UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIËNCIA DA COMPUTAÇÃO

IANN CARVALHO BARBOSA

AVALIAÇÃO DO MODELO T5 NA DETECÇÃO DE BUG REPORTS SIMILARES: UMA ABORDAGEM HÍBRIDA COM TF-IDF

CAMPINA GRANDE - PB
2024

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Avaliação do Modelo T5 na Detecção de Bug Reports Similares: Uma Abordagem Híbrida com TF-IDF

Iann Carvalho Barbosa

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Recuperação da Informação, PLN, Engenharia de Software

João Arthur Brunet Monteiro e Franklin de Souza Ramalho
(Orientadores)

Campina Grande, Paraíba, Brasil

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO EM CIENCIA DA COMPUTACAO
Rua Aprígio Veloso, 882, Edifício Telmo Silva de Araújo, Bloco CG1, - Bairro Universitário, Campina Grande/PB, CEP 58429-900
Telefone: 2101-1122 - (83) 2101-1123 - (83) 2101-1124
Site: http://computacao.ufcg.edu.br - E-mail: secpg@computacao.ufcg.edu.br

FOLHA DE ASSINATURA PARA TESES E DISSERTAÇÕES

IANN CARVALHO BARBOSA

AVALIAÇÃO DO MODELO T5 NA DETECÇÃO DE BUG REPORTS SIMILARES: UMA ABORDAGEM HÍBRIDA COM TF-IDF

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação como pré-requisito para obtenção do título de Mestre em Ciência da Computação.

Aprovada em: 13/11/2024

Prof. Dr. JOÃO ARTHUR BRUNET MONTEIRO, Orientador, UFCG

Prof. Dr. FRANKLIN DE SOUZA RAMALHO, Orientador, UFCG

Prof. Dr. TIAGO LIMA MASSONI, Examinador Interno, UFCG

Prof. Dr. UIRA KULESZA, Examinador Externo, UFRN

Documento assinado eletronicamente por **JOAO ARTHUR BRUNET MONTEIRO**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 14/11/2024, às 17:58, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **TIAGO LIMA MASSONI**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 16/11/2024, às 10:35, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Uirá Kulesza**, **Usuário Externo**, em 18/11/2024, às 08:57, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **FRANKLIN DE SOUZA RAMALHO**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 18/11/2024, às 17:01, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **5012897** e o código CRC **EC1DE4D4**.

Referência: Processo nº 23096.080550/2024-63 SEI nº 5012897

B238a Barbosa, Iann Carvalho.

Avaliação do modelo T5 na detecção de *bug reports* similares: uma abordagem híbrida com TF-IDF / Iann Carvalho Barbosa. – Campina Grande, 2024.

134 f. : il. color.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2024.

"Orientação: Prof. Dr. João Arthur Brunet Monteiro, Prof. Dr. Franklin de Souza Ramalho".

Referências.

1. Bug Report. 2. Similaridade Textual Semântica (STS). 3. Modelo T5. 4. Método TF-IDF (*Term Frequency-Inverse Document Frequency*). 5. Processamento de Linguagem Natural (PLN). 6. Engenharia de Software. 7. Bugzilla. 8. Vetorização. 9. Ajuste Fino. I. Monteiro, João Arthur Brunet. II. Ramalho, Franklin de Souza. III. Título.

CDU 004.41(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA MEIRE EMANUELA DA SILVA MELO CRB-15/568

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO EM CIENCIA DA COMPUTACAO
Rua Aprígio Veloso, 882, Edifício Telmo Silva de Araújo, Bloco CG1, - Bairro Universitário, Campina Grande/PB, CEP 58429-900
Telefone: 2101-1122 - (83) 2101-1123 - (83) 2101-1124
Site: http://computacao.ufcg.edu.br - E-mail: secpg@computacao.ufcg.edu.br

REGISTRO DE PRESENÇA E ASSINATURAS

ATA Nº 022/2024 (DISSERTAÇÃO N° 736)

Aos treze (13) dias do mês de novembro do ano de dois mil e vinte e quatro (2024), às oito horas e trinta minutos (08:30), no AUDITÓRIO DO SPLAB, da Universidade Federal de Campina Grande - UFCG, nesta cidade, reuniu-se a Comissão Examinadora composta pelos Professores JOÃO ARTHUR BRUNET MONTEIRO, Dr., Orientador, UFCG, funcionando neste ato como Presidente, FRANKLIN DE SOUZA RAMALHO, Dr., Orientador, UFCG, TIAGO LIMA MASSONI, Dr., UFCG, UIRA KULESZA, Dr., UFRN, este com participação por videoconferência. Constituída a mencionada Comissão Examinadora pela Portaria Nº 036/2024 da Coordenação do Programa de Pós-Graduação em Ciência da Computação, tendo em vista a deliberação do Colegiado do Curso, tomada em reunião de 31 de Outubro de 2024 e com fundamento no Regulamento Geral dos Cursos de Pós-Graduação da Universidade Federal de Campina Grande - UFCG, juntamente com o Sr(a) IANN CARVALHO BARBOSA, candidato(a) ao grau de MESTRE em Ciência da Computação, presentes ainda professores e alunos do referido centro e demais presentes. Abertos os trabalhos, o(a) Senhor(a) Presidente da Comissão Examinadora anunciou que a reunião tinha por finalidade a apresentação e julgamento da dissertação "AVALIAÇÃO DO MODELO T5 NA DETECÇÃO DE BUG REPORTS SIMILARES: UMA ABORDAGEM HÍBRIDA COM TF-IDF", elaborada pelo(a) candidato(a) acima designado, sob a orientação do(s) Professor(es) JOÃO ARTHUR BRUNET MONTEIRO e FRANKLIN DE SOUZA RAMALHO, com o objetivo de atender as exigências do Regulamento Geral dos Cursos de Pós-Graduação da Universidade Federal de Campina Grande - UFCG. A seguir, concedeu a palavra ao (a) candidato(a), o qual, após salientar a importância do assunto desenvolvido, defendeu o conteúdo da dissertação. Concluída a exposição e defesa do candidato, passou cada membro da Comissão Examinadora a arguir o mestrando sobre os vários aspectos que constituíram o campo de estudo tratado na referida dissertação. Terminados os trabalhos de arguição, o(a) Senhor(a) Presidente da Comissão Examinadora determinou a suspensão da sessão pelo tempo necessário ao julgamento da dissertação. Reunidos, em caráter secreto, no mesmo recinto, os membros da Comissão Examinadora passaram à apreciação da dissertação. Reaberta a sessão, o(a) Presidente da Comissão Examinadora anunciou o resultado do julgamento, tendo assim, o candidato obtida o Conceito APROVADO. Na sequência, o(a) Presidente da Comissão Examinadora anunciou o resultado do julgamento, tendo a seguir encerrado a sessão, da qual lavrei a presente ata, que vai assinada por mim, Lyana Silva e Cavalcante Nascimento, pelos membros da Comissão Examinadora e pelo candidato. Campina Grande, 13 de Novembro de 2024.

Documento assinado eletronicamente por **JOAO ARTHUR BRUNET MONTEIRO**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 14/11/2024, às 17:57, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **LYANA SILVA E CAVALCANTE NASCIMENTO**, **ASSISTENTE EM ADMINISTRACAO**, em 15/11/2024, às 11:51, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Iann Carvalho Barbosa**, **Aluno**, em 15/11/2024, às 16:54, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **TIAGO LIMA MASSONI**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 16/11/2024, às 10:35, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Uirá Kulesza**, **Usuário Externo**, em 18/11/2024, às 08:57, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **FRANKLIN DE SOUZA RAMALHO**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 18/11/2024, às 17:00, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **5012895** e o código CRC **5E7F35A3**.

Referência: Processo nº 23096.080550/2024-63 SEI nº 5012895

Resumo

No contexto do desenvolvimento de *software*, *bug reports* (BRs) são fundamentais para identificar e descrever falhas que impactam a qualidade e estabilidade do produto final. O crescente volume de BRs em grandes projetos de *software* torna a identificação manual de BRs similares uma tarefa demorada e propensa a erros, levando a uma menor eficiência no processo de desenvolvimento. Visando melhorar a alocação de recursos, agilizar a resolução de problemas recorrentes e otimizar o desenvolvimento de *software*, examinamos a aplicação de técnicas de aprendizagem de máquina relevantes aos problemas. Para tal, foi utilizado o modelo T5 (*Text-to-Text Transfer Transformer*), o método TF-IDF (*Term Frequency-Inverse Document Frequency*) e uma abordagem híbrida, aproveitando a eficácia do T5 em tarefas de Similaridade Textual Semântica (STS) e a versatilidade do TF-IDF em análises léxicas, combinando-os para potencializar a identificação de BRs similares. O *pipeline* divide-se em recuperação dos dados, pré-processamento, vetorização, normalização, treinamento das redes neurais e avaliação dos resultados obtidos. Foram avaliados os desempenhos de 56 modelos, aplicando diversas estratégias de modelagem. Esta análise detalhada revela que o uso de vetores completos como *features* é mais eficaz do que a distância de cosseno. Já a abordagem híbrida proposta demonstra resultados promissores, muitas vezes superando as abordagens individuais. O estudo também realiza um ajuste fino em 14 modelos promissores, testando 168 combinações de hiperparâmetros, com os otimizadores *Adam* e *RMSprop* apresentando os melhores desempenhos. As contribuições deste trabalho incluem uma avaliação do desempenho do T5 e do TF-IDF no contexto de BRs, a concepção e validação de uma abordagem híbrida, e a exploração de várias estratégias de modelagem. A pesquisa oferece sugestões para implementações futuras, potencialmente melhorando a eficiência e a eficácia no desenvolvimento e facilitando a alocação de recursos. As descobertas sobre o desempenho do T5 e a eficácia da abordagem híbrida impulsionam pesquisas futuras e aplicações em sistemas de recomendação para gestão de *bugs* e desenvolvimento de *software*, ressaltando a importância do seu aprimoramento contínuo.

Abstract

In the context of software development, bug reports (BRs) are fundamental for identifying and describing flaws that impact the quality and stability of the final product. The growing volume of BRs in large software projects makes manual identification of similar BRs a time-consuming and error-prone task, leading to reduced efficiency in the development process. Aiming to improve resource allocation, expedite the resolution of recurring problems, and optimize software development, we examined the application of machine learning techniques relevant to these issues. To this end, we utilized the T5 (Text-to-Text Transfer Transformer) model, the TF-IDF (Term Frequency-Inverse Document Frequency) method, and a hybrid approach, leveraging the effectiveness of T5 in Semantic Textual Similarity (STS) tasks and the versatility of TF-IDF in lexical analyses, combining them to enhance the identification of similar BRs. The pipeline is divided into data retrieval, preprocessing, vectorization, normalization, neural network training, and evaluation of obtained results. We evaluated the performance of 56 models, applying various modeling strategies. This detailed analysis reveals that using complete vectors as features is more effective than using cosine distance. The proposed hybrid approach demonstrates promising results, often outperforming individual approaches. The study also performs fine-tuning on 14 promising models, testing 168 hyperparameter combinations, with Adam and RMSprop optimizers showing the best performance. The contributions of this work include an evaluation of T5 and TF-IDF performance in the context of BRs, the conception and validation of a hybrid approach, and the exploration of various modeling strategies. The research offers suggestions for future implementations, potentially improving efficiency and effectiveness in development and facilitating resource allocation. The findings on T5 performance and the effectiveness of the hybrid approach drive future research and applications in recommendation systems for bug management and software development, highlighting the importance of their continuous improvement.

Agradecimentos

Sou grato à Universidade Federal de Campina Grande e à Unidade Acadêmica de Sistemas e Computação, onde pude ter a oportunidade de conhecer e aprender com excelentes professores e colegas.

Desejo expressar minha sincera gratidão por ter sido abençoado com a orientação de Franklin Ramalho e João Arthur. Meus orientadores sempre mantiveram uma presença constante e uma disposição inabalável para o diálogo em todos os momentos, mesmo nos mais desafiadores. Desde a graduação, tive o prazer de trabalhar e ser orientado por João que, para mim, é um exemplo de firmeza, competência, franqueza e objetividade. Após o ingresso no mestrado, tive a equivalente satisfação com Franklin, um exemplo de humanidade, proatividade, didática e organização.

Sou grato pelas amizades que já me davam suporte antes e pelas novas que conquistei durante o mestrado. Gostaria de reconhecer mais uma vez meu parceiro de jornada acadêmica e pessoal, Júlio Barreto. Agradeço por todas as madrugadas no Discord de estudo, conversas sinceras e momentos de descontração que compartilhamos ao longo da graduação, projetos, mestrado e vida. Deixo clara minha gratidão à minha namorada Alícia Beatriz, por me fazer ver o mundo com uma perspectiva diferente, tornando-me um homem mais forte, completo e obstinado. Um agradecimento especial ao meu amigo de longa data Arthur, que me ajudou nos momentos finais de desespero com a correção da dissertação. Seu olhar atento e sugestões valiosas foram fundamentais para aprimorar meu trabalho. Destaco também meus agradecimentos aos meus amigos mais antigos como Matheus, Gustavo, Pedro, João, Iago, Queirós, Erick, Kaio, Igor, Daniella e outros que me acompanharam por todo esse tempo.

Destaco também minha gratidão pelos dois anos trabalhados no SPLAB, onde tive o prazer de conhecer pessoas incríveis no CodeSQ, ePol e Dark Stores. Agradeço a Manoel Ferreira pelo companheirismo no projeto ePol e pelo suporte no desenvolvimento da pesquisa. Também sou grato a Lucas Fernandes que, apesar de todas as intempéries e dificuldades, sempre foi carinhoso e acolhedor com qualquer dificuldade que eu tivesse. Gostaria também de deixar claros meus agradecimentos a Lilian, Paloma e Jackson, pois sei que a atenção que tiveram comigo me ajudou a ser mais calmo e resiliente nos momentos de maior ansiedade.

Enquanto escrevo isso, nomes de pessoas importantes passam pela minha cabeça, como Guilherme, Victor, Daniel, Ruan, Sheila, Anna Beatriz, Igor, Hellen, entre outros que, infelizmente, seria impossível não esquecer de alguém. Acredito que eu poderia escrever uma dissertação só de agradecimentos por tudo e todos que tive contato no melhor momento da minha vida, que foi o mestrado.

Não poderia deixar de expressar minha profunda gratidão à equipe do projeto da IBM que, mesmo sem me conhecerem pessoalmente, me acolheram como se fôssemos velhos amigos. Nos momentos mais desafiadores da minha pesquisa, quando precisei de recursos computacionais para rodar meus experimentos, eles prontamente disponibilizaram suas máquinas, demonstrando uma generosidade e um espírito colaborativo admirável. Gostaria de agradecer nominalmente a Áxel, Higor, André, Ian, Ekarani, Filipe, Rian e Jamilly. Cada um de vocês contribuiu de forma significativa para o sucesso deste trabalho, oferecendo não apenas recursos técnicos, mas também encorajamento.

Por fim, e não menos importante, sou grato à minha família, principalmente pela oportunidade que tive de conviver com minha avó Rita Silva. Apesar de debilitada, ela esteve ao meu lado fazendo do impossível o possível no momento em que eu mais precisei. Gostaria de agradecer ao meu núcleo familiar Linaldo, Kyvânia, Igor e Iago, afinal tudo que eu sou e me tornei eu devo a eles. Sou grato pela disponibilidade e pela inspiração que minhas tias Kissia e Kênia são para mim, grandes educadoras, pesquisadoras e acadêmicas.

Conteúdo

Lista de Símbolos

BR - *Bug Report*

STS - *Semantic Textual Similarity/Similaridade Textual Semântica*

NLP/PLN - *Natural Language Processing/Processamento de Linguagem Natural*

LLM - *Large Language Model*

T5 - *Text-To-Text Transfer Transformer*

NN/RN - *Neural Network/Rede Neural*

TF-IDF - *Term Frequency-Inverse Document Frequency*

AUC - *Area Under the Curve/Área Sob a Curva*

ML/AM - *Machine Learning/Aprendizagem de Máquina*

AI/IA - *Artificial Inteligence/Inteligência Artificial*

BoW - *Bag of Words*

TF - *Term Frequency*

IDF - *Inverse Document Frequency*

GloVe - *Global Vectors for Word Representation*

C4 - *Colossal Clean Crawled Corpus*

BERT - *Bidirectional Encoder Representations for Transformers*

S-BERT - *Sentence-BERT*

LDA - *Linear Discriminant Analysis*

DC-CNN - *Dual-Channel Convolutional Neural Network*

MLP - *Multilayer Perceptron*

SVC - *Support Vector Classifier*

NB - *Naive Bayes*

DT - *Decision Tree*

KNN - *K-nearest Neighbors*

TP - *True Positive*

TN - *True Negative*

FP - *False Positive*

FN - *False Negative*

TPR - *True Positive Rate*

TNR - *True Negative Rate*

FPR - *False Positive Rate*

FNR - *False Negative Rate*

ROC - *Receiver Operating Characteristic*

PR - *Precision-Recall*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Nos últimos anos, a demanda por *softwares* tem crescido exponencialmente, impulsionada pela transformação digital e pela necessidade de soluções tecnológicas em diversos setores. De acordo com um estudo recente, o mercado global de *software* deve alcançar um faturamento de US$ 698,80 bilhões em 2024, refletindo um aumento significativo na demanda por soluções tecnológicas [21]. Essa crescente demanda destaca a importância dos *softwares* como ferramentas essenciais para inovação e eficiência em áreas como saúde, educação, finanças e entretenimento.

O desenvolvimento de software é um processo estruturado que envolve várias etapas, incluindo planejamento, *design*, codificação, teste e manutenção. Cada etapa desempenha um papel crítico para garantir que o produto final atenda aos padrões desejados de qualidade e funcionalidade [71]. Durante o planejamento, são definidas as necessidades do projeto e os requisitos do *software*. O *design* envolve a criação da arquitetura do sistema, enquanto a codificação é o processo de escrever o código que implementa essa arquitetura. A fase de teste é crucial para identificar falhas e garantir que o *software* funcione conforme o esperado. Finalmente, a manutenção abrange a correção desses comportamentos inesperados e a atualização do *software* ao longo do tempo.

Dentro do processo de criação de sistemas, um elemento inevitável são os *bugs* [58]. *Bugs* são faltas no *software* que podem causar resultados inesperados em operações computacionais, conhecidos como falhas [50]. Detectar e corrigir *bugs* é uma parte essencial do desenvolvimento e evolução, pois eles podem comprometer a funcionalidade e a segurança do *software*.

Ao longo do processo de identificação do *bug*, usualmente é criado um documento chamado de *bug report* (BR), que descreve a falha encontrada e como reproduzi-la, facilitando a correção do *bug* pelos desenvolvedores. Um relatório de *bug* bem elaborado é crucial para que os desenvolvedores possam entender e corrigir o problema de maneira eficiente [5].

Neste contexto, existem ferramentas que auxiliam na descrição e gestão de *Bug Reports* (BRs). Uma das mais amplamente utilizadas é o *Bugzilla*[1], que serve como um sistema de descrição e rastreamento de *bugs* e é empregado por muitas organizações de desenvolvimento de *software*, como o *Eclipse Foundation*. O *Bugzilla* oferece um ambiente centralizado para registrar, monitorar e resolver *bugs*, facilitando o processo de desenvolvimento do sistema através de BRs. O *Bugzilla* organiza os BRs em três tipos de campos: anexos e dependências, pré-definidos (categorizações e identificações) e abertos (textuais).

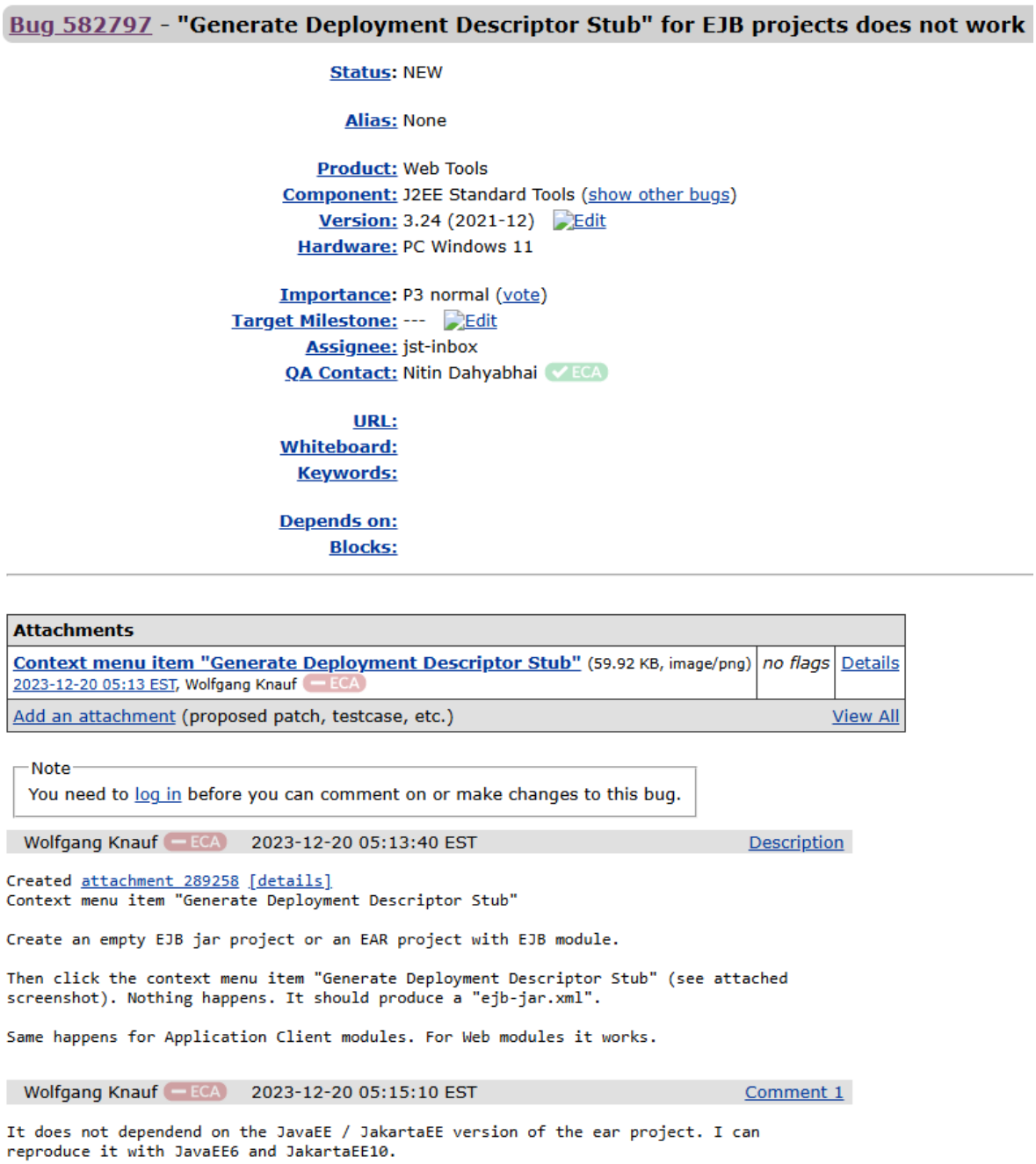
No campo de anexos e dependências, o relator pode anexar arquivos, como *screenshots*, *logs* ou códigos-fonte, que auxiliam na compreensão do problema. Além disso, o BR pode depender de outros BRs previamente registrados, indicando uma relação de dependência entre as falhas. Na Figura 1.1, é exemplificado esses anexos e dependências na seção de *attachments*, os quais são: "*Eclipse 4.9, GTK2 (29.58 KB, image/png)*"e "*Eclipse 4.9, GTK3 (31.98 KB, Image/png)*".

Já os campos pré-definidos fornecem informações categóricas e de identificação, preenchidos com valores pré-determinados para oferecer estrutura e organização ao relatório. Na Figura 1.1, é possível observar como exemplo de campos pré-definidos o *status* e *product*, os quais indicam o estado do *bug* e a área do sistema onde ocorreu a falha, respectivamente. Por fim, os campos abertos são destinados a informações descritivas e livres, permitindo que o relator detalhe o problema encontrado de forma completa e organizada. Os exemplos que podemos observar na Figura 1.1, são o título e a descrição do BR.

Os campos pré-definidos incluem informações como a severidade do *bug*, a prioridade, o componente afetado e a versão do *software*. Já os campos textuais abertos são projetados para fornecer uma explicação detalhada e em linguagem natural sobre o problema, permitindo uma rápida reprodução e compreensão da falha.

Esses campos textuais ajudam a simplificar o processo de rastreamento, entendimento, comunicação e correção de uma falha específica. Ao fornecer informações claras e detalha-

[1] https://www.bugzilla.org/



Bug 582797 - "Generate Deployment Descriptor Stub" for EJB projects does not work

Status: NEW

Alias: None

Product: Web Tools
Component: J2EE Standard Tools (show other bugs)
Version: 3.24 (2021-12) Edit
Hardware: PC Windows 11

Importance: P3 normal (vote)
Target Milestone: --- Edit
Assignee: jst-inbox
QA Contact: Nitin Dahyabhai ECA

URL:
Whiteboard:
Keywords:

Depends on:
Blocks:

Attachments

Context menu item "Generate Deployment Descriptor Stub" (59.92 KB, image/png) 2023-12-20 05:13 EST, Wolfgang Knauf ECA	*no flags*	Details
Add an attachment (proposed patch, testcase, etc.)		View All

Note
You need to log in before you can comment on or make changes to this bug.

Wolfgang Knauf ECA 2023-12-20 05:13:40 EST Description

```
Created attachment 289258 [details]
Context menu item "Generate Deployment Descriptor Stub"

Create an empty EJB jar project or an EAR project with EJB module.

Then click the context menu item "Generate Deployment Descriptor Stub" (see attached
screenshot). Nothing happens. It should produce a "ejb-jar.xml".

Same happens for Application Client modules. For Web modules it works.
```

Wolfgang Knauf ECA 2023-12-20 05:15:10 EST Comment 1

```
It does not dependend on the JavaEE / JakartaEE version of the ear project. I can
reproduce it with JavaEE6 and JakartaEE10.
```

Figura 1.1: Exemplo de um *bug report*.

das, os BRs no *Bugzilla* tornam o desenvolvimento colaborativo dentro de uma organização mais eficiente, permitindo que os desenvolvedores identifiquem rapidamente o defeito e implementem soluções eficazes. Além disso, o uso de anexos e dependências permite uma melhor gestão de como os *bugs* interagem entre si, garantindo que as correções sejam feitas de maneira coordenada e com um impacto mínimo em outras partes do sistema.

1.1 Motivação

Apesar da eficiência proporcionada pela documentação de BRs, a gestão desses documentos em grandes projetos pode ser onerosa e demorada devido à grande quantidade gerada diariamente. Por exemplo, apenas na plataforma *Bugzilla* do *Eclipse*, foram alterados 9001 BRs e criados 1537 BRs somente no ano de 2023.

Além da quantidade, é importante destacar que diversos desenvolvedores, frequentemente distribuídos em equipes diferentes, e até mesmo usuários, podem criar BRs. Nesse cenário, é comum que muitos desses relatórios sejam semelhantes ou até mesmo duplicados, o que pode resultar em redundâncias e comprometer a eficiência no processo de resolução de problemas.

Para mitigar essa ineficiência na resolução do problema, surgiram técnicas automatizadas que visam agrupar BRs semelhantes, permitindo que sejam tratados de maneira padronizada [68] [78] [13] [**?**]. Essas técnicas se baseiam em *Inteligência Artificial* (IA) [69], principalmente em *Processamento de Linguagem Natural* (PLN) [14], que permite a compreensão e manipulação automatizada de textos. O uso de PLN possibilita a identificação de padrões e similaridades nos textos dos BRs, facilitando a categorização e priorização dos problemas relatados.

Com as técnicas automatizadas para agrupar BRs semelhantes, não apenas se reduz o tempo necessário para gerenciar e analisar os BRs, mas também se cria a oportunidade de utilizar BRs similares como modelos para a resolução de problemas futuros. Isso pode levar a uma padronização nos métodos de correção e a uma melhoria contínua nos processos de desenvolvimento, resultando em um *software* mais robusto e confiável. Além disso, a automação desse processo libera os desenvolvedores para focarem em tarefas mais complexas e criativas, aumentando a eficiência geral da equipe.

Para facilitar o gerenciamento dos BRs semelhantes, os sistemas de recomendação surgem como uma solução promissora, utilizando técnicas de PLN. Duas abordagens fundamentais neste campo são o *TF-IDF* (*Term Frequency-Inverse Document Frequency*) [51] e os *word embeddings* [54].

O *TF-IDF* (*Term Frequency-Inverse Document Frequency*) é uma técnica que avalia a importância de uma palavra em um documento dentro de um conjunto, enquanto os *word embeddings* são representações vetoriais de palavras que capturam relações semânticas. Visando explorar a recomendação de BRs similares, diversos modelos têm sido desenvolvidos utilizando estas técnicas, juntamente com cálculos de distância entre vetores. O *TF-IDF* tem sido amplamente utilizado para identificar termos relevantes, enquanto os *word embeddings* oferecem uma compreensão mais profunda da semântica textual, permitindo alcançar melhores resultados na análise e recomendação de BRs similares.

1.2 Problema

A identificação de *bug reports* (BRs) similares é uma tarefa essencial em projetos de desenvolvimento de software devido à sua natureza colaborativa e à grande quantidade de BRs gerados diariamente. No entando, realizar essa identificação manualmente é um processo demorado, sujeito a erros e que pode impactar negativamente a agilidade na alocação de atividades e na resolução de problemas recorrentes. Nesse contexto, a detecção automática de BRs similares surge como uma solução promissora para otimizar o fluxo de trabalho.

Embora existam trabalhos que abordam a recomendação de BRs similares [68] [78] [13] [12], ainda há uma lacuna significativa na exploração do estado da arte em *Semantic Textual Similarity* (STS) para este problema. De acordo com o benchmark do *Massive Textual Embedding Benchmark* (MTEB) [57], o modelo T5 é considerado o estado da arte em STS, destacando-se por sua precisão e eficácia em tarefas de similaridade textual.

O uso do estado da arte em STS, traz avanços potenciais significativos para a detecção de BRs similares. Por sua capacidade de capturar relações semânticas profundas entre textos, o T5 pode superar métodos tradicionais, como *TF-IDF* e *word embeddings*, ao proporcionar maior precisão e robustez na recomendação de BRs. Dessa forma, a integração de soluções

baseadas em T5 tem o potencial de melhorar substancialmente a eficiência do processo de desenvolvimento, reduzindo redundâncias e agilizando a resolução de problemas.

1.3 Abordagem Proposta

Dada a importância de identificar automaticamente BRs similares com qualidade, o modelo T5 foi destacado como referência na literatura para tarefas de *Similaridade Textual Semântica* (STS) [57]. Através dos dados de BRs recuperados durante o desenvolvimento do trabalho, a pesquisa investiga o uso de uma rede neural para classificar e ranquear BRs similares, empregando três abordagens de vetorização: o método TF-IDF, um módulo de vetorização do modelo T5 e uma abordagem híbrida que combina ambas. Optou-se pelo módulo de vetorização T5 por lidar bem com sequências longas e captar o sentido semântico textual. O *TF-IDF*, por sua vez, é uma técnica clássica que avalia a importância relativa de termos em documentos.

Para otimizar a rede neural, foram utilizados como *features* os vetores e as distâncias de cosseno entre eles. Além disso, foram testadas diferentes configurações de balanceamento, pré-processamento, pesos de classe, otimizadores e camadas de ativação para encontrar a melhor configuração possível.

Para avaliar o treinamento e o ajuste fino dos modelos, foram utilizadas várias métricas, como acurácia, precisão, revocação e *F1-Score* [64], que ajudam a medir a capacidade do modelo de classificar corretamente os dados. A curva ROC e sua área sob a curva (AUC) [24] avaliam a capacidade do modelo de distinguir entre classes, enquanto a AUC PR [17] é usada em cenários de desbalanceamento extremo. Para ranqueamento, métricas como *Precision@K* e *Recall@K* [51] medem a precisão e a revocação das recomendações do sistema, proporcionando uma avaliação abrangente do modelo.

1.4 Objetivos

Este trabalho tem como objetivo aplicar técnicas de aprendizado de máquina para aprimorar a gestão de BRs no contexto do *Bugzilla* do *Eclipse*. Especificamente, a pesquisa busca propor e avaliar um modelo que utiliza a arquitetura T5 para sugerir BRs, ajustando-o por

meio de diferentes estratégias. Desta forma, descrevemos os seguintes objetivos específicos:

- **Utilizar do modelo T5 no contexto de *bug reports* similares**: Identificar se o uso do T5 ou da abordagem híbrida são promissores para a classificação de BRs similares.
- **Realizar ajustes no modelo no contexto de BRs, buscando melhorias**: Explorar diferentes formas de melhorar o modelo proposto inicialmente, principalmente no desempenho de métricas de avaliação do modelo.
- **Conduzir uma série de experimentos para analisar os modelos comparando-os**: realizar experimentos para avaliar a performance das abordagens e compará-las entre si em termos de acurácia dos modelos.

