6060	0,6107	0,6068
SVC	81,48	0,6729	0,6685	0,6729	0,6654
OVR_VC1	1455,62	0,7340	0,7321	0,7340	0,7322
OVR_VC2	2108,63	0,7313	0,7289	0,7313	0,7291
OVR_SC1	3710,54	0,7210	0,7182	0,7210	0,7188
OVR_SC2	3652,07	0,7339	0,7314	0,7339	0,7316
OVR_SC3	2239,39	0,7501	0,7479	0,7501	0,7479
OVR_SCS	453,53	0,7736	0,7745	0,7736	0,7734
OVR_XGBC	26,40	0,7540	0,7544	0,7540	0,7540

Tabela 5.25: Resultados do teste 25

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVR_MPC	192,80	0,8576	0,8580	0,8576	0,8577
OVR_ABC	215,51	0,8126	0,8138	0,8126	0,8123
OVR_DTC	120,47	0,7771	0,8051	0,7771	0,7824
OVR_GNB	12,41	0,3576	0,5638	0,3576	0,3330
OVR_BNB	6,23	0,7081	0,7204	0,7081	0,7074
OVR_MNB	2,87	0,7675	0,7780	0,7675	0,7705
OVR_RFC	8,85	0,8677	0,8686	0,8677	0,8680
OVR_ETC	20,81	0,8750	0,8759	0,8750	0,8753
OVR_LR	3,48	0,8436	0,8438	0,8436	0,8433
SVC	6788,65	0,8723	0,8723	0,8723	0,8720
OVR_VC1	31358,64	0,8732	0,8740	0,8732	0,8733
OVR_VC2	32197,46	0,8712	0,8720	0,8712	0,8713
OVR_SC1	70928,25	0,8720	0,8727	0,8720	0,8720
OVR_SC2	70359,86	0,8731	0,8738	0,8731	0,8731
OVR_SC3	39329,07	0,8764	0,8767	0,8764	0,8763
OVR_SCS	7244,08	0,8793	0,8803	0,8793	0,8795
OVR_XGBC	422,26	0,8711	0,8724	0,8711	0,8714

Tabela 5.26: Resultados do teste 26

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVR_RFC	22,23	0,8925	0,8927	0,8925	0,8924
OVR_ETC	74,82	0,8918	0,8920	0,8918	0,8918
OVR_LR	45,70	0,8554	0,8589	0,8554	0,8551
OVR_SC1	142,76	0,8949	0,8951	0,8949	0,8948
OVR_SC2	97,06	0,8918	0,8920	0,8918	0,8918
OVR_SCS	862,66	0,8963	0,8965	0,8963	0,8963
OVR_XGBC	141,68	0,8644	0,8681	0,8644	0,8640

Tabela 5.27: Resultados do teste 27

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVR_RFC	21.65	0.8852	0.8853	0.8852	0.8851
OVR_ETC	77.87	0.8864	0.8865	0.8864	0.8863
OVR_LR	3.49	0.8549	0.8567	0.8549	0.8546
OVR_SC1	103.01	0.8874	0.8875	0.8874	0.8873
OVR_SC2	99.52	0.8852	0.8855	0.8852	0.8853
OVR_SCS	652.57	0.8890	0.8891	0.8890	0.8889
OVR_XGBC	547.93	0.8668	0.8691	0.8668	0.8665

Tabela 5.28: Resultados do teste 28

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVR_RFC	221,31	0,9206	0,9190	0,9206	0,9189
OVR_ETC	577,55	0,9210	0,9194	0,9210	0,9195
OVR_LR	57,01	0,8951	0,8926	0,8951	0,8913
OVR_SC1	855,89	0,9226	0,9211	0,9226	0,9210
OVR_SC2	798,87	0,9223	0,9209	0,9223	0,9212
OVR_SCS	3286,86	0,9233	0,9219	0,9233	0,9221
OVR_XGBC	56,30	0,9060	0,9039	0,9060	0,9028

Tabela 5.29: Resultados do teste 29

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVR_RFC	164,73	0,9165	0,9147	0,9165	0,9145
OVR_ETC	531,24	0,9173	0,9157	0,9173	0,9154
OVR_LR	12,81	0,8912	0,8885	0,8912	0,8872
OVR_SC1	708,79	0,9180	0,9164	0,9180	0,9160
OVR_SC2	695,98	0,9192	0,9177	0,9192	0,9179
OVR_SCS	3037,48	0,9196	0,9180	0,9196	0,9182
OVR_XGBC	307,92	0,9055	0,9034	0,9055	0,9025

Tabela 5.30: Resultados do teste 30

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVO_MPC	117,82	0,8516	0,8529	0,8516	0,8520
OVO_ABC	14,23	0,8385	0,8404	0,8385	0,8388
OVO_DTC	4,62	0,8328	0,8356	0,8328	0,8338
OVO_GNB	108,24	0,5296	0,5902	0,5296	0,5165
OVO_BNB	52,75	0,8158	0,8171	0,8158	0,8156
OVO_MNB	16,15	0,8109	0,8122	0,8109	0,8110
OVO_RFC	26,69	0,8752	0,8760	0,8752	0,8754
OVO_ETC	43,81	0,8808	0,8815	0,8808	0,8810
OVO_LR	1,83	0,8542	0,8552	0,8542	0,8545
OVO_SVC	227,60	0,8792	0,8806	0,8792	0,8796
OVO_VC1	2230,14	0,8826	0,8836	0,8826	0,8829
OVO_VC2	927,05	0,8801	0,8811	0,8801	0,8804
OVO_SC1	3770,97	0,8811	0,8819	0,8811	0,8813
OVO_SC2	3464,25	0,8814	0,8824	0,8814	0,8817
OVO_SC3	1347,61	0,8861	0,8868	0,8861	0,8862
OVO_SCS	132,67	0,8880	0,8888	0,8880	0,8882
OVO_XGBC	6,43	0,8709	0,8726	0,8709	0,8714

Tabela 5.31: Resultados do teste 31

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVR_RFC	4.17	0.8855	0.8857	0.8855	0.8852
OVR_ETC	11.13	0.8910	0.8912	0.8910	0.8908
OVR_LR	1.41	0.8591	0.8603	0.8591	0.8591
OVR_SC1	16.72	0.8898	0.8901	0.8898	0.8895
OVR_SC2	15.30	0.8874	0.8874	0.8874	0.8869
OVR_SCS	58.27	0.8890	0.8898	0.8890	0.8891
OVR_XGBC	4.79	0.8789	0.8798	0.8789	0.8790

Tabela 5.32: Resultados do teste 32

Algoritmo	**Tempo de execução**	**Acurácia**	**Precisão**	**Recall**	**F1_Score**
OVR_RFC	4.40	0.8855	0.8859	0.8855	0.8854
OVR_ETC	11.11	0.8895	0.8896	0.8895	0.8893
OVR_LR	1.45	0.8592	0.8605	0.8592	0.8593
OVR_SC1	16.97	0.8890	0.8893	0.8890	0.8888
OVR_SC2	15.51	0.8872	0.8873	0.8872	0.8868
OVR_SCS	58.78	0.8883	0.8894	0.8883	0.8885
OVR_XGBC	4.92	0.8732	0.8745	0.8732	0.8735

One Versus Rest (OVR).

80% dos dados foram utilizados para treinamento e 20% para teste, conforme descrição dos resultados apresentada a seguir:

- **Teste 1**: Esse primeiro teste foi executado com os textos sem pré-processamento e com 3000 registros por classe de problema para identificar qual acurácia os algoritmos iriam atingir e qual a melhoria que seria obtida com a aplicação das técnicas descritas nessa pesquisa. Os resultados desse primeiro teste estão na tabela 5.1.
- **Teste 2**: Esse segundo teste foi executado removendo-se apenas as *stop words* da biblioteca *NLTK* e com 3000 registros por classe de problema. Os resultados desse teste estão na tabela 5.2.
- **Teste 3**: Esse teste foi executado removendo-se apenas as *stop words* da biblioteca *NLTK* e com 7000 registros por classe de problema. Os resultados desse teste estão na tabela 5.3.
- **Teste 4**: Esse teste foi executado removendo-se apenas as *stop words* da biblioteca *NLTK*, removendo as 6 palavras mais frequentes da base de dados e com 3000 registros por classe de problema. Os resultados desse teste estão na tabela 5.4.
- **Teste 5**: Esse teste foi executado removendo-se apenas as *stop words* da biblioteca *NLTK*, utilizando apenas as 700 palavras mais frequentes da base de dados, removendo todas as restantes e com 3000 registros por classe de problema. Os resultados desse teste estão na tabela 5.5.
- **Teste 6**: Esse teste foi executado removendo-se apenas as *stop words* da biblioteca *NLTK*, utilizando apenas as 700 palavras mais frequentes da base de dados, removendo todas as restantes e com 7000 registros por classe de problema. Os resultados desse teste estão na tabela 5.6.
- **Teste 7**: Esse teste foi executado removendo-se apenas as *stop words* da biblioteca *NLTK*, utilizando apenas as 4000 palavras mais frequentes da base de dados, removendo todas as restantes e com 3000 registros por classe de problema. Os resultados desse teste estão na tabela 5.7.
- **Teste 8**: Esse teste foi executado removendo-se apenas as *stop words* da biblioteca *NLTK*, utilizando apenas as 4000 palavras mais frequentes da base de dados, removendo todas as restantes e com 7000 registros por classe de problema. Os resultados desse teste estão na tabela 5.8.