1.5 Escopo

Para o desenvolvimento do trabalho, utilizamos dados do *Bugzilla* do *Eclipse*, filtrando BRs que estão marcados como *FIXED* ou *CLOSED*. Escolhemos lidar apenas com BRs marcados dessa forma, porque esses *status* indicam que o problema foi resolvido. Isso garante que os dados analisados sejam consistentes e completos, permitindo uma avaliação precisa das alterações feitas para corrigir o *bug*, finalizando o ciclo de vida do BR.

Além das marcações de *FIXED* ou *CLOSED*, utilizamos apenas BRs que tivessem dados do *Gerrit*[2] disponíveis. O *Gerrit* é uma ferramenta que ajuda equipes de desenvolvimento de *software* a revisarem e colaborarem em código fonte. Considerando que a similaridade entre os BRs é determinada pela modificação dos mesmos arquivos de código ao corrigir um *bug* descrito por um BR, a utilização dos dados do *Gerrit* é fundamental para o estudo, pois foca na alteração dos arquivos de cada BR.

O escopo deste trabalho é delimitado pela utilização de três abordagens principais: vetores T5, TF-IDF e redes neurais. Os vetores T5, derivados do modelo T5 (*Text-to-Text Transfer Transformer*) [65], são utilizados para capturar representações semânticas ricas dos textos dos BRs. O TF-IDF (*Term Frequency-Inverse Document Frequency*) [51] é empregado para avaliar a importância relativa das palavras nos documentos. Por fim, as redes neurais

[2] https://www.gerritcodereview.com/

(RNs) [30] são aplicadas para aprender padrões complexos e relações não lineares entre os BRs.

1.6 Hipóteses

Com base nos objetivos e na abordagem proposta descritos anteriormente, formulamos as seguintes hipóteses para orientar nossa investigação:

Hipótese 1: Desempenho dos Modelos

- $\mathbf{H}_{01}$: Não há diferença significativa no desempenho entre o modelo T5, o método *TF-IDF* ou a abordagem híbrida na classificação ou no ranqueamento de *Bug Reports* (BRs) similares.

- $\mathbf{H}_{11}$: O modelo T5, o método *TF-IDF* ou a abordagem híbrida apresentam desempenho significativamente superior aos outros na classificação ou no ranqueamento de BRs similares.

Hipótese 2: Impacto das Estratégias de Modelagem

- $\mathbf{H}_{02}$: As diferentes estratégias de balanceamento, pré-processamento, pesos de classe, otimizadores e camadas de ativação não afetam significativamente o desempenho do modelo na detecção de BRs similares.

- $\mathbf{H}_{12}$: Pelo menos uma das estratégias de balanceamento, pré-processamento, pesos de classe, otimizadores ou camadas de ativação afeta significativamente o desempenho do modelo na detecção de BRs similares.

Hipótese 3: Vetores Completos vs. Distância de Cosseno

- $\mathbf{H}_{03}$: O uso de vetores completos como *features* não apresenta vantagem significativa em relação ao uso da distância de cosseno na classificação ou na recomendação de BRs similares.

- $\mathbf{H}_{13}$: O uso de vetores completos como *features* apresenta vantagem significativa em relação ao uso da distância de cosseno na classificação ou na recomendação de BRs similares.

Estas hipóteses foram formuladas com base na revisão da literatura e nos objetivos específicos do estudo. Elas serão testadas através dos experimentos e análises descritos na metodologia, utilizando as métricas de avaliação mencionadas anteriormente, como acurácia, precisão, revocação, *F1-Score*, curva ROC, AUC, AUC PR, *Precision@K* e *Recall@K*.

A hipótese nula (H_0) representa a ausência de efeito ou diferença significativa entre as abordagens testadas. As hipóteses alternativas (H_1) propõem relações específicas e efeitos esperados com base em nossa abordagem proposta.

Ao testar estas hipóteses, buscamos não apenas validar a eficácia de nossa abordagem proposta, mas também contribuir para o avanço do conhecimento na área de detecção automática de *Bug Reports* similares.

1.7 Contribuições

Desenvolvemos um modelo que recomenda *Bug Reports* (BRs) similares, com o objetivo de auxiliar a área de engenharia de *software* em grandes projetos, tornando as atividades de desenvolvimento mais eficientes. Além disso:

- Realizamos um estudo comparativo entre diversas abordagens de detecção de BRs similares, colaborando com a comunidade científica de IA para avaliar a eficácia da abordagem híbrida e do modelo T5 no contexto de similaridade para BRs.

- Avaliamos 56 combinações de modelos aplicando diversas estratégias de vetorização, balanceamento de dados e pré-processamento. Os resultados mostraram que:

  - O uso de vetores completos como *features* foi mais eficaz que a distância de cosseno.

  - As abordagens híbridas e *TF-IDF* isolada se destacaram como promissoras.

  - O pré-processamento não teve impacto significativo.

- No ajuste fino, testamos 168 combinações de hiperparâmetros, com os otimizadores *Adam* e *RMSprop* apresentando os melhores desempenhos.

1.8 Relevância

A relevância do trabalho tem foco em dar suporte para alocação de desenvolvedores em suas atribuições, evitando um esforço manual humano que poderá ser realizado por IA. Ao final deste trabalho, busca-se auxiliar grandes projetos, tornando o planejamento das próximas atividades do desenvolvedor mais fácil e rápido, contribuindo em:

- Facilitar o Trabalho de Desenvolvedores: Desenvolvedores geralmente podem corrigir *bugs* semelhantes em um tempo mais curto e com uma qualidade superior, pois poderão se concentrar em menos arquivos de código.

- Apoiar a Tomada de Decisões: Gerentes poderão usar a análise de *bugs* similares para atribuir um programador a resolver um problema similar ao que ele já tenha resolvido anteriormente, melhorando a alocação de recursos.

- Fomentar a Comunicação Eficiente entre Equipes: Identificação de BRs similares poderá ser útil em fomentar a comunicação entre o desenvolvedor atribuído para resolver o problema e o programador que solucionou o *bug* similar.

A detecção de itens similares é um campo em constante evolução, impulsionado pelos avanços tecnológicos e pela emergência de novas metodologias. Devido à vastidão de dados disponíveis e à complexidade dos problemas de similaridade, pesquisadores frequentemente exploram abordagens diversificadas em variados contextos de aplicação. O uso de técnicas avançadas, como redes neurais e modelos de linguagem de ponta, tem revolucionado a maneira os textos são analisados, permitindo uma identificação de padrões de similaridade, não captados por métodos tradicionais. Embora já existam métodos automatizados para detectar similaridades BRs, as pesquisas nessa área continuam avançando com o desenvolvimento de algoritmos cada vez mais eficientes e precisos.

A análise detalhada das estratégias de vetorização, balanceamento de dados e pré-processamento revelou *insights* valiosos sobre o desempenho dos modelos em diferentes

métricas. A escolha criteriosa de hiperparâmetros, como otimizadores e funções de ativação, demonstrou ter um impacto significativo na eficácia dos modelos, destacando a importância de um ajuste fino. Esses achados não apenas respondem às questões de pesquisa propostas, mas também fornecem diretrizes práticas para futuras implementações, contribuindo para a melhoria contínua da qualidade e eficiência dos processos de desenvolvimento de *software*.

1.9 Organização do Documento

Esta dissertação está estruturada de forma a guiar o leitor através de nossa investigação sobre a recomendação de BRs similares. No Capítulo 2, apresentamos a fundamentação teórica e os conceitos essenciais, seguidos por uma análise dos trabalhos relacionados no Capítulo 5. A metodologia detalhada de nosso estudo é descrita no Capítulo 3, enquanto os resultados das análises e experimentos, incluindo o desempenho das redes neurais, são apresentados e discutidos no Capítulo 4. Finalmente, no Capítulo 6, sintetizamos nossas descobertas e oferecemos sugestões para pesquisas futuras.

Capítulo 2

Fundamentação teórica

Este capítulo oferece uma visão geral de 2.1 Bugzilla, 2.2 Gerrit, 2.3 Aprendizado de Máquina, 2.4 Processamento de Linguagem Natural (PLN), 2.5 Vetorização Textual e 2.6 *Word Embeddings*. Essa exposição é essencial para uma compreensão mais aprofundada das abordagens propostas nesta pesquisa para a detecção e recomendação de BRs similares.

2.1 Bugzilla

Bugzilla é um sistema de rastreamento de *bugs* de código aberto, popular entre desenvolvedores e gerentes de *software* [56]. Foi criado em 1998 por Terry Weissman como uma ferramenta interna para a *Netscape Communications Corporation*, mas posteriormente foi lançado como um projeto de código aberto sob a Licença Pública Mozilla.

O Bugzilla fornece uma interface baseada na *web* para submissão, rastreamento e gerenciamento de *bugs* e outros problemas no desenvolvimento do *software*. Ele permite que desenvolvedores e interessados relatem, rastreiem e resolvam falhas de forma colaborativa. Essa plataforma disponibiliza recursos como campos personalizados, notificações por e-mail e ferramentas avançadas de pesquisa, configurando-se como uma solução versátil para o rastreamento de *bugs* em projetos de *software*.

Normalmente, esses relatos são feitos através de um documento chamado *bug report* (BR) na própria plataforma do Bugzilla. Além de ajudar os desenvolvedores a identificar e corrigir *bugs*, os BRs também servem como um *feedback* para os usuários do *status* de correção do *bug* em questão [81].

BRs normalmente incluem informações sobre o produto, o ambiente no qual o *bug* ocorreu e os passos tomados para reproduzir o problema. Em teoria, quanto mais informações o BR tem, mais fácil mapear e resolver a falha. No formulário do Bugzilla do Eclipse, por exemplo, os campos de criação de um BR são:

1. **Summary**: Título ou descrição resumida sobre o *bug*, tendo como objetivo identificar de forma mais sucinta um problema.

2. **Product**: Específica o *software* ou projeto afetado pelo bug, visando encontrar a equipe responsável por resolver o problema.

3. **Version**: a versão em particular do *software* na qual o *bug* foi encontrado.

4. **Component**:: Detalha qual módulo ou componente o *bug* está presente, facilitando o processo de *debugging*.

5. **What did you do? (steps to reproduce)**: Descreve os passos de reprodução de um determinado *bug*, com finalidade de facilitar a reprodução de determinada falha.

6. **What happened? (actual result)**: Descreve o que aconteceu depois de realizar o processo de reprodução.

7. **What should have happened (expected results)**: Qual era o resultado correto que deveria ter acontecido após os passos de reprodução.

8. **Attach a file**: permite anexar arquivos para ajudar na compreensão do *bug*.

9. **Bug type**: permite classificar o *bug* em sua gravidade, prioridade ou outras classificações predefinidas.

10. **Security**: Indica se esse BR deve manter-se longe do alcance público até ser resolvido.

Após a criação de um BR, para acompanhar o progresso na resolução de problemas e facilitar a comunicação entre desenvolvedores, testadores e usuários, existem os campos "*resolution*" e "*status*". Os possíveis valores para o campo "*resolution*" têm como objetivo descrever o desfecho final de cada relatório, podendo ser:

1. **Fixed**: indica que o problema foi corrigido e integrado ao código-fonte.

2. **Duplicate**: o mesmo problema já foi reportado em outro relatório.

3. **Invalid**: o relatório não descreve um problema real.

4. **Won"t Fix**: a equipe decidiu não corrigir o problema.

5. **Works as Designed**: o comportamento observado é intencional.

6. **Incomplete**: falta de informações suficientes para investigar.

7. **Cannot Reproduce**: a equipe não conseguiu reproduzir o problema.

Já o "*status*" do BR tem como objetivo passar o progresso geral da resolução de um determinado *bug*. podendo ser descrito por:

1. **UNCONFIRMED**: indica que o BR foi submetido, mas ainda não foi confirmado como um problema genuíno ou que o BR precise de mais informações ou que ainda não tenha sido revisado pela equipe responsável pelo gerenciamento.

2. **NEW**: indica que o BR foi revisado e aceito, mas ainda não foi atribuído a alguém para correção.

3. **ASSIGNED**: indica que o *bug* é atribuído para o responsável pela correção.

4. **RESOLVED**: indica que o responsável pela correção acredita ter corrigido o problema.

5. **REOPENED**: indica que *bug* marcado como resolvido foi encontrado novamente ou a solução implementada não resolveu completamente o problema.

6. **VERIFIED**: Indica que solução implementada pelo desenvolvedor ou equipe passou nos testes.

7. **CLOSED**: Problema foi completamente resolvido e não requer mais atenção.

Na Figura 2.1 [11], é descrito o ciclo de vida do BR e suas várias etapas do campo "*status*" e "*resolution*". A importância dos BRs do Bugzilla reside no fato de que eles constituem a fonte primária de dados para este estudo. Para desenvolver uma inteligência capaz de identificar BRs semelhantes, é fundamental dispor de informações adequadas para treinar, validar e testar os modelos aplicados nesta análise.

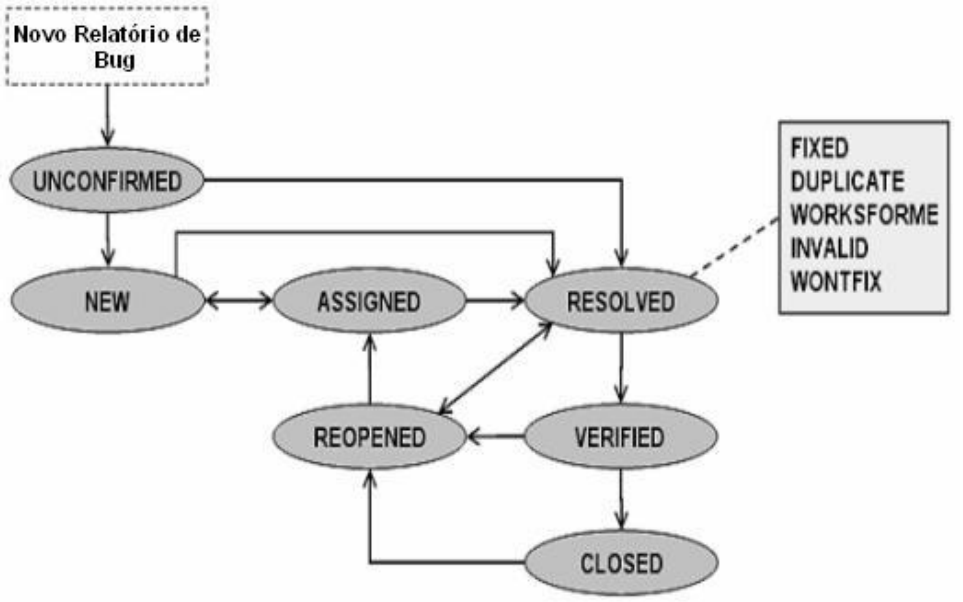


Figura 2.1: Ciclo de vida de *bug report* no Bugzilla [11].

2.2 Gerrit

Desenvolvido inicialmente pelo Google, o Gerrit[1] é uma plataforma de revisão de código utilizada em ambientes de desenvolvimento de *software* colaborativo. Nessa plataforma, as alterações submetidas pelos desenvolvedores são avaliadas por outros desenvolvedores que exercerão o papel de revisores.

O objetivo do Gerrit é facilitar a comunicação entre os revisores e o autor da alteração, permitindo discussões detalhadas e interativas sobre o código proposto. Portanto, essa plataforma contribui para a promoção da qualidade do código ao permitir que revisores façam sugestões de forma rápida e eficiente, antes que as alterações sejam integradas ao código-base.

Nos BRs do Bugzilla, existe um campo responsável por guardar *links* úteis, como *links* para: BRs relacionados, *patches*, revisões de código, discussões e comentários e documentação relacionada. Parte desses BRs tem entre seus *links* úteis revisões realizadas através do Gerrit.

A relevância dos dados do processo de correção de *bugs* disponíveis no Gerrit reside na capacidade de identificar quais arquivos foram modificados para corrigir o *bug*. Ao ter

[1] https://www.gerritcodereview.com/

acesso aos arquivos alterados, podemos determinar se os BRs são ou não semelhantes com base nos arquivos modificados em comum.

2.3 Aprendizagem de Máquina

O aprendizado de máquina (AM) é um ramo da inteligência artificial (IA) que se concentra no desenvolvimento de algoritmos capazes de aprender a partir de dados [40]. Em vez de serem explicitamente programados para realizar uma tarefa, esses algoritmos são treinados usando grandes volumes de dados, permitindo-lhes aprender padrões e realizar previsões ou tomar decisões com base nesses padrões. Existem quatro principais modelos de AM:

1. **Supervisionado**: Neste tipo, os modelos aprendem a partir de um conjunto de dados rotulados, onde cada entrada possui uma resposta correta associada. O modelo é treinado para prever essas respostas com base nos dados de entrada. Por exemplo, classificar e-mails como *spam* ou não *spam* com base em exemplos previamente rotulados.

2. **Não supervisionado**: Aqui, os modelos lidam com dados não rotulados e tentam encontrar padrões ou estrutura neles [41]. Eles agrupam dados semelhantes em *clusters* ou detectam anomalias sem a necessidade de exemplos rotulados. Um exemplo é a segmentação de clientes com base em seus hábitos de compra, onde o modelo identifica naturalmente diferentes grupos de clientes.

3. **Semi-supervisionado**: Esta abordagem combina elementos dos dois anteriores. O modelo é treinado com um conjunto de dados que contém tanto exemplos rotulados quanto não rotulados. A ideia é que os dados não rotulados forneçam informações adicionais úteis para melhorar o desempenho do modelo ao prever os rótulos dos dados rotulados.

4. **Por reforço**: Neste caso, os modelos aprendem através da interação com um ambiente, recebendo *feedback* na forma de recompensas ou penalidades [3]. Eles tomam decisões sequenciais para maximizar a recompensa ao longo do tempo. Um exemplo é um algoritmo de aprendizado de máquina treinado para jogos, onde recebe recompensas

ao fazer movimentos corretos e penalidades por movimentos errados, ajustando assim sua estratégia ao longo do tempo.

As redes neurais (RNs) [30], fundamentais no campo do aprendizado de máquina, são estruturas computacionais inspiradas no funcionamento do cérebro humano. Essas redes consistem em unidades básicas chamadas neurônios artificiais, organizadas em camadas interconectadas. Cada neurônio processa informações recebidas, transmitindo-as para os neurônios subsequentes na rede [33].

As RNs são uma das abordagens mais poderosas dentro do AM, devido à sua capacidade de aprender representações complexas e hierárquicas dos dados. Ao serem expostas a exemplos de entrada e saída, as redes neurais ajustam seus parâmetros internos por meio de aprendizado supervisionado, minimizando a diferença entre as respostas previstas e as respostas reais.

Essa diferença é quantificada por uma função de perda (em inglês, *loss function*), que serve como um guia para o ajuste dos pesos da rede durante o treinamento. A função de perda mais comum em problemas de classificação é a entropia cruzada (*cross-entropy*), que avalia a diferença entre as distribuições preditas e as reais. No caso de classificação binária, essa função de perda assume a forma de entropia cruzada binária (*binary cross-entropy*).

Para evitar o *overfitting*, que ocorre quando um modelo se ajusta excessivamente aos dados de treinamento, técnicas como o early stopping são empregadas. O *early stopping* monitora a performance do modelo em um conjunto de validação e interrompe o treinamento quando a métrica de validação não melhora após um número pré-definido de épocas. Isso assegura que o modelo mantenha a capacidade de generalização para novos dados.

Essa capacidade de aprender a partir dos dados torna as redes neurais extremamente versáteis e aplicáveis em uma ampla gama de domínios. Desde reconhecimento de padrões em imagens e processamento de linguagem natural até diagnóstico médico e previsão de séries temporais, as redes neurais têm demonstrado um desempenho excepcional em uma variedade de tarefas complexas.

Explicar os conceitos de AM e RNs neste trabalho é fundamental para oferecer uma compreensão mais aprofundada do contexto teórico e metodológico subjacente ao desenvolvimento do modelo de IA. Esses conceitos não apenas fornecem uma estrutura teórica para interpretar o desempenho do modelo em relação às métricas de avaliação, mas também

possibilitam uma análise mais sólida e abrangente dos resultados obtidos.

2.4 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN) é uma área interdisciplinar entre linguística e a computação que tem como objetivo permitir que sistemas computacionais reconheçam, interpretem e gerem linguagem humana [14]. Essencialmente, o PLN funciona como uma interface entre a linguagem humana e as operações computacionais, com o objetivo de capacitar os sistemas a compreender e gerar textos de forma semelhante à maneira como os humanos o fazem.

Para as máquinas, decifrar nuances na linguagem humana é uma tarefa desafiadora, tendo em vista a necessidade de compreender contextos complexos, tonalidades de voz, extrair informações, interpretar significados, analisar sentimentos e realizar diversas operações linguísticas, como análise sintática, semântica, lexical e morfológica.

Atualmente, o PLN engloba uma variedade de aplicações práticas, como assistentes virtuais, motores de busca, sistemas de recomendação, análise de sentimentos, processamento automático de documentos, entre outros. À medida que a tecnologia avança e os dados se tornam mais abundantes, o campo do PLN continua a evoluir, tornando-se mais preciso e sofisticado.

Uma etapa fundamental no PLN é o pré-processamento textual, que prepara e estrutura o texto para análise subsequente. Esta etapa é crucial para melhorar a qualidade dos dados e a eficácia dos algoritmos de PLN. Algumas das técnicas mais comuns de pré-processamento incluem:

1. **Tokenização**: estratégia mais básica de pré-processamento e consiste em dividir o texto em tokens, que podem ser palavras, sílabas ou caracteres, de acordo com a granularidade desejada. Partindo do texto "All filter in Assign dialog for Custom Categories does not show term definitions", o texto tokenizado por palavras seria: ["All", "filter", "in", "Assign", "dialog", "for", "Custom", "Categories", "does", "not", "show", "term", "definitions"].

2. **Remoção de *stopwords***: *Stopwords* são palavras que ocorrem frequentemente em um

texto, mas não carregam significado semântico importante para a análise, como por exemplo "the", "and", "or", entre outros. Para realizar a remoção de *stopwords*, precisamos primeiro identificá-las para depois excluí-las, com o objetivo de reduzir o ruído nos dados e concentrar o foco nas palavras mais significativas para a análise. Após a remoção de *stopwords* o exemplo citado ficaria ["All", "filter", "Assign", "dialog", "Custom", "Categories", "show", "term", "definitions"], removendo os tokens: "in", "for", "does", "not".

3. **Lematização**: técnica que visa reduzir as palavras flexionadas à sua forma base, representando sua raiz lexical, Por exemplo, o lema das palavras "correr", "corria"e "corrido"é "correr". O objetivo por trás da lematização é normalizar o texto e reduzir a variabilidade das palavras, agrupando variantes morfológicas da mesma palavra, facilitando a interpretação dos dados. Após o processo de remoção de *stopwords* e lematização e ficaria: ["All", "filter", "Assign", "dialog", "Custom", "Category", "show", "term", "definition"], transofrmando o "Categories"em "Category"e "definitions"em "definition".

Após o pré-processamento, o texto tem sua dimensionalidade reduzida, removendo informações irrelevantes ou redundantes. Como resultado, o texto pré-processado é mais adequado para ser representado numericamente, através de diferentes técnicas de vetorização, que são essenciais para converter a linguagem natural em uma forma que os modelos computacionais possam compreender.

2.5 Vetorização textual

A vetorização textual é uma técnica de PLN que permite representar dados textuais em formato numérico compreensível por algoritmos de aprendizado de máquina. Na maioria das aplicações, parte dos dados são constituídos por uma grande variedade de textos, como documentos, *tweets*, e-mails, etc. Apesar disso, os modelos de aprendizado de máquina, incluindo aqueles usados em PLN, operam essencialmente com números. Isso ocorre porque os algoritmos de aprendizado de máquina são baseados em cálculos matemáticos que envolvem manipulação de números. Portanto, converter texto em vetores numéricos é crucial para que

esses modelos possam compreender e processar a linguagem. Por exemplo, as técnicas de vetorização discutidas nas subseções subsequentes que abordam *Bag of Words* (BoW) e *Term Frequency Inverse Document Frequency* (TF-IDF).

2.5.1 Bag of Words (BoW)

Bag of Words (BoW) ou "Saco de Palavras"é uma técnica de PLN para representar texto de forma simples. Primeiramente, cria-se um vocabulário, composto por todas as palavras distintas encontradas nos documentos [80]. Em seguida, é criado um vetor para cada documento, onde as dimensões correspondem às palavras do vocabulário. Para cada aparição de uma palavra no documento, o valor da dimensão correspondente é incrementado em 1, como é ilustrado na Figura 2.2 [61].

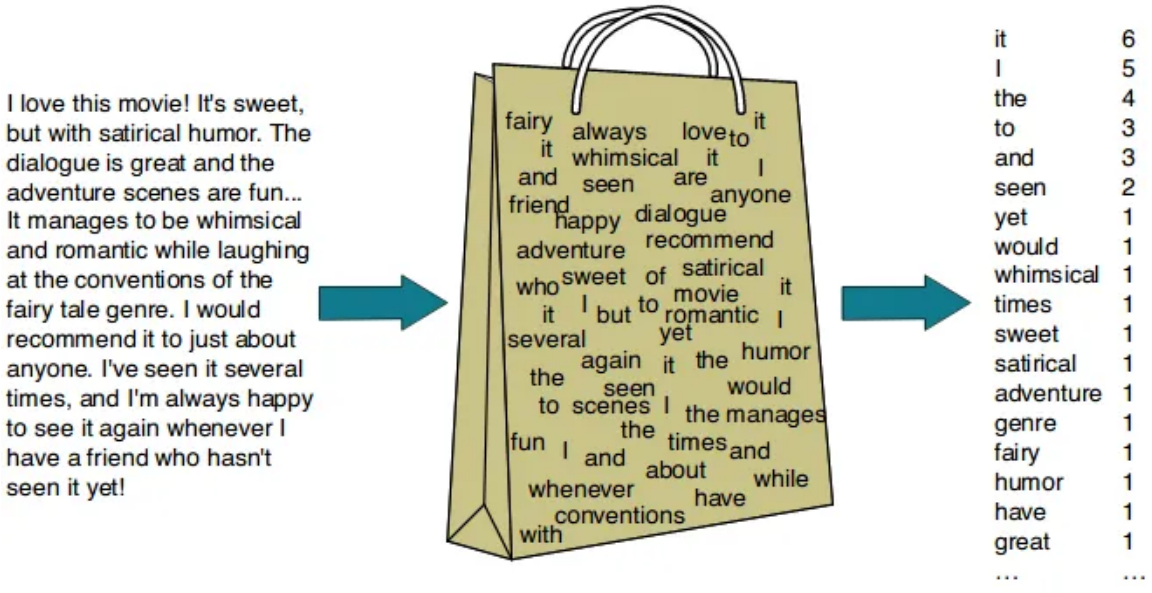


Figura 2.2: Exemplo gráfico do BoW [61].

Apesar de muito simples, prático e eficaz, BoW pode ser aplicado em muitas situações e ser aplicado com técnicas mais avançadas. Entretanto, destacam-se como pontos negativos que ele não considera a ordem das palavras, não considera o sentido semântico da palavra e o tamanho do vetor gerado, caso seja comparado a um conjunto de dados com vocabulário extenso. Por exemplo, digamos que o conjunto de dados consiste em dois documentos tokenizados, como mostrado na Tabela 2.1.

Tabela 2.1: Exemplo de texto tokenizado.

ID	Documento tokenizado
460290	["Send", "Dialog", "preview", "should", "not", "try", "to", "render", "null", "exceptions"]
470891	["[Rhino]", "NPE", "when", "script", "returns", "null"]

Primeiramente precisa ser criado o vocabulário com todas as palavras únicas nos dois documentos, constituído por: ["Send", "Dialog", "preview", "should", "not", "try", "to", "render", "null", "exceptions", "[Rhino]", "NPE", "when", "script", "returns"]. Depois gera-se um vetor BoW para cada documento, o qual cada vetor representa a frequência de cada palavra do vocabulário nos documentos correspondentes, descrito na Tabela 2.2.

Tabela 2.2: Exemplo de documentos vetorizados com BoW

ID	Documento vetorizado com BoW
460290	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]
470891	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1]

2.5.2 TF-IDF

TF-IDF, abreviação de *Term Frequency-Inverse Document Frequency*, é uma técnica fundamental no processamento de linguagem natural (PLN) para avaliar a importância de palavras em documentos. Esta técnica é amplamente utilizada na vetorização textual, transformando palavras em vetores numéricos que representam sua relevância [47].

Conceito e Cálculo

O valor TF-IDF de uma palavra aumenta proporcionalmente com sua frequência no documento (TF) e diminui com sua frequência em toda a coleção de documentos (IDF). O cálculo é realizado em três etapas:

1. **Term Frequency (TF):** Contagem do número de ocorrências de uma palavra no documento, dividida pelo número total de palavras no documento.

2. **Inverse Document Frequency (IDF):** Logaritmo do número total de documentos dividido pelo número de documentos que contêm a palavra.

3. **TF-IDF:** Multiplicação dos valores de TF e IDF.

A fórmula completa do TF-IDF é representada pela Equação 2.1:

$$TF - IDF = TF \times IDF = \frac{f(d,t)}{f(d)} \times \log\left(\frac{N(D)}{N(D,T)}\right) \tag{2.1}$$

Onde:

- **t**: palavra ou token escolhida para o cálculo.
- **d**: Documento em que a palavra foi encontrada.
- **f(d, t)**: Frequência de t em d.
- **f(d)**: Quantidade total de palavras em d.
- **D**: Conjunto completo de documentos.
- **N(D)**: Número total de documentos.
- **N(D, t)**: Número de documentos contendo t.

Exemplo Prático

Para ilustrar o cálculo do TF-IDF, consideremos os documentos descritos na Tabela 2.1. Analisaremos em detalhes duas palavras: "Dialog"e "null".

Palavra "Dialog":

- TF: $1/7$ no documento 460290 (1 ocorrência em 7 palavras).
- IDF: $\log(2/1) \approx 0.301$ (aparece em 1 de 2 documentos).
- TF-IDF: $1/7 \times 0.301 \approx 0.043$ para o documento 460290.

Palavra "null":

- TF: $1/7$ no documento 460290 e 1/5 no documento 470891.
- IDF: $\log(2/2) = 0$ (aparece em todos os documentos).
- TF-IDF: 0 para ambos os documentos.

Observe que "null", por ser comum a todos os documentos, recebe um TF-IDF de 0, demonstrando como esta técnica reduz a importância de palavras muito frequentes.

Resultados e Vetorização

As Tabelas 2.3, 2.4, e 2.5 apresentam os valores de TF, IDF e TF-IDF calculados para cada palavra nos documentos do exemplo.

Tabela 2.3: Valores de TF por palavra.

Palavra	ID	Valor de TF
Dialog	460290	1/7
NPE	470891	1/5
Send	460290	1/7
Rhino	470891	1/5
exceptions	460290	1/7
null	460290	1/7
null	470891	1/5
preview	460290	1/7
render	460290	1/7
returns	470891	1/5
script	470891	1/5
try	460290	1/7

A Tabela 2.6 mostra a representação vetorial final dos documentos usando TF-IDF.

Limitações e Considerações

Apesar de sua eficácia em representar a importância das palavras, o TF-IDF compartilha algumas limitações com o BoW:

Tabela 2.4: Valores de IDF por palavra.

Palavra	Valor de IDF
Dialog	$\log(2/1) \approx 0.693$
NPE	$\log(2/1) \approx 0.693$
Send	$\log(2/1) \approx 0.693$
Rhino	$\log(2/1) \approx 0.693$
exceptions	$\log(2/1) \approx 0.693$
null	$\log(2/2) = 0$
preview	$\log(2/1) \approx 0.693$
render	$\log(2/1) \approx 0.693$
returns	$\log(2/1) \approx 0.693$
script	$\log(2/1) \approx 0.693$
try	$\log(2/1) \approx 0.693$

Tabela 2.5: Valores de TF-IDF por palavra.

Palavra	ID	Valor de TF-IDF
Dialog	460290	$(1/7) * 0.693 \approx 0.099$
NPE	470891	$(1/5) * 0.693 \approx 0.139$
Send	460290	$(1/7) * 0.693 \approx 0.099$
Rhino	470891	$(1/5) * 0.693 \approx 0.139$
exceptions	460290	$(1/7) * 0.693 \approx 0.099$
null	460290	$(1/7) * 0 = 0$
null	470891	$(1/5) * 0 = 0$
preview	460290	$(1/7) * 0.693 \approx 0.099$
render	460290	$(1/7) * 0.693 \approx 0.099$
returns	470891	$(1/5) * 0.693 \approx 0.139$
script	470891	$(1/5) * 0.693 \approx 0.139$
try	460290	$(1/7) * 0.693 \approx 0.099$

Tabela 2.6: Vetorização TF-IDF para os documentos.

ID	Documento vetorizado com TF-IDF
460290	[0.099, 0, 0.099, 0, 0.099, 0, 0.099, 0.099, 0, 0, 0.099]
470891	[0, 0.139, 0, 0.139, 0, 0, 0, 0, 0.139, 0.139, 0]

- Não considera o contexto semântico das palavras.
- Pode gerar vetores de alta dimensionalidade para vocabulários extensos.

A compreensão dessas técnicas de vetorização textual, incluindo pré-processamento, BoW, e TF-IDF, é crucial para este trabalho. Elas formam a base para transformar os textos dos BRs em representações que podem ser eficientemente processadas e analisadas por modelos de aprendizado de máquina, facilitando a análise automatizada e a tomada de decisões baseadas nos conteúdos dos BRs.

2.6 Word Embeddings

Na última década, os avanços na área de PLN foram impulsionados por uma inovação crucial: os *word embeddings*. Essas representações de texto em um espaço vetorial têm sido uma verdadeira revolução, permitindo que os modelos compreendam a linguagem humana de uma maneira mais refinada [54]. Antes dos *word embeddings*, métodos mais tradicionais, como o BoW e TF-IDF, eram amplamente utilizados para representar o texto. Embora úteis em muitos cenários, essas abordagens não consideram a semântica das palavras nem a relação entre elas, tratando cada palavra de forma independente.

Com os *word embeddings*, as palavras são representadas como vetores em um espaço multidimensional, onde palavras semanticamente semelhantes estão próximas umas das outras. Isso significa que palavras com significados semelhantes são mapeadas para regiões próximas no espaço vetorial, permitindo que os modelos compreendam relações semânticas e contextuais, como mostrado na Figura 2.3 [31].

O principal ponto negativo dessa abordagem é a exigência de um treinamento longo e o uso de uma grande quantidade de dados para que o modelo construa esse espaço multidimensional. Para contornar essa limitação, foram desenvolvidos modelos pré-treinados, que permitem reutilizar um modelo que já resolve um problema semelhante como ponto de partida para resolver outro, otimizando assim o processo de treinamento.

Uma das abordagens que utilizam *word embeddings* e possui sua versão pré-treinada proposta por Mikolov em 2013 [54] [55], o Word2Vec aprende representações vetoriais de palavras a partir de grandes coleções de texto não rotulados, captando não apenas a frequên-

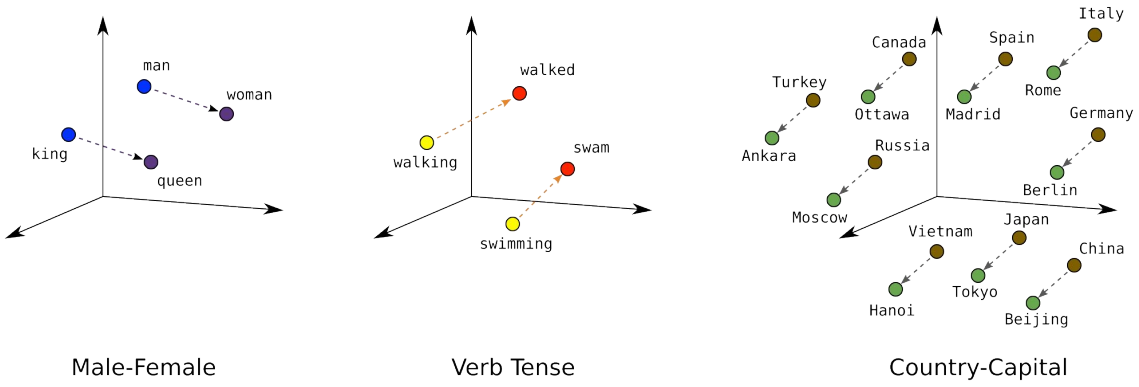


Figura 2.3: Exemplo gráfico do Word Embeddings [31].

cia das palavras, mas também as relações contextuais entre elas. Outra abordagem influente nesse contexto, desenvolvida em 2014 por Pennington é o Global Vectors for Word Representation (GloVe) [63], que combina a contagem global de co-ocorrência de palavras em um corpus com uma técnica de fatoração de matriz para produzir representações vetoriais que capturam tanto a semântica quanto a sintaxe das palavras.

Entretanto, mesmo com esses modelos, ainda persistiam desafios em compreender a estrutura e o contexto mais amplo dos textos. Foi então que, em 2017, surgiu a arquitetura *transformer* [76], que gerou uma mudança significativa no mundo de PLN, sendo utilizada como base para vários modelos famosos e avançados, como BERT [20] e GPT-3 [9]. Os *transformers* são uma arquitetura de redes neurais amplamente utilizada no processamento de linguagem natural (PLN). Eles introduzem um mecanismo de atenção que permite processar a entrada de texto simultaneamente, aprendendo as relações contextuais entre as palavras. Um *transformer* é composto por dois componentes principais: o codificador, que lê o texto de entrada e gera uma representação para cada palavra, e o decodificador, que gera o texto traduzido a partir dessa representação [76], como mostrado na Figura 2.4.

No codificador, descrito a direita na Figura 2.4, cada camada possui duas subcamadas: um mecanismo de atenção multi-head e uma rede feed-forward. A primeira subcamada permite que o codificador veja outras palavras enquanto processa uma palavra específica. A saída de cada subcamada é normalizada e são aplicadas conexões residuais. O decodificador, descrito a esquerda na Figura 2.4, tem uma subcamada adicional de atenção sobre a saída do codificador, permitindo que ele se concentre nas partes relevantes da entrada.

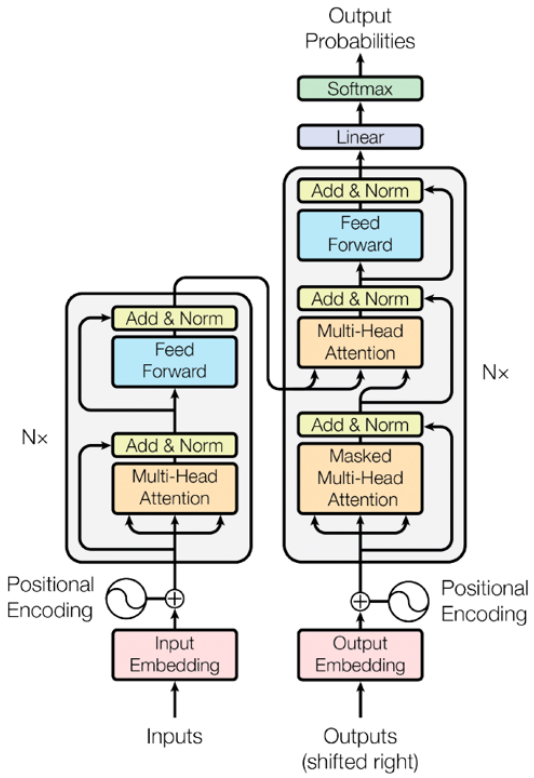


Figura 2.4: Saída da arquitetura transformers detalhada.

O processo de atenção é realizado com o cálculo de pontuações entre as palavras, onde cada palavra tem um vetor de chave, valor e consulta. A atenção é calculada através de um produto escalar entre os vetores de consulta e chave, determinando o quanto de foco cada palavra deve receber. A pontuação é normalizada com a função softmax para garantir que todas as pontuações sejam positivas e somem 1. O resultado é multiplicado pelos vetores de valor, gerando a saída ponderada, que é passada para a rede *feed-forward* [76].

Com o avanço dos modelos baseados na arquitetura de *Transformers*, foi conduzida uma pesquisa para identificar e aplicar técnicas eficazes de aprendizagem, culminando na criação do *Text to Text Transfer Transformers* (T5). Este modelo, treinado no *Colossal Clean Crawled Corpus* (C4)[2], obteve resultados de ponta em múltiplos *benchmarks* de Processamento de Linguagem Natural (PLN) [66].

Uma característica marcante do T5 é sua estrutura de *Text-to-Text*, possibilitando a formulação unificada de todas as tarefas de PLN, com entrada e saída sempre em formato textual, possibilitando a realização de diferentes tarefas com entradas e saídas textuais, como

[2] https://commoncrawl.org

mostrado na Figura 2.5 [32]. Além disso, o T5 pode ser ajustado finamente para tarefas específicas através do processo de ajuste fino, treinando o modelo em conjuntos de dados específicos da tarefa para otimizar seu desempenho .

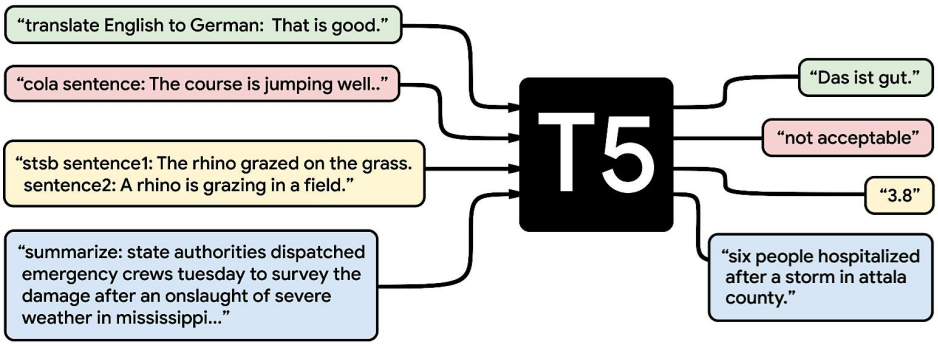


Figura 2.5: Exemplo gráfico do T5 [32].

Posteriormente, um estudo em 2021 [60] avaliou diferentes formas de representar os textos utilizando a arquitetura T5 e realizou um *benchmark* comparando-os com outras estratégias de representações textuais. A conclusão foi que uma dessas arquiteturas T5 exploradas no artigo, estabeleceu um novo estado da arte em similaridade textual semântica (STS).

Discutir sobre *word embeddings*, modelos pré-treinados, Word2Vec, GloVe, Transformers e T5 neste trabalho tem como propósito enfatizar como essas técnicas sofisticadas de representação textual possibilitam uma compreensão mais profunda da linguagem natural, sendo fundamentais para capturar nuances semânticas nos BRs.

2.7 Análise Estatística: Testes de Hipótese e Comparação de Variáveis

A aplicação de testes de hipótese permite verificar a existência de diferenças estatisticamente significativas entre duas ou mais variáveis analisadas. Este processo inicia-se com a formulação de hipóteses: a hipótese nula (H0), que geralmente afirma a ausência de efeito ou diferença, e a hipótese alternativa (H1), que sugere a presença de uma diferença significativa.

Para a seleção adequada do tipo de teste de hipótese a ser realizado, é fundamental verificar premissas básicas sobre a amostra, como a normalidade dos dados e a homogeneidade das variâncias. A normalidade dos dados postula que os dados de cada grupo devem seguir

uma distribuição normal, enquanto a homogeneidade das variâncias afirma que as variâncias entre os grupos devem ser equivalentes.

Após a verificação da normalidade e da homogeneidade dos dados, procede-se à escolha do teste de hipótese mais apropriado, considerando os resultados obtidos nestas análises preliminares. Nos casos em que se observa diferença significativa entre os dados, realizam-se testes *post-hoc* adicionais, com o objetivo de identificar precisamente onde essas diferenças se manifestam.

2.7.1 Verificação da Normalidade

A verificação da normalidade é um passo fundamental para o processo de decisão sobre o teste de hipótese mais adequado. Dentre as opções de verificação de normalidade, existe o teste de Kolmogorov-Smirnov (KS), que compara a distribuição dos dados com uma distribuição normal teórica, avaliando se há discrepâncias significativas [52].

O teste KS produz um valor p, que representa a probabilidade de observar os dados sob a premissa de que a hipótese nula seja verdadeira. A interpretação deste valor segue um critério estabelecido: se o valor p exceder o nível de significância adotado, não se rejeita a hipótese nula, indicando que os dados podem ser considerados normalmente distribuídos. Inversamente, um valor p inferior ao nível de significância adotado (comumente 0,05) sugere que os dados não aderem a uma distribuição normal. O teste de normalidade tem como objetivo dar suporte à escolha da melhor estratégia de teste de hipótese a ser utilizada [52].

2.7.2 Verificação de Homogeniedade

O teste de homogeneidade é um procedimento estatístico utilizado para avaliar se diferentes grupos ou amostras de uma população possuem a mesma variância. Um exemplo desse teste é o teste de Levene, desenvolvido por Howard Levene em 1960 [48], este teste é robusto contra desvios da normalidade, o que o torna particularmente útil em situações onde a distribuição dos dados não é perfeitamente normal.

O teste de Levene avalia a homogeneidade das variâncias entre grupos, calculando as diferenças entre cada valor e a média do seu grupo, e então analisando estatisticamente essas diferenças. O objetivo é verificar se essas diferenças são semelhantes entre os grupos. Se

o teste resultar em um *p-value* menor que 0,05 (nível de significância comumente usado), isso indica que as variâncias provavelmente não são iguais entre os grupos [29]. O teste de homogeniedade ajuda a decidir se podem prosseguir com análises que assumem variâncias iguais entre grupos, como a ANOVA, ou se precisam considerar métodos alternativos [70].

2.7.3 Testes de Hipótese

Após a verificação da normalidade e da homogeneidade das variâncias, podemos escolher o teste adequado para a análise. Se os dados forem normais e as variâncias homogêneas, uma das opções é utilizar o ANOVA, que compara as médias entre os grupos [26]. Esse teste é útil para identificar se pelo menos um grupo difere significativamente dos outros, baseando-se na razão entre a variabilidade entre os grupos e a variabilidade dentro deles [37].

Caso as variâncias sejam heterogêneas e os dados normais, uma das alternativas é utilizar o ANOVA de Welch [77]. Essa versão ajusta os pesos dos grupos de acordo com suas variâncias, permitindo uma análise precisa quando os pressupostos da ANOVA tradicional não são atendidos [19]. Já nos casos em que os dados não seguem uma distribuição normal, adotamos o teste de Kruskal-Wallis [44], um teste não paramétrico que compara as medianas entre grupos. Esse teste é especialmente útil quando as suposições de normalidade e homogeneidade das variâncias não podem ser satisfeitas. O teste de Kruskal-Wallis avalia se há diferenças significativas nas medianas nos grupos, oferecendo uma alternativa válida para situações onde os diferentes tipos de ANOVA não são apropriados.

2.7.4 Testes Post-Hoc

Quando um teste de hipótese indica a existência de diferenças significativas entre grupos, torna-se necessário identificar precisamente quais grupos apresentam essas diferenças [38]. Para este fim, empregam-se testes *post-hoc*, que permitem comparações múltiplas entre pares de grupos [36].

Entre os métodos mais amplamente utilizados, destacam-se o teste de Tukey [75], adequado para comparações após uma ANOVA com variâncias homogêneas, o teste de Games-Howell [27], recomendado quando as variâncias são heterogêneas, e o teste de Dunn [22], aplicável após o teste não paramétrico de Kruskal-Wallis. A escolha do teste *post-hoc* apro-

priado depende das características dos dados e do testes realizados inicialmente, visando maximizar a precisão das conclusões [18].

Capítulo 3

Metodologia

Este capítulo descreve a metodologia deste trabalho, cujo objetivo é realizar um estudo exploratório para identificar a melhor forma de definir automaticamente a similaridade entre BRs utilizando diferentes abordagens em modelos de *machine learning*. Nosso foco é investigar como técnicas contemporâneas podem aprimorar essa tarefa, indo além de métodos previamente explorados na literatura.

Após estudos iniciais e uma imersão no tema, elaboramos as questões de pesquisa a serem respondidas por meio de um processo metodológico dividido em cinco grandes etapas: 3.2 recuperação dos dados, 3.3 pré-processamento e vetorização, 3.4 concepção dos arcos de similaridade, 3.5 normalização das *features* e 3.6 treinamento e avaliação dos modelos, conforme ilustrado na Figura 3.1.

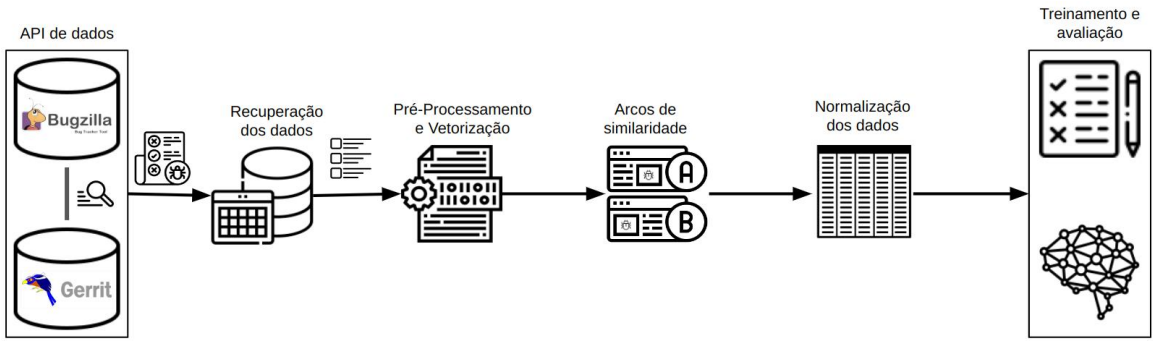


Figura 3.1: *Pipeline* experimental.

Na primeira e na segunda etapa, coletamos e pré-processamos os dados através da *tokenização*, remoção de *stopwords* e lematização, detalhando como foi realizada a vetorização

do texto com o T5 e o TF-IDF. Em seguida, criamos e normalizamos estruturas que representam as relações entre os BRs (arcos). Posteriormente, realizamos a divisão dos dados, o treinamento da rede neural (RN), a permutação das configurações encontradas e a escolha dos melhores modelos com base na avaliação. Por fim, analisamos o impacto das *features* no processo de aprendizagem da RN.

Para promover a transparência e reprodutibilidade da pesquisa, disponibilizamos publicamente todos os códigos utilizados[1] e os modelos treinados[2]. O restante deste capítulo descreve detalhadamente as questões de pesquisa e cada etapa mencionada anteriormente.

3.1 Questões de pesquisa

Para avaliar uma variedade de modelos, utilizamos diversas técnicas e comparamos a eficácia de cada um na detecção de BRs similares, por meio de uma análise minuciosa do impacto das features em cada modelo. Testamos o TF-IDF isolado, o T5 isolado, e uma abordagem híbrida combinando T5 e TF-IDF. Dessa forma, surgiram as seguintes questões de pesquisa:

QP1. Qual o desempenho comparativo dos modelos TF-IDF, T5 isolado e T5 + TF-IDF na identificação de BRs similares? O objetivo é avaliar e comparar a eficácia de três abordagens distintas: o TF-IDF como baseline, o T5 isolado representando uma técnica avançada de processamento de linguagem natural, e a abordagem híbrida T5 + TF-IDF. Para responder esta questão, realizaremos um experimento comparando as três abordagens. Através de várias métricas como acurácia, precisão, *recall*, F1, AUC ROC, AUC PR, *precision@n* e *recall@n* seremos capazes de determinar qual modelo tem melhor desempenho na identificação de bug reports similares. Esta análise nos permitirá entender as vantagens e desvantagens de cada abordagem, incluindo se as capacidades de compreensão contextual do T5 oferecem benefícios significativos sobre a abordagem estatística do TF-IDF, e se a combinação das técnicas na abordagem híbrida resulta em melhorias adicionais.

QP2. Como a escolha de hiperparâmetros impacta no desempenho dos modelos T5 e T5 + TF-IDF? Visamos realizar uma análise para entender como a alteração dos pa-

[1] https://drive.google.com/drive/folders/1aDTHFJlUvzTm5ZZI6LCfZcZ-2j-ULfxC

[2] https://drive.google.com/drive/folders/1U2iIjhRKCjS6I9m52X7y5keKAv4nJNz3

râmetros dos modelos impacta seu desempenho. Esta análise nos ajuda a compreender os pontos fortes e fracos dos modelos selecionados e fornece insights para trabalhos futuros. Além disso, permitirá entender se a otimização de hiperparâmetros afeta de maneira diferente o modelo T5 isolado e o modelo híbrido T5 + TF-IDF.

3.2 Etapa 1: Recuperação dos dados

Para o treinamento de um modelo de *machine learning*, é fundamental disponibilizar dados pertinentes que permitam aprender a executar uma tarefa específica. Estes dados, frequentemente denominados como *features* ou características, são representações dos dados de entrada contendo informações cruciais e úteis para resolver o problema. Na pesquisa em questão, foi crucial identificar e coletar *features* que sustentassem o desenvolvimento do modelo para abordar o desafio da similaridade entre BRs. Durante o processo de seleção dessas *features*, priorizamos informações que incluíssem descrições textuais, detalhes sobre os responsáveis pelo bug e a localização do bug no código.

A inclusão de descrições textuais foi fundamental para explorar a similaridade entre os textos e, por consequência, entre os BRs. A identificação dos colaboradores envolvidos em cada BR permitiria inferir a semelhança de quem trabalhou naqueles BRs. Da mesma forma, o ambiente em que o bug aconteceu poderia sugerir uma relação de similaridade entre eles. Portanto, as características selecionadas foram:

1. ***summary***: o título ou descrição curta de um BR.
2. ***description***: descrição detalhada do BR, geralmente inclui uma explicação clara e objetiva do bug, os passos necessários para reproduzir o comportamento observado, a descrição do comportamento esperado em contraste com o comportamento real, e quaisquer informações adicionais relevantes.
3. ***component***: componente no qual o BR foi encontrado.
4. ***version***: versão na qual o BR foi encontrado.
5. **asssigned_to**: para quem a resolução do bug foi atribuída.
6. ***creator***: quem criou o BR.

7. ***qa_contact***: o contato para o QA responsável.
8. ***platform***: sistema operacional ou plataforma de hardware na qual o BR foi encontrado.
9. ***severity***: a gravidade do BR.
10. **priority**: a prioridade para a resolução do BR.
11. ***files_changed***: lista de todos os arquivos alterados durante o processo de bug fixing.

Para coletar as *features*, foram utilizadas as plataformas do Bugzilla e do Gerrit. O Bugzilla é um sistema de código aberto para submissão, rastreamento e gerenciamento de bugs em projetos de software. Já o Gerrit, trata-se de uma plataforma de revisão de código que facilita a comunicação entre desenvolvedores, permitindo revisões detalhadas e interativas das alterações propostas.

Utilizamos o Bugzilla para obter informações que descrevem o bug através do bug report e o Gerrit para mapear as informações do bug report com a alteração dos arquivos no processo de bug fixing. Além disso, destacam-se entre os motivos da utilização da plataforma Bugzilla: a variedade de empresas que as utilizam, a quantidade de informações que possuem e serem plataformas *open source*.

Além de ser possível obter informações descritivas presentes no bug report do Bugzilla, como *summary*, *description*, *component* etc, existe um campo chamado "see_also". Nesse campo, podemos encontrar *links* para: outros BRs relacionados, *patches*, revisões de código, discussões e comentários e documentação relacionada. Dentre os BRs recuperados, alguns podem conter *links* que associam as revisões através do Gerrit, o qual será utilizado para obter as informações sobre os arquivos alterados.

Na prática, para recuperar a informação foi criado um *script* Python que conecta-se ao Bugzilla[3] por meio da biblioteca Bugzilla e busca os IDs dos relatórios de bugs para um intervalo de anos especificado, fazendo solicitações HTTP para a API do Bugzilla e extraindo os IDs dos relatórios de bugs da resposta HTML usando BeautifulSoup[4].

Para cada ID de BR recuperado, o *script* busca os dados detalhados do relatório de bug correspondente no Bugzilla. Caso algo inesperado ocorra, é lançada uma exceção para indicar problemas como: problemas de conexão ou bugs inexistentes. Por meio dos IDs do

[3] `https://pypi.org/project/bugzilla/`
[4] `https://beautiful-soup-4.readthedocs.io`

Gerrit obtidos do campo "see_also"dos BRs, o *script* obtém, através de solicitações *HTTP* as alterações de arquivos durante o processo de bug *fixing*.

Para melhorar a eficiência, o *script* aproveita o processamento concorrente utilizando *ProcessPoolExecutor*[5], distribuindo a recuperação e o processamento de BRs em vários processos para uma execução mais rápida. Esse processamento foi realizado em um conjunto de 13 máquinas com diferentes configurações: onze delas com *13th Gen Intel(R) Core(TM) i7-1355U* e 16 GB de RAM; outra com *Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz*, 48 GB de RAM e uma *GPU RX5500XT*; e outra com *11th Gen Intel(R) Core(TM) i7-1165G7* e 32 GB de RAM. Os dados de BRs processados, juntamente com as alterações do Gerrit associadas, são então inseridos em um banco de dados para posterior análise.

Foram recuperados 10.265 BRs públicos de 80 produtos do Bugzilla do Eclipse que estão marcados como "FIXED"e "CLOSED", juntamente com informações do Gerrit. Esses estados foram escolhidos porque indicam BRs cuja solução foi implementada e verificada, garantindo que os dados coletados representem casos concluídos e validados no ciclo de desenvolvimento. Essa decisão minimiza ruídos associados a BRs ainda em aberto ou em discussão, permitindo uma análise mais consistente e confiável dos padrões de resolução de problemas.

A coleta abrange desde o BR mais antigo que atende a esses critérios até a data de recuperação dos dados (11 de novembro de 2023, às 05:36 (GMT-3)). O primeiro BR recuperado foi criado em 11 de outubro de 2001 às 01:34:46 UTC, com sua última alteração registrada em 5 de fevereiro de 2015 às 16:48:37 UTC, demonstrando a longevidade e a manutenção contínua dos registros no sistema. Esses dados permitem investigar a evolução dos processos de desenvolvimento ao longo de mais de uma década.

É importante notar que a integração do Gerrit com o Bugzilla do Eclipse ocorreu posteriormente à criação dos primeiros BRs. Consequentemente, os BRs mais antigos podem não conter todas as informações relacionadas ao Gerrit que estão presentes nos BRs mais recentes. Esta particularidade na evolução do sistema deve ser considerada na análise dos dados, especialmente ao examinar tendências temporais ou comparar BRs de diferentes períodos.

Como detalhado na seção 2.1, os BRs selecionados para este estudo atendem simultaneamente aos critérios de *status* "CLOSED"e *resolution* "FIXED", representando etapas com-

[5] `https://docs.python.org/3/library/concurrent.futures.html`

plementares no ciclo de vida de um *bug*. Um BR com *resolution* "FIXED"indica que a correção foi implementada e integrada ao código-fonte, enquanto o *status* "CLOSED"confirma que essa solução foi validada e o problema foi efetivamente resolvido.

A aplicação conjunta desses dois filtros assegura que analisamos apenas BRs cujo processo de correção foi concluído de forma ideal, eliminando casos pendentes ou inconclusos. Essa abordagem é essencial para garantir a estabilidade da variável "*files_changed*", já que BRs que atendem a esses critérios não sofrerão alterações posteriores. Assim, é possível realizar uma análise confiável e consistente dos padrões de resolução e impacto das mudanças de código.

3.3 Etapa 2: Pré-processamento e vetorização

A vetorização textual é uma peça fundamental na análise de dados textuais para o aprendizado de máquina, permitindo que palavras, sentenças e documentos sejam transformados em representações numéricas, facilitando assim o processamento por algoritmos. Além disso, técnicas como TF-IDF requerem pré-processamento, incluindo tokenização e remoção de *stopwords*, para preparar adequadamente os dados textuais para análise.

Durante essa etapa, o texto é pré-processado com técnicas como tokenização, remoção de *stopwords* e lematização, e em seguida, é vetorizado usando estratégias como T5 e TF-IDF. Com esses vetores disponíveis, é possível treinar modelos usando as abordagens TF-IDF e T5 separadamente ou juntamente (abordagem híbrida), permitindo responder a questão de pesquisa 1 (**QP1**).

3.3.1 Pré-processamento

Antes de realizar estratégias de vetorização mais antigas, como TF-IDF, utilizamos diferentes técnicas de pré-processamento, como: a tokenização, a remoção de *stopwords* e a lematização. A tokenização visa fragmentar o texto em sequências de caracteres denominadas tokens, podendo representar fonemas, sílabas, letras, entre outros.

A remoção de *stopwords* tem como intuito eliminar palavras com pouca relevância contextual e que são muito frequentes em todas as frases, como "the", "and", "that", entre outras. Já a lematização é um processo que reduz palavras flexionadas para sua forma base, por

exemplo: "running", "ran", e "runs"podem ser todas reduzidas ao lema "run".

3.3.2 TF-IDF

Como descrito anteriormente na seção 2.5.2, o TF-IDF é uma técnica de PLN que avalia a importância das palavras em um documento. Ele combina a frequência da palavra no documento (TF) com sua raridade em todos os documentos (IDF) para destacar a relevância das palavras. No entanto, assim como o BoW, o TF-IDF não considera o sentido das palavras e pode gerar vetores grandes em grandes vocabulários.

Para realizar a vetorização TF-IDF, foi desenvolvido um código que, primeiramente, realiza o pré-processamento nas *features* textuais removendo caracteres especiais por meio de expressões regulares e faz tokenização, remoção de *stopwords* e lematização utilizando a biblioteca NLTK[6]. Posteriormente, as *features* textuais *summary* e *description* são vetorizadas com TF-IDF através da biblioteca scikit-learn[7] e vinculadas a um ID de BR armazenado no formato pickle.

3.3.3 T5

Por ser capaz de lidar com texto bruto, devido à sua natureza *text-to-text*, a abordagem T5 não necessita de um pré-processamento extenso, simplificando o fluxo de vetorização para esta abordagem. Para realizar a vetorização T5, utilizando ST5 [60] foi desenvolvido um *script* que faz uso do modelo "sentence-transformers/sentence-t5-xxl"[8] disponível na biblioteca *sentence_transformers*[9]. Essa biblioteca é especialmente designada para a vetorização de texto, permitindo a conversão dos BRs em representações vetoriais.

Nesse *script*, cada BR é processado individualmente, e tanto o *summary* quanto a *description* são transformados em vetores utilizando o modelo T5. Esses dois campos foram selecionados para a vetorização por serem os mais representativos em termos de conteúdo textual descritivo, enquanto as outras variáveis são categóricas. O *summary* oferece uma visão condensada e objetiva do problema reportado, enquanto a *description* detalha o contexto,

[6] https://www.nltk.org/
[7] https://scikit-learn.org/
[8] https://huggingface.co/sentence-transformers/sentence-t5-xxl
[9] https://sbert.net/

as condições e os passos necessários para reproduzir o problema, fornecendo informações mais completas.

Ao transformar esses campos em vetores, busca-se capturar tanto o contexto geral quanto os detalhes específicos de cada BR, facilitando as análises de similaridade semântica entre os itens. Os vetores gerados são associados ao ID único de cada BR, organizados em um dicionário e armazenados em um arquivo no formato *pickle* para uso posterior.

3.3.4 Abordagem Híbrida (T5+TF-IDF)

A abordagem híbrida combina as vantagens das técnicas T5 e TF-IDF, visando capturar tanto a semântica profunda quanto as características específicas do domínio presentes nos BRs. Na prática, os vetores T5 e TF-IDF são gerados separadamente para cada BR, conforme descrito nas subseções anteriores 3.3.2 3.3.3. Em seguida, os vetores são utilizados para formar uma representação híbrida mais rica.

3.4 Etapa 3: Definição de ground truth

Na etapa de definição do *ground truth*, são criados os arcos de similaridade que têm como objetivo definir as relações e quantificar o grau de similaridade entre os BRs previamente recuperados e processados. Nesta fase, são construídas estruturas denominadas "arcos", que incorporam duas categorias principais de informações. A primeira é a distância do cosseno [51] entre os vetores gerados a partir das *features* textuais, proporcionando uma medida de similaridade semântica entre os BRs. A segunda é o grau de igualdade entre as demais *features* não textuais selecionadas na Etapa 1, avaliando a semelhança em aspectos estruturais e metadados dos BRs.

Para estabelecer o *ground truth* de similaridade entre BRs, definimos como similares os BRs que possuem pelo menos 50% dos arquivos alterados em comum. Essa métrica foi adotada porque, segundo Yang et al. [78], quando um desenvolvedor resolve bugs semelhantes, ele tende a resolver os problemas de forma mais eficiente, concentrando-se em um subconjunto reduzido de linhas de código. Essa definição serve como referência para validar modelos e métodos que buscam identificar similaridade entre BRs, garantindo que o critério adotado reflete casos práticos e relevantes no contexto do desenvolvimento de software.