- **Teste 9**: Esse teste foi executado removendo-se apenas as *stop words* da biblioteca *NLTK*, utilizando apenas as 5000 palavras mais frequentes da base de dados, removendo todas as restantes e com 7000 registros por classe de problema. Os resultados desse teste estão na tabela 5.9.

- **Teste 10**: Esse teste foi executado removendo-se apenas as *stop words* da biblioteca *NLTK*, utilizando apenas as 10000 palavras mais frequentes da base de dados, removendo todas as restantes e com 7000 registros por classe de problema. Os resultados desse teste estão na tabela 5.10.

- **Teste 11**: Esse teste foi executado removendo-se as *stop words* de forma personalizada, sem remover palavras menos frequentes da base de dados, com 7000 registros por classe de problema, utilizando o TF-IDF como método de vetorização. Os resultados desse teste estão na tabela 5.11.

- **Teste 12**: Esse teste foi executado removendo-se as *stop words* de forma personalizada, sem remover palavras menos frequentes da base de dados, com 7000 registros por classe de problema, utilizando o *Count Vectorizer* como método de vetorização. Os resultados desse teste estão na tabela 5.12.

- **Teste 13**: Esse teste foi executado removendo-se as *stop words* de forma personalizada, sem remover palavras menos frequentes da base de dados, com 7000 registros por classe de problema, utilizando o *Count Vectorizer* como método de vetorização e utilizando a técnica de Stemização. Os resultados desse teste estão na tabela 5.13.

- **Teste 14**: Esse teste foi executado removendo-se as *stop words* de forma personalizada, utilizando apenas as 10000 palavras mais frequentes da base de dados, removendo todas as restantes, com 7000 registros por classe de problema, utilizando o *TF-IDF* como método de vetorização e utilizando a técnica *One versus One*. Os resultados desse teste estão na tabela 5.14.

- **Teste 15**: Esse teste foi executado removendo-se as *stop words* de forma personalizada, utilizando apenas as 10000 palavras mais frequentes da base de dados, removendo todas as restantes, com 7000 registros por classe de problema, utilizando o *TF-IDF* como método de vetorização e utilizando a técnica *One versus Rest*. Os resultados desse teste estão na tabela 5.15.

- **Teste 16**: Esse teste foi executado removendo-se as *stop words* de forma personalizada, sem remover palavras mais frequentes, com 7000 registros por classe de

problema, utilizando o *Count Vectorizer* como método de vetorização e utilizando a técnica *One versus Rest*. Os resultados desse teste estão na tabela 5.16.

- **Teste 17**: Esse teste foi executado removendo-se as *stop words* de forma personalizada, sem remover palavras mais frequentes, com 7000 registros por classe de problema, com a transformação dos textos para forma numérica utilizando um corpus construído com a biblioteca *gensim* e a técnica *bag of words*. Além disso, foi utilizada a técnica de criação de novas *fetures* a partir da similaridade entre pequenos conjuntos de palavras escolhidos para representar cada problema, utilizando o *Word Embedding Similarity Index* da biblioteca *gensim*. Adicionalmente, foi utilizada a técnica *One versus Rest* para tentar elevar a precisão do modelo. Os resultados desse teste estão na tabela 5.17.

- **Testes 18 ao 25**: Esses testes foram executados removendo-se as *stop words* de forma personalizada, sem remover palavras mais frequentes, com 7000 registros por classe de problema, utilizando TF-IDF ou *Count Vectorizer* como método de vetorização, variando a escolha do que seria o elemento mais básico: palavra, caractere e combinações geradas por palavras ou caracteres. Todos esses testes foram executados utilizando a técnica *One versus Rest*, que, de acordo com os testes anteriores, eleva os valores de precisão dos resultados. O teste na tabela 5.18 foi realizado utilizando *Count Vectorizer*, com palavra como elemento mais básico. O teste na tabela 5.19 foi realizado utilizando *Count Vectorizer*, porém utilizando como elemento mais básico a combinação de palavras. O teste na tabela 5.20 foi realizado utilizando *Count Vectorizer*, com caractere como elemento mais básico. O teste na tabela 5.21 foi realizado utilizando *Count Vectorizer*, porém utilizando como elemento mais básico a combinação de caracteres. O teste na tabela 5.22 foi realizado utilizando TF-IDF, com palavra como elemento mais básico. O teste na tabela 5.23 foi realizado utilizando TF-IDF, porém utilizando como elemento mais básico a combinação de palavras. O teste na tabela 5.24 foi realizado utilizando TF-IDF, com caractere como elemento mais básico. O teste na tabela 5.25 foi realizado utilizando TF-IDF, porém utilizando como elemento mais básico a combinação de caracteres.

- **Testes 26 a 29**: Além dos testes numerados de 1 a 25, que foram realizados com a mesma base de dados com os campos "RELATOCLIENTE"e "PROBLEMA", foram realizados os testes numerados de 26 a 29 com a base de dados com os campos "RELATOCLIENTE"e "MOTIVO3". Esses testes foram executados removendo-se as *stop words* de forma personalizada, sem remover palavras mais frequentes, uti-

lizando a técnica *One versus Rest*, que, de acordo com os testes anteriores, eleva os valores de precisão dos resultados. A diferença entre os testes está no número de registros utilizados em cada teste e no tipo de transformação para a forma numérica: TF-IDF ou *Count Vectorizer*. Esses testes não foram realizados com todos os algoritmos, pois alguns demoravam muito tempo para executar, se tornando inviáveis para um uso prático. Dessa forma, escolheu-se os algoritmos que obtiveram melhores performances nos testes anteriores, levando-se em conta também o tempo de execução.

Os testes nas tabelas 5.26 e 5.27 foram feitos com a base de dados reduzida para que ficasse balanceada, com mesmo número de registros para cada classe de problema. Dessa forma, esses dois testes foram realizados com uma base com 36400 registros para cada uma das três classes de problemas descritos na seção 4.1 do capítulo 4. O teste na tabela 5.26 foi feito utilizando *Count Vectorizer* e o teste na tabela 5.27 foi feito utilizando TF-IDF.

Os testes nas tabelas 5.28 e 5.29 foram feitos com a base de dados completa e não balanceada, com 192825 registros para classe "Sem Sincronismo", com 48775 registros para classe "Parâmetros Ruins"e com 36456 registros para classe "Exige Técnico". O teste na tabela 5.28 foi feito utilizando *Count Vectorizer* e o teste na tabela 5.29 foi feito utilizando TF-IDF.

- **Teste 30**: Os testes na tabela 5.30 foram feitos utilizando a base de dados com seis classes de problemas, com os campos "RELATOCLIENTE"e "PROBLEMA". Esses testes se diferenciam dos demais, pois foram realizados seguindo todo o pré-processamento personalizado descrito no capítulo 4 com atenção para remoção de *stop words* de forma personalizada, remoção de nomes próprios e aplicação do método descrito para correção de pequenos erros ortográficos, gerando um aumento de acurácia.

- **Teste 31**: Os testes na tabela 5.31 também foram realizados com a base de dados com seis classes de problemas, com os campos "RELATOCLIENTE"e "PROBLEMA", seguindo todo o pré-processamento personalizado descrito na metodologia, utilizando a técnica *One versus Rest* com *Count Vectorizer*, porém considerando apenas os chamados cujas descrições têm mais de 5 palavras, mantendo os termos identificadores das palavras removidas, tais como: "emailremoved", "nameremoved"e "urlremoved".

- **Teste 32**: Os testes na tabela 5.32 foram realizados nas mesmas condições dos testes

da tabela 5.31, porém removendo os termos identificadores das palavras removidas, tais como: "emailremoved", "nameremoved"e "urlremoved".

5.2 Análise dos resultados

Por meio dos testes feitos e com base nos resultados obtidos, esse trabalho propõe uma abordagem para classificação de problemas de rede em seis tipos diferentes com base em dados reais fornecidos por uma grande empresa de telecomunicações que atua no Brasil.