Além das características mencionadas anteriormente, também foi calculada a porcentagem média de arquivos alterados em comum, representada pela Fórmula 3.1. Por exemplo, se o BRx alterar os arquivos 1 e 3, e o BRy alterar os arquivos 1, 2, 3, 4 e 5, o BRx terá 100% dos arquivos alterados em comum com o BRy, enquanto o BRy terá apenas 40%. A média desses valores resulta em 70%.

$$similaridade = \frac{\frac{ArquivosAlterados(BRx)}{ArquivosAlterados(BRy)} + \frac{ArquivosAlterados(BRy)}{ArquivosAlterados(BRx)}}{2} \tag{3.1}$$

Nesse contexto, utilizaremos os dados do Gerrit para calcular o grau de similaridade através dos arquivos alterados de dois bugs após o processo de bug fixing. Como estamos desconsiderando a direcionalidade dos arcos, iremos calcular a similaridade entre dois bug reports aos quais denominaremos X e Y.

Visando não prejudicar a aprendizagem do modelo, utilizamos duas estratégias para ajustar os dados. As estratégias baseiam-se em desconsiderar comparações de um BR com ele mesmo e comparações iguais com direções diferentes, considerando o princípio da simetria similaridade(XY) = similaridade(YX), que pode ser observado na tabela 3.1.

Tabela 3.1: Tabela de arcos desconsiderados marcados com X.

╲	BR 1	BR 2	BR 3
BR 1	**X**		
BR 2	**X**	**X**	
BR 3	**X**	**X**	**X**

Considerando o número total 10.265 de BRs obtidos como explicado em 3.3.1 e 3.3.2 e a combinação em pares considerando a simetria, geramos uma combinação total de 52.679.980 arcos como pode ser verificado calculando pela equação 3.2:

$$C(Nb, t) = \frac{Nb!}{(Nb - t)! \cdot t!} = \frac{10.265!}{(10.265 - 2)! \cdot 2!} = 52.679.980 \tag{3.2}$$

Considerando que:

- **Nb**: Número de bug reports (10.265).
- **t**: O tamanho da combinação (Nesse caso, dois por vez).

Para a concepção das estruturas de relação chamadas de arcos, foi desenvolvido um código responsável por calcular a similaridade entre BRs com uma abordagem paralela para lidar eficientemente com grandes volumes de dados. No *script* há uma iteração sobre os BRs, calculando as métricas de similaridade, incluindo similaridade entre os vetores T5 e TF-IDF.

Para lidar com grandes volumes de dados de maneira eficiente, o processamento dos dados é dividido em *batches*, culminando no armazenamento em uma instância do MongoDB.

Dos $52.679.980$ arcos gerados, apenas 36.676 foram identificados como similares, representando uma proporção extremamente desequilibrada de apenas $0,07\%$. Desses $52.679.980$ arcos totais, $5.482.354$ pertencem a arcos que tratam de BRs de um mesmo produto, dos quais 35.823 são identificados como similares, representando uma proporção de $0,65\%$. Considerando que a maioria dos arcos similares pertence a projetos iguais, evidenciou-se a necessidade de realizar uma análise concentrada exclusivamente nesses casos.

3.5 Etapa 4: Normalização dos dados

A etapa de normalização dos dados tem como objetivo padronizar as *features*, eliminando discrepâncias nas escalas e unidades de medida. Este processo visa evitar que o modelo seja sensível às diferenças de escala entre as *features*, o que poderia comprometer a acurácia do treinamento. Para alcançar esse objetivo, aplicou-se a técnica de normalização *min-max*[10]. Esta abordagem garante que todos os dados de entrada da Rede Neural (RN) sejam ajustados para um intervalo uniforme entre 0 e 1.

No *script* desenvolvido para normalizar os dados, são buscados os valores mínimos e máximos de cada feature presente no banco de dados e cada valor é normalizado através da Equação 3.3 e enviado para uma instância do MongoDB.

$$X_{normalizado} = \frac{X - min}{max - min} \tag{3.3}$$

[10] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

3.6 Etapa 5: Treinamento e avaliação dos modelos

A etapa de treinamento e avaliação dos modelos é fundamental para otimizar o desempenho preditivo dos algoritmos investigados. Esta etapa envolve três fases principais: treinamento, avaliação e ajuste fino. Inicialmente, múltiplos modelos são treinados para aprender padrões nos dados normalizados. Em seguida, realiza-se uma avaliação usando métricas de desempenho, permitindo uma comparação entre os modelos. Por fim, executa-se o *fine-tuning* dos hiperparâmetros dos modelos mais promissores.

Para fins didáticos, essa etapa foi subdividida em quatro sub-etapas 3.6.1 divisão dos dados, 3.6.2 métricas de avaliação, 3.6.3 treinamento e escolha de configurações e 3.6.4 definição da arquitetura de rede neural, com o objetivo de dar suporte para responder às questões de pesquisa **RQ1** e **RQ2** descritas na seção 3.1.

3.6.1 Divisão dos dados

Inicialmente, replicamos o *dataset* de arcos em duas instâncias distintas para permitir uma análise separada em cada uma. Em uma delas, filtramos os arquivos que estão presentes apenas no mesmo produto, enquanto na outra não aplicamos nenhum filtro. Em seguida, com o objetivo de ajustar os dados para o treinamento, dividimos de forma estratificada em conjuntos distintos. O conjunto de treinamento, representando 60% dos dados, é utilizado no processo de treinamento, permitindo que ele aprenda padrões nos dados.

O conjunto de validação, que corresponde a 20% dos dados, é utilizado para avaliar o desempenho do modelo durante o treinamento. Essa avaliação contínua permite detectar possíveis problemas de *overfitting*, garantindo que a rede neural aprenda de forma eficaz. Quando detectado um possível sobreajuste nos dados, o método de *early stopping* [4] é aplicado, não permitindo que haja mais uma época de treinamento.

Por fim, o conjunto de testes, também com 20% dos dados, serve para ter uma avaliação final sobre a capacidade de generalização do modelo com dados não vistos anteriormente após o fim de todas as épocas ou do *early stopping*.

Como descrito na seção 3.4, mais de 99% dos dados são compostos por arcos não similares, foram aplicadas técnicas de balanceamento com o objetivo de melhorar o treinamento do modelo. A técnica escolhida para a nossa abordagem, considerando a grande quantidade de

dados que temos ao dispor, foi o *undersampling* [25]. Essa técnica consiste em reduzir o número de exemplos da classe majoritária (não similares) para igualar ao número de exemplos da classe minoritária (similares). Adicionalmente, testamos duas proporções de distribuição entre BRs não similares e similares: uma razão de 2:1 e outra mais desbalanceada de 9:1.

O código desenvolvido para dividir os dados recupera as informações previamente armazenadas no MongoDB e, em seguida, realiza a separação dos dados em treino, validação e teste de forma estratificada. Além disso, cada uma dessas três partes dos dados é subdividida em *batches* de tamanho 2048, também de forma estratificada.

Após salvar o *dataset* completo em disco, o *script* faz *undersampling* gerando três conjuntos de dados de treino que variam o número de não similares em relação aos similares: um nove vezes maior, outro duas vezes maior e outro igual ao número de similares. Por fim, todas as formas de balanceamento de treinamento e os dados de validação e teste são salvos em memória de leitura e escrita do computador. Sendo todo esse processo realizado com multiprocessamento e para as duas instâncias: com o filtro do mesmo produto e sem filtro.

3.6.2 Métricas de avaliação

Após o processo de treinamento de um modelo, a avaliação de um modelo de aprendizado de máquina é essencial para medir seu desempenho e identificar possíveis problemas como *overfitting* e *underfitting*. Para avaliar os modelos treinados, utilizamos métricas de suporte à decisão para identificar quanto o modelo realmente é capaz de classificar os arcos corretamente, e métricas de avaliação focadas em ranqueamento, tendo como objetivo observar se os modelos identificados como mais similares realmente são relevantes.

Para testar o desempenho de um modelo de classificação, é comumente utilizada a matriz de confusão. Esta matriz é uma tabela que compara as previsões do modelo com os rótulos de classe reais, por meio de quatro elementos, conforme descrito na Figura 3.2 [8].

Considere as variáveis na tabela como:

- **TP**: Verdadeiros Positivos (do inglês: *True Positive*)
  - São valores classificados como classe positiva corretamente
- **FP**: Falsos Positivos (do inglês: *False Positive*)

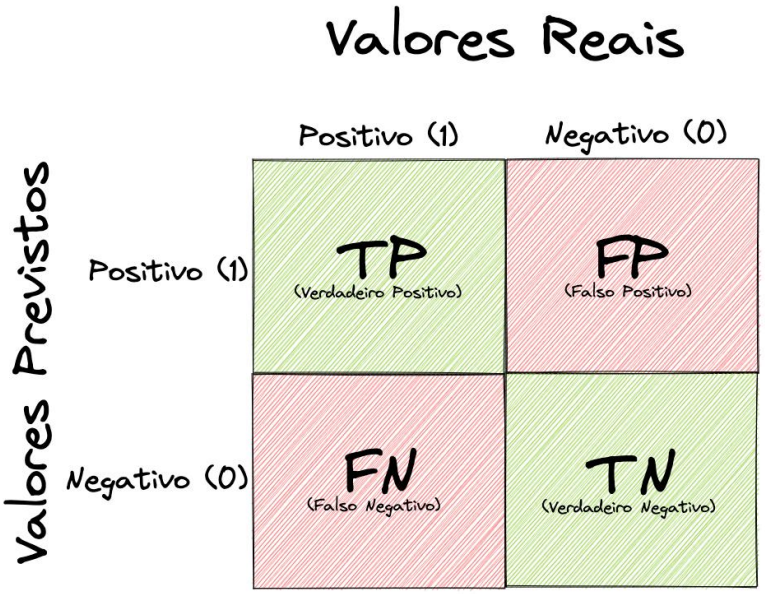


Figura 3.2: Matriz de confusão [8].

  - São valores classificados erroneamente como classe positiva

- **TN**: Verdadeiros Negativos (do inglês: *True Negative*)

  - São valores classificados corretamente como classe negativa

- **FN**: Falsos Negativos (do inglês: *False Negative*)

  - São valores classificados erroneamente como classe negativa

A partir das variáveis expostas na Figura 3.2 também podemos avaliar o modelo através de métricas de suporte à decisão. Essas métricas têm como objetivo medir o quão bem um modelo ajuda os usuários a tomar boas decisões. A acurácia indica como o modelo se comporta de forma geral, retornando quantas instâncias foram classificadas corretamente 3.4.

$$\text{Acurácia} = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.4}$$

A precisão calcula a proporção de verdadeiros positivos classificados corretamente pelo modelo 3.5.

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{3.5}$$

A revocação ou sensibilidade ou true positive rate (TPR) calcula a proporção de verdadeiros positivos em relação a todos os reais positivos 3.6.

$$\text{Revocação} = \frac{TP}{TP + FN} \tag{3.6}$$

O *False Positive Rate* (FPR) é o complemento da revocação, ou seja, a proporção de falsos positivos com relação a todos os positivos 3.7.

$$\text{FPR} = \frac{FP}{FP + TN} \tag{3.7}$$

O *True Negative Rate* (TNR) ou especificidade é a proporção de verdadeiros negativos ao total de instâncias negativas 3.8.

$$\text{TNR} = \frac{TN}{TN + FP} \tag{3.8}$$

O *False Negative Rate* (FNR) é o complemento da especificidade, representando a proporção de falsos negativos em relação a todos os casos negativos, conforme descrito na equação 3.9.

$$\text{FNR} = \frac{FN}{FN + TP} \tag{3.9}$$

O *F1-Score* busca calcular uma média harmônica ponderada entre as métricas de precisão e revocação 3.10.

$$\text{F1} = 2 * \frac{\text{Precisão} * \text{Revocação}}{\text{Precisão} + \text{Revocação}} \tag{3.10}$$

De acordo com as equações, é possível que um modelo apresente bons valores de precisão e revocação sem, no entanto, ser um modelo eficaz. Isso ocorre porque, mesmo que o modelo tenha muitos falsos negativos, a precisão pode ser alta, e, mesmo com muitos falsos positivos, a revocação pode ser boa. Diante disso, concluímos que o *F1-Score* oferece uma avaliação mais fiel dos modelos, pois considera ambas as métricas em conjunto.

Apesar do *F1-Score* ser métrica para a avaliação de um modelo de forma geral, essa métrica é bastante sensível ao desequilíbrio de classe em cenário em que as classes estão desproporcionalmente distribuídas, como é o caso do problema de similaridade que estamos

abordando. Para reavaliar o modelo com uma métrica específica, utilizamos os cálculos de Áreas Abaixo da Curva (AUC, do inglês *Area Under the Curve*).

A curva ROC (*Receiver Operating Characteristic*) é uma ferramenta comumente usada para avaliar a performance de modelos de classificação binária em diferentes limiares de decisão. Essa curva é gerada através do TPR e da FPR, nos eixos Y e X respectivamente. Cada ponto na curva ROC representa um limiar de classificação diferente. A ROC AUC é uma medida da probabilidade de que o modelo classifica corretamente uma instância positiva aleatória mais alta do que uma instância negativa aleatória. Quanto maior o tamanho da área abaixo da curva, melhor o modelo é, considerando que 0.5 é um modelo aleatório. Mais ludicamente explicado na Figura 3.3 [79].

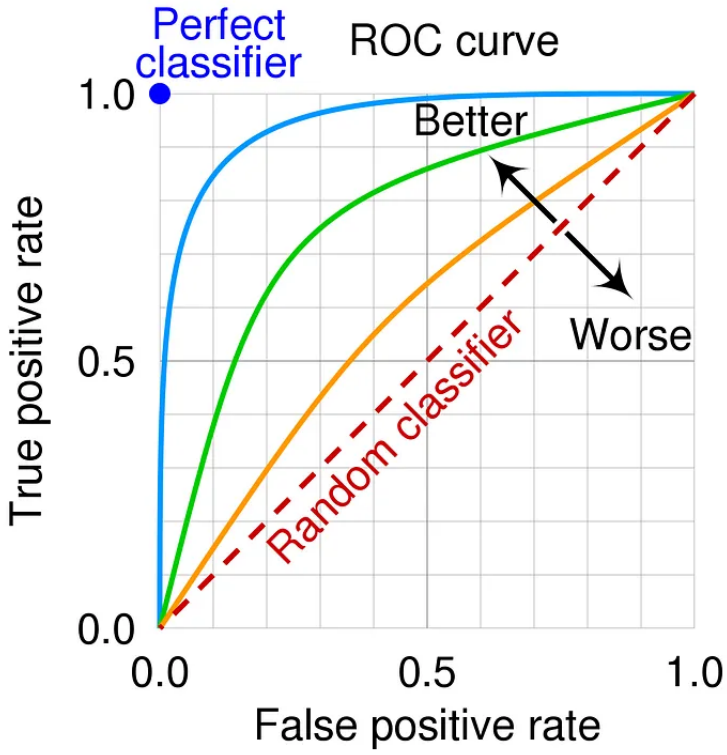


Figura 3.3: Curva ROC [79].

Apesar do ROC AUC ser uma boa métrica para classificação binária em cenários desbalanceados, quando essa distorção de distribuição entre as classes é muito acentuada há um risco dessa métrica gerar resultados muito otimistas. Para esses cenários de desbalanceamento extremo, é sugerido o uso da métrica AUC PR (Precision-*Recall*) [45].

Diferente da AUC ROC, a AUC PR utiliza a relação entre precisão e o *recall* do modelo

para diferentes limiares de probabilidade. Quanto maior a área da AUC PR, melhor é o desempenho do modelo, como é exemplificado na Figura 3.4 [10].

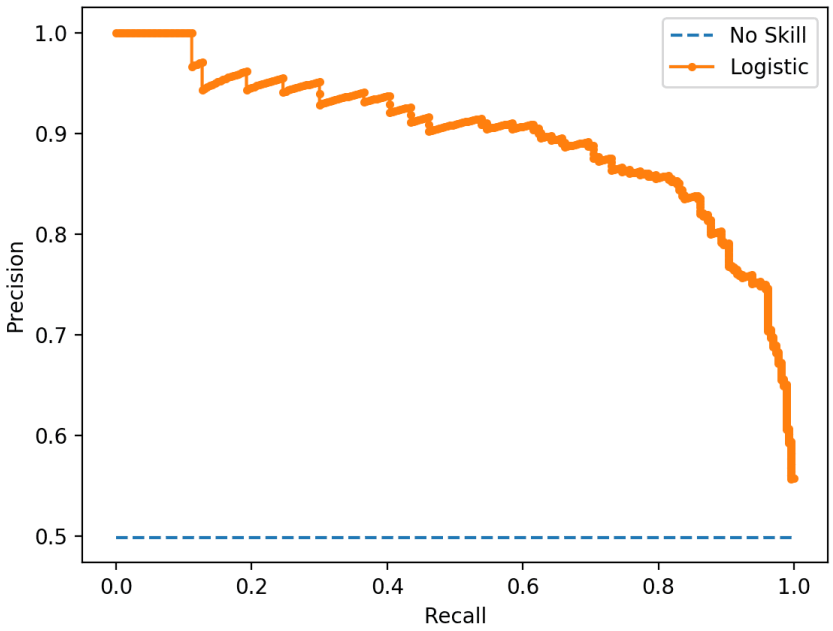


Figura 3.4: Curva PR [10].

Com o objetivo de ver se a RN também é capaz de recomendar além de classificar os BRs similares, utilizaremos a *Precision@N* e o *Recall@N*, que mede a precisão das principais recomendações feitas pelo sistema. Em resumo, *Precision@N* calcula a porcentagem de itens relevantes em relação a todos os top N recomendados pelo sistema [23]. Para definir também a capacidade da recomendação obter itens relevantes com relação ao total de todos os itens dispostos, é calculado o *Recall@N* [23].

Com o objetivo de garantir uma avaliação justa do modelo, o *Precision@N* e o *Recall@N* serão calculados apenas para requisições que contenham ao menos N itens relevantes. Ao final, para assegurar uma análise abrangente dos modelos propostos, utilizamos todas as métricas apresentadas, a fim de obter diferentes perspectivas sobre a capacidade do modelo em classificar e ranquear corretamente os BRs em diversos contextos e cenários.

3.6.3 Treinamento e Escolha de Configurações

Visando encontrar o melhor modelo, realizamos uma série de treinamentos em duas etapas distintas, seguidas de uma avaliação criteriosa para identificar o modelo mais eficaz. O processo foi estruturado da seguinte forma:

1. **Primeira etapa**: Treinamento e Seleção Inicial

    - **Como utilizar os vetores**: Comparamos o desempenho dos modelos utilizando duas abordagens:
        - Distância de cosseno previamente calculada entre os vetores
        - Vetor completo como feature
    - **Qual *dataset* utilizar**: Testamos quatro distribuições de balanceamento diferentes:
        - Dados completos desbalanceados
        - Dados completamente balanceados
        - Proporção de dois não similares para um similar
        - Proporção de nove não similares para um similar
    - **Qual estratégia de vetorização utilizar**: Avaliamos três abordagens:
        - TF-IDF
        - T5
        - Abordagem híbrida (TF-IDF com T5)
    - **Qual a melhor configuração de pré-processamento**: Analisamos três configurações, aplicáveis apenas para as abordagens TF-IDF e híbrida, conforme explicado na seção 4.3.3:
        - Apenas tokenização
        - Tokenização combinada com remoção de *stopwords*
        - Tokenização combinada com remoção de *stopwords* e lematização

      É importante notar que, como explicado na seção 4.3.3, a abordagem isolada do T5 não requer pré-processamento adicional.

Após esta etapa, avaliamos os modelos utilizando as métricas descritas na seção 3.6.2. Selecionamos os modelos com os melhores resultados para prosseguir para a segunda etapa.

2. **Segunda etapa**: *Fine Tuning* [11] [12]

   - **Definir o peso das classes**: Comparamos duas abordagens:
     - Pesos de acordo com a distribuição do dado de treino
     - Pesos balanceados
   - **Qual melhor otimizador**: Testamos três otimizadores comumente utilizados em problemas semelhantes:
     - SGD (Stochastic Gradient Descent)
     - RMSprop
     - Adam
   - **Qual a melhor camada de ativação final**: Avaliamos duas variantes da função Sigmoid:
     - Sigmoid Tradicional
     - Hard Sigmoid

     A escolha de funções sigmoid foi feita para utilizar a saída como heurística para a recomendação.
   - **Qual a melhor métrica de *loss***: Mantivemos a BinaryCrossentropy como *loss* em todas as circunstâncias devido à sua adequação perfeita para o problema de classificação binária.

As combinações exploradas nessas duas etapas resultaram em um total de 56 configurações diferentes na primeira etapa e 12 configurações na segunda etapa, considerando que a métrica de loss foi mantida constante. O cálculo das 56 configurações na primeira etapa pode ser detalhado da seguinte forma:

[11] https://www.tensorflow.org/tutorials/structured_data/imbalanced_data
[12] https://www.tensorflow.org/versions/r1.15/api_docs/python/tf/keras

- Para TF-IDF e abordagem híbrida: 2 (vetores) x 4(datasets) x 2 (TF-IDF e híbrida) x 3 (pré-processamento) = 48 configurações
- Para T5 isolado: 2 (vetores) x 4 (datasets) = 8 configurações
- Total: $48 + 8 = 56$ configurações

Na segunda etapa, temos 12 configurações (2 x 3 x 2 = 12). Após a primeira etapa, selecionamos as N melhores configurações com base nas métricas de avaliação. Essas configurações selecionadas foram então submetidas ao processo de *fine tuning* na segunda etapa, resultando em N x 12 modelos finais para avaliação. A avaliação final desses modelos nos permitiu identificar a combinação ótima de parâmetros e técnicas para o nosso problema específico de classificação e recomendação.

3.6.4 Definição da Arquitetura de Rede Neural

Para realizar o experimento, foram definidas diferentes arquiteturas de rede neural (RN), dependendo da forma de utilização e do tipo de vetores empregados. Quando são utilizadas distâncias de cosseno, a arquitetura da RN é descrita na Figura 3.5.

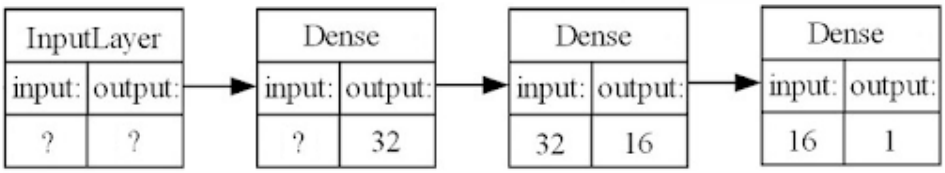


Figura 3.5: Arquitetura de distância de cosseno.

A *InputLayer* receberá uma quantidade variável de *features* (representada por "?" na Figura 3.5), dependendo da abordagem utilizada no treinamento. Na abordagem híbrida (TF-IDF + T5), são utilizadas 14 *features*, devido à combinação de dois vetores, uma vez que ambas as vetorização (TF-IDF e T5) são aplicadas em conjunto. Por outro lado, quando a abordagem é isolada (somente TF-IDF ou T5), são utilizadas 12 *features*. Em seguida, a arquitetura inclui três camadas *dense* com 32, 16 e 1 neurônios, cujo objetivo final é determinar a similaridade do arco.

Caso sejam utilizados os embeddings, o número de camadas que recebem *features* e o tamanho do *input* dessas camadas variam. Com uma abordagem isolada, há 5 camadas para

receber as *features*, sendo 4 apenas para os dois vetores de *summary* e depois vetores de *description*. Após esse processo, os dois vetores serão passados para uma camada *Concatenate* e depois por camadas 32, 16 e 1 neurônio(s), como mostrado na metade da Figura 3.6.

Para a abordagem híbrida, há 9 camadas para entrada das *features*, sendo 8 só para vetores, pois serão os quatro vetores TF-IDF e os quatro vetores T5. De forma análoga, cada par de vetor irá para uma camada Concatenate e depois para camadas de 32, 16 e 1 neurônio(s), como mostrado na Figura 3.6.

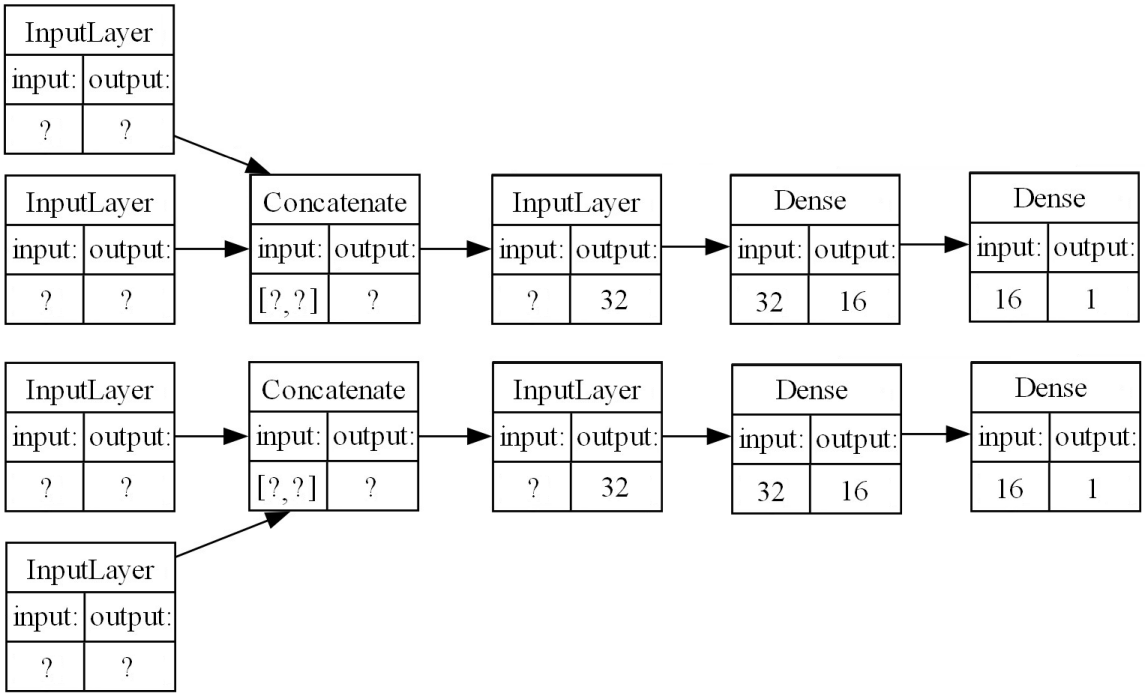


Figura 3.6: Arquitetura recebendo oito vetores.

Por fim, as saídas das camadas *Dense* são utilizadas para alimentar uma camada *Concatenate*, junto com as *features* restantes, representadas pela *InputLayer* de tamanho 10. Isso resulta em um output de tamanho 12 ou 14, dependendo da abordagem, que será utilizado para novas camadas com 32, 16 e 1 neurônios, como mostrado na Figura 3.7.

Para treinar todas as redes neurais (RN) mencionadas, foi desenvolvido um código que implementa uma classe configurável chamada "Similarity Model Trainer", permitindo ajustes em diversas opções durante o treinamento, além de suportar processamento paralelo. A biblioteca utilizada para o treinamento das RNs é o *TensorFlow*[13], responsável pela concep-

[13] https://www.tensorflow.org/versions/r1.15/api_docs/python/

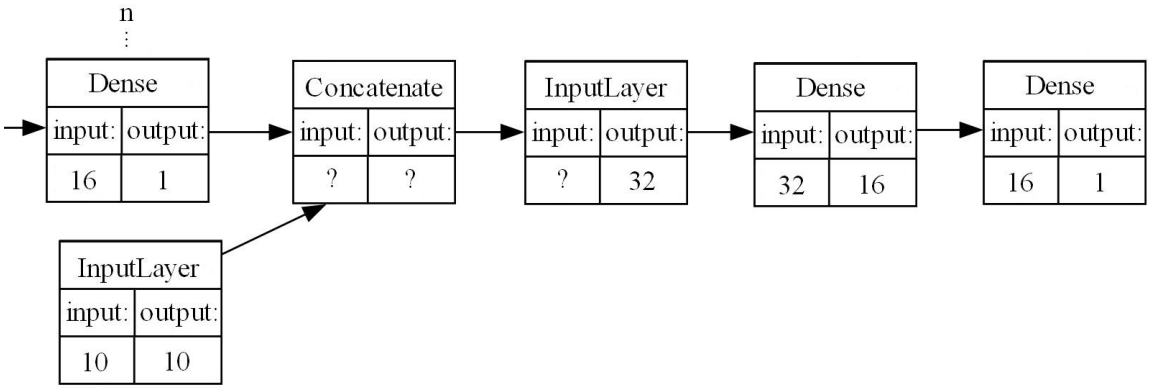


Figura 3.7: Final da arquitetura de vetores.

ção, treinamento e avaliação dos modelos. Dentro da classe *SimilarityModelTrainer*, são definidos os parâmetros de configuração, como a escolha de usar ou não vetores, o pré-processamento de texto a ser aplicado, o número de épocas de treinamento, entre outros.

3.7 Testes de Hipótese

Como descrito na seção 2.7, a análise estatística seguiu um fluxo sistemático para garantir a aplicação apropriada dos testes de hipótese. Esse processo consistiu em quatro etapas principais: o teste de normalidade, o teste de homogeneidade das variâncias, o teste de hipótese adequado e, quando necessário, o teste *post-hoc* correspondente.

Inicialmente, realizou-se o teste de normalidade para verificar se os dados de cada grupo seguiam uma distribuição normal. Para esta finalidade, foi empregado o teste de Kolmogorov-Smirnov, que é adequado para amostras de diversos tamanhos. A decisão subsequente baseou-se nos resultados deste teste. Se os dados apresentassem distribuição normal, prosseguia-se para o teste de homogeneidade das variâncias. Caso contrário, considerava-se a aplicação de testes não paramétricos.

Para os dados que demonstraram normalidade, o próximo passo foi verificar a homogeneidade das variâncias entre os grupos. Utilizou-se o teste de Levene para esta avaliação. A interpretação dos resultados deste teste guiou a seleção do teste de hipótese apropriado:

- Para dados normais com variâncias homogêneas: ANOVA tradicional
- Para dados normais com variâncias heterogêneas: ANOVA de Welch

- Para dados não normais: Teste de Kruskal-Wallis

Após a aplicação dos testes de hipótese, procedeu-se com testes *post-hoc* quando o resultado do teste principal indicou diferenças estatisticamente significativas entre os grupos ($p < 0,05$). Os testes *post-hoc* foram utilizados para identificar especificamente quais grupos diferiam entre si. A escolha do teste *post-hoc* depende do teste de hipótese utilizado:

- Após ANOVA com resultado significativo: Teste de Tukey
- Após ANOVA de Welch com resultado significativo: Teste de Games-Howell
- Após Kruskal-Wallis com resultado significativo: Teste de Dunn

Capítulo 4

Resultados

Este capítulo apresenta os resultados e análises derivados da identificação e recomendação de *Bug Reports* (BRs) similares usando diversas abordagens de treinamento de redes neurais. A primeira questão de pesquisa (QP1) procura determinar qual estratégia de vetorização é mais eficaz no treinamento de uma rede neural para identificar e recomendar BRs semelhantes. Avaliamos os modelos treinados usando várias métricas, incluindo acurácia, precisão, *recall*, F1, entre outras, para medir sua capacidade preditiva na determinação da similaridade entre BRs.

A segunda questão de pesquisa (QP2) visa realizar uma análise para entender como a alteração dos parâmetros do modelo afeta seu desempenho. Nessa análise, utilizamos os melhores modelos encontrados para realizar o processo de *fine tuning* e entender a melhor forma de montar essa rede, fornecendo *insights* para trabalhos futuros.

Conforme discutido nas seções 3.6.2 e 3.6.3, realizamos uma série de treinamentos com o objetivo de identificar o melhor modelo para identificação e recomendação de BRs similares, subdividindo-se em uma primeira etapa de treinamento e segunda etapa de ajuste fino. A primeira etapa envolve considerações sobre como empregar os vetores, qual método de balanceamento de dados adotar, qual abordagem de vetorização utilizar e qual configuração de pré-processamento seria mais eficaz.

Como mostrado na Figura 4.1, exploramos duas formas de utilizar os vetores, quatro técnicas de balanceamento de dados, três estratégias de vetorização e três configurações de pré-processamento, totalizando potencialmente uma permutação de 72 modelos distintos. No entanto, como explicado na seção 3.3.3, os modelos que dependem exclusivamente do

T5 não requerem pré-processamento adicional, resultando na criação de 56 modelos.

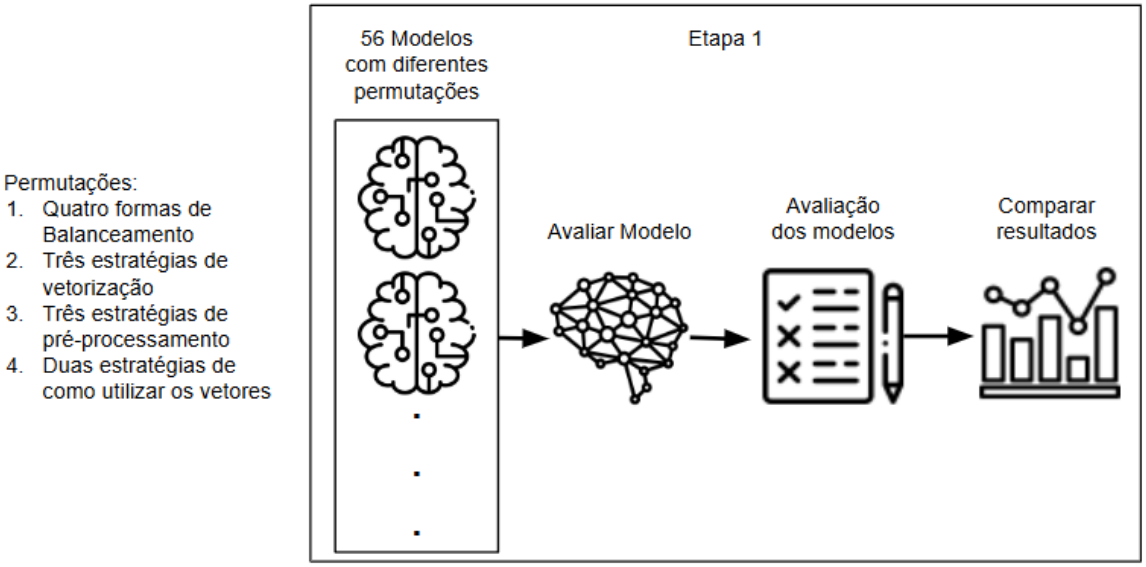


Figura 4.1: Detalhamento da Primeira Etapa de avaliação dos modelos

Para simplificar a compreensão, atribuímos um nome a cada modelo com base na estratégia utilizada para sua criação:

1. **Utilização de vetores**:
    - **CD**: utilização da distância de cosseno dos vetores como *feature*.
    - **VT**: utilização dos vetores como *feature*.
2. **Balanceamento**:
    - **UNB**: uso de dados desbalanceados para o treinamento do modelo.
    - **B11**: uso de dados balanceados na proporção de um similar para um não similar no treinamento do modelo.
    - **B12**: uso de dados balanceados na proporção de um similar para dois não similares no treinamento do modelo.
    - **B19**: uso de dados balanceados na proporção de um similar para nove não similares no treinamento do modelo.
3. **Abordagem de vetorização**:

- **TFIDF**: se utiliza o TF-IDF isoladamente.
- **T5**: se utiliza o T5 isoladamente.
- **HYBRID**: se utiliza a abordagem híbrida.

4. **Pré-processamento**:

- **TOK**: se o texto foi tokenizado.
- **STO**: se o texto foi tokenizado e removido as *stopwords*.
- **LEM**: se o texto foi tokenizado, removido as *stopwords* e lematizado.

Assim, quando nos referimos ao modelo treinado com dados desbalanceados (UNB), com utilização de distância de cosseno (CD) e TD-IDF (TFIDF) com tokenização (TOK), o designamos como UNB-CD-TFIDF-TOK. Como destacado anteriormente, o T5 não requer pré-processamento adicional, portanto, não haverá nenhuma sigla descrevendo o pré-processamento, por exemplo: B12-VT-T5.

Ao final da primeira etapa, selecionamos os modelos mais promissores e iniciamos a segunda etapa, referente ao ajuste fino dos modelos. Primeiramente, experimentamos três otimizadores populares e eficazes para problemas de classificação, devido à sua capacidade de adaptação às características dos dados e eficiência na convergência: Adam [42], SGD [7] e RMSprop [74]. Adam é conhecido por sua rápida convergência e adaptação individual das taxas de aprendizado. O SGD, por sua vez, é eficiente em termos de memória e pode escapar de mínimos locais. Já o RMSprop adapta a taxa de aprendizado, dividindo a taxa de aprendizado pelo valor médio dos gradientes recentes, o que ajuda a manter a estabilidade durante o treinamento.

Em seguida, avaliamos quais das duas funções de ativação, sigmoid [6] ou hard_sigmoid [72], se aplicam melhor para gerar valores entre 0 e 1. A função sigmoid é uma escolha comum em problemas de classificação binária, pois mapeia qualquer valor real para um intervalo entre 0 e 1, facilitando a interpretação probabilística das saídas. A hard_sigmoid, por sua vez, é uma aproximação da sigmoid que é computacionalmente mais eficiente, pois evita a utilização de exponenciais, resultando em um cálculo mais rápido e menos propenso a problemas de saturação. Por fim, por se tratar de uma classificação com

classes altamente desbalanceadas, também experimentamos a eficácia da classificação ponderada [73], onde atribuímos pesos relativos ou pesos iguais às classes.

Para avaliar a capacidade dos modelos, dividimos a análise em duas partes: classificação e recomendação. Na avaliação da capacidade de classificação, analisamos o desempenho dos modelos na tarefa de determinar se um par de BRs pertence à classe "similar" ou "não similar". A análise dos resultados foi realizada utilizando diversas métricas de desempenho, incluindo AUC-ROC, AUC-PR, FNR, FPR, TNR, TPR (*Recall*), precisão e *F1-Score*. Com exceção de AUC-ROC e AUC-PR, as métricas foram calculadas considerando que todas as previsões com valores iguais ou superiores a 0.5 foram classificadas como 1, e as previsões abaixo desse valor foram classificadas como 0. A escolha do *threshold* de 0.5 é comumente utilizada pois a função sigmoid retorna valores entre 0 e 1, e 0.5 é o ponto de corte natural para distinguir duas classes de forma balanceada [6].

Para avaliar a capacidade dos modelos em recomendar BRs similares a partir de um BR, com o objetivo de obter mais *insights* sobre seu desempenho, utilizamos as métricas de *precision@n* e *recall@n*. Primeiramente, para determinar os valores de n a serem analisados, examinamos a distribuição do número de similares que cada BR possui no *dataset* de testes.

Além de observar que só fazia sentido analisar valores de $n \geq 10$, os resultados indicaram que mais da metade dos BRs têm apenas um ou nenhum BR semelhante. Com base nos resultados, optamos por avaliar as métricas de *precision@n* e *recall@n* apenas para os BRs que possuem n ou mais similares. Isso significa que, ao invés de considerar todos os BRs, focamos apenas naqueles que possuem uma quantidade suficiente de BRs similares. Por exemplo, apenas avaliamos a capacidade do modelo de recomendar dois BRs com dados que contenham dois ou mais BRs similares disponíveis para recomendação, garantindo que as métricas de precisão e *recall* sejam calculadas de maneira mais significativa e representativa.

Portanto, neste capítulo, apresentamos e discutimos os resultados obtidos ao abordar as questões de pesquisa. Comparamos o desempenho de cada modelo de aprendizado de máquina, destacando aquele que superou os demais na previsão da similaridade entre BRs. Adicionalmente, exploramos as características e recursos que contribuíram significativamente para o desempenho do melhor modelo de classificação, elucidando os fatores que levaram ao seu sucesso. Além disso, discutimos as características do BR fortemente ligadas à similaridade, enfatizando sua importância na melhoria da qualidade e eficácia dos BRs.

4.1 Primeira Etapa

Na primeira etapa, dedicada ao treinamento e à exploração de diferentes estratégias para alimentar os dados nas redes neurais, foram testadas 56 combinações de parâmetros. Entre os modelos treinados, o número de épocas até a melhor época encontrada variou de 3 a 170, até que o critério de *early stopping* com paciência 5 fosse atingido [30].

4.1.1 Utilização da Vetorização

Para avaliar o impacto da forma como utilizamos vetores em nossos modelos, exploramos duas abordagens distintas no treinamento. Na primeira abordagem, os modelos foram treinados utilizando diretamente os vetores gerados pelos processos de vetorização TF-IDF e T5. Na segunda abordagem, os modelos foram alimentados com a distância do cosseno calculada previamente entre pares desses vetores, a fim de medir a similaridade entre os textos.

O objetivo desta análise é entender como a forma de utilizar os vetores influencia a performance do modelo na identificação e recomendação de BRs similares. Queremos determinar se o uso da distância do cosseno, calculada previamente, ou o uso dos vetores completos como *features* traz melhores resultados para a tarefa de identificação e recomendação de BRs similares.

Na Tabela 4.1, a utilização da distância de cosseno pré-calculada como *feature* apresenta uma faixa de valores de AUC-ROC que varia entre $77,71\%$ e $81,93\%$, com uma média de $80,13\%$. Por outro lado, a utilização de vetores completos como *features* apresenta uma faixa de valores de AUC-ROC que varia entre $81,01\%$ e $91,62\%$, com uma média de $88,35\%$.

Tabela 4.1: AUC-ROC de utilização de vetorização

Métrica	**CD**	**VT**
Média	$80,13\%$	$88,35\%$
Desvio padrão	$1,08\%$	$2,69\%$
Mínimo	$77,71\%$	$81,01\%$
Mediana	$80,06\%$	$88,90\%$
Máximo	$81,93\%$	$91,62\%$

Na Tabela 4.2, a utilização da distância de cosseno como *feature* apresenta uma faixa de

valores de AUC-PR, que varia entre $4,40\%$ e $5,87\%$, com uma média de $5,18\%$. Por outro lado, a utilização de vetores como *feature* apresenta uma faixa de valores de AUC-PR, que varia entre $6,69\%$ e $36,74\%$, com uma média de $17,73\%$. Portanto, fica evidente através das duas métricas que *a utilização de vetores como features demonstra ser mais eficiente com relação à apenas a utilização da distância de cosseno pré-calculada.*

Tabela 4.2: AUC-PR de utilização de vetorização

Métrica	**CD**	**VT**
Média	$5,18\%$	$17,73\%$
Desvio padrão	$0,39\%$	$10,27\%$
Mínimo	$4,40\%$	$6,69\%$
Mediana	$5,19\%$	$12,49\%$
Máximo	$5,87\%$	$36,74\%$

4.1.2 Balanceamento

O balanceamento de dados em um treinamento de modelo é uma técnica fundamental para melhorar a performance de modelos de aprendizado de máquina, especialmente quando se lida com conjuntos de dados desbalanceados. Portanto, é imprescindível saber definir qual é a melhor estratégia para o balanceamento dos dados, considerando sua interação com as estratégias de utilização dos vetores, vetorização e pré-processamento. Na pesquisa, além das outras análises, buscamos explorar quatro cenários diferentes de balanceamento: totalmente desbalanceado, balanceamento nove não similares para um similar, balanceamento dois não similares para um similar e balanceamento um não similares para um.

Após treinar os modelos com diferentes tipos de balanceamento, observamos que os modelos totalmente desbalanceados tendem a apresentar altos valores de FNR, TNR e precisão, enquanto os valores de FPR e TPR são baixos. Isso sugere que esses modelos tendem a classificar a maioria dos casos como negativos, mas quando identificam um caso como positivo, frequentemente estão corretos. Além disso, identificamos um *trade-off*: à medida que os dados são mais balanceados, os resultados de TPR (*recall*) melhoram, enquanto a precisão tende a diminuir, como ilustrado na Figura 4.2.

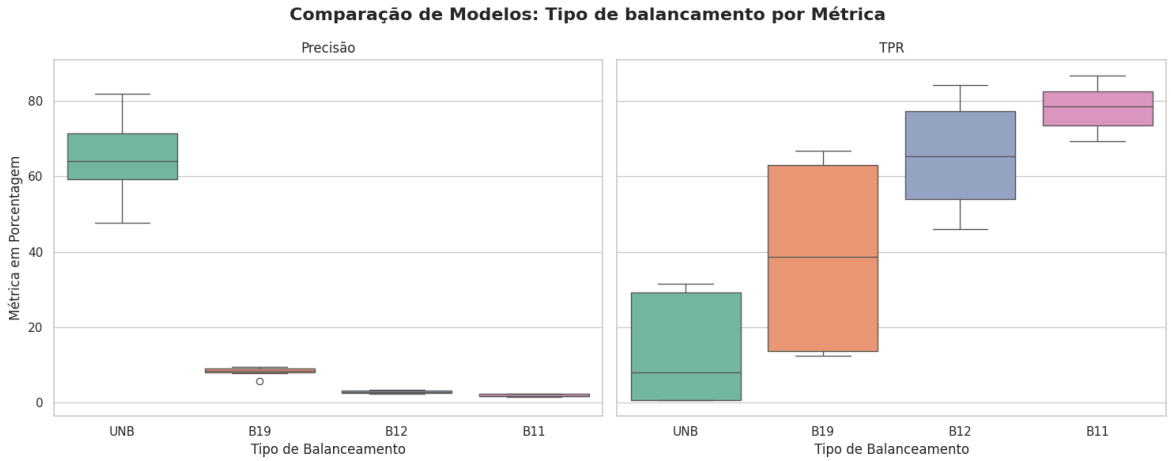


Figura 4.2: *Trade-off* de *precision* e *recall* relacionado ao balanceamento

Na Tabela 4.3, o balanceamento UNB apresenta uma faixa de valores entre $77,71\%$ e $88,49\%$, com uma média de $81,91\%$. O balanceamento B19 mostra uma faixa de valores mais ampla, variando de $79,67\%$ a $91,62\%$, com uma média de $86,02\%$. O balanceamento B12 tem uma faixa de valores entre $79,15\%$ e $89,68\%$, com uma média de $84,85\%$, enquanto o balanceamento B11 apresenta uma faixa de valores entre $78,97\%$ e $89,38\%$, com uma média de $84,19\%$. A variação nos valores sugere que os modelos com balanceamento B19 e B12 podem alcançar valores mais altos de AUC-ROC.

Tabela 4.3: AUC-ROC de balanceamento

Métrica	**UNB**	**B19**	**B12**	**B11**
Média	$81,91\%$	$86,02\%$	$84,85\%$	$84,19\%$
Desvio padrão	$3,42\%$	$5,43\%$	$4,66\%$	$4,21\%$
Mínimo	$77,71\%$	$79,67\%$	$79,15\%$	$78,97\%$
Mediana	$80,57\%$	$86,37\%$	$85,05\%$	$84,20\%$
Máximo	$88,49\%$	$91,62\%$	$89,68\%$	$89,38\%$

Para a métrica AUC-PR, exibido na Tabela 4.4, o balanceamento UNB apresenta uma faixa de valores que varia de $5,14\%$ a $36,74\%$, com uma média de $19,59\%$. O balanceamento B19 tem uma faixa de valores entre $4,88\%$ e $19,41\%$, com uma média de $11,30\%$. O balanceamento B12 mostra uma faixa de valores entre $4,71\%$ e $11,60\%$, com uma média de $8,03\%$, e o balanceamento B11 apresenta uma faixa de valores entre $4,40\%$ e $9,89\%$,

com uma média de $6,91\%$. Assim como na métrica AUC-ROC, a variação nos valores de AUC-PR sugere que o balanceamento UNB pode alcançar valores mais altos.

Tabela 4.4: AUC-PR de balanceamento

Métrica	**UNB**	**B19**	**B12**	**B11**
Média	$19,59\%$	$11,30\%$	$8,03\%$	$6,91\%$
Desvio padrão	$15,10\%$	$6,30\%$	$3,17\%$	$2,26\%$
Mínimo	$5,14\%$	$4,88\%$	$4,71\%$	$4,40\%$
Mediana	$13,79\%$	$9,62\%$	$7,25\%$	$5,96\%$
Máximo	$36,74\%$	$19,41\%$	$11,60\%$	$9,89\%$

Comparando as duas métricas, observamos que o balanceamento B19 e B12 tendem a ter um desempenho médio mais alto em AUC-ROC, enquanto o balanceamento UNB tem potencial para alcançar os maiores valores em AUC-PR. Em outras palavras, *os modelos com balanceamento B19 e B12 tendem a ter um melhor desempenho na discriminação entre classes*, conforme indicado pelos valores mais altos de AUC-ROC. Por outro lado, *o balanceamento UNB tem potencial para alcançar melhores resultados na identificação da classe positiva*, conforme indicado pelos valores mais altos de AUC-PR.

4.1.3 Estratégia de Vetorização

A escolha da estratégia de vetorização é um aspecto crucial na definição do treinamento do modelo, pois define a forma como o texto é transformado em dados numéricos que podem ser utilizados pelos algoritmos de aprendizado de máquina. A vetorização é o processo pelo qual representamos as palavras ou documentos em um espaço vetorial, e a escolha da técnica adequada pode impactar significativamente a performance do modelo. Para a pesquisa, variamos a utilização de apenas vetores TF-IDF, apenas vetores T5 e a estratégia híbrida, que consiste na utilização das duas vetorizações em conjunto.

Na Tabela 4.5, os valores médios de AUC-ROC são muito semelhantes, em torno de 84%. O intervalo de variação para T5 é de $77,71\%$ a $91,60\%$, para TFIDF é de $77,71\%$ a $91,62\%$, e para HYBRID é de $78,54\%$ a $91,62\%$.

Já na Tabela 4.6, as estratégias TFIDF e HYBRID apresentam valores médios de AUC-

Tabela 4.5: AUC-ROC de estratégia de vetorização

Métrica	**T5**	**TFIDF**	**HYBRID**
Média	$84,20\%$	$84,09\%$	$84,18\%$
Desvio padrão	$4,73\%$	$4,67\%$	$4,67\%$
Mínimo	$77,71\%$	$78,14\%$	$78,54\%$
Mediana	$83,16\%$	$82,46\%$	$81,59\%$
Máximo	$91,60\%$	$91,25\%$	$91,62\%$

PR mais altos, de $11,98\%$ e $11,86\%$, respectivamente, em comparação com T5, que tem um valor médio de $8,84\%$. No entanto, os intervalos de variação para TFIDF e HYBRID são mais amplos, de $4,40\%$ a $36,74\%$, sugerindo que os modelos mais promissores estão entre eles.

Tabela 4.6: AUC-PR de estratégia de vetorização

Métrica	**T5**	**TFIDF**	**HYBRID**
Média	$8,84\%$	$11,80\%$	$11,99\%$
Desvio padrão	$6,00\%$	$10,08\%$	$10,23\%$
Mínimo	$4,63\%$	$4,40\%$	$5,04\%$
Mediana	$5,98\%$	$7,47\%$	$7,38\%$
Máximo	$21,71\%$	$36,08\%$	$36,74\%$

Em resumo, em termos de AUC-ROC, todas as três estratégias de vetorização têm desempenho semelhante. Já em termos de AUC-PR, *as estratégias TFIDF e HYBRID alcançam melhores resultados que T5*, tanto no melhor resultado quanto no desempenho médio.

4.1.4 Pré-processamento

A análise do impacto do pré-processamento de dados é uma etapa crucial no treinamento de aprendizado de máquina, tendo em vista que envolve a transformação dos dados brutos em um formato adequado para a modelagem. No caso da pesquisa, testamos o treinamento do texto apenas separado por tokens, separado por tokens e com stopwords removidas e, por fim, separado por tokens, com stopwords removidas e lematizadas.

Na Tabela 4.7, observa-se que todas as três estratégias apresentam valores médios de AUC-ROC muito semelhantes, em torno de 84%. O intervalo de variação para todas as três abordagens também demonstra ser bem parecido, variando aproximadamente entre 77% a 91%.

Tabela 4.7: AUC-ROC de estratégias de pré-processamento

Métrica	**TOK**	**STO**	**LEM**
Média	$84,20\%$	$84,09\%$	$84,18\%$
Desvio padrão	$4,73\%$	$4,67\%$	$4,67\%$
Mínimo	$77,71\%$	$78,14\%$	$78,54\%$
Mediana	$83,16\%$	$82,46\%$	$81,59\%$
Máximo	$91,60\%$	$91,25\%$	$91,62\%$

Na Tabela 4.8, as estratégias também demonstram valores médios de AUC-PR semelhantes, com médias aproximadamente de 11% e um intervalo entre $4,40\%$ e $36,74\%$. Em concordância com as conclusões obtidas nas métricas de AUC-ROC, os intervalos de variação das três métricas também são bem parecidos. Em resumo, em termos de AUC-ROC e AUC-PR, *todas as três estratégias de vetorização têm desempenho semelhante*.

Tabela 4.8: AUC-PR de estratégias de pré-processamento

Métrica	**TOK**	**STO**	**LEM**
Média	$11,15\%$	$11,98\%$	$11,86\%$
Desvio padrão	$9,73\%$	$10,46\%$	$10,20\%$
Mínimo	$4,40\%$	$4,62\%$	$4,80\%$
Mediana	$6,16\%$	$7,46\%$	$7,36\%$
Máximo	$36,70\%$	$36,74\%$	$35,76\%$

Em resumo, em termos de AUC-ROC, todas as três estratégias de vetorização têm desempenho semelhante. Já em termos de AUC-PR, *as estratégias TFIDF e HYBRID alcançam melhores resultados que T5*, tanto no melhor resultado quanto no desempenho médio.

4.1.5 Recomendação

Durante a análise dos resultados das métricas de recomendação, observou-se uma concordância entre as conclusões obtidas a partir das métricas de recomendação e dos resultados de classificação discutidos anteriormente, pois os que alcançaram melhores resultados em *precision@n* e *recall@n* foram aqueles que também atingiram os melhores resultados em AUC-PR. Os resultados das métricas de ranqueamento revelaram que *os modelos mais promissores não usaram balanceamento ou utilizaram balanceamento B19 e usaram vetores como features*.

Para uma melhor visualização dos resultados de *precision@n* e *recall@n* variando o valor de n de 1 a 10, foram criados vários gráficos de linha dos modelos que utilizam vetores como *features*. Cada gráfico representa os resultados separados pela estratégia de balanceamento utilizada. Assim, os modelos que utilizaram dados desbalanceados estão representados na Figura 4.3 e 4.4, enquanto os que utilizaram balanceamento B19 estão na Figura 4.5 e 4.6.

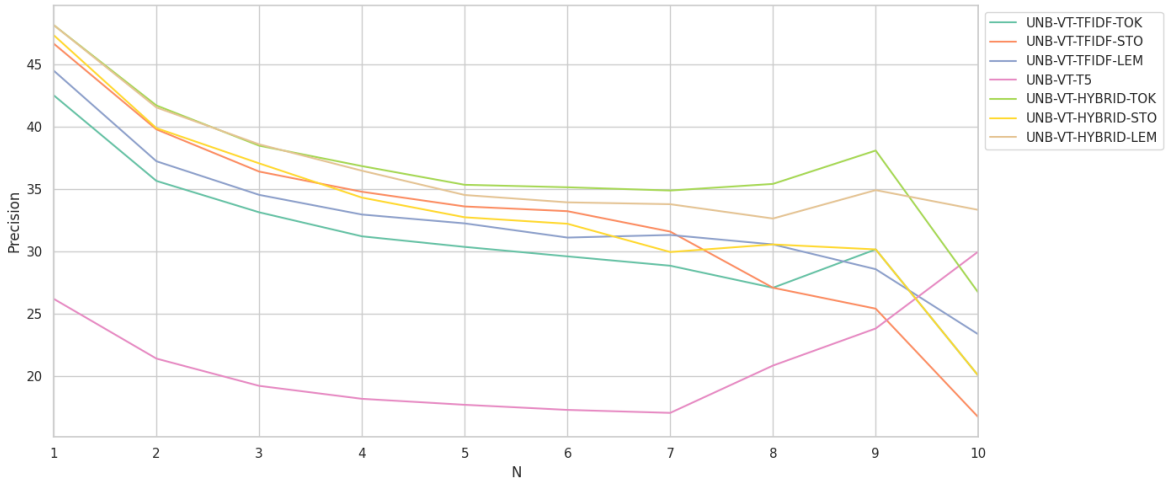


Figura 4.3: Gráfico de *precision@n* para balanceamento UNB

Também em concordância com os resultados obtidos anteriormente, podemos ver claramente que *a vetorização T5 isolada demonstrou desempenho inferior ao TF-IDF isolado ou híbrido*. Embora uma abordagem híbrida tenha superado o TF-IDF isolado em muitos casos, o TF-IDF isolado também mostrou resultados promissores em certos cenários.

Além disso, o cenário desbalanceado, em geral, obteve os melhores resultados de *recall@n* quando $n < 3$ e *precision@n* quando $n \leq 3$. Por exemplo, o modelo UNB-VT-HYBRID-Tokens alcançou mais de 48% de *precision@1* e mais de 25% de *recall@1*, indi-

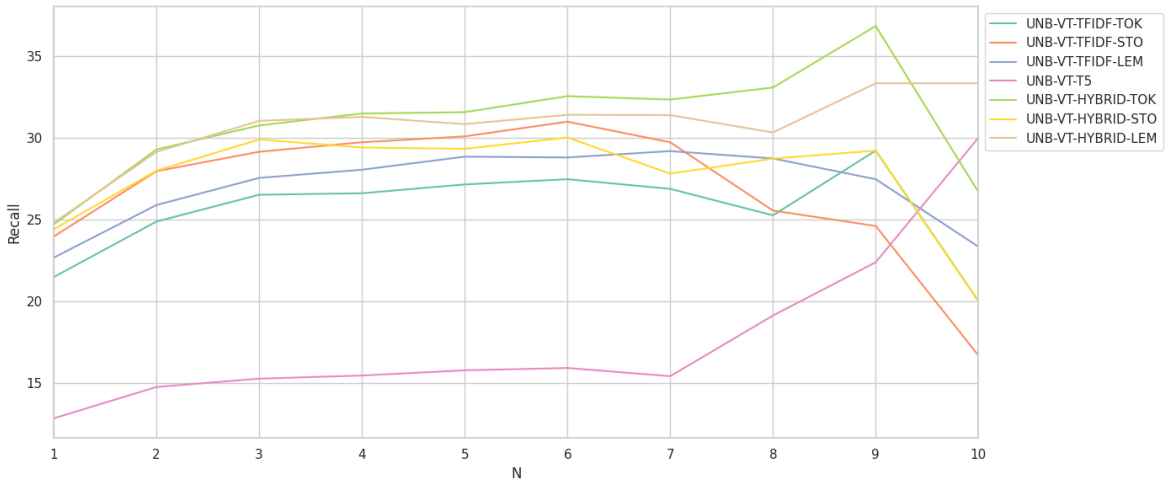


Figura 4.4: Gráfico de *recall@n* para balanceamento UNB

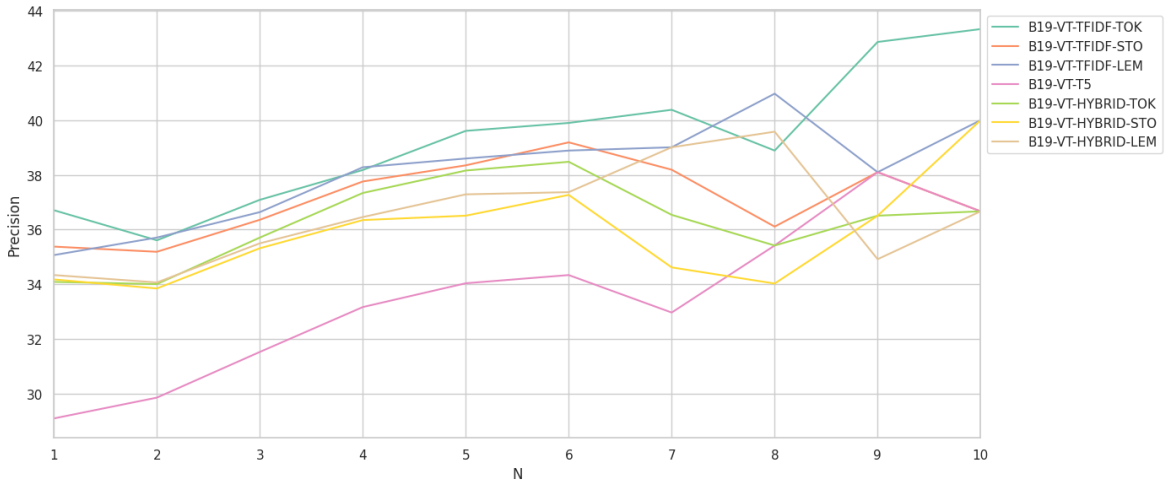


Figura 4.5: Gráfico de *precision@n* para balanceamento B19

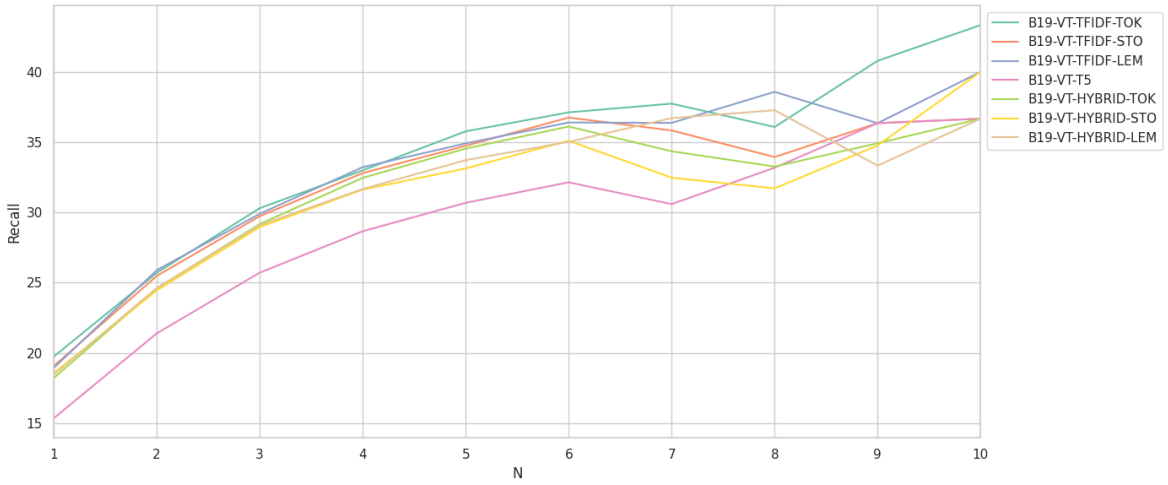


Figura 4.6: Gráfico de *recall@n* para balanceamento B19

cando que quase metade das recomendações foram corretas e mais de um quarto delas foram identificadas corretamente. No entanto, quando $n \geq 4$, a utilização do balanceamento B19 mostrou melhores resultados, alcançando quase 45% de *precision@10* e *recall@10*. Este resultado também está em concordância com a observação anterior de que modelos desbalanceados tendem a ter alta precisão, mas baixo *recall*, resultando em desempenho inferior em n elevados.

4.1.6 Teste de Hipótese

A análise estatística do teste de hipótese revelou que balanceamento de dados teve um impactando significativamente na maioria das métricas avaliadas. Diferentes técnicas de balanceamento se destacaram em aspectos específicos: o não-balanceamento (UNB) mostrou-se superior em métricas como AUC-PR, TNR, FPR e Precision, enquanto o balanceamento B11 apresentou melhor desempenho em FNR e TPR. Também notou-se que a utilização de vetores como *features* demonstrou um impacto positivo consistente, superando outras abordagens em métricas como AUC-ROC, AUC-PR, FNR e TPR.

Quanto às técnicas de vetorização, observou-se um impacto menos pronunciado na maioria das métricas, com exceção das medidas Precision@N e Recall@N. Nestas, o método T5 apresentou um desempenho significativamente diferente dos demais. É importante ressaltar que esta diferença não implica necessariamente em superioridade ou inferioridade do T5. As diferentes abordagens de pré-processamento também não resultaram em diferenças estatisticamente significativas para nenhuma das métricas analisadas. Isso sugere que, para este conjunto de dados e tarefas específicas, o impacto do pré-processamento pode ser menos crítico em comparação com outras variáveis do modelo.

4.1.7 Seleção de Modelos Mais Promissores

De forma geral, os modelos que utilizam o balanceamento UNB e B19 com a utilização de vetores como *features* se destacam devido à sua eficácia em várias dimensões para a classificação. Esta conclusão é evidenciada pelo desempenho superior desses modelos nas métricas que avaliam o desempenho geral: AUC-ROC, AUC-PR e *F1-Score*.