O teste 1 foi realizado sem qualquer pré-processamento para identificar qual seria a acurácia com a simples aplicação de algoritmos de classificação. A partir do teste 2, diversas técnicas de Processamento de Linguagem Natural foram utilizadas visando identificar quais trariam aumento de acurácia. Assim, primeiramente, no teste 2, foram removidas dos textos de descrição dos problemas as *stop words* para linguagem portuguesa contidas na biblioteca NLTK. Em seguida, todos os algoritmos de classificação foram executados novamente. Comparando os resultados dos testes 1 e 2, mostrados nas tabelas 5.1 e 5.2, respectivamente, não se observou aumento significativo de performance, mas sim diminuição de performance para alguns algoritmos. Dessa forma, concluiu-se que nem todas as *stop words* contidas na biblioteca NLTK poderiam ser removidas, pois algumas acrescentavam informações importantes para definição dos problemas de rede, como, por exemplo, as palavras: com, fora, há, haja, não, nem, sem, tem, têm, esta, está...

O teste 3 foi realizado nas mesmas condições do teste 2, porém com uma quantidade de registros maior para cada classe de problema. Para o teste 2, foram utilizados 3000 registros para cada classe de problema e no teste 3 foram utilizados 7000. Comparando os resultados apresentados das tabelas 5.2 e 5.3, observou-se um pequeno aumento de performance, concluindo-se que um número maior de registros geraria aumento de performance.

No teste 4, foi utilizada a abordagem de remoção de algumas palavras mais frequentes da base de dados. Em alguns casos de Processamento de Linguagem Natural, essas palavras que aparecem com uma frequência muito elevada em relação às frequências de outras palavras do texto, causam diminuição de performance, pois podem aparecer em grande quantidade em todas as classes de problemas, fazendo os algoritmos entenderem que os problemas são de uma mesma classe quando não o são. Comparando os resultados apresentados nas tabelas 5.1 e 5.4, que se diferenciam apenas por essa remoção de palavras mais frequentes, conclui-se que, para a base de dados dessa pesquisa, a remoção de palavras mais frequentes não traz aumento de performance.

No teste 5, utilizou-se a abordagem de remoção das palavras menos frequentes, considerando para o teste apenas as 700 palavras mais frequentes nos textos, removendo-se todas as outras de cada registro de chamado. Comparando os resultados apresentados nas tabelas 5.2 e 5.5, que se diferenciam apenas por essa remoção das palavras menos frequentes, conclui-se que, para a base de dados dessa pesquisa, a remoção de palavras menos frequentes não traz aumento de performance. Isso também pode ser observado comparando-se os testes 3 e 6, semelhantes aos testes 2 e 5, porém com 7000 registros por classe de problema.

Nos testes 7 e 8, também se utilizou a abordagem de remoção das palavras menos frequentes, porém, considerando para o teste apenas as 4000 palavras mais frequentes nos textos, removendo-se todas as outras de cada registro de chamado. Comparando os resultados apresentados nas tabelas 5.2 e 5.7, bem como, os das tabelas 5.3 e 5.8, ratifica-se novamente que remover as palavras menos frequentes não trouxe aumento de performance.

Nos testes 9 e 10, também se utilizou a abordagem de remoção das palavras menos frequentes, porém, considerando para o teste apenas as 5000 e as 10000 palavras mais frequentes nos textos, respectivamente, removendo-se todas as outras de cada registro de chamado. Contrariando as conclusões obtidas nas comparações entre os testes 2 com 5, 3 com 6, 2 com 7 e 3 com 8, a comparação dos testes 9 e 10 com o teste 3 mostra que utilizando apenas as 10000 palavras mais frequentes, removendo-se todas as outras, obtém-se aumento de performance.

Os testes 11 e 12 foram realizados para se comparar a performance de dois métodos de transformação dos textos para forma numérica: TF-IDF e Count Vectorizer. Comparando os resultados apresentados nas tabelas 5.11 e 5.12, conclui-se que a utilização do Count Vectorizer gera uma acurácia maior.

O teste 13 foi realizado utilizando a técnica de Stemização, reduzindo cada palavra dos textos dos chamados à sua raiz. Comparando os resultados apresentados nas tabelas 5.12 e 5.13, conclui-se que essa técnica não gerou nem aumento nem piora nos resultados.

Os testes 14 e 15 foram realizados utilizando as técnicas *One versus One* e *One versus Rest*. Comparando os resultados apresentados nas tabelas 5.14 e 5.15 com os resultados apresentados na tabela 5.10, todos utilizando TF-IDF, não se observou aumento de performance. Porém, no teste 16, que combina a técnica *One versus Rest* com *Count Vectorizer*, observa-se aumento de performance, com valores de acurácia em torno de 88,37%.

O teste 17 foi realizado para avaliar o comportamento da técnica de criação de novas *features* a partir da similaridade entre pequenos conjuntos de palavras escolhidos para representar cada classe de problema. Tudo foi feito utilizando a biblioteca *Gensim*. Comparando os resultados apresentados nas tabelas 5.16 e 5.17, os valores de acurácia do teste 17 não superaram os do teste 16.

Os testes numerados de 18 a 25 foram realizados para comparar a performance dos testes variando o que seria o elemento mais básico: palavra, caractere ou combinações geradas por palavras ou caracteres. Porém, os valores de acurácia obtidos não superaram os do teste 16, com a combinação as técnicas *One versus Rest* com *Count Vectorizer* tomando a palavra como elemento mais básico.

Os testes numerados de 1 a 25 foram realizados com uma base de dados de um período de um mês. Como já foi mostrado nos testes descritos anteriormente, quanto maior o número de registros contidos na base de dados, maior seria a acurácia obtida. Para ratificar esse fato observado, novos dados para outros períodos foram solicitados para empresa, no entanto, esses novos dados vieram sem o campo "PROBLEMA". Analisando as novas bases de dados disponibilizadas, o único campo de classificação de um chamado comum a todas as novas bases foi "MOTIVO3", composto por três classes de problemas. Dessa forma, com a nova base de dados constituída por um número maior de registros, foram realizados os testes cujos resultados estão apresentados nas tabelas numeradas de 26 a 29. Para esse conjunto de testes, optou-se por utilizar apenas os algoritmos que já tinham apresentado maiores acurácias nos testes anteriores. Os resultados ratificaram as conclusões previamente obtidas sobre a influência do número de registros contidos na base de dados. Para esses testes com uma base de dados maior, utilizando a mesma abordagem do teste 16, que apresentou maior acurácia até o momento, obteve-se acurácias em torno de 92,33%.

Nos testes realizados, por inspeção manual, observou-se que os principais erros de classificação ocorriam por descrições erradas, ou que continham erros de ortografia, ou seja, se um especialista fosse classificar manualmente o chamado, iria atribuir a mesma classificação obtida por um algoritmo. Nesse cenário, visando melhorar a acurácia do modelo, como não seria possível mudar a questão das descrições erradas, decidiu-se utilizar as principais bibliotecas em *Python* para correção ortográfica, conforme descrição na metodologia. Contudo, a utilização das bibliotecas prontas em *Python* não produziram resultados satisfatórios para a língua portuguesa. Dessa forma, optou-se por um método simples de correção de pequenos erros ortográficos de forma personalizada, conforme

detalhamento na metodologia. Após a aplicação desse método de correção ortográfica, remoção de nomes próprios e seguindo todos os passos descritos na seção 4.2.11, obteve-se os resultados apresentados na tabela 5.30, atingindo acurácia de 88,80%.

Outro fato observado quanto aos erros de classificação foi que muitas descrições continham poucas palavras ou poucas informações sobre o problema. Dessa forma, o teste 31 foi realizado com a base de dados com registros contendo mais que 5 palavras, removendo-se todos os registros com 5 palavras ou menos. Os resultados apresentados na tabela 5.31 mostram um aumento de acurácia, atingindo valor de 89,10%. Esse teste serve para mostrar a importância de as descrições dos problemas serem inseridas no sistema com o máximo de detalhe possível, pois, de acordo com os resultados, a base de dados com chamados que possuem mais de 5 palavras facilita o processo e classificação, tornando-o mais preciso.

O teste 32 foi realizado nas mesmas condições do teste 31, porém, removendo as palavras que identificam os e-mails, nomes próprios e URLs removidos. Os resultados apresentados na tabela 5.32 mostram que manter identificadores de palavras e termos removidos, tais como, "*emailremoved*", "*nameremoved*"e "*urlremoved*", geram melhoria na acurácia. Dessa forma, saber que o texto original tinha um e-mail gera maior precisão do que simplesmente remover o e-mail.

Capítulo 6

Conclusão

Esse trabalho analisou o desempenho de diversos algoritmos de Aprendizado de Máquina aplicados à classificação de problemas de rede a partir do Processamento de Linguagem Natural dos relatos de clientes registrados em um *Call Center* pelos atendentes de uma grande empresa de telecomunicações que atua no Brasil. Nesse contexto, foram realizados diversos testes aplicando técnicas encontradas nas literaturas relacionadas, bem como, foram realizados testes aplicando técnicas de pré-processamento textual personalizadas para o tipo de estilo de linguagem encontrada nas descrições dos chamados, considerando jargões e palavras específicas utilizadas pelos membros da empresa. Com base nas diversas técnicas testadas de Processamento de Linguagem Natural e de Classificação utilizando algoritmos de Aprendizado de Máquina, esse trabalho alcançou o objetivo de elaborar uma metodologia customizada de acordo com os resultados obtidos por meio dos testes com dados reais.

Com base nos testes de classificação de problemas de rede no contexto descrito nesse trabalho, as seguintes conclusões foram encontradas:

- Algumas *stop words* contidas na biblioteca NLTK para a língua portuguesa não podem ser removidas, pois possuem significado dentro do contexto de identificação de um problema de rede. Dessa forma, se faz necessária uma remoção de *stop words* personalizada considerando jargões e palavras específicas utilizadas pela empresa.
- Quanto maior for a base de dados e quanto mais detalhadas e completas forem as descrições dos problemas de rede, maior será a performance dos algoritmos de classificação. Com o maior conjunto de dados disponível, o algoritmo *Stacking Classifier do Scikit-learn* atingiu a acurácia de 92,33%.
- Com base nos resultados obtidos, constatou-se também que a remoção de palavras

mais frequentes ou a remoção de palavras menos frequentes não gerou melhorias significativas na performance dos algoritmos de classificação.

- Quanto ao método de transformação dos textos para forma numérica, testou-se o TF-IDF e *Count Vectorizer*, com esse último contribuindo de forma mais efetiva.

- Foram testadas técnicas de Stemização para língua Portuguesa, porém não aumentaram a performance do modelo.

- As técnicas *One versus One* e *One versus Rest* foram testadas em conjunto com TF-IDF e *Count Vectorizer*. A combinação que gerou melhor resultado foi a composta por *One versus Rest* e *Count Vectorizer*, atingindo acurácia de 88,37% para o algoritmo *Custom Stacking Classifier 3* descrito na metodologia.

- A técnica de criação de novas *features* a partir da similaridade entre pequenos conjuntos de palavras escolhidos para representar cada classe de problema, utilizando a biblioteca *Gensim*, foi testada e não gerou aumento de performance.

- Também foram realizados testes alterando o que seria o elemento mais básico de um texto: palavra, caractere ou combinações geradas por palavras ou caracteres. Os testes utilizando palavra como elemento mais básico tiveram melhor desempenho.

- Foi observado que os algoritmos erravam as classificações para descrições que continham erros ortográficos. Inicialmente, as bibliotecas *pyspellchecker, autocorrect* e *textblob* foram testadas, porém não geraram resultados satisfatórios. Dessa forma, foi desenvolvida uma técnica personalizada para correção de pequenos erros ortográficos descrita na metodologia. Essa técnica gerou melhores resultados que as bibliotecas citadas.

- Outra análise feita por esse trabalho está relacionada às descrições que continham poucas palavras ou poucas informações sobre o problema. Dessa forma, foi realizado um teste com a base de dados com registros contendo mais que 5 palavras, removendo todos os registros com 5 palavras ou menos. Esse teste serve para mostrar a importância de as descrições dos problemas serem inseridas no sistema com o máximo de detalhe possível, pois, de acordo com os resultados, a base de dados com chamados que possuem mais de 5 palavras facilita o processo e classificação, tornando-o mais preciso.

- Também foi realizado um teste que mostra que manter identificadores de palavras e termos removidos, tais como, "*emailremoved*", "*nameremoved*"e "*urlremoved*",

gera melhoria na acurácia. Dessa forma, saber que o texto original tinha um e-mail gera maior precisão na classificação do que simplesmente remover o e-mail.

Referências

[1] Mieliestronk's list of more than 58 000 english words. `http://www.mieliestronk.com/wordlist.html`. (acessado em 09-outubro-2022).

[2] Al Sharou, K.; Li, Z.; Specia, L. Towards a better understanding of noise in natural language processing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)* (2021), pp. 53–62.

[3] Ali, M.; Haider, M. N.; Lashari, S. A.; Sharif, W.; Khan, A.; Ramli, D. A. Stacking classifier with random forest functioning as a meta classifier for diabetes diseases classification. *Procedia Computer Science 207* (2022), 3459–3468.

[4] Alkhaldi, F. M.; Hammami, S. M.; Ahmar Uddin, M. Understanding value characteristics toward a robust it governance application in private organizations using cobit framework. *International Journal of Engineering Business Management 9* (2017), 1847979017703779.

[5] Anatel. Relatório semestral de reclamações - 1º semestre de 2021, 2021. `https://sistemas.anatel.gov.br/anexar-api/publico/anexos/download/42a4e995c27217df60269e481476e7fc`. Acessado em 12/01/2022.

[6] Asres, M. W.; Mengistu, M. A.; Castrogiovanni, P.; Bottaccioli, L.; Macii, E.; Patti, E.; Acquaviva, A. Supporting telecommunication alarm management system with trouble ticket prediction. *IEEE Transactions on Industrial Informatics 17*, 2 (2020), 1459–1469.

[7] Aziz, M. V. G.; Prihatmanto, A. S.; Henriyan, D.; Wijaya, R. Design and implementation of natural language processing with syntax and semantic analysis for extract traffic conditions from social media data. In *2015 5th IEEE International Conference on System Engineering and Technology (ICSET)* (2015), pp. 43–48.

[8] Bayer, M.; Kaufhold, M.-A.; Buchhold, B.; Keller, M.; Dallmeyer, J.; Reuter, C. Data augmentation in natural language processing: a novel text generation approach for long and short text classifiers. *International journal of machine learning and cybernetics 14*, 1 (2023), 135–150.

[9] Bentes, O. M. *Atendimento ao cliente*. IESDE BRASIL SA, 2012.

[10] Beroiz, M.; Cabral, J. B.; Sanchez, B. Astroalign: A python module for astronomical image registration. *Astronomy and Computing 32* (2020), 100384.

[11] Boser, B. E.; Guyon, I. M.; Vapnik, V. N. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (1992), pp. 144–152.

[12] Bou-Hamad, I.; Jamali, I. Forecasting financial time-series using data mining models: A simulation study. *Research in International Business and Finance 51* (2020), 101072.

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

[14] Breiman, L.; Freedman, D. How many variables should be entered in a regression equation? *Journal of the American Statistical Association 78*, 381 (1983), 131–136.

[15] Broby, D. The use of predictive analytics in finance. *The Journal of Finance and Data Science* (2022).

[16] Browne, M. Predictive validity of a linear regression equation. *British Journal of Mathematical and Statistical Psychology 28*, 1 (1975), 79–87.

[17] Buber, E.; Diri, B.; Sahingoz, O. K. Nlp based phishing attack detection from urls. In *Intelligent Systems Design and Applications: 17th International Conference on Intelligent Systems Design and Applications (ISDA 2017) held in Delhi, India, December 14-16, 2017* (2018), Springer, pp. 608–618.

[18] Budiharto, W.; Meiliana, M. Prediction and analysis of indonesia presidential election from twitter using sentiment analysis. *Journal of Big data 5*, 1 (2018), 1–10.

[19] Buonanno, M.; Grilj, V.; Brenner, D. J. Biological effects in normal cells exposed to flash dose rate protons. *Radiotherapy and Oncology 139* (2019), 51–55.

[20] Burges, C. J. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery 2*, 2 (1998), 121–167.

[21] Busemann, S.; Schmeier, S.; Arens, R. G. Message classification in the call center. *arXiv preprint cs/0003060* (2000).

[22] Cai, H.; Jia, X.; Feng, J.; Li, W.; Hsu, Y.-M.; Lee, J. Gaussian process regression for numerical wind speed prediction enhancement. *Renewable energy 146* (2020), 2112–2123.

[23] Carcillo, F.; Moro, S.; Epifani, P.; Deiro, A. Lessons from adoption of open source trouble ticket system in turin municipality to manage citizens' requests. In *eChallenges e-2010 Conference* (2010), IEEE, pp. 1–6.

[24] Carducci, G.; Rizzo, G.; Monti, D.; Palumbo, E.; Morisio, M. Twitpersonality: Computing personality traits from tweets using word embeddings and supervised learning. *Information 9*, 5 (2018), 127.

[25] Carvalho, D. V.; Pereira, E. M.; Cardoso, J. S. Machine learning interpretability: A survey on methods and metrics. *Electronics 8*, 8 (2019), 832.

[26] Chandra, M. A.; Bedi, S. Survey on svm and their application in image classification. *International Journal of Information Technology 13*, 5 (2021), 1–11.

[27] Chen, J.; de Hoogh, K.; Gulliver, J.; Hoffmann, B.; Hertel, O.; Ketzel, M.; Bauwelinck, M.; Van Donkelaar, A.; Hvidtfeldt, U. A.; Katsouyanni, K., et al. A comparison of linear regression, regularization, and machine learning algorithms to develop europe-wide spatial models of fine particles and nitrogen dioxide. *Environment international 130* (2019), 104934.