O AUC-ROC, que mede a capacidade do modelo de distinguir entre classes, apresenta va-

lores notavelmente altos para os modelos mencionados. O B19-VT-TFIDF-LEM, por exemplo, atinge um impressionante AUC-ROC de 91.62%, com outras variantes do B19-VT superando consistentemente 90.81%. Esta performance indica uma excelente capacidade de discriminação entre classes positivas e negativas, um feito não alcançado por outros grupos de modelos.

Quanto ao AUC-PR, métrica fundamental para conjuntos de dados desbalanceados, os modelos UNB-VT se destacam significativamente. O UNB-VT-HYBRID-STO alcança um AUC-PR de 36.74%, com outras variantes UNB-VT apresentando resultados acima de 33.38%, com exceção do que utiliza a vetorização T5 que atingiu 21.71%. Desconsiderando os modelos UNB-VT, os demais modelos atingem 19.41% valores de AUC-PR. Através destes valores, torna-se notável a superioridade dos modelos UNB-VT ao demonstrar um equilíbrio robusto entre precisão e *recall*.

O F1-Score, representando a média harmônica entre precisão e *recall*, também evidencia a superioridade destes modelos. Os UNB-VT consistentemente apresentam os *F1-Scores* mais elevados, com o UNB-VT-HYBRID-TOK atingindo 40.38%. Esta métrica oferece uma visão equilibrada do desempenho, considerando tanto falsos positivos quanto falsos negativos. Novamente, apenas os modelos UNB-VT alcançam F1-Scores acima de 23.16%, um nível não atingido por outros grupos.

Além dessas métricas gerais, é importante destacar outros resultados excepcionais que esses modelos alcançam:

1. Precisão e TNR (*True Negative Rate*): Os modelos UNB atingem níveis de precisão extraordinariamente altos, com o UNB-CD-HYBRID-STO chegando a 81.94%. Além disso, os modelos UNB apresentam TNRs próximos ou iguais a 100%. Nenhum outro grupo de modelos se aproxima desse nível de precisão. Isso indica uma habilidade quase perfeita em identificar corretamente os casos negativos. Que faz sentido tendo em vista que são os modelos que tem em seu treinamento mais casos da classe não similar.

2. TPR/*Recall* (*True Positive Rate*): Os modelos que utilizam o balanceamento B19 e B11, juntamente com vetores como *features* apresentam TPRs altos, com o B11-VT-T5-TOK atingindo 86.68%. Isso demonstra uma capacidade superior de identificar

corretamente os casos positivos. Que faz sentido se analisar que os modelos balanceados foram treinados com uma quantidade relativa maior de similares.

Como descrito na seção 4.1.5, os modelos mais promissores no contexto de recomendação foram os que tinham o balanceamento UNB e B19 com a utilização de vetores como *features*. Em conjunto com estas informações, é notável que os modelos UNB, B19 e VT não apenas lideram em uma métrica específica, mas mantêm um desempenho superior consistente em todas as métricas gerais mencionadas.

A notável consistência dos modelos UNB e B19 com vetores como *features* é evidenciada por seu desempenho superior em múltiplas métricas: AUC-ROC, AUC-PR, F1-Score, *Recall*@n e *Precision*@n. Esta versatilidade os torna ideais para aplicações práticas, onde o desempenho otimizado em várias dimensões simultaneamente é crucial. Consequentemente, dentre os 56 modelos inicialmente treinados e analisados, selecionamos para a etapa de ajuste fino os 14 modelos que combinam a utilização de vetores como *features* com os balanceamentos B19 ou UNB, reconhecendo seu potencial superior.

4.2 Segunda Etapa

Durante a segunda etapa, que envolveu o ajuste fino dos 14 melhores modelos identificados com base nas métricas de classificação e recomendação, foram testadas 12 configurações diferentes para cada modelo, totalizando 168 combinações. A escolha desses 14 modelos foi baseada no desempenho observado, considerando apenas aqueles que apresentaram AUC-ROC superior a 90% e AUC-PR superior a 20%, garantindo a seleção de modelos com boa capacidade discriminativa e de predição. Observamos que alguns modelos atingiram seu desempenho máximo entre 1 e 199 épocas, chegando ao limite pré-definido de 200 épocas sem acionar o *early stopping*.

Ao analisar os resultados, notamos que todos os modelos que ultrapassaram 100 épocas utilizaram o otimizador SGD, que requer muitas épocas para convergir. Isso indica que *modelos que utilizaram o SGD como otimizador poderiam obter melhores resultados caso fossem treinados por mais épocas*. Além disso, notamos que, com a variação das configurações, alguns modelos não conseguiram aprender corretamente. Como resultado, esses modelos classificaram quase todos os BRs como não similares, apresentando métricas de

AUC-ROC próximas de 50%. Isso sugere que *parte dos modelos treinados com algumas combinações de hiperparâmetros apresentam um desempenho equivalente ao de um modelo aleatório*.

4.2.1 Otimizador

A escolha do otimizador tem papel fundamental no treinamento de uma rede neural, influenciando a velocidade, estabilidade e precisão do modelo, além de sua capacidade de generalizar para novos dados. Ele também afeta a robustez dos dados ruidosos e a eficiência computacional. No contexto da pesquisa foram experimentados os otimizadores SGD, RMSprop e Adam. Conforme citado anteriormente, para uma análise mais acurada dos resultados, removemos os modelos equivalentes a modelos aleatórios.

Na Tabela 4.9 de AUC-ROC, o otimizador Adam apresenta uma variação de $76,08\%$ a $91,66\%$, com uma média de $87,06\%$, indicando uma boa performance geral. O RMSprop varia de $80,98\%$ a $92,14\%$, com média de $89,33\%$. O SGD, por sua vez, varia de $71,32\%$ a $90,34\%$, com uma média em torno de $84,61\%$.

Tabela 4.9: AUC-ROC de otimizadores

Métrica	**Adam**	**SGD**	**RMSprop**
Média	$87,06\%$	$84,61\%$	$89,33\%$
Desvio padrão	$4,50\%$	$6,50\%$	$3,16\%$
Mínimo	$76,08\%$	$71,32\%$	$80,98\%$
Mediana	$89,06\%$	$88,08\%$	$90,78\%$
Máximo	$91,66\%$	$90,34\%$	$92,14\%$

Na Tabela 4.10 de AUC-PR, o otimizador Adam apresenta uma variação de $5,10\%$ a $37,11\%$, com uma média de $23,97\%$, indicando uma boa performance geral com variação significativa. O RMSprop varia de $11,14\%$ a $24,24\%$, com uma média de $16,86\%$, mostrando uma performance razoável com variação moderada. O SGD varia de $1,58\%$ a $18,62\%$, com uma média de $9,45\%$, indicando a pior performance.

Em resumo, *Adam e RMSprop tendem a atingir melhores performances com relação ao SGD, tanto levando em consideração AUC-PR quanto AUC-ROC. Em termos de AUC-PR*

Tabela 4.10: AUC-PR de otimizadores

Métrica	**Adam**	**SGD**	**RMSprop**
Média	$23,97\%$	$9,45\%$	$16,86\%$
Desvio padrão	$9,57\%$	$5,86\%$	$3,44\%$
Mínimo	$5,10\%$	$1,58\%$	$11,14\%$
Mediana	$18,88\%$	$9,75\%$	$16,88\%$
Máximo	$37,11\%$	$18,62\%$	$24,24\%$

isoladamente, Adam demonstra a capacidade de atingir valores mais altos. Observamos que a quantidade de modelos aleatórios também foi maior entre os modelos treinados com otimizador Adam e RMSprop, sendo 8 com RMSprop, 8 com Adam e 7 com SGF.

4.2.2 Função de Ativação

A escolha da função de ativação tem papel fundamental na aprendizagem das redes neurais pois introduz não-linearidade, permitindo que a rede aprenda e modele relações complexas entre entradas e saídas. Sem essa não-linearidade, a rede seria equivalente a uma única camada linear, independentemente de sua profundidade. Em resumo, elas tornam as redes neurais capazes de resolver problemas complexos e aprender padrões nos dados. Na pesquisa exploramos a utilização das funções hard_sigmoid e sigmoid, tendo em vista que buscamos encontrar valores entre 0 e 1.

Nas Tabelas 4.11 e 4.12, percebemos que chegamos às mesmas conclusões comparativas, que as funções apresentam um desempenho similar. Percebemos que a média da sigmoid tem valores ligeiramente superiores, que a variação da sigmoid também é menor e o melhor resultado da hard_sigmoid seja também ligeiramente superior para as duas métricas. Em termos de AUC-ROC, os intervalos entre aproximadamente 71% e 92% e uma média $85,56\%$ e $88,02\%$ para hard_sigmoid e sigmoid, respectivamente. Já AUC-PR, os intervalos tendem a ser entre aproximadamente 1% e 37%, enquanto as médias são aproximadamente $14,22\%$ e $18,52\%$ para hard_sigmoid e sigmoid.

Portanto, a partir dos resultados obtidos, podemos concluir que *apesar da função hard_sigmoid mostrar os melhores valores, a função sigmoid tende a ser mais constante*

na entrega de bons resultados.

Tabela 4.11: AUC-ROC de função de ativação

Métrica	**sigmoid**	**hard_sigmoid**
Média	$88,02\%$	$85,56\%$
Desvio padrão	$4,39\%$	$6,03\%$
Mínimo	$73,45\%$	$71,32\%$
Mediana	$90,02\%$	$88,94\%$
Máximo	$91,66\%$	$92,14\%$

Tabela 4.12: AUC-PR de função de ativação

Métrica	**sigmoid**	**hard_sigmoid**
Média	$18,52\%$	$14,22\%$
Desvio padrão	$8,62\%$	$8,95\%$
Mínimo	$1,88\%$	$1,58\%$
Mediana	$17,52\%$	$14,92\%$
Máximo	$36,74\%$	$37,11\%$

Corroborando com a conclusão acima, percebemos que *todos os modelos considerados aleatórios ou quase aleatórios são obtidos através da função hard_sigmoid*, totalizando 23 modelos aleatórios.

4.2.3 Classificação Ponderada

No contexto de problemas de classificação com classes desbalanceadas, é fundamental distinguir entre duas abordagens: o balanceamento de dados e a classificação ponderada. Enquanto o balanceamento de dados altera a distribuição das classes no conjunto de treinamento, a classificação ponderada ajusta a importância de cada classe durante o processo de aprendizagem, sem modificar o conjunto de dados original.

O balanceamento de dados refere-se à manipulação direta do conjunto de treinamento, utilizando técnicas como as mencionadas anteriormente (UNB, B19, B12 e B11). O objetivo

principal é ajustar o conjunto de dados para que as classes estejam equilibradas, evitando que o modelo se torne tendencioso em favor da classe majoritária, o que poderia comprometer sua capacidade de generalizar e fazer predições precisas para todas as classes.

Por outro lado, a classificação ponderada é uma técnica que não altera o conjunto de dados, mas ajusta o processo de aprendizagem do modelo. O parâmetro *class_weight* no TensorFlow é um exemplo desta abordagem. Ele permite atribuir pesos diferentes às classes durante o treinamento, influenciando a função de perda para penalizar mais fortemente os erros nas classes menos representadas.

Nas Tabelas 4.13 e 4.13, que mostram o desempenho das métricas AUC-ROC e AUC-PR para diferentes esquemas de ponderação de classes, observa-se que ambos os *tipos de pesos têm uma faixa de valores e médias semelhantes, demonstrando não ter muito impacto na sua escolha*. Para AUC-ROC, percebemos nos dois casos intervalos entre aproximadamente 71% e 92% e médias aproximadamente de 87%. Tratando-se de AUC-PR, os valores tendem a variar aproximadamente entre 1% e 37% com média de 16%.

Tabela 4.13: AUC-ROC classificação ponderada

Métrica	**Pesos Relativos**	**Pesos Iguais**
Média	$87,11\%$	$86,86\%$
Desvio padrão	$5,20\%$	$5,36\%$
Mínimo	$71,32\%$	$73,45\%$
Mediana	$89,27\%$	$89,17\%$
Máximo	$92,14\%$	$92,14\%$

Tabela 4.14: AUC-PR classificação ponderada

Métrica	**Pesos Relativos**	**Pesos Iguais**
Média	$16,94\%$	$16,49\%$
Desvio padrão	$8,90\%$	$9,12\%$
Mínimo	$1,58\%$	$1,75\%$
Mediana	$16,96\%$	$16,89\%$
Máximo	$37,11\%$	$37,11\%$

Corroborando com os resultados acima, *a distribuição de modelos aleatórios é quase igualitária entre os dois pesos*, tendo apenas um modelo a mais nos modelos que utilizaram a função *class_weight* com pesos relativos, sendo 12 modelos aleatórios utilizando pesos relativos e 11 utilizando pesos iguais.

É importante notar que, enquanto o balanceamento de dados (UNB, B19) foi aplicado na preparação do conjunto de treinamento, a classificação ponderada foi implementada durante o processo de treinamento do modelo. Ambas as técnicas visam abordar o problema de classes desbalanceadas, mas em diferentes estágios do processo de modelagem e de maneiras distintas.

4.2.4 Recomendação

Ao contrário das métricas de classificação, as métricas de recomendação revelaram que muitos dos melhores desempenhos dos modelos foram alcançados utilizando RMSprop como otimizador durante o treinamento. Nas Figuras 4.7 e 4.8, é possível comparar as melhores configurações de cada modelo, onde claramente o modelo UNB-VT-TFIDF-TOK com Adam como otimizador, hard_sigmoid como função de saída e sem *class_weight* obteve o melhor resultado. Como mencionado anteriormente, os modelos desbalanceados demonstram uma precisão destacada em prever corretamente para valores baixos de *n*. No caso deste melhor modelo, ele conseguiu alcançar quase 55% de *precision@1* e quase 30% de *recall@1*, indicando que, dos aproximadamente 30% de itens similares recuperados, mais da metade foram corretamente identificados.

Em concordância com as métricas de classificação, as métricas de recomendação também indicaram que *os melhores modelos utilizaram Adam ou RMSprop como otimizador durante o treinamento*. Nas Figuras 4.9 e 4.10, é possível comparar as melhores configurações de cada modelo, onde claramente o modelo B19-VT-HYBRID-STO com RMSprop como otimizador, sigmoid como função de saída e sem *class_weight* obteve o melhor resultado. Observamos que, em geral, os modelos apresentam resultados similares quando $n \leq 7$ em precisão e *recall*, com exceção do modelo T5 isolado (representado pela linha azul claro). Para valores de $n > 7$, o modelo se destacou significativamente, alcançando mais de 45% de *precision@10* e *recall@10*.

De forma geral, podemos eleger o modelo UNB-VT-TFIDF-Tokens com a configuração

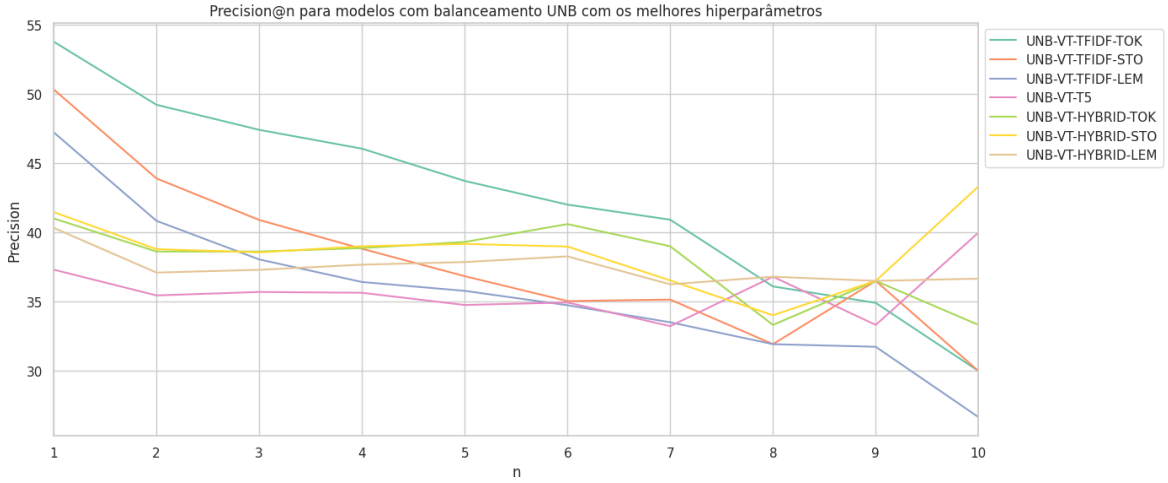


Figura 4.7: Gráfico de *precision@n* de modelos *tunados* com balancemanto UNB

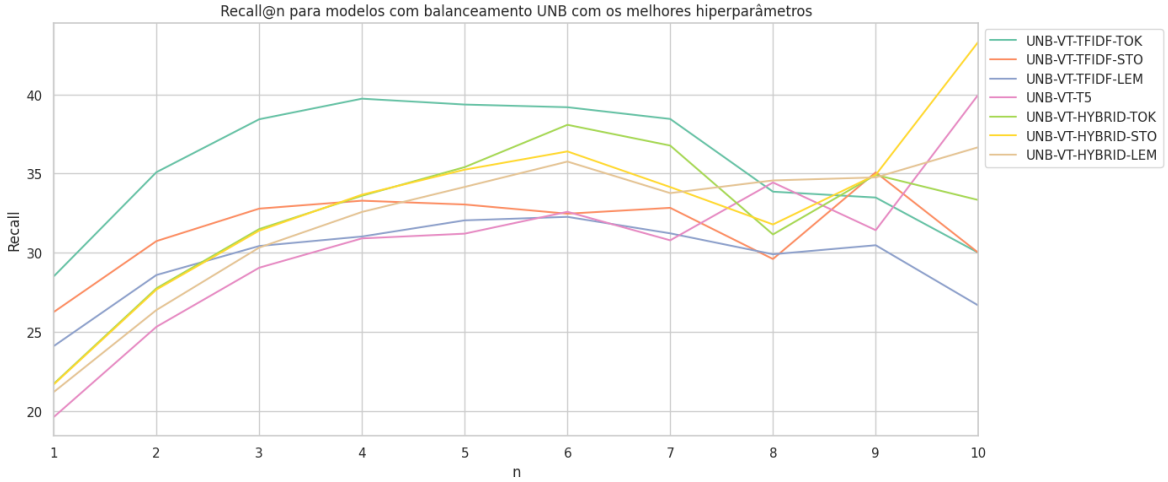


Figura 4.8: Gráfico de *recall@n* de modelos *tunados* com balancemanto UNB

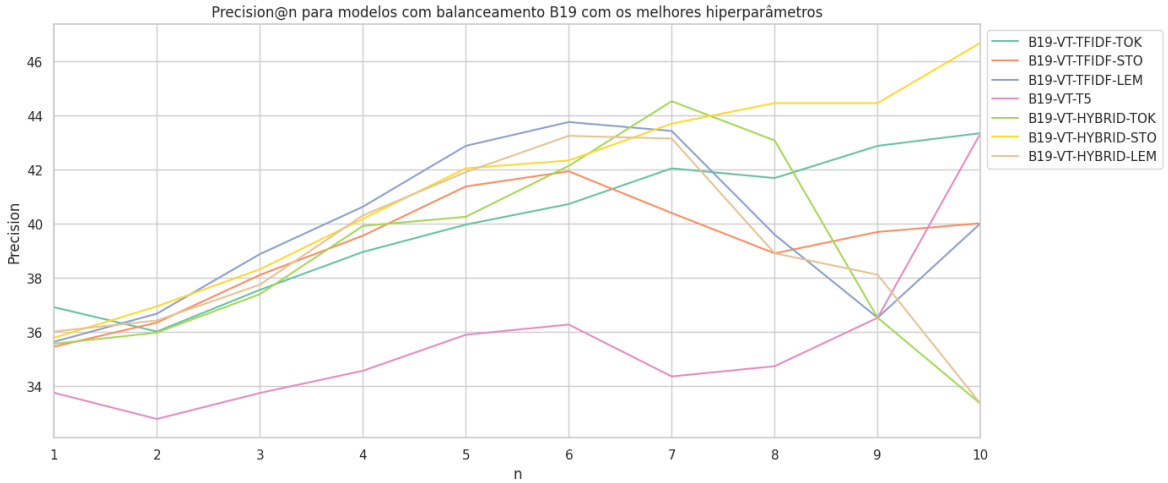


Figura 4.9: Gráfico de *precision@n* de modelos *tunados* com balancemanto B19

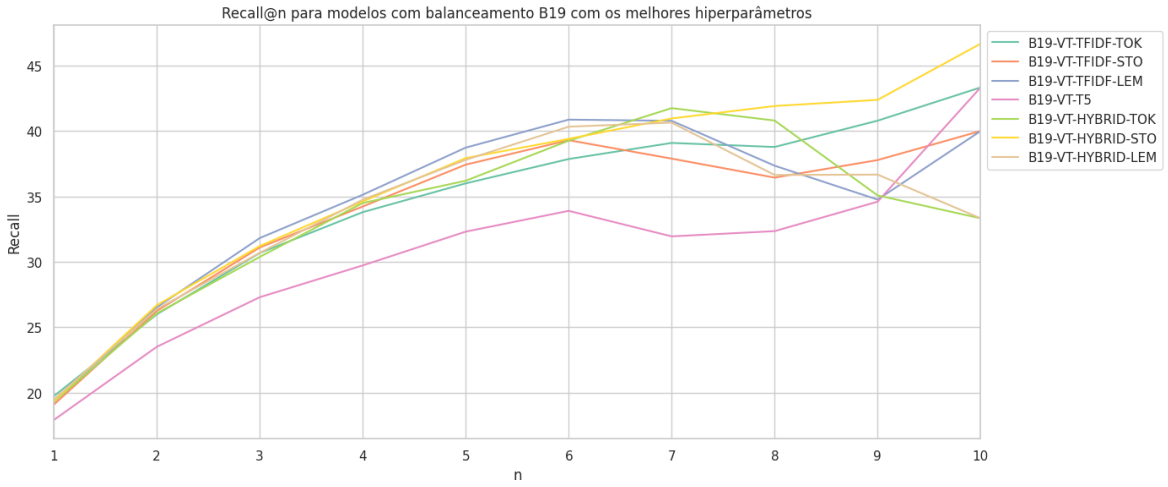


Figura 4.10: Gráfico de *recall@n* de modelos *tunados* com balancemanto B19

Adam, *hard_sigmoid* e com *class_weight* default como melhor modelo, tendo valores altos em quase todas as métricas, sendo elas AUC-PR $34,76\%$, AUC-ROC de $83,38\%$, F1-Score de $37,30\%$ para o *threshold* de $0,5$ e possuindo melhores *precision@n* e *recall@n* para qualquer valor de $n \leq 7$. Porém, se utilizarmos $n < 7$, o modelo B19-VT-HYBRID-STO com a configuração RMSprop, sigmoid e com *class_weight default* apresenta melhores resultados.

Qual o desempenho comparativo dos modelos TF-IDF, T5 isolado e T5 + TF-IDF na identificação de BRs similares?

Segundo os testes de hipótese realizados, percebemos uma diferença substancial entre os modelos T5 e os demais modelos. Em termos de métricas de classificação, percebemos que os modelos T5 apresentaram um desempenho consideravelmente inferior aos que utilizaram TF-IDF ou a abordagem híbrida. Os modelos que utilizam TF-IDF sozinho, excluindo resultados de AUC-ROC considerados aleatórios ou quase aleatórios, apresentaram valores de AUC-ROC variando de 73% a 90% e AUC-PR variando de 32% a 37%, com os melhores desempenhos ocorrendo com o otimizador Adam. Por outro lado, os modelos híbridos (UNB-VT-HYBRID), também excluindo resultados de AUC-ROC considerados aleatórios ou quase aleatórios, tiveram valores de AUC-ROC variando entre 71% e 91% e AUC-PR entre 35% e 36%. Portanto, em termos de métricas de classificação, percebemos a inferioridade do modelo T5 isolado, mas não é possível identificar um modelo claramente superior neste

contexto, uma vez que há sobreposição nos intervalos de confiança e os valores são bastante próximos.

Quanto às métricas de recomendação, também reparamos um desempenho inferior dos modelos que utilizaram apenas o T5 isolado com relação aos demais modelos. Observamos que a maioria dos modelos utilizando TF-IDF isolado, após ajustes, obteve melhores resultados quando $n \leq 3$. Para a configuração UNB-VT-TFIDF, ele manteve os melhores resultados de *precision@n* e *recall@n* até $n < 7$. No entanto, a partir de $n \geq 7$, os modelos híbridos demonstraram um desempenho superior em comparação com os modelos que utilizam TF-IDF isoladamente. *De maneira geral, os modelos híbridos apresentaram um desempenho inferior em relação aos modelos que utilizam TF-IDF isoladamente, embora os modelos híbridos demonstrem serem promissores*.

Em concordância com os resultados obtidos, os testes de hipóteses de Kruskal Wallis e ANOVA, juntamente com os post-hoc de Dunn e Tukey, demostraram que o T5 isolado possui uma diferença significativa dos demais.

Como a escolha de hiperparâmetros impacta no desempenho dos modelos testados?

Os otimizadores são cruciais na configuração dos modelos, sendo um dos hiperparâmetros mais impactantes. Nos testes realizados, os otimizadores Adam e RMSprop frequentemente apresentaram os melhores desempenhos. Isso indica que *Adam é eficaz em ajustar os pesos do modelo para maximizar a precisão e a recuperação de verdadeiros positivos*. Em contraste, o otimizador SGD mostrou convergência mais lenta em termos de número de épocas, demonstrando a necessidade de mais épocas para treinamento. Esses resultados foram evidenciados por meio dos testes de hipótese, da avaliação das métricas de classificação e da análise das métricas de recomendação.

As funções de saída, como sigmoid e hard_sigmoid, também desempenham papel importante no desempenho dos modelos. Percebemos que a função *hard_sigmoid*, tendeu a gerar modelos comparáveis a modelos aleatórios, classificando todos os resultados como não similares. Por outro lado, a função de saída sigmoid, quando combinada com otimizadores como Adam ou RMSprop, resultou em ótimos desempenhos tanto em AUC-ROC quanto em

AUC-PR. *De maneira geral, isso sugere que o sigmoid é mais eficaz em ajustar os limiares de classificação para maximizar a recuperação de verdadeiros positivos, embora parte dos melhores resultados também tenham utilizado hard_sigmoid*. Essa instabilidade dos modelos que utilizaram hard_sigmoid foi confirmada através dos testes de hipótese como uma diferença significativa.

O *class_weight*, em alguns casos, demonstrou melhorar os resultados alcançados, mas em geral, foi confirmado pelos testes de hipótese e pela análise das métricas que o *uso da classificação ponderada não impactou significativamente na melhoria dos modelos*. Em resumo, *a escolha de hiperparâmetros no geral demonstrou ter um impacto substancial no desempenho dos modelos testados*, afetando as métricas de desempenho, como AUC-ROC, AUC-PR, *precision@n* e *recall@n*.

4.3 Análise Geral

Este capítulo tem como propósito apresentar os resultados e análises derivados da identificação e recomendação de BRs similares usando diversas abordagens de treinamento de redes neurais. A partir dos resultados apresentados, o objetivo é responder às questões de pesquisa RQ1 e RQ2 através de uma série de treinamentos, subdividindo-se em uma primeira etapa de treinamento e uma segunda etapa de ajuste fino.

Durante a primeira etapa, exploramos diferentes estratégias de vetorização, balanceamento de dados e pré-processamento. A partir dos resultados dos 56 modelos testados, observamos que a utilização de vetores completos como features foi mais eficiente do que a distância de cosseno. Modelos treinados com balanceamento B19 e B12 apresentaram melhor desempenho em AUC-ROC, enquanto o balanceamento UNB foi superior em AUC-PR. Além disso, a utilização das abordagens híbridas e TF-IDF isolada apresentou melhores resultados. O pré-processamento, por sua vez, não demonstrou diferenças significativas entre as diferentes estratégias testadas.

Na segunda etapa, foram selecionados 14 modelos para o ajuste fino, testando 12 configurações para cada, totalizando 168 treinamentos. Durante essa etapa, alteramos os hiperparâmetros: otimizador, função de ativação e classificação ponderada. A partir dos experimentos, observamos que os otimizadores Adam e RMSprop apresentaram os melhores resultados,

embora grande parte dos modelos que utilizaram SGD tenham demonstrado precisar de mais épocas para atingir seu potencial máximo. Além disso, percebemos que a função sigmoid comportou-se bem na maioria dos cenários, fornecendo resultados consistentes de AUC-ROC e AUC-PR, enquanto a hard_sigmoid gerou muitos modelos aleatórios, mas também parte dos melhores modelos. Por fim, concluímos que, no contexto de pesquisa utilizado, o *class_weight* não demonstrou um impacto significativo na melhoria dos modelos.

De forma geral, os resultados de classificação, recomendação e do teste de hipóteses corroboram entre si na escolha da melhor estratégia de treinamento e na escolha dos melhores hiperparâmetros. Diante disso, elegemos o modelo UNB-VT-TFIDF-Tokens com os hiperparâmetros Adam, hard_sigmoid e pesos iguais como o que obteve os melhores resultados em *precision@n* e *recall@n* para $n \leq 7$. Para $n > 7$, o modelo B19-VT-HYBRID-STO com RMSprop, sigmoid e pesos iguais mostrou melhor desempenho.

Capítulo 5

Trabalhos Relacionados

Neste capítulo, apresentamos uma revisão abrangente dos estudos para esta pesquisa, destacando suas contribuições para o embasamento teórico e o estado da arte na área de classificação e recomendação de BRs similares. Esses trabalhos foram fundamentais para situar nossa investigação no contexto atual de pesquisa, identificar lacunas de conhecimento e oportunidades de inovação, bem como fornecer uma base sólida para o desenvolvimento de nossa metodologia e a interpretação dos resultados obtidos.

Ao longo deste capítulo, avaliamos cuidadosamente as métricas, os oráculos e os conjuntos de dados empregados nesses estudos, contrastando-os com nossas escolhas metodológicas. Essa discussão proporciona uma compreensão aprofundada das vantagens e limitações de cada abordagem, subsidiando uma interpretação mais robusta e contextualizada de nossos próprios resultados.

Portanto, este capítulo não apenas situa nosso trabalho no panorama atual de pesquisa, mas também evidencia sua relevância, originalidade e potencial impacto para o avanço do conhecimento e da prática na área de classificação e recomendação de BRs similares. Ao final desta revisão, teremos uma visão clara de como nossa investigação se distingue e complementa os estudos anteriores, pavimentando o caminho para uma discussão aprofundada de nossas contribuições nos capítulos subsequentes.

BRs duplicados são relatórios distintos que descrevem o mesmo *bug* em um *software*. Identificá-los é essencial para evitar que duas pessoas trabalhem no mesmo problema ao mesmo tempo e manter um registro organizado dos problemas a serem corrigidos. Para realizar essa detecção de forma automática, foram desenvolvidos vários estudos explorando

diferentes estratégias e abordagens para a detecção de BRs duplicados.

Em 2005, Anvik et. al. [2] destacou que os desenvolvedores de projetos *open source* frequentemente enfrentam o desafio de identificar *bug reports* (BRs) duplicados. Para abordar essa questão, ele propôs um modelo estatístico de classificação de BRs duplicados, que utiliza vetorização textual aplicada às descrições dos relatórios, medindo a similaridade de cosseno entre essas representações. Esse modelo foi treinado usando como base de dados BRs provenientes Bugzilla do Mozilla e Eclipse e alcançou uma precisão de 90% na distinção entre BRs únicos e duplicados.

Utilizando técnicas mais sofisticadas, foram desenvolvidos dois trabalhos utilizando estratégias baseadas em *transformers* para a tarefa de detecção de BRs duplicados. Em 2021, Isotani et. al. [39] publicou um estudo que conduziu um estudo para detecção de duplicidade automatizada de BRs por meio do *fine tuning* em um modelo Sentence-BERT (SBERT) [67], mediante uma base de dados de *bug reports* em japonês separada em treino e teste por intermédio de técnicas de agrupamento. Nesse processo, os autores variam cálculos de similaridade atribuindo diferentes pesos ao título e à descrição de um *bug* específico e aplicam como *baseline* o *linear discriminant analysis* (LDA) e o TF-IDF. Por fim, é mostrado que o sistema proposto alcançou um *score* de avaliação MAP de 0.829, enquanto os sistemas *baselines* com TF-IDF e LDA alcançaram *scores* de avaliação MAP de 0.751 e 0.308, respectivamente.