[28] Chen, P.-H. Essential elements of natural language processing: what the radiologist should know. *Academic Radiology 27*, 1 (2020), 6–12.

[29] Chen, S.; Webb, G. I.; Liu, L.; Ma, X. A novel selective naïve bayes algorithm. *Knowledge-Based Systems 192* (2020), 105361.

[30] Chicco, D.; Warrens, M. J.; Jurman, G. The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. *PeerJ Computer Science 7* (2021), e623.

[31] Chiddarwar, A. S.; Midhun, K. Qoe improvement in telecom networks using dynamic preload based content caching. In *2014 International Conference on Signal Processing and Communications (SPCOM)* (2014), IEEE, pp. 1–6.

[32] Chowdhary, K.; Chowdhary, K. Natural language processing. *Fundamentals of artificial intelligence* (2020), 603–649.

[33] Christian, H.; Agus, M. P.; Suhartono, D. Single document automatic text summarization using term frequency-inverse document frequency (tf-idf). *ComTech: Computer, Mathematics and Engineering Applications 7*, 4 (2016), 285–294.

[34] Christodoulou, E.; Ma, J.; Collins, G. S.; Steyerberg, E. W.; Verbakel, J. Y.; Van Calster, B. A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models. *Journal of clinical epidemiology 110* (2019), 12–22.

[35] Claro, D. P.; Fragoso, A. F. G. R.; Laban, S. A.; Claro, P. B. d. O. Consumer complaints and company market value. *BAR-Brazilian Administration Review 11* (2014), 248–263.

[36] Coelho, B.; Queiroz, V.; Calazans, C.; Silva, R. A consolidação de sites de reclamação online como uma alternativa eficaz no intermédio das relações de consumo: um estudo de caso do site reclame aqui. In *Anais do Congresso de Ciências da Comunicação na Região Nordeste, Caruaru, PE, Brasil* (2016), vol. 18.

[37] Cohn, J. B.; Liu, Z.; Wardlaw, M. I. Count (and count-like) data in finance. *Journal of Financial Economics 146*, 2 (2022), 529–551.

[38] Cottam, J. A.; Heller, N. C.; Ebsch, C. L.; Deshmukh, R.; Mackey, P.; Chin, G. Evaluation of alignment: Precision, recall, weighting and limitations. In *2020 IEEE International Conference on Big Data (Big Data)* (2020), IEEE, pp. 2513–2519.

[39] Cristianini, N.; Shawe-Taylor, J., et al. *An introduction to support vector machines and other kernel-based learning methods.* Cambridge university press, 2000.

[40] da Ponte Junior, L. A.; Saade, D. C. M.; de Carvalho, A. P.; de Cássia Alves, R.; Portugal, L. C. L.; de Oliveira, L.; Pereira, M. G. Identifying post-traumatic stress symptoms using physiological signals and data mining. In *2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS)* (2020), IEEE, pp. 233–238.

[41] Daneshvar, S.; Inkpen, D. Gender identification in twitter using n-grams and lsa. In *proceedings of the ninth international conference of the CLEF association (CLEF 2018)* (2018), CEUR-WS.

[42] Davidov, D.; Tsur, O.; Rappoport, A. Enhanced sentiment learning using twitter hashtags and smileys. In *Coling 2010: Posters* (2010), pp. 241–249.

[43] de Lima Júnior, M. L. *Previsao De Integradores E Tempo De Vida De Pull Requests*. Tese de Doutorado, Ph. D. Dissertation. Universidade Federal Fluminense, 2017.

[44] de Oliveira, D. N.; Merschmann, L. H. d. C. Joint evaluation of preprocessing tasks with classifiers for sentiment analysis in brazilian portuguese language. *Multimedia Tools and Applications 80*, 10 (2021), 15391–15412.

[45] Deringer, V. L.; Bartók, A. P.; Bernstein, N.; Wilkins, D. M.; Ceriotti, M.; Csányi, G. Gaussian process regression for materials and molecules. *Chemical Reviews 121*, 16 (2021), 10073–10141.

[46] Duong, H.-T.; Nguyen-Thi, T.-A. A review: preprocessing techniques and data augmentation for sentiment analysis. *Computational Social Networks 8*, 1 (2021), 1–16.

[47] Eisenstein, J. What to do about bad language on the internet. In *Proceedings of the 2013 conference of the North American Chapter of the association for computational linguistics: Human language technologies* (2013), pp. 359–369.

[48] El-Kenawy, E.-S. M.; Ibrahim, A.; Mirjalili, S.; Eid, M. M.; Hussein, S. E. Novel feature selection and voting classifier algorithms for covid-19 classification in ct images. *IEEE access 8* (2020), 179317–179335.

[49] El Kourdi, M.; Bensaid, A.; Rachidi, T.-e. Automatic arabic document categorization based on the naïve bayes algorithm. In *proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages* (2004), pp. 51–58.

[50] Fahmy, A. F.; Yousef, A. H.; Mohamed, H. K. The application of data mining for the trouble ticket prediction in telecom operators. In *2017 12th International Conference on Computer Engineering and Systems (ICCES)* (2017), IEEE, pp. 227–232.

[51] Feng, W.; Sun, J.; Zhang, L.; Cao, C.; Yang, Q. A support vector machine based naive bayes algorithm for spam filtering. In *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)* (2016), IEEE, pp. 1–8.

[52] Feofiloff, P. Lista de todas as palavras do português brasileiro. `https://www.ime.usp.br/~pf/dicios/`. (acessado em 09-outubro-2022).

[53] Ferland, N.; Sun, W.; Fan, X.; Yu, L.; Yang, J. Automatically resolve trouble tickets with hybrid nlp. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)* (2020), IEEE, pp. 1334–1340.

[54] Fleuren, L. M.; Klausch, T. L.; Zwager, C. L.; Schoonmade, L. J.; Guo, T.; Roggeveen, L. F.; Swart, E. L.; Girbes, A. R.; Thoral, P.; Ercole, A., et al. Machine learning for the prediction of sepsis: a systematic review and meta-analysis of diagnostic test accuracy. *Intensive care medicine 46*, 3 (2020), 383–400.

[55] Forte Martins, A. D.; Cabral, L.; Chaves Mourão, P. J.; Monteiro, J. M.; Machado, J. Detection of misinformation about covid-19 in brazilian portuguese whatsapp messages. In *Natural Language Processing and Information Systems: 26th International Conference on Applications of Natural Language to Information Systems, NLDB 2021, Saarbrücken, Germany, June 23–25, 2021, Proceedings* (2021), Springer, pp. 199–206.

[56] Freedman, D. A.; Freedman, D. A. A note on screening regression equations. *the american statistician 37*, 2 (1983), 152–155.

[57] Fuada, S. Incident management of information technology in the indonesia higher education based on cobit framework: A review. *EAI Endorsed Transactions on Energy Web 6*, 22 (2019), e3–e3.

[58] Gambino, G.; Pirrone, R. Chilab@ haspeede 2: Enhancing hate speech detection with part-of-speech tagging. *EVALITA Evaluation of NLP and Speech Tools for Italian-December 17th, 2020* (2020), 165.

[59] Geurts, P.; Irrthum, A.; Wehenkel, L. Supervised learning with decision tree-based methods in computational and systems biology. *Molecular Biosystems 5*, 12 (2009), 1593–1605.

[60] Goel, P.; Kumar, S. S. Certain class of starlike functions associated with modified sigmoid function. *Bulletin of the Malaysian Mathematical Sciences Society 43*, 1 (2020), 957–991.

[61] Gunawan, H. Strategic management for it services using the information technology infrastructure library (itil) framework. In *2019 International Conference on Information Management and Technology (ICIMTech)* (2019), vol. 1, pp. 362–366.

[62] Hackeling, G. *Mastering Machine Learning with scikit-learn*. Packt Publishing Ltd, 2017.

[63] Hamdan, Y. B., et al. Construction of statistical svm based recognition model for handwritten character recognition. *Journal of Information Technology 3*, 02 (2021), 92–107.

[64] Han, J.; Kamber, M.; Pei, J. Data mining: Concepts and techniques [internet]. waltham, 2011.

[65] Han, J.; Pei, J.; Tong, H. *Data mining: concepts and techniques*. Morgan kaufmann, 2022.

[66] Hand, D.; Christen, P. A note on using the f-measure for evaluating record linkage algorithms. *Statistics and Computing 28*, 3 (2018), 539–547.

[67] Handelman, G. S.; Kok, H. K.; Chandra, R. V.; Razavi, A. H.; Huang, S.; Brooks, M.; Lee, M. J.; Asadi, H. Peering into the black box of artificial intelligence: evaluation metrics of machine learning methods. *American Journal of Roentgenology 212*, 1 (2019), 38–43.

[68] Hao, J.; Ho, T. K. Machine learning made easy: a review of scikit-learn package in python programming language. *Journal of Educational and Behavioral Statistics 44*, 3 (2019), 348–361.

[69] Hartono, S.; Tjahyadi, R.; Cassandra, C. Analysis of trouble ticket system using cobit 5 framework (a case study approach). In *2019 International Conference on Information Management and Technology (ICIMTech)* (2019), vol. 1, pp. 420–425.

[70] Hasan, M. R.; Maliha, M.; Arifuzzaman, M. Sentiment analysis with nlp on twitter data. In *2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2)* (2019), IEEE, pp. 1–4.

[71] Hasan, T.; Nishat, M. M.; Faisal, F.; Islam, A.; Al Mehadi, A.; Nasrullah, S. M.; Islam, M. R. Exploring the performances of stacking classifier in predicting patients having stroke. In *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)* (2021), IEEE, pp. 242–247.

[72] Haykin, S. *Neural Networks: A Comprehensive Reference and Index*. Prentice Hall, 1999.

[73] Heidari, M.; Jones, J. H.; Uzuner, O. Deep contextualized word embedding for text-based online user profiling to detect social bots on twitter. In *2020 International Conference on Data Mining Workshops (ICDMW)* (2020), IEEE, pp. 480–487.

[74] Huang, M. Theory and implementation of linear regression. In *2020 International conference on computer vision, image and deep learning (CVIDL)* (2020), IEEE, pp. 210–217.

[75] Huang, M.-W.; Chen, C.-W.; Lin, W.-C.; Ke, S.-W.; Tsai, C.-F. Svm and svm ensembles in breast cancer prediction. *PloS one 12*, 1 (2017), e0161501.

[76] Huang, S.; Cai, N.; Pacheco, P. P.; Narrandes, S.; Wang, Y.; Xu, W. Applications of support vector machine (svm) learning in cancer genomics. *Cancer genomics & proteomics 15*, 1 (2018), 41–51.

[77] Hutchins, W. J. The georgetown-ibm experiment demonstrated in january 1954. In *Conference of the Association for Machine Translation in the Americas* (2004), Springer, pp. 102–114.

[78] Imam, T.; Ting, K. M.; Kamruzzaman, J. z-svm: An svm for improved classification of imbalanced data. In *Australasian joint conference on artificial intelligence* (2006), Springer, pp. 264–273.

[79] Indurkhya, N.; Damerau, F. J. *Handbook of natural language processing*. Chapman and Hall/CRC, 2010.

[80] Jan, E.-E.; Chen, K.-Y.; Idé, T. Probabilistic text analytics framework for information technology service desk tickets. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)* (2015), IEEE, pp. 870–873.

[81] Jianqiang, Z.; Xiaolin, G.; Xuejun, Z. Deep convolution neural networks for twitter sentiment analysis. *IEEE access 6* (2018), 23253–23260.

[82] Juba, B.; Le, H. S. Precision-recall versus accuracy and the role of large data sets. In *Proceedings of the AAAI conference on artificial intelligence* (2019), vol. 33, pp. 4039–4048.

[83] Juluru, K.; Shih, H.-H.; Keshava Murthy, K. N.; Elnajjar, P. Bag-of-words technique in natural language processing: a primer for radiologists. *Radiographics 41*, 5 (2021), 1420–1426.

[84] Justen, Gênero dos nomes. `https://brasil.io/dataset/genero-nomes/nomes/`. (acessado em 07-dezembro-2022).

[85] Kakwani, N. C. The unbiasedness of zellner's seemingly unrelated regression equations estimators. *Journal of the American Statistical Association 62*, 317 (1967), 141–142.

[86] Kantardzic, M. *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons, 2011.

[87] Karpukhin, V.; Levy, O.; Eisenstein, J.; Ghazvininejad, M. Training on synthetic noise improves robustness to natural noise in machine translation. *arXiv preprint arXiv:1902.01509* (2019).

[88] Kaur, G.; Chhabra, A. Improved j48 classification algorithm for the prediction of diabetes. *International journal of computer applications 98*, 22 (2014).

[89] Kembhavi, A.; Pattnaik, R. Machine learning in astronomy. *Journal of Astrophysics and Astronomy 43*, 2 (2022), 1–10.

[90] Khader, M.; Awajan, A.; Al-Naymat, G. The impact of natural language preprocessing on big data sentiment analysis. *Int. Arab J. Inf. Technol. 16*, 3A (2019), 506–513.

[91] Khayrallah, H.; Koehn, P. On the impact of various types of noise on neural machine translation. *arXiv preprint arXiv:1805.12282* (2018).

[92] Kim, H.; Kim, S. Data mining based army repair parts demand forecast. *The Korean Data & Information Science Society 30*, 2 (2019), 429–444.

[93] Kirasich, K.; Smith, T.; Sadler, B. Random forest vs logistic regression: binary classification for heterogeneous datasets. *SMU Data Science Review 1*, 3 (2018), 9.

[94] Kotsiantis, S. B. Decision trees: a recent overview. *Artificial Intelligence Review 39*, 4 (2013), 261–283.

[95] Krashniak, A.; Lamm, E. Francis galton's regression towards mediocrity and the stability of types. *Studies in History and Philosophy of Science Part A 86* (2021), 6–19.

[96] Kulkarni, A.; Shivananda, A. *Natural language processing recipes*. Springer, 2019.

[97] Kumar, U. K.; Nikhil, M. S.; Sumangali, K. Prediction of breast cancer using voting classifier technique. In *2017 IEEE international conference on smart technologies and management for computing, communication, controls, energy and materials (ICSTM)* (2017), IEEE, pp. 108–114.

[98] Kumari, K.; Yadav, S., et al. Linear regression analysis study. *Journal of the practice of Cardiovascular Sciences 4*, 1 (2018), 33.

[99] Kuyumcu, B.; Aksakalli, C.; Delil, S. An automated new approach in fast text classification (fasttext) a case study for turkish text classification without preprocessing. In *Proceedings of the 2019 3rd International Conference on Natural Language Processing and Information Retrieval* (2019), pp. 1–4.

[100] Langkjær-Bain, R. The troubling legacy of francis galton. *Significance 16*, 3 (2019), 16–21.

[101] Las Casas, A. L. *Excelência em atendimento ao cliente: atendimento e serviço ao cliente como fator estratégico e diferencial competitivo*. M. Books, 2021.

[102] Lee, D.; Hosanagar, K.; Nair, H. S. Advertising content and consumer engagement on social media: Evidence from facebook. *Management Science 64*, 11 (2018), 5105–5131.

[103] Lento, G. C. *Random forest em dados desbalanceados: uma aplicação na modelagem de churn em seguro saúde*. Tese de Doutorado, 2017.

[104] Lewis, L.; Dreo, G. Extending trouble ticket systems to fault diagnostics. *IEEE network 7*, 6 (1993), 44–51.

[105] Li, X.; Wang, Z.; Yan, J. Prognostic health condition for lithium battery using the partial incremental capacity and gaussian process regression. *Journal of power sources 421* (2019), 56–67.

[106] Li, Y.; Thomas, M. A.; Liu, D. From semantics to pragmatics: where is can lead in natural language processing (nlp) research. *European Journal of Information Systems 30*, 5 (2021), 569–590.

[107] Librelotto, S. R.; Mozzaquatro, P. M. Análise dos algoritmos de mineração j48 e apriori aplicados na detecção de indicadores da qualidade de vida e saúde. *Revista Interdisciplinar de Ensino, Pesquisa e Extensão-RevInt 1*, 1 (2014).

[108] Liddy, E. D. *Natural language processing*, 2nd ed. NY. Marcel Decker, Inc., 2001.

[109] Lin, D.; Raghu, R.; Ramamurthy, V.; Yu, J.; Radhakrishnan, R.; Fernandez, J. Unveiling clusters of events for alert and incident management in large-scale enterprise it. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), pp. 1630–1639.

[110] Lin, X.; Yacoub, S.; Burns, J.; Simske, S. Performance analysis of pattern classifier combination by plurality voting. *Pattern Recognition Letters 24*, 12 (2003), 1959–1969.

[111] Liu, K.; Hu, X.; Wei, Z.; Li, Y.; Jiang, Y. Modified gaussian process regression models for cyclic capacity prediction of lithium-ion batteries. *IEEE Transactions on Transportation Electrification 5*, 4 (2019), 1225–1236.

[112] Liu, Y.; Wu, Y.; Su, L.; Li, W.; Lei, J. Stacking-based ensemble learning method for house price prediction. In *Software Engineering Application in Informatics: Proceedings of 5th Computational Methods in Systems and Software 2021, Vol. 1.* Springer, 2021, pp. 224–237.

[113] Liu, Z.; Wu, D.; Liu, Y.; Han, Z.; Lun, L.; Gao, J.; Jin, G.; Cao, G. Accuracy analyses and model comparison of machine learning adopted in building energy consumption prediction. *Energy Exploration & Exploitation 37*, 4 (2019), 1426–1451.