Em 2022, propondo um estudo comparativo, Messaoud et. al. [53] sugeriu o *fine-tuning* de diferentes modelos de geração de *embeddings*, como Sentence-BERT/SBERT [67], RoBERTa [49], *dual-channel convolutional neural network* (DC-CNN) [34] e o BERT [20], e também compara o desempenho de diferentes classificadores para a detecção de BRs duplicados, como Regressão Logística [43], *Multilayer Perceptron* (MLP) [28], *Support Vector Classifier* (SVC) [15], *Naive Bayes* (NB) [1], *Decision Tree* (DT) [59], *Random Forest* (RF) [35], *K-nearest Neighbors* (KNN) [16] e *XGBoost* [62]. Para realizar esse experimento, usou-se um *dataset* das plataformas Mozilla, Eclipse e Thunderbird com tamanho de aproximadamente 47.000 pares de BRs, separando treino e teste pelo método *holdout*. Por meio de confusão e métricas, como *accuracy*, *precision*, *recall* e *F1-score*, o artigo definiu que a melhor configuração para esse problema era utilizar o vetorizador BERT tradicional acoplado ao classificador MLP.

Diferente do problema de duplicidade, na teoria, a definição de BR similar é quando dois

BRs descrevem problemas encontrados com características ou sintomas semelhantes. Nessas descrições podem ser encontradas semelhanças como: comportamentos de *software*, código-fonte afetado ou circunstâncias em que o *bug* aconteceu. A identificação de BRs similares é útil para os desenvolvedores, pois pode ajudá-los a entender melhor a causa raiz do problema e a encontrar soluções mais rapidamente. Até agora, na literatura, foram exploradas diferentes técnicas e alternativas para essa identificação e recomendação de similares.

Em 2015, Rocha et. al. [68] desenvolveu uma pesquisa que criou uma extensão de navegador, chamada NextBug. Essa extensão tem como objetivo realizar recomendações de BRs similares através de uma expressão matemática considerando as representações vetoriais TF-IDF. Visando avaliar o NextBug, foi utilizado um conjunto de dados de 130.495 BRs do Mozilla, alcançando resultados de *feedback* de 65% e *precision* em torno de 31%. Também foi realizada uma pesquisa com desenvolvedores do Mozilla, os quais 77% classificaram as recomendações como relevantes e 85% confirmaram que o NextBug seria útil para a comunidade da Mozilla.

Publicado em 2016, um artigo de Yang et. al. [78] propõe um sistema de recomendação de BRs similares que considera várias características, incluindo representações vetoriais, distâncias entre *word embeddings* gerados pelo *skip-gram* [55] e comparações de campos específicos nos BRs. Para o conjunto de dados de BRs do Eclipse, o modelo atingiu aproximadamente 49%, 27% e 45%, demonstrando uma melhora nas métricas de *recall@10*, MAP e MRR respectivamente, demonstrando uma melhora de 42%, 57% e 57% com relação à abordagem utilizada na extensão do NextBug.

Disponibilizado em 2018, Chen et. al. [13] conduziu uma pesquisa para detecção de similares em quatro fatores: distância dos cossenos entre vetores TF-IDF, *word embeddings*, *document embeddings* [46] e a similaridade dos campos de produto e componente do BR. Nesse artigo, usou-se como base de dados dos BRs os projetos do JDT, Birt e Eclipse Platform e os avaliou por meio de *recall@n*, MAP e MRR e alcançou os resultados descritos nas Tabela 5.1, 5.2 e 5.3.

Desenvolvido em 2023, Carneiro et. al. [12] desenvolveu um sistema de recomendação de BRs que se baseia na similaridade textual, destacando-se pelo uso do avançado modelo de compreensão textual BERT como um dos fatores na avaliação da similaridade. Para a avaliação da estratégia, usou-se como base de dados dos BRs os projetos do Mozilla e utilizou

Tabela 5.1: Métricas de avaliação nos dados do JDT.

Métrica	Yang et ai's	Chen et ai's	Melhoria
Recall-Rate@1	8,83%	9,61%	8,12%
Recall-Rate@5	20,07%	21,80%	8,62%
Recall-Rate@10	26,11%	28,59%	9,50%
MAP	6,22%	6,75%	8,52%
MRR	11,74%	12,80%	9,03%

Tabela 5.2: Métricas de avaliação nos dados do BIRT.

Métrica	Yang et ai's	Chen et ai's	Melhoria
Recall-Rate@1	8,97%	9,54%	6,35%
Recall-Rate@5	23,08%	23,79%	3,08%
Recall-Rate@10	30,72%	31,54%	2,67%
MAP	6,72%	7,02%	4,46%
MRR	12,65%	13,14%	3,73%

Tabela 5.3: Métricas de avaliação nos dados do Eclipse.

Métrica	Yang et ai's	Chen et ai's	Melhoria
Recall-Rate@1	8,83%	9,58%	8,49%
Recall-Rate@5	20,07%	21,80%	8,77%
Recall-Rate@10	26,23%	28,30%	7,89%
MAP	6,23%	6,75%	8,35%
MRR	11,72%	12,77%	8,96%

precisão@n, *feedback* e *likelihood* como métricas, os valores são descritos na Tabela 5.4.

Tabela 5.4: Métricas obtidas pela técnica proposta por Carneiro.

K	Feedback	Likelihood	Precisão
1	25,07%	25,07%	99,81%
3	22,09%	41,73%	99,51%
5	20,49%	49,97%	99,22%
10	18,11%	60,34%	98,29%
20	15,35%	69,79%	96,32%

Os estudos que identificam e sugerem *bug reports* (BRs) duplicados ou similares têm um papel fundamental neste contexto de pesquisa. Embora a combinação de vetores TF-IDF com *embeddings* já tenha sido utilizada por Yang et al. [78] em sua métrica de similaridade, nosso objetivo é explorar abordagens alternativas que empreguem técnicas mais contemporâneas.

Entre as novas estratégias analisadas, utilizamos redes neurais e a arquitetura T5 para vetorização textual. Diferentemente de estudos anteriores, nossa metodologia incorpora o modelo T5 para fornecer os *embeddings*, reconhecido em alguns *benchmarks* como o estado da arte para *Semantic Textual Similarity* (STS) [57]. Adicionalmente, investigamos o impacto de várias abordagens, incluindo: o uso de vetores completos em comparação à aplicação apenas da distância de cosseno, diferentes estratégias de pré-processamento, balanceamento de dados, combinações de TF-IDF com T5, e o ajuste fino da rede neural. Por fim, avaliamos os resultados com um conjunto ampliado de métricas e analisamos como as *features* utilizadas influenciam o desempenho na classificação e recomendação de BRs similares.

Neste estudo, utilizamos como *features*: *summary*, *description*, *component*, *version*, *assigned_to*, *creator*, *qa_contact*, *platform*, *severity*, *priority* e *files_changed*. Em comparação, a maioria dos outros estudos restringe-se ao uso das features *summary*, *description*, *product* e *component*.

Quanto às métricas de avaliação, utilizamos um conjunto abrangente, incluindo AUC-ROC, AUC-PR, precisão, *recall*, *F1-score*, além de métricas de recomendação como *Precision@N* e *Recall@N*. Em comparação, Rocha et al. [68] focou em *feedback* de usuários e precisão, enquanto Yang et al. [78] e Chen et al. [13] usaram *Recall@N*, MAP e MRR.

Carneiro et al. [12] avaliou com *Precision@N*, *feedback* e *likelihood*. Nossos modelos se destacaram especialmente na avaliação da capacidade do modelo em classificar, como por exemplo através da métrica AUC-PR, que é crucial para conjuntos de dados desbalanceados, atingindo até 37,11%.

Em relação ao oráculo utilizado para definir a similaridade entre *bug reports*, inovamos ao considerar a proporção de arquivos modificados em comum durante a correção dos *bugs*, utilizando dados obtidos diretamente da plataforma Gerrit. Essa abordagem oferece uma base empírica e objetiva, diferenciando-se de trabalhos anteriores que utilizaram critérios menos transparentes. Por exemplo, Yang et al. [78] mencionam o uso de uma métrica baseada em 50% de sobreposição dos arquivos alterados como indicador de similaridade, mas não detalham como essas informações foram coletadas ou calculadas. Nossa metodologia aborda essa lacuna ao empregar fontes verificáveis e ao proporcionar maior clareza no processo de análise.

Vale ressaltar os resultados do estudo de Rocha et al. [68], que desenvolveu a extensão *NextBug* e realizou uma avaliação com engenheiros de *software* da Mozilla. Nessa avaliação, 77% dos desenvolvedores classificaram as recomendações da extensão como relevantes, e 85% confirmaram que o *NextBug* seria útil para a comunidade da Mozilla. Esses resultados evidenciam a aplicabilidade prática de sistemas de recomendação de BRs similares no desenvolvimento de *software*.

Em resumo, nosso trabalho se destaca pelo uso de técnicas avançadas como o T5, a exploração de abordagens híbridas de vetorização, a utilização de métricas abrangentes e a proposição de um oráculo baseado em informações de modificações de código obtidas do Gerrit. A análise detalhada de estratégias de vetorização e balanceamento contribui para o avanço do estado da arte na detecção de *bug reports* similares, complementando e expandindo os achados de estudos anteriores. Ademais, a avaliação positiva do *NextBug* por engenheiros de *software* ressalta o potencial impacto prático dessas técnicas no desenvolvimento de *software* real.

Capítulo 6

Conclusões

Ao encerrarmos este estudo sobre a identificação e recomendação de *bug reports* (BRs) similares, também refletimos sobre as principais descobertas e contribuições realizadas. Durante a pesquisa, avaliamos a eficácia de diversas abordagens para a identificação e recomendação de relatórios de BRs e investigamos os fatores que influenciam esse processo. Este capítulo sintetiza as principais descobertas, contribuições e limitações deste trabalho, destacando suas implicações para a prática de desenvolvimento de software. Adicionalmente, são apresentadas recomendações detalhadas e direcionadas para profissionais e pesquisadores da área de *machine learning*, com o propósito de apoiar o aprimoramento contínuo e promover a excelência no campo.

6.1 Contribuições

Este trabalho realizou experimentações abrangentes com diversas configurações de modelo, proporcionando uma análise detalhada de seu desempenho. Com base nos resultados obtidos, foram levantadas as seguintes observações:

- **Abordagens de vetorização**: As abordagens híbridas, que combinam TF-IDF e *embeddings* (T5), e as abordagens isoladas com TF-IDF destacaram-se como estratégias eficazes para representar os *bug reports*.
- **Utilização de vetores como *features***: A combinação de diferentes tipos de vetores permitiu capturar aspectos semânticos e sintáticos dos *bug reports*, proporcionando

maior riqueza nas representações textuais.

- **Estratégias de balanceamento de dados**: O treinamento com dados desbalanceados e o balanceamento utilizando uma proporção de 9 não similares para cada similar (B19) apresentaram desempenhos superiores, dependendo do cenário e das métricas consideradas.

- **Escolha do otimizador**: Os otimizadores *Adam* e *RMSprop* destacaram-se pela maior estabilidade e eficiência no treinamento, superando outras alternativas testadas.

Essas descobertas reforçam a importância de selecionar cuidadosamente as *features*, estratégias de balanceamento de dados e otimizadores, evidenciando sua influência significativa no desempenho das redes neurais treinadas.

O estudo contribuiu significativamente ao identificar os modelos mais eficientes em diferentes cenários de *N*. Para $N \leqslant 7$, o modelo treinado com dados desbalanceados, utilizando vetores TF-IDF tokenizados (UNB-VT-TFIDF-Tokens) e otimizado com *Adam*, apresentou o melhor desempenho. Por outro lado, para $N > 7$, o modelo que utilizou dados balanceados (proporção de 9 não similares para cada similar), vetores híbridos TF-IDF e T5 tokenizados, com remoção de *stopwords* (B19-VT-HYBRID-STO), e otimizado com *RMSprop*, destacou-se como o mais eficiente. Esses resultados oferecem *insights* valiosos para a escolha de estratégias de treinamento e ajuste de hiperparâmetros.

Além disso, os resultados indicam que a identificação e recomendação de *bug reports* similares têm grande potencial para melhorar a eficiência do desenvolvimento de software. Um modelo eficiente pode auxiliar os desenvolvedores a localizar soluções baseadas em *bugs* semelhantes, economizando tempo e esforços. Para os gerentes, essa abordagem oferece suporte na alocação de recursos humanos, atribuindo tarefas a programadores com experiência prévia em problemas similares.

Adicionalmente, a investigação do impacto das estratégias híbridas de vetorização revelou a importância de combinar TF-IDF, que captura informações estatísticas, com os *embeddings* T5, capazes de compreender nuances semânticas mais complexas. Embora o TF-IDF isolado tenha apresentado resultados competitivos com ajuste fino, a combinação híbrida foi essencial para capturar diferentes dimensões dos *bug reports*.

Durante os experimentos, explorou-se o impacto do uso de vetores completos em comparação com a distância de cosseno para calcular similaridades. Os resultados demonstraram que vetores completos fornecem maior riqueza informacional, contribuindo para melhores desempenhos nas métricas avaliadas. No entanto, percebeu-se que os dados desbalanceados frequentemente superaram os balanceados, indicando que o equilíbrio artificial pode prejudicar a diversidade dos exemplos representados.

As técnicas de pré-processamento, como remoção de *stopwords*, lematização e tokenização, não produziram ganhos significativos no desempenho geral, mas contribuíram para a estabilidade dos modelos. No caso dos otimizadores, *Adam* e *RMSprop* foram consistentemente mais rápidos e eficazes, enquanto *SGD* mostrou-se menos eficiente, exigindo mais épocas para atingir um desempenho satisfatório.

Em relação às funções de ativação, foi observado que alternativas como *sigmoid* e *hard_sigmoid* impactaram significativamente o desempenho, com algumas combinações resultando em modelos com desempenhos próximos a escolhas aleatórias. Por fim, a abordagem de classificação ponderada, amplamente utilizada em problemas de classes desbalanceadas, não demonstrou impactos significativos neste contexto, sugerindo que ajustes adicionais podem ser necessários para explorar seu potencial.

Essas análises detalham os avanços obtidos neste estudo, oferecendo direções claras para futuras pesquisas e implementações práticas na detecção de *bug reports* similares.

6.2 Limitações

Este estudo apresenta várias limitações que devem ser consideradas ao interpretar os resultados. Primeiramente, o foco exclusivo na base de dados do *Bugzilla* do *Eclipse* pode limitar a generalização dos resultados para outras plataformas ou ambientes de desenvolvimento. Para superar essa limitação, seria necessário expandir a análise para outras plataformas baseadas em *Bugzilla* e investigar suas conexões com o *Gerrit*, ou buscar acesso aos dados do *Jira* de grandes projetos.

A análise restrita aos *Bug Reports* (BRs) em inglês limita a aplicabilidade dos resultados a BRs em outros idiomas, sugerindo a necessidade de desenvolver um modelo multilíngue capaz de identificar similaridades independentemente da língua. Além disso, a exclusão de

dados sem conexão com o *Gerrit* resultou na perda de um volume significativo de informações potencialmente úteis para o treinamento do modelo.

Os BRs podem conter erros e imprecisões, afetando a qualidade dos dados usados para treinar o modelo. Idealmente, seria benéfico identificar empresas com processos mais rigorosos na criação de BRs descritivos. Embora os *scripts* de coleta de dados tenham sido extensivamente testados, ainda existe o risco de danos aos dados durante este processo, sugerindo a necessidade de acesso direto aos dados brutos da plataforma.

O estudo pode ter desconsiderado características potencialmente importantes para a identificação de BRs similares, indicando a necessidade de experimentar o modelo com uma variedade maior de *features* e selecionar as mais relevantes. A utilização exclusiva de BRs *FIXED* limita o volume de dados disponíveis para análise e treinamento, uma limitação que poderia ser superada utilizando *snapshots* de dados antigos de BRs abertos e fechados.

A escolha de um número máximo de 200 épocas durante a experimentação pode ter desfavorecido o otimizador *SGD*, que potencialmente necessitaria de mais épocas para atingir um resultado ótimo. Similarmente, o número de paciência de 5 épocas no *early stopping* pode ter sido insuficiente, possivelmente levando à parada prematura do treinamento antes de atingir um máximo absoluto na métrica alvo.

Por fim, o ajuste fino foi realizado apenas nos 14 modelos que apresentaram os melhores resultados na primeira etapa. Embora esses modelos parecessem os mais promissores, não há garantia absoluta de que seriam os melhores após um processo de ajuste fino mais abrangente.

Reconhecer essas limitações não apenas contextualiza os resultados obtidos, mas também oferece direções valiosas para pesquisas futuras, visando superar estas restrições e expandir ainda mais o conhecimento no campo da detecção automática de BRs similares.

6.3 Trabalhos Futuros

Neste estudo, como em qualquer pesquisa, existem oportunidades para refinamentos e novas investigações. Isso abre várias direções promissoras para pesquisas futuras, permitindo que outros pesquisadores construam e ampliem as descobertas apresentadas, promovendo avanços significativos na eficiência e eficácia dos resultados. A seguir, destacamos algumas áreas

potenciais para o refinamento e exploração deste estudo.

Uma das direções é expandir a base de dados para incluir plataformas como *Jira* e projetos de outras empresas além do *Eclipse*. Isso pode aumentar a generalização dos resultados, ajudando a verificar se as conclusões são aplicáveis a uma variedade mais ampla de ambientes de desenvolvimento e práticas de gerenciamento de *bugs*. Além disso, desenvolver ou treinar modelos em várias línguas aumentaria a aplicabilidade dos resultados para projetos internacionais, permitindo que um modelo multilíngue identifique relatórios de *bugs* similares independentemente da língua, melhorando a versatilidade da abordagem.

O acesso aos dados do *Jira*, que também gerencia *pull requests*, poderia resolver a limitação de dependência do oráculo, proporcionando uma fonte de dados mais rica e diversificada para análise. Investigar e incluir novas características nos relatórios de *bugs* poderia melhorar a precisão do modelo, revelando fatores adicionais que influenciam a resolução de *bugs*. Implementar técnicas de redução de dimensionalidade, como PCA (Análise de Componentes Principais), poderia simplificar o modelo e melhorar seu desempenho, reduzindo o ruído nos dados e focando nas características mais relevantes. Experimentar diferentes métodos de vetorização, como BERT, poderia melhorar a representação dos relatórios de *bugs* e, consequentemente, a eficácia do modelo.

Além disso, investigar técnicas de balanceamento híbrido ou outras distribuições pode otimizar o *trade-off* entre precisão e *recall*, levando a um modelo mais robusto. Métodos de *oversampling* ou híbrido (*oversampling* com *undersampling*) podem ser considerados. Aumentar ou não limitar o número de épocas poderia ajudar a garantir que os modelos atingissem o potencial máximo sem ser limitado, evitando o que aconteceu com o otimizador *SGD*. É importante salientar que não limitar o número máximo de épocas deveria ter uma condição de parada, como por exemplo com *early stopping*.

Aumentar o número de épocas de treinamento ou a paciência do *early stopping* pode permitir que o modelo alcance um desempenho melhor, evitando máximos locais. Sem o *early stopping*, será necessário salvar o melhor modelo encontrado e estabelecer uma nova condição de parada, como o número máximo de épocas. Analisar as curvas de perda de validação e teste por época é importante para identificar o *overfitting* e *underfitting*, ajustando o modelo para melhor desempenho. Essas curvas ajudam a determinar o ponto ideal para parar o treinamento, evitando que o modelo se ajuste demais aos dados de treinamento ou

não se ajuste o suficiente, além de oferecer informações importantes para hiperparâmetros, como a taxa de aprendizado e o número de épocas.

Por fim, explorar o ajuste fino de todos os modelos treinados poderia gerar um modelo ótimo, mesmo que não tenha aparentado ser promissor na primeira etapa. Embora o estudo tenha testado 12 configurações diferentes, explorar uma gama maior de hiperparâmetros poderia levar a melhorias adicionais, incluindo testar mais combinações de otimizadores, funções de ativação e métodos de balanceamento, apesar do aumento na complexidade e no tempo de ajuste fino. Avaliar o tempo de execução e a eficiência dos modelos pode fornecer *insights* valiosos sobre a viabilidade prática das soluções propostas, especialmente em ambientes de produção onde o tempo de resposta é crítico.

Bibliografia

[1] Charu C. Aggarwal and ChengXiang Zhai. *A Survey of Text Classification Algorithms*, pages 163–222. Springer US, Boston, MA, 2012.

[2] John Anvik, Lyndon Hiew, and Gail C. Murphy. Coping with an open bug repository. In *Proceedings of the 2005 OOPSLA Workshop on Eclipse Technology EXchange*, eclipse '05, page 35–39, New York, NY, USA, 2005. Association for Computing Machinery.

[3] Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. A comprehensive study of class incremental learning algorithms for visual tasks. *Neural Networks*, 135:38–54, 2021.

[4] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures, 2012.

[5] Nicolas Bettenburg, Sascha Just, Adrian Schröter, Christian Weiss, Rahul Premraj, and Thomas Zimmermann. What makes a good bug report? In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 308–318. ACM, 2008.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. The Logistic Sigmoid Function.

[7] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[8] Brains. Medidas de performance para modelos de classificação, 2023.

[9] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini

Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[10] Jason Brownlee. Roc curves and precision-recall curves for imbalanced classification. https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-imbalanced-classificatio Acesso em: 20 Maio de 2024.

[11] Bugzilla. Bugzilla: Life cycle of a bug. https://www.bugzilla.org/docs/2.18/html/lifecycle.html. Acesso em: 12 Maio de 2024.

[12] Guilherme Carneiro, José Ferreira, Franklin Ramalho, and Tiago Massoni. Similar bug reports recommendation system using bert. In *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*, SBES '23, page 378–387, New York, NY, USA, 2023. Association for Computing Machinery.

[13] Ming Chen, Dongyang Hu, Tao Wang, Jun Long, Gang Yin, Yue Yu, and Yang Zhang. Using document embedding techniques for similar bug reports recommendation. In *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, pages 811–814, 2018.

[14] Gobinda G. Chowdhury. Natural language processing. *Annual Review of Information Science and Technology*, 37(1):51–89, 2003.

[15] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[16] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[17] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.

[18] Yvonne Day. *Analysing data using SPSS*. SAGE Publications, 2007.

[19] Marie Delacre, Christophe Leys, Youri L Mora, and Daniël Lakens. Taking parametric assumptions seriously: Arguments for the use of welch's f-test instead of the classical f-test in one-way anova. *International Review of Social Psychology*, 32(1), 2019.

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[21] DINO. Mercado de softwares: estudo prevê us$ 698 bi em receitas, 2 2024.

[22] Olive Jean Dunn. Multiple comparisons using rank sums. *Technometrics*, 6(3):241–252, 1964.

[23] EvidentlyAI. Evaluating recommender systems. https://www.evidentlyai.com/ranking-metrics/evaluating-recommender-systems. Acesso em: 20 Maio de 2024.

[24] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[25] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C. Prati, Bartosz Krawczyk, and Francisco Herrera. *Learning from Imbalanced Data Sets*. Springer International Publishing, 1st edition, 2018.

[26] Ronald Aylmer Fisher. *Statistical methods for research workers*. Oliver and Boyd, 1925.

[27] Paul A Games and John F Howell. Pairwise multiple comparison procedures with unequal n's and/or variances: a monte carlo study. *Journal of Educational Statistics*, 1(2):113–125, 1976.

[28] M.W Gardner and S.R Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14):2627–2636, 1998.

[29] Joseph L Gastwirth, Yulia R Gel, and Weiwen Miao. The impact of levene's test of equality of variances on statistical theory and practice. *Statistical Science*, 24(3):343–360, 2009.

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[31] Google Developers. Translating to a lower-dimensional space. https://developers.google.com/machine-learning/crash-course/embeddings/translating-to-a-lower-dimensional-space. Acesso em: 19 Maio de 2024.

[32] Google Research. Exploring transfer learning with t5: The text-to-text transfer transformer. https://research.google/blog/exploring-transfer-learning-with-t5-the-text-to-text-transfer-trans 2020. Acesso em: 20 Maio de 2024.

[33] Simon Haykin. *Redes neurais: princípios e prática*. Bookman, Porto Alegre, 2 edition, 2001.

[34] Jianjun He, Ling Xu, Meng Yan, Xin Xia, and Yan Lei. Duplicate bug report detection using dual-channel convolutional neural networks. In *Proceedings of the 28th International Conference on Program Comprehension*, ICPC '20, page 117–127, New York, NY, USA, 2020. Association for Computing Machinery.

[35] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[36] Yosef Hochberg and Ajit C Tamhane. Multiple comparison procedures. *Wiley Series in Probability and Mathematical Statistics*, 1987.

[37] David C Howell. *Statistical methods for psychology*. Cengage Learning, 2012.

[38] Jason Hsu. *Multiple comparisons: theory and methods*. Chapman and Hall/CRC, 1996.

[39] Haruna Isotani, Hironori Washizaki, Yoshiaki Fukazawa, Tsutomu Nomoto, Saori Ouji, and Shinobu Saito. Duplicate bug report detection by using sentence embedding and

fine-tuning. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 535–544, 2021.

[40] John D. Kelleher, Brian MacNamee, and Aoife D'Arcy. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. MIT Press, Cambridge, MA, 2015.

[41] Memoona Khanam, Tahira Mahboob, Warda Imtiaz, Humaraia Ghafoor, and Rabeea Sehar. A survey on unsupervised machine learning algorithms for automation, classification and maintenance. *International Journal of Computer Applications*, 119:34–39, June 2015.

[42] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[43] David G. Kleinbaum and Mitchel Klein. Introduction to logistic regression. In *Logistic regression*, pages 1–39. Springer, 2010.

[44] William H Kruskal and W Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[45] Daria J. Kuss and Mark D. Griffiths. Internet addiction in psychotherapy. *PLOS ONE*, 10(2):e0118432, 2015.

[46] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1188–II–1196. JMLR.org, 2014.

[47] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, Cambridge, 2014.

[48] Howard Levene. Robust tests for equality of variances. *Contributions to probability and statistics*, 1:278–292, 1960.

[49] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[50] M. R. Lyu. *Handbook of Software Reliability Engineering*. McGraw-Hill, Inc., Hightstown, NJ, USA, 1996.

[51] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[52] Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

[53] Montassar Ben Messaoud, Asma Miladi, Ilyes Jenhani, Mohamed Wiem Mkaouer, and Lobna Ghadhab. Duplicate bug report detection using an attention-based neural language model. *IEEE Transactions on Reliability*, 72(2):846–858, 2023.

[54] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[55] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

[56] Lloyd Montgomery, Clara Lüders, and Walid Maalej. An alternative issue tracking dataset of public jira repositories. In *Proceedings of the 19th International Conference on Mining Software Repositories*, MSR '22, page 73–77, New York, NY, USA, 2022. Association for Computing Machinery.

[57] Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. MTEB: Massive text embedding benchmark. In Andreas Vlachos and Isabelle Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics.

[58] Glenford J. Myers and Corey Sandler. *The Art of Software Testing*. John Wiley & Sons, Hoboken, New Jersey, 2 edition, 2004.

[59] Anthony Myles, Robert Feudale, Yang Liu, Nathaniel Woody, and Steven Brown. An introduction to decision tree modeling. *Journal of Chemometrics*, 18:275 – 285, 06 2004.

[60] Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models, 2021.

[61] ogre51. Nlp explain: Bag-of-words. https://ogre51.medium.com/nlp-explain-bag-of-words-3b9fc4f211e8. Acesso em: 15 Maio de 2024.

[62] Adeola Ogunleye and Qing-Guo Wang. Xgboost model for chronic kidney disease diagnosis. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 17(6):2131–2140, dec 2020.

[63] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[64] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.

[65] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[66] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[67] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[68] Henrique Rocha, Guilherme de Oliveira, Humberto Marques-Neto, and Marco Tulio Valente. Nextbug: a bugzilla extension for recommending similar bugs. *Journal of Software Engineering Research and Development*, 3(1):3, April 2015.

[69] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition, 2021.

[70] Barry B Schultz. Levene's test for relative variation. *Systematic Biology*, 34(4):449–456, 1985.

[71] Ian Sommerville. *Software Engineering*. Pearson, Harlow, England, 10 edition, 2016.

[72] TensorFlow. Tensorflow: Hard sigmoid activation function. https://www.tensorflow.org/versions/r1.15/api_docs/python/tf/keras/activations/hard_sigmoid, 2019. Accessed: 2024-10-08.

[73] TensorFlow. Tutorial tensorflow: Lidando com dados desbalanceados. https://www.tensorflow.org/tutorials/structured_data/imbalanced_data?hl=pt-br, 2023. Accessed: 2024-10-08.

[74] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[75] John W Tukey. Comparing individual means in the analysis of variance. *Biometrics*, 5(2):99–114, 1949.

[76] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[77] Bernard L Welch. On the comparison of several mean values: an alternative approach. *Biometrika*, 38(3/4):330–336, 1951.

[78] Xinli Yang, David Lo, Xin Xia, Lingfeng Bao, and Jianling Sun. Combining word embedding with information retrieval to recommend similar bug reports. pages 127–137, 10 2016.

[79] Ilker Yurek. Roc curve and auc: Evaluating model performance. https://medium.com/@ilyurek/roc-curve-and-auc-evaluating-model-performance-c2178008b02. Acesso em: 20 Maio de 2024.

[80] Y. Zhang, R. Jin, and ZH. Zhou. Understanding bag-of-words model: a statistical framework. *Int. J. Mach. Learn. & Cyber.*, 1:43–52, 2010.

[81] Thomas Zimmermann, Rahul Premraj, and Nicolas Bettenburg. What makes a good bug report? In *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*, pages 308–318. ACM, 2007.

Apêndice A

Resultados da Etapa 1

Neste Apêndice, são exibidos os valores dos resultados obtidos nos experimentos da Etapa 1.