[114] Locke, S.; Bashall, A.; Al-Adely, S.; Moore, J.; Wilson, A.; Kitchen, G. B. Natural language processing in medicine: a review. *Trends in Anaesthesia and Critical Care 38* (2021), 4–9.

[115] Locke, W. N.; Booth, A. D. *Machine translation of languages: fourteen essays.* Published jointly by Technology Press of the Massachusetts Institute of Technology, 1955.

[116] Lombardo, L.; Mai, P. M. Presenting logistic regression-based landslide susceptibility results. *Engineering geology 244* (2018), 14–24.

[117] LORENA, A. C. Carvalho, acplf uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada 14*, 2 (2007), 43–67.

[118] Luo, E.; Zhang, D.; Luo, H.; Liu, B.; Zhao, K.; Zhao, Y.; Bian, Y.; Wang, Y. Treatment efficacy analysis of traditional chinese medicine for novel coronavirus pneumonia (covid-19): an empirical study from wuhan, hubei province, china. *Chinese medicine 15*, 1 (2020), 1–13.

[119] Machado, M. R.; Karray, S.; de Sousa, I. T. Lightgbm: An effective decision tree gradient boosting method to predict customer loyalty in the finance industry. In *2019 14th International Conference on Computer Science & Education (ICCSE)* (2019), IEEE, pp. 1111–1116.

[120] Magalhães, I. L.; Pinheiro, W. B. *Gerenciamento de serviços de TI na prática: uma abordagem com base na ITIL: inclui ISO/IEC 20.000 e IT Flex*. Novatec Editora, 2007.

[121] Mahabub, A. A robust technique of fake news detection using ensemble voting classifier and comparison with other classifiers. *SN Applied Sciences 2*, 4 (2020), 525.

[122] Maldonado, S.; López, J.; Vairetti, C. An alternative smote oversampling strategy for high-dimensional datasets. *Applied Soft Computing 76* (2019), 380–389.

[123] Maneejuk, P.; Yamaka, W. An analysis of the impacts of telecommunications technology and innovation on economic growth. *Telecommunications Policy 44*, 10 (2020), 102038.

[124] Manjunatha, K.; Kempanna, M. Count vectorizer model based web application vulnerability detection using artificial intelligence approach. *Journal of Discrete Mathematical Sciences and Cryptography 25*, 7 (2022), 2039–2048.

[125] Manning, C.; Schutze, H. *Foundations of statistical natural language processing*. MIT press, 1999.

[126] Maulud, D.; Abdulazeez, A. M. A review on linear regression comprehensive in machine learning. *Journal of Applied Science and Technology Trends 1*, 4 (2020), 140–147.

[127] Medem, A.; Akodjenou, M.-I.; Teixeira, R. Troubleminer: Mining network trouble tickets. In *2009 IFIP/IEEE International Symposium on Integrated Network Management-Workshops* (2009), IEEE, pp. 113–119.

[128] Mehrolia, S.; Alagarsamy, S.; Solaikutty, V. M. Customers response to online food delivery services during covid-19 outbreak using binary logistic regression. *International journal of consumer studies 45*, 3 (2021), 396–408.

[129] Mhamdi, C.; Al-Emran, M.; Salloum, S. A. Text mining and analytics: A case study from news channels posts on facebook. In *Intelligent Natural Language Processing: Trends and Applications*. Springer, 2018, pp. 399–415.

[130] Miao, J.; Zhu, W. Precision–recall curve (prc) classification trees. *Evolutionary intelligence 15*, 3 (2022), 1545–1569.

[131] Mielke, S. J.; Alyafeai, Z.; Salesky, E.; Raffel, C.; Dey, M.; Gallé, M.; Raja, A.; Si, C.; Lee, W. Y.; Sagot, B., et al. Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp. *arXiv preprint arXiv:2112.10508* (2021).

[132] Mimura, M.; Miura, H. Detecting unseen malicious vba macros with nlp techniques. *Journal of Information Processing 27* (2019), 555–563.

[133] Moayedi, H.; Osouli, A.; Nguyen, H.; Rashid, A. S. A. A novel harris hawks' optimization and k-fold cross-validation predicting slope stability. *Engineering with Computers 37*, 1 (2021), 369–379.

[134] Müller, M.; Salathé, M.; Kummervold, P. E. Covid-twitter-bert: A natural language processing model to analyse covid-19 content on twitter. *arXiv preprint arXiv:2005.07503* (2020).

[135] Ngo, P. T. T.; Panahi, M.; Khosravi, K.; Ghorbanzadeh, O.; Kariminejad, N.; Cerda, A.; Lee, S. Evaluation of deep learning algorithms for national scale landslide susceptibility mapping of iran. *Geoscience Frontiers 12*, 2 (2021), 505–519.

[136] Nguyen, P. T.; Ha, D. H.; Avand, M.; Jaafari, A.; Nguyen, H. D.; Al-Ansari, N.; Van Phong, T.; Sharma, R.; Kumar, R.; Le, H. V., et al. Soft computing ensemble models based on logistic regression for groundwater potential mapping. *Applied Sciences 10*, 7 (2020), 2469.

[137] Oliinyk, V.-A.; Vysotska, V.; Burov, Y.; Mykich, K.; Fernandes, V. B. Propaganda detection in text data based on nlp and machine learning. In *MoMLeT+ DS* (2020), pp. 132–144.

[138] Olusegun, R.; Oladunni, T.; Audu, H.; Houkpati, Y.; Bengesi, S. Text mining and emotion classification on monkeypox twitter dataset: A deep learning-natural language processing (nlp) approach. *IEEE Access* (2023), 1–1.

[139] Oshiro, T. M. *Uma abordagem para a construção de uma única árvore a partir de uma Random Forest para classificação de bases de expressão gênica.* Tese de Doutorado, Universidade de São Paulo, 2013.

[140] Otchere, D. A.; Ganat, T. O. A.; Gholami, R.; Ridha, S. Application of supervised machine learning paradigms in the prediction of petroleum reservoir properties: Comparative analysis of ann and svm models. *Journal of Petroleum Science and Engineering 200* (2021), 108182.

[141] Otter, D. W.; Medina, J. R.; Kalita, J. K. A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems 32*, 2 (2021), 604–624.

[142] Pariyani, B.; Shah, K.; Shah, M.; Vyas, T.; Degadwala, S. Hate speech detection in twitter using natural language processing. In *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)* (2021), IEEE, pp. 1146–1152.

[143] Parmar, M.; Jain, N.; Jain, P.; Jayakrishna Sahit, P.; Pachpande, S.; Singh, S.; Singh, M. Nlpexplorer: exploring the universe of nlp papers. In *European Conference on Information Retrieval* (2020), Springer, pp. 476–480.

[144] Parveen, H.; Pandey, S. Sentiment analysis on twitter data-set using naive bayes algorithm. In *2016 2nd international conference on applied and theoretical computing and communication technology (iCATccT)* (2016), IEEE, pp. 416–419.

[145] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V., et al. One vs one classifier. `https://scikit-learn.org/stable/modules/`

generated/sklearn.multiclass.OneVsOneClassifier.html. (acessado em 07-dezembro-2022).

[146] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V., et al. One vs rest classifier. https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html. (acessado em 07-dezembro-2022).

[147] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V., et al. Stacking classifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html. (acessado em 07-dezembro-2022).

[148] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V., et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research 12* (2011), 2825–2830.

[149] Platt, J. Sequential minimal optimization: A fast algorithm for training support vector machines.

[150] Plisson, J.; Lavrac, N.; Mladenic, D., et al. A rule based approach to word lemmatization. In *Proceedings of IS* (2004), vol. 3, pp. 83–86.

[151] Poongodi, M.; Sharma, A.; Vijayakumar, V.; Bhardwaj, V.; Sharma, A. P.; Iqbal, R.; Kumar, R. Prediction of the price of ethereum blockchain cryptocurrency in an industrial finance system. *Computers & Electrical Engineering 81* (2020), 106527.

[152] Potharaju, R.; Jain, N.; Nita-Rotaru, C. Juggling the jigsaw: Towards automated problem inference from network trouble tickets. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)* (2013), pp. 127–141.

[153] Powers, D. M. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061* (2020).

[154] Priyadarshini, S. B. B.; Bagjadab, A. B.; Mishra, B. K. A brief overview of natural language processing and artificial intelligence. *Natural Language Processing in Artificial Intelligence* (2020), 211–224.

[155] Raina, V.; Krishnamurthy, S.; Raina, V.; Krishnamurthy, S. Natural language processing. *Building an Effective Data Science Practice: A Framework to Bootstrap and Manage a Successful Data Science Practice* (2022), 63–73.

[156] Ranganathan, G., et al. A study to find facts behind preprocessing on deep learning algorithms. *Journal of Innovative Image Processing (JIIP) 3*, 01 (2021), 66–74.

[157] Ronran, C.; Lee, S. Effect of character and word features in bidirectional lstm-crf for ner. In *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)* (2020), IEEE, pp. 613–616.

[158] Rubio, J. L.; Arcilla, M. How to optimize the implementation of itil through a process ordering algorithm. *Applied Sciences 10*, 1 (2019), 34.

[159] Ruta, D.; Gabrys, B. Classifier selection for majority voting. *Information fusion 6*, 1 (2005), 63–81.

[160] Rymarczyk, T.; Kozłowski, E.; Kłosowski, G.; Niderla, K. Logistic regression for machine learning in process tomography. *Sensors 19*, 15 (2019), 3400.

[161] Sahingoz, O. K.; Buber, E.; Demir, O.; Diri, B. Machine learning based phishing detection from urls. *Expert Systems with Applications 117* (2019), 345–357.

[162] Santos, T. Corretor ortográfico em python: aplicando técnicas de nlp. `https://www.alura.com.br/curso-online-nlp-corretor-ortografico`. (acessado em 09-outubro-2022).

[163] Sarica, S.; Luo, J. Stopwords in technical language processing. *Plos one 16*, 8 (2021), e0254937.

[164] Schober, P.; Boer, C.; Schwarte, L. A. Correlation coefficients: appropriate use and interpretation. *Anesthesia & Analgesia 126*, 5 (2018), 1763–1768.

[165] Schulz, E.; Speekenbrink, M.; Krause, A. A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology 85* (2018), 1–16.

[166] Semolini, R., et al. Support vector machines, inferência transdutiva e o problema de classificação. *Campinas, SP* (2002).

[167] Senaviratna, N.; Cooray, T., et al. Diagnosing multicollinearity of logistic regression model. *Asian Journal of Probability and Statistics 5*, 2 (2019), 1–9.

[168] Shao, Q.; Chen, Y.; Tao, S.; Yan, X.; Anerousis, N. Easyticket: A ticket routing recommendation engine for enterprise problem resolution. *Proceedings of the VLDB Endowment 1*, 2 (2008), 1436–1439.

[169] Sharifzadeh, M.; Sikinioti-Lock, A.; Shah, N. Machine-learning methods for integrated renewable power generation: A comparative study of artificial neural networks, support vector regression, and gaussian process regression. *Renewable and Sustainable Energy Reviews 108* (2019), 513–538.

[170] Silva, C.; Ribeiro, B. The importance of stop word removal on recall values in text categorization. In *Proceedings of the International Joint Conference on Neural Networks, 2003.* (2003), vol. 3, IEEE, pp. 1661–1666.

[171] Singh, J.; Gupta, V. Text stemming: Approaches, applications, and challenges. *ACM Computing Surveys (CSUR) 49*, 3 (2016), 1–46.

[172] Situmeang, S. Impact of text preprocessing on named entity recognition based on conditional random field in indonesian text. *Jurnal Mantik 6*, 1 (2022), 423–430.

[173] Srivastava, V. K.; Giles, D. E. *Seemingly unrelated regression equations models: Estimation and inference.* CRC press, 2020.

[174] Subramaniam, L. V.; Roy, S.; Faruquie, T. A.; Negi, S. A survey of types of text noise and techniques to handle noisy text. In *Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data* (2009), pp. 115–122.

[175] Suits, D. B. Use of dummy variables in regression equations. *Journal of the American Statistical Association 52*, 280 (1957), 548–551.

[176] Sun, W.; Xue, M.; Yu, H.; Tang, H.; Lin, A. Augmentation of fingerprints for indoor wifi localization based on gaussian process regression. *IEEE Transactions on Vehicular Technology 67*, 11 (2018), 10896–10905.

[177] Tang, L.; Li, T.; Shwartz, L.; Grabarnik, G. Recommending resolutions for problems identified by monitoring. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)* (2013), IEEE, pp. 134–142.

[178] Temprado, Y.; Molinero, F. J.; Garcia, C.; Gomez, J. Knowledge discovery from trouble ticketing reports in a large telecommunication company. In *2008 International Conference on Computational Intelligence for Modelling Control & Automation* (2008), IEEE, pp. 37–42.

[179] Üstün, B.; Melssen, W. J.; Buydens, L. M. Facilitating the application of support vector regression by using a universal pearson vii function based kernel. *Chemometrics and Intelligent Laboratory Systems 81*, 1 (2006), 29–40.

[180] Vaibhav, V.; Singh, S.; Stewart, C.; Neubig, G. Improving robustness of machine translation with synthetic noise. *arXiv preprint arXiv:1902.09508* (2019).

[181] Vajjala, S.; Majumder, B.; Gupta, A.; Surana, H. *Practical natural language processing: a comprehensive guide to building real-world NLP systems.* O'Reilly Media, 2020.

[182] Vapnik, V. Statistical learning theory, new york etc, 1998.

[183] Vapnik, V. *The nature of statistical learning theory.* Springer science & business media, 1999.

[184] Velásquez, R. M. A.; Lara, J. V. M. Forecast and evaluation of covid-19 spreading in usa with reduced-space gaussian process regression. *Chaos, Solitons & Fractals 136* (2020), 109924.

[185] Vembandasamy, K.; Sasipriya, R.; Deepa, E. Heart diseases detection using naive bayes algorithm. *International Journal of Innovative Science, Engineering & Technology 2*, 9 (2015), 441–444.

[186] Wang, D.; Zhong, D.; Li, L. A comprehensive study of the role of cloud computing on the information technology infrastructure library (itil) processes. *Library Hi Tech 40*, 6 (2022), 1954–1975.

[187] Wang, W.; Lu, Y. Analysis of the mean absolute error (mae) and the root mean square error (rmse) in assessing rounding model. In *IOP conference series: materials science and engineering* (2018), vol. 324, IOP Publishing, p. 012049.

[188] Watanabe, A.; Ishibashi, K.; Toyono, T.; Watanabe, K.; Kimura, T.; Matsuo, Y.; Shiomoto, K.; Kawahara, R. Workflow extraction for service operation using multiple unstructured trouble tickets. *IEICE Transactions on Information and Systems 101*, 4 (2018), 1030–1041.

[189] Weiying, K.; Pham, D. N.; Eftekharypour, Y.; Pheng, A. J. Benchmarking nlp toolkits for enterprise application. In *Pacific Rim International Conference on Artificial Intelligence* (2019), Springer, pp. 289–294.

[190] Witten, I. H.; Frank, E. Data mining: practical machine learning tools and techniques with java implementations. *Acm Sigmod Record 31*, 1 (2002), 76–77.

[191] Wu, L.; Dodoo, N. A.; Wen, T. J.; Ke, L. Understanding twitter conversations about artificial intelligence in advertising based on natural language processing. *International Journal of Advertising 41*, 4 (2022), 685–702.

[192] Xu, J.; He, R.; Zhou, W.; Li, T. Trouble ticket routing models and their applications. *IEEE Transactions on Network and Service Management 15*, 2 (2018), 530–543.

[193] Yogish, D.; Manjunath, T.; Hegadi, R. S. Review on natural language processing trends and techniques using nltk. In *Recent Trends in Image Processing and Pattern Recognition: Second International Conference, RTIP2R 2018, Solapur, India, December 21–22, 2018, Revised Selected Papers, Part III 2* (2019), Springer, pp. 589–606.

[194] Young, T.; Hazarika, D.; Poria, S.; Cambria, E. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine 13*, 3 (2018), 55–75.

[195] Youssef, Y. B.; Afif, M.; Ksantini, R.; Tabbane, S. A novel online qoe prediction model based on multiclass incremental support vector machine. In *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)* (2018), IEEE, pp. 334–341.

[196] Zeng, C.; Li, T.; Shwartz, L.; Grabarnik, G. Y. Hierarchical multi-label classification over ticket data using contextual loss. In *2014 IEEE Network Operations and Management Symposium (NOMS)* (2014), IEEE, pp. 1–8.

[197] Zhang, Y.; Xu, X. Predicting doped mgb2 superconductor critical temperature from lattice parameters using gaussian process regression. *Physica C: Superconductivity and its Applications 573* (2020), 1353633.

[198] Zhao, R.; Mu, Y.; Zou, L.; Wen, X. A hybrid intrusion detection system based on feature selection and weighted stacking classifier. *IEEE Access 10* (2022), 71414–71426.

[199] Zhao, Y. Research on personal credit evaluation of internet finance based on blockchain and decision tree algorithm. *EURASIP Journal on Wireless Communications and Networking 2020*, 1 (2020), 1–12.

[200] Zhou, J.; Gandomi, A. H.; Chen, F.; Holzinger, A. Evaluating the quality of machine learning explanations: A survey on methods and metrics. *Electronics 10*, 5 (2021), 593.

[201] Zhou, Z.; Guan, H.; Bhat, M. M.; Hsu, J. Fake news detection via nlp is vulnerable to adversarial attacks. *arXiv preprint arXiv:1901.09657* (2019).