Model	AUC-ROC	AUC-PR
UNB-CD-TFIDF-TOK	77.71	5.14
UNB-CD-TFIDF-STO	78.14	5.23
UNB-CD-TFIDF-LEM	78.54	5.37
UNB-CD-T5	79.33	5.28
UNB-CD-HYBRID-TOK	79.78	5.62
UNB-CD-HYBRID-STO	79.98	5.87
UNB-CD-HYBRID-LEM	80.13	5.81
UNB-VT-TFIDF-TOK	86.21	33.38
UNB-VT-TFIDF-STO	83.65	36.08
UNB-VT-TFIDF-LEM	85.95	35.53
UNB-VT-T5	88.49	21.71
UNB-VT-HYBRID-TOK	84.68	36.70
UNB-VT-HYBRID-STO	83.16	36.74
UNB-VT-HYBRID-LEM	81.01	35.76
B19-CD-TFIDF-TOK	79.67	5.07
B19-CD-TFIDF-STO	79.97	5.15
B19-CD-TFIDF-LEM	80.19	5.23
B19-CD-T5	80.61	4.88
B19-CD-HYBRID-TOK	81.63	5.62
B19-CD-HYBRID-STO	81.77	5.76
B19-CD-HYBRID-LEM	81.93	5.85
B19-VT-TFIDF-TOK	91.60	19.41
B19-VT-TFIDF-STO	91.25	18.18
B19-VT-TFIDF-LEM	91.62	18.30
B19-VT-T5	91.06	13.38
B19-VT-HYBRID-TOK	90.81	17.01
B19-VT-HYBRID-STO	91.11	17.39
B19-VT-HYBRID-LEM	91.08	17.00

Tabela A.1: AUC-ROC e AUC-PR para modelos UNB e B19

Model	AUC-ROC	AUC-PR
B12-CD-TFIDF-TOK	79.15	4.71
B12-CD-TFIDF-STO	79.60	4.83
B12-CD-TFIDF-LEM	79.99	4.99
B12-CD-T5	80.90	4.95
B12-CD-HYBRID-TOK	80.82	5.14
B12-CD-HYBRID-STO	81.10	5.24
B12-CD-HYBRID-LEM	81.24	5.31
B12-VT-TFIDF-TOK	89.16	11.53
B12-VT-TFIDF-STO	88.95	10.98
B12-VT-TFIDF-LEM	88.86	11.04
B12-VT-T5	89.36	9.20
B12-VT-HYBRID-TOK	89.54	11.52
B12-VT-HYBRID-STO	89.68	11.60
B12-VT-HYBRID-LEM	89.56	11.36
B11-CD-TFIDF-TOK	78.97	4.40
B11-CD-TFIDF-STO	79.19	4.62
B11-CD-TFIDF-LEM	79.63	4.80
B11-CD-T5	80.49	4.63
B11-CD-HYBRID-TOK	81.03	5.24
B11-CD-HYBRID-STO	81.33	5.24
B11-CD-HYBRID-LEM	80.82	5.04
B11-VT-TFIDF-TOK	87.84	9.89
B11-VT-TFIDF-STO	87.79	9.69
B11-VT-TFIDF-LEM	87.70	9.57
B11-VT-T5	87.07	6.69
B11-VT-HYBRID-TOK	89.38	8.95
B11-VT-HYBRID-STO	88.72	9.05
B11-VT-HYBRID-LEM	88.64	8.88

Tabela A.2: AUC-ROC e AUC-PR para modelos B12 e B11

Model	FNR	TNR	FPR	TPR	Precision	F1-Score
UNB-CD-TFIDF-TOK	99.15	100.00	0.00	0.85	64.89	1.68
UNB-CD-TFIDF-STO	99.19	100.00	0.00	0.81	74.36	1.60
UNB-CD-TFIDF-LEM	99.22	100.00	0.00	0.78	76.71	1.54
UNB-CD-T5	99.15	100.00	0.00	0.85	70.11	1.68
UNB-CD-HYBRID-TOK	99.18	100.00	0.00	0.82	70.24	1.62
UNB-CD-HYBRID-STO	99.18	100.00	0.00	0.82	81.94	1.63
UNB-CD-HYBRID-LEM	99.11	100.00	0.00	0.89	71.91	1.76
UNB-VT-TFIDF-TOK	73.47	99.88	0.12	26.53	59.25	36.65
UNB-VT-TFIDF-STO	70.27	99.87	0.13	29.73	59.40	39.63
UNB-VT-TFIDF-LEM	72.46	99.89	0.11	27.54	63.33	38.39
UNB-VT-T5	84.72	99.89	0.11	15.28	47.78	23.16
UNB-VT-HYBRID-TOK	68.62	99.84	0.16	31.38	56.59	40.38
UNB-VT-HYBRID-STO	70.03	99.88	0.12	29.97	61.66	40.33
UNB-VT-HYBRID-LEM	68.50	99.82	0.18	31.50	53.87	39.75
B19-CD-TFIDF-TOK	87.53	99.13	0.87	12.47	8.64	10.21
B19-CD-TFIDF-STO	87.35	99.10	0.90	12.65	8.50	10.17
B19-CD-TFIDF-LEM	87.18	99.05	0.95	12.82	8.17	9.98
B19-CD-T5	86.93	98.98	1.02	13.07	7.77	9.74
B19-CD-HYBRID-TOK	84.50	98.85	1.15	15.50	8.17	10.70
B19-CD-HYBRID-STO	84.11	98.83	1.17	15.89	8.23	10.84
B19-CD-HYBRID-LEM	83.48	98.77	1.23	16.52	8.13	10.89
B19-VT-TFIDF-TOK	36.59	95.80	4.20	63.41	9.05	15.84
B19-VT-TFIDF-STO	36.49	95.72	4.28	63.51	8.92	15.64
B19-VT-TFIDF-LEM	36.10	95.78	4.22	63.90	9.08	15.89
B19-VT-T5	33.25	92.80	7.20	66.75	5.76	10.60
B19-VT-HYBRID-TOK	38.11	96.10	3.90	61.89	9.48	16.44
B19-VT-HYBRID-STO	38.69	96.11	3.89	61.31	9.42	16.32
B19-VT-HYBRID-LEM	38.93	96.07	3.93	61.07	9.31	16.15

Tabela A.3: Demais métrica de classificação para modelos UNB e B19

Model	FNR	TNR	FPR	TPR	Precision	F1-Score
B12-CD-TFIDF-TOK	54.04	88.81	11.19	45.96	2.64	4.99
B12-CD-TFIDF-STO	52.67	89.09	10.91	47.33	2.78	5.25
B12-CD-TFIDF-LEM	52.50	89.36	10.64	47.50	2.86	5.40
B12-CD-T5	46.08	86.66	13.34	53.92	2.60	4.95
B12-CD-HYBRID-TOK	45.61	86.20	13.80	54.39	2.53	4.84
B12-CD-HYBRID-STO	44.79	86.54	13.46	55.21	2.63	5.03
B12-CD-HYBRID-LEM	45.12	86.69	13.31	54.88	2.65	5.05
B12-VT-TFIDF-TOK	21.60	84.12	15.88	78.40	3.15	6.06
B12-VT-TFIDF-STO	24.50	86.12	13.88	75.50	3.46	6.62
B12-VT-TFIDF-LEM	24.31	85.85	14.15	75.69	3.41	6.52
B12-VT-T5	15.82	78.23	21.77	84.18	2.49	4.83
B12-VT-HYBRID-TOK	23.40	85.45	14.55	76.60	3.36	6.43
B12-VT-HYBRID-STO	22.51	85.27	14.73	77.49	3.35	6.43
B12-VT-HYBRID-LEM	22.51	85.18	14.82	77.49	3.33	6.39
B11-CD-TFIDF-TOK	29.53	72.06	27.94	70.47	1.64	3.20
B11-CD-TFIDF-STO	30.63	73.87	26.13	69.37	1.72	3.36
B11-CD-TFIDF-LEM	30.35	73.87	26.13	69.65	1.73	3.37
B11-CD-T5	24.20	69.46	30.54	75.80	1.61	3.15
B11-CD-HYBRID-TOK	24.84	70.64	29.36	75.16	1.66	3.25
B11-CD-HYBRID-STO	24.97	71.40	28.60	75.03	1.70	3.33
B11-CD-HYBRID-LEM	26.90	72.55	27.45	73.10	1.73	3.37
B11-VT-TFIDF-TOK	18.75	78.99	21.01	81.25	2.49	4.83
B11-VT-TFIDF-STO	18.40	78.45	21.55	81.60	2.44	4.73
B11-VT-TFIDF-LEM	18.46	78.32	21.68	81.54	2.42	4.70
B11-VT-T5	13.32	70.00	30.00	86.68	1.87	3.66
B11-VT-HYBRID-TOK	15.09	77.61	22.39	84.91	2.44	4.74
B11-VT-HYBRID-STO	17.04	78.09	21.91	82.96	2.44	4.74
B11-VT-HYBRID-LEM	17.06	77.80	22.20	82.94	2.41	4.67

Tabela A.4: Demais métrica de classificação para modelos B12 e B11

Model	N	Precision_n	Recall_n
UNB-VT-TFIDF-TOK	1	42.54	21.47
UNB-VT-TFIDF-TOK	2	35.66	24.87
UNB-VT-TFIDF-TOK	3	33.14	26.51
UNB-VT-TFIDF-TOK	4	31.21	26.6
UNB-VT-TFIDF-TOK	5	30.36	27.14
UNB-VT-TFIDF-TOK	6	29.6	27.46
UNB-VT-TFIDF-TOK	7	28.85	26.87
UNB-VT-TFIDF-TOK	8	27.08	25.25
UNB-VT-TFIDF-TOK	9	30.16	29.21
UNB-VT-TFIDF-TOK	10	20.00	20.00
UNB-VT-TFIDF-STO	1	46.69	23.95
UNB-VT-TFIDF-STO	2	39.8	27.95
UNB-VT-TFIDF-STO	3	36.41	29.14
UNB-VT-TFIDF-STO	4	34.79	29.72
UNB-VT-TFIDF-STO	5	33.61	30.08
UNB-VT-TFIDF-STO	6	33.23	30.98
UNB-VT-TFIDF-STO	7	31.59	29.72
UNB-VT-TFIDF-STO	8	27.08	25.54
UNB-VT-TFIDF-STO	9	25.40	24.60
UNB-VT-TFIDF-STO	10	16.67	16.67
UNB-VT-TFIDF-LEM	1	44.52	22.66
UNB-VT-TFIDF-LEM	2	37.24	25.88
UNB-VT-TFIDF-LEM	3	34.54	27.54
UNB-VT-TFIDF-LEM	4	32.96	28.04
UNB-VT-TFIDF-LEM	5	32.25	28.84
UNB-VT-TFIDF-LEM	6	31.11	28.79
UNB-VT-TFIDF-LEM	7	31.32	29.18
UNB-VT-TFIDF-LEM	8	30.56	28.73
UNB-VT-TFIDF-LEM	9	28.57	27.46
UNB-VT-TFIDF-LEM	10	23.33	23.33

Tabela A.5: Precision@n e Recall@n - Parte 1

Model	N	Precision_n	Recall_n
UNB-VT-T5	1	26.19	12.83
UNB-VT-T5	2	21.39	14.75
UNB-VT-T5	3	19.20	15.26
UNB-VT-T5	4	18.16	15.45
UNB-VT-T5	5	17.68	15.77
UNB-VT-T5	6	17.27	15.91
UNB-VT-T5	7	17.03	15.41
UNB-VT-T5	8	20.83	19.12
UNB-VT-T5	9	23.81	22.38
UNB-VT-T5	10	30.00	30.00
UNB-VT-HYBRID-TOK	1	48.18	24.69
UNB-VT-HYBRID-TOK	2	41.72	29.28
UNB-VT-HYBRID-TOK	3	38.49	30.75
UNB-VT-HYBRID-TOK	4	36.85	31.48
UNB-VT-HYBRID-TOK	5	35.35	31.56
UNB-VT-HYBRID-TOK	6	35.15	32.54
UNB-VT-HYBRID-TOK	7	34.89	32.33
UNB-VT-HYBRID-TOK	8	35.42	33.07
UNB-VT-HYBRID-TOK	9	38.10	36.83
UNB-VT-HYBRID-TOK	10	26.67	26.67
UNB-VT-HYBRID-STO	1	47.37	24.39
UNB-VT-HYBRID-STO	2	39.90	27.98
UNB-VT-HYBRID-STO	3	37.07	29.89
UNB-VT-HYBRID-STO	4	34.32	29.40
UNB-VT-HYBRID-STO	5	32.74	29.32
UNB-VT-HYBRID-STO	6	32.22	30.01
UNB-VT-HYBRID-STO	7	29.95	27.80
UNB-VT-HYBRID-STO	8	30.56	28.72
UNB-VT-HYBRID-STO	9	30.16	29.21
UNB-VT-HYBRID-STO	10	20.00	20.00

Tabela A.6: Precision@n e Recall@n - Parte 2

Model	N	Precision_n	Recall_n
UNB-VT-HYBRID-LEM	1	48.18	24.79
UNB-VT-HYBRID-LEM	2	41.56	29.13
UNB-VT-HYBRID-LEM	3	38.60	31.03
UNB-VT-HYBRID-LEM	4	36.48	31.27
UNB-VT-HYBRID-LEM	5	34.53	30.83
UNB-VT-HYBRID-LEM	6	33.94	31.40
UNB-VT-HYBRID-LEM	7	33.79	31.38
UNB-VT-HYBRID-LEM	8	32.64	30.32
UNB-VT-HYBRID-LEM	9	34.92	33.33
UNB-VT-HYBRID-LEM	10	33.33	33.33
B19-VT-TFIDF-TOK	1	36.71	19.74
B19-VT-TFIDF-TOK	2	35.61	25.7
B19-VT-TFIDF-TOK	3	37.09	30.3
B19-VT-TFIDF-TOK	4	38.18	32.99
B19-VT-TFIDF-TOK	5	39.61	35.77
B19-VT-TFIDF-TOK	6	39.9	37.12
B19-VT-TFIDF-TOK	7	40.38	37.74
B19-VT-TFIDF-TOK	8	38.89	36.08
B19-VT-TFIDF-TOK	9	42.86	40.79
B19-VT-TFIDF-TOK	10	43.33	43.33
B19-VT-TFIDF-STO	1	35.38	19.07
B19-VT-TFIDF-STO	2	35.19	25.48
B19-VT-TFIDF-STO	3	36.36	29.73
B19-VT-TFIDF-STO	4	37.76	32.8
B19-VT-TFIDF-STO	5	38.35	34.74
B19-VT-TFIDF-STO	6	39.19	36.75
B19-VT-TFIDF-STO	7	38.19	35.83
B19-VT-TFIDF-STO	8	36.11	33.94
B19-VT-TFIDF-STO	9	38.1	36.35
B19-VT-TFIDF-STO	10	36.67	36.67

Tabela A.7: Precision@n e Recall@n - Parte 3

Model	N	Precision_n	Recall_n
B19-VT-TFIDF-LEM	1	35.07	18.93
B19-VT-TFIDF-LEM	2	35.71	25.89
B19-VT-TFIDF-LEM	3	36.64	29.9
B19-VT-TFIDF-LEM	4	38.28	33.22
B19-VT-TFIDF-LEM	5	38.6	34.91
B19-VT-TFIDF-LEM	6	38.89	36.4
B19-VT-TFIDF-LEM	7	39.01	36.37
B19-VT-TFIDF-LEM	8	40.97	38.58
B19-VT-TFIDF-LEM	9	38.1	36.35
B19-VT-TFIDF-LEM	10	40.0	40.0
B19-VT-T5	1	29.1	15.35
B19-VT-T5	2	29.86	21.39
B19-VT-T5	3	31.53	25.71
B19-VT-T5	4	33.17	28.66
B19-VT-T5	5	34.04	30.68
B19-VT-T5	6	34.34	32.14
B19-VT-T5	7	32.97	30.59
B19-VT-T5	8	35.42	33.18
B19-VT-T5	9	38.1	36.35
B19-VT-T5	10	36.67	36.67
B19-VT-HYBRID-TOK	1	34.09	18.21
B19-VT-HYBRID-TOK	2	34.01	24.59
B19-VT-HYBRID-TOK	3	35.71	29.17
B19-VT-HYBRID-TOK	4	37.34	32.46
B19-VT-HYBRID-TOK	5	38.16	34.54
B19-VT-HYBRID-TOK	6	38.48	36.11
B19-VT-HYBRID-TOK	7	36.54	34.35
B19-VT-HYBRID-TOK	8	35.42	33.26
B19-VT-HYBRID-TOK	9	36.51	34.92
B19-VT-HYBRID-TOK	10	36.67	36.67

Tabela A.8: Precision@n e Recall@n - Parte 4

Model	N	Precision_n	Recall_n
B19-VT-HYBRID-STO	1	34.18	18.56
B19-VT-HYBRID-STO	2	33.85	24.45
B19-VT-HYBRID-STO	3	35.32	28.95
B19-VT-HYBRID-STO	4	36.35	31.62
B19-VT-HYBRID-STO	5	36.51	33.13
B19-VT-HYBRID-STO	6	37.27	35.1
B19-VT-HYBRID-STO	7	34.62	32.47
B19-VT-HYBRID-STO	8	34.03	31.71
B19-VT-HYBRID-STO	9	36.51	34.76
B19-VT-HYBRID-STO	10	40.0	40.0
B19-VT-HYBRID-LEM	1	34.34	18.45
B19-VT-HYBRID-LEM	2	34.07	24.62
B19-VT-HYBRID-LEM	3	35.5	29.09
B19-VT-HYBRID-LEM	4	36.46	31.66
B19-VT-HYBRID-LEM	5	37.29	33.72
B19-VT-HYBRID-LEM	6	37.37	35.02
B19-VT-HYBRID-LEM	7	39.01	36.71
B19-VT-HYBRID-LEM	8	39.58	37.27
B19-VT-HYBRID-LEM	9	34.92	33.33
B19-VT-HYBRID-LEM	10	36.67	36.67
B12-VT-TFIDF-TOK	1	26.72	14.16
B12-VT-TFIDF-TOK	2	27.21	19.58
B12-VT-TFIDF-TOK	3	27.78	22.78
B12-VT-TFIDF-TOK	4	27.84	24.19
B12-VT-TFIDF-TOK	5	28.28	25.68
B12-VT-TFIDF-TOK	6	28.89	27.09
B12-VT-TFIDF-TOK	7	26.65	25.05
B12-VT-TFIDF-TOK	8	28.47	26.60
B12-VT-TFIDF-TOK	9	28.57	27.14
B12-VT-TFIDF-TOK	10	33.33	33.33

Tabela A.9: Precision@n e Recall@n - Parte 5

Model	N	Precision_n	Recall_n
B12-VT-TFIDF-STO	1	25.80	13.68
B12-VT-TFIDF-STO	2	25.95	18.77
B12-VT-TFIDF-STO	3	26.36	21.59
B12-VT-TFIDF-STO	4	27.04	23.38
B12-VT-TFIDF-STO	5	27.99	25.41
B12-VT-TFIDF-STO	6	27.98	26.22
B12-VT-TFIDF-STO	7	25.55	23.94
B12-VT-TFIDF-STO	8	26.39	24.66
B12-VT-TFIDF-STO	9	28.57	27.14
B12-VT-TFIDF-STO	10	30.00	30.00
B12-VT-TFIDF-LEM	1	25.74	13.68
B12-VT-TFIDF-LEM	2	26.31	19.02
B12-VT-TFIDF-LEM	3	26.77	21.94
B12-VT-TFIDF-LEM	4	27.27	23.63
B12-VT-TFIDF-LEM	5	27.80	25.19
B12-VT-TFIDF-LEM	6	28.59	26.84
B12-VT-TFIDF-LEM	7	27.47	25.76
B12-VT-TFIDF-LEM	8	25.69	23.98
B12-VT-TFIDF-LEM	9	28.57	26.98
B12-VT-TFIDF-LEM	10	33.33	33.33
B12-VT-T5	1	22.50	12.08
B12-VT-T5	2	22.99	16.54
B12-VT-T5	3	24.12	19.79
B12-VT-T5	4	25.16	21.74
B12-VT-T5	5	26.78	24.10
B12-VT-T5	6	28.59	26.71
B12-VT-T5	7	27.20	25.27
B12-VT-T5	8	28.47	26.59
B12-VT-T5	9	31.75	30.48
B12-VT-T5	10	26.67	26.67

Tabela A.10: Precision@n e Recall@n - Parte 6

Model	N	Precision_n	Recall_n
B12-VT-HYBRID-TOK	1	26.78	14.48
B12-VT-HYBRID-TOK	2	27.07	19.62
B12-VT-HYBRID-TOK	3	27.75	22.77
B12-VT-HYBRID-TOK	4	27.97	24.29
B12-VT-HYBRID-TOK	5	28.81	26.24
B12-VT-HYBRID-TOK	6	28.38	26.68
B12-VT-HYBRID-TOK	7	27.20	25.71
B12-VT-HYBRID-TOK	8	22.92	21.48
B12-VT-HYBRID-TOK	9	25.40	24.13
B12-VT-HYBRID-TOK	10	33.33	33.33
B12-VT-HYBRID-STO	1	26.84	14.44
B12-VT-HYBRID-STO	2	27.28	19.81
B12-VT-HYBRID-STO	3	27.73	22.72
B12-VT-HYBRID-STO	4	27.90	24.23
B12-VT-HYBRID-STO	5	29.06	26.41
B12-VT-HYBRID-STO	6	28.69	26.99
B12-VT-HYBRID-STO	7	28.02	26.40
B12-VT-HYBRID-STO	8	23.61	22.11
B12-VT-HYBRID-STO	9	26.98	25.40
B12-VT-HYBRID-STO	10	33.33	33.33
B12-VT-HYBRID-LEM	1	26.69	14.40
B12-VT-HYBRID-LEM	2	27.24	19.75
B12-VT-HYBRID-LEM	3	27.62	22.67
B12-VT-HYBRID-LEM	4	27.66	24.01
B12-VT-HYBRID-LEM	5	29.01	26.36
B12-VT-HYBRID-LEM	6	28.48	26.76
B12-VT-HYBRID-LEM	7	26.92	25.39
B12-VT-HYBRID-LEM	8	22.92	21.56
B12-VT-HYBRID-LEM	9	22.22	20.79
B12-VT-HYBRID-LEM	10	30.00	30.00

Tabela A.11: Precision@n e Recall@n - Parte 7

Model	N	Precision_n	Recall_n
B11-VT-TFIDF-TOK	1	23.38	12.38
B11-VT-TFIDF-TOK	2	24.66	17.77
B11-VT-TFIDF-TOK	3	25.53	20.96
B11-VT-TFIDF-TOK	4	25.68	22.30
B11-VT-TFIDF-TOK	5	27.12	24.60
B11-VT-TFIDF-TOK	6	27.68	26.00
B11-VT-TFIDF-TOK	7	26.10	24.49
B11-VT-TFIDF-TOK	8	25.69	24.03
B11-VT-TFIDF-TOK	9	26.98	25.71
B11-VT-TFIDF-TOK	10	30.00	30.00
B11-VT-TFIDF-STO	1	23.65	12.61
B11-VT-TFIDF-STO	2	24.58	17.73
B11-VT-TFIDF-STO	3	25.42	20.81
B11-VT-TFIDF-STO	4	25.86	22.40
B11-VT-TFIDF-STO	5	27.17	24.63
B11-VT-TFIDF-STO	6	27.17	25.44
B11-VT-TFIDF-STO	7	25.82	24.13
B11-VT-TFIDF-STO	8	25.00	23.41
B11-VT-TFIDF-STO	9	26.98	25.71
B11-VT-TFIDF-STO	10	26.67	26.67
B11-VT-TFIDF-LEM	1	23.36	12.50
B11-VT-TFIDF-LEM	2	24.01	17.32
B11-VT-TFIDF-LEM	3	25.01	20.46
B11-VT-TFIDF-LEM	4	25.44	21.98
B11-VT-TFIDF-LEM	5	26.49	23.96
B11-VT-TFIDF-LEM	6	27.17	25.45
B11-VT-TFIDF-LEM	7	25.82	24.28
B11-VT-TFIDF-LEM	8	24.31	22.79
B11-VT-TFIDF-LEM	9	26.98	25.71
B11-VT-TFIDF-LEM	10	26.67	26.67

Tabela A.12: Precision@n e Recall@n - Parte 8

Model	N	Precision_n	Recall_n
B11-VT-T5	1	17.53	9.43
B11-VT-T5	2	17.88	12.79
B11-VT-T5	3	19.25	15.78
B11-VT-T5	4	19.60	17.01
B11-VT-T5	5	20.39	18.44
B11-VT-T5	6	21.21	19.83
B11-VT-T5	7	20.88	19.36
B11-VT-T5	8	21.53	19.94
B11-VT-T5	9	23.81	22.70
B11-VT-T5	10	26.67	26.67
B11-VT-HYBRID-TOK	1	22.28	11.90
B11-VT-HYBRID-TOK	2	22.52	16.32
B11-VT-HYBRID-TOK	3	24.23	19.85
B11-VT-HYBRID-TOK	4	25.44	22.19
B11-VT-HYBRID-TOK	5	24.84	22.47
B11-VT-HYBRID-TOK	6	25.25	23.68
B11-VT-HYBRID-TOK	7	23.08	21.62
B11-VT-HYBRID-TOK	8	25.00	23.63
B11-VT-HYBRID-TOK	9	23.81	22.70
B11-VT-HYBRID-TOK	10	30.00	30.00
B11-VT-HYBRID-STO	1	22.31	12.14
B11-VT-HYBRID-STO	2	22.69	16.37
B11-VT-HYBRID-STO	3	23.80	19.42
B11-VT-HYBRID-STO	4	25.00	21.67
B11-VT-HYBRID-STO	5	25.71	23.34
B11-VT-HYBRID-STO	6	25.45	23.95
B11-VT-HYBRID-STO	7	23.08	21.74
B11-VT-HYBRID-STO	8	20.14	19.04
B11-VT-HYBRID-STO	9	17.46	16.83
B11-VT-HYBRID-STO	10	16.67	16.67

Tabela A.13: Precision@n e Recall@n - Parte 9

Model	N	Precision_n	Recall_n
B11-VT-HYBRID-LEM	1	21.95	11.92
B11-VT-HYBRID-LEM	2	22.76	16.43
B11-VT-HYBRID-LEM	3	23.51	19.20
B11-VT-HYBRID-LEM	4	24.69	21.38
B11-VT-HYBRID-LEM	5	25.38	23.00
B11-VT-HYBRID-LEM	6	25.76	24.20
B11-VT-HYBRID-LEM	7	23.35	22.13
B11-VT-HYBRID-LEM	8	19.44	18.50
B11-VT-HYBRID-LEM	9	17.46	16.83
B11-VT-HYBRID-LEM	10	16.67	16.67

Tabela A.14: Precision@n e Recall@n - Parte 10

Apêndice B

Resultados da Etapa 2

Neste Apêndice, são exibidos os valores dos resultados obtidos nos experimentos da Etapa 2.

Model	Config	AUC-ROC	AUC-PR
UNB-VT-TFIDF-TOK	AHF	83.38	34.76
UNB-VT-TFIDF-TOK	AHT	83.38	34.76
UNB-VT-TFIDF-TOK	ASF	86.21	33.38
UNB-VT-TFIDF-TOK	AST	86.21	32.90
UNB-VT-TFIDF-TOK	RHF	50.00	0.66
UNB-VT-TFIDF-TOK	RHT	50.00	0.66
UNB-VT-TFIDF-TOK	RSF	87.44	13.37
UNB-VT-TFIDF-TOK	RST	87.29	14.00
UNB-VT-TFIDF-TOK	SHF	50.11	0.67
UNB-VT-TFIDF-TOK	SHT	50.11	0.67
UNB-VT-TFIDF-TOK	SSF	89.91	18.62
UNB-VT-TFIDF-TOK	SST	89.90	18.60
UNB-VT-TFIDF-STO	AHF	76.08	37.11
UNB-VT-TFIDF-STO	AHT	76.08	37.11
UNB-VT-TFIDF-STO	ASF	83.65	36.08
UNB-VT-TFIDF-STO	AST	83.61	36.71
UNB-VT-TFIDF-STO	RHF	51.25	2.40
UNB-VT-TFIDF-STO	RHT	51.25	2.40
UNB-VT-TFIDF-STO	RSF	86.83	12.83
UNB-VT-TFIDF-STO	RST	86.84	13.61
UNB-VT-TFIDF-STO	SHF	50.21	0.68
UNB-VT-TFIDF-STO	SHT	50.21	0.67
UNB-VT-TFIDF-STO	SSF	90.25	18.44
UNB-VT-TFIDF-STO	SST	90.23	18.45

Tabela B.1: AUC-ROC e AUC-PR para os Modelos UNB-VT-TFIDF-TOK e UNB-VT-TFIDF-STO.

Model	Config	AUC-ROC	AUC-PR
UNB-VT-TFIDF-LEM	AHF	77.95	33.63
UNB-VT-TFIDF-LEM	AHT	77.95	33.63
UNB-VT-TFIDF-LEM	ASF	85.95	35.53
UNB-VT-TFIDF-LEM	AST	83.97	36.19
UNB-VT-TFIDF-LEM	RHF	51.32	2.49
UNB-VT-TFIDF-LEM	RHT	51.32	2.49
UNB-VT-TFIDF-LEM	RSF	86.80	13.35
UNB-VT-TFIDF-LEM	RST	87.08	14.70
UNB-VT-TFIDF-LEM	SHF	75.48	1.91
UNB-VT-TFIDF-LEM	SHT	50.27	0.68
UNB-VT-TFIDF-LEM	SSF	73.45	1.88
UNB-VT-TFIDF-LEM	SST	90.04	18.60
UNB-VT-T5	AHF	50.00	0.66
UNB-VT-T5	AHT	50.00	0.66
UNB-VT-T5	ASF	88.49	21.71
UNB-VT-T5	AST	86.10	24.77
UNB-VT-T5	RHF	50.00	0.66
UNB-VT-T5	RHT	50.00	0.66
UNB-VT-T5	RSF	90.41	20.22
UNB-VT-T5	RST	90.01	20.62
UNB-VT-T5	SHF	49.95	0.65
UNB-VT-T5	SHT	49.95	0.65
UNB-VT-T5	SSF	82.90	8.55
UNB-VT-T5	SST	82.90	8.55

Tabela B.2: AUC-ROC e AUC-PR para os Modelos UNB-VT-TFIDF-LEM e UNB-VT-T5.

Model	Config	AUC-ROC	AUC-PR
UNB-VT-HYBRID-TOK	AHF	50.00	0.66
UNB-VT-HYBRID-TOK	AHT	50.00	0.66
UNB-VT-HYBRID-TOK	ASF	84.68	36.70
UNB-VT-HYBRID-TOK	AST	85.87	36.03
UNB-VT-HYBRID-TOK	RHF	80.98	14.92
UNB-VT-HYBRID-TOK	RHT	83.12	11.14
UNB-VT-HYBRID-TOK	RSF	91.57	24.24
UNB-VT-HYBRID-TOK	RST	91.57	24.24
UNB-VT-HYBRID-TOK	SHF	74.74	2.07
UNB-VT-HYBRID-TOK	SHT	74.74	2.07
UNB-VT-HYBRID-TOK	SSF	87.76	16.08
UNB-VT-HYBRID-TOK	SST	87.72	16.08
UNB-VT-HYBRID-STO	AHF	50.00	0.66
UNB-VT-HYBRID-STO	AHT	50.00	0.66
UNB-VT-HYBRID-STO	ASF	83.16	36.74
UNB-VT-HYBRID-STO	AST	83.16	36.74
UNB-VT-HYBRID-STO	RHF	83.79	11.72
UNB-VT-HYBRID-STO	RHT	81.17	11.19
UNB-VT-HYBRID-STO	RSF	91.45	23.44
UNB-VT-HYBRID-STO	RST	91.11	23.81
UNB-VT-HYBRID-STO	SHF	74.97	1.85
UNB-VT-HYBRID-STO	SHT	71.32	1.58
UNB-VT-HYBRID-STO	SSF	89.84	17.64
UNB-VT-HYBRID-STO	SST	89.85	17.63

Tabela B.3: AUC-ROC e AUC-PR para os Modelos UNB-VT-HYBRID-TOK e UNB-VT-HYBRID-STO.

Model	Config	AUC-ROC	AUC-PR
UNB-VT-HYBRID-LEM	AHF	50.00	0.66
UNB-VT-HYBRID-LEM	AHT	50.00	0.66
UNB-VT-HYBRID-LEM	ASF	81.01	35.76
UNB-VT-HYBRID-LEM	AST	81.01	35.76
UNB-VT-HYBRID-LEM	RHF	81.48	12.15
UNB-VT-HYBRID-LEM	RHT	81.48	12.15
UNB-VT-HYBRID-LEM	RSF	91.30	23.15
UNB-VT-HYBRID-LEM	RST	90.31	22.50
UNB-VT-HYBRID-LEM	SHF	74.13	1.75
UNB-VT-HYBRID-LEM	SHT	71.84	1.64
UNB-VT-HYBRID-LEM	SSF	88.75	18.14
UNB-VT-HYBRID-LEM	SST	89.76	17.42
B19-VT-TFIDF-TOK	AHF	89.88	19.35
B19-VT-TFIDF-TOK	AHT	89.88	19.35
B19-VT-TFIDF-TOK	ASF	91.60	19.41
B19-VT-TFIDF-TOK	AST	91.63	19.76
B19-VT-TFIDF-TOK	RHF	90.06	16.92
B19-VT-TFIDF-TOK	RHT	90.06	16.92
B19-VT-TFIDF-TOK	RSF	90.71	16.89
B19-VT-TFIDF-TOK	RST	90.71	16.89
B19-VT-TFIDF-TOK	SHF	89.24	10.15
B19-VT-TFIDF-TOK	SHT	89.24	10.15
B19-VT-TFIDF-TOK	SSF	74.87	1.94
B19-VT-TFIDF-TOK	SST	74.87	1.94

Tabela B.4: AUC-ROC e AUC-PR para os Modelos UNB-VT-HYBRID-LEM e B19-VT-TFIDF-TOK.

Model	Config	AUC-ROC	AUC-PR
B19-VT-TFIDF-STO	AHF	90.35	18.30
B19-VT-TFIDF-STO	AHT	90.35	18.30
B19-VT-TFIDF-STO	ASF	91.25	18.18
B19-VT-TFIDF-STO	AST	91.25	18.18
B19-VT-TFIDF-STO	RHF	91.47	17.21
B19-VT-TFIDF-STO	RHT	91.47	17.21
B19-VT-TFIDF-STO	RSF	91.38	15.91
B19-VT-TFIDF-STO	RST	91.38	15.91
B19-VT-TFIDF-STO	SHF	89.17	10.40
B19-VT-TFIDF-STO	SHT	89.17	10.40
B19-VT-TFIDF-STO	SSF	90.34	13.47
B19-VT-TFIDF-STO	SST	90.34	13.47
B19-VT-TFIDF-LEM	AHF	90.52	17.30
B19-VT-TFIDF-LEM	AHT	90.52	17.30
B19-VT-TFIDF-LEM	ASF	91.62	18.30
B19-VT-TFIDF-LEM	AST	91.66	18.40
B19-VT-TFIDF-LEM	RHF	92.14	18.01
B19-VT-TFIDF-LEM	RHT	92.14	18.01
B19-VT-TFIDF-LEM	RSF	91.55	15.97
B19-VT-TFIDF-LEM	RST	91.55	15.97
B19-VT-TFIDF-LEM	SHF	89.12	10.30
B19-VT-TFIDF-LEM	SHT	89.12	10.30
B19-VT-TFIDF-LEM	SSF	90.02	13.16
B19-VT-TFIDF-LEM	SST	90.04	13.16

Tabela B.5: AUC-ROC e AUC-PR para os Modelos B19-VT-TFIDF-STO e B19-VT-TFIDF-LEM.

Model	Config	AUC-ROC	AUC-PR
B19-VT-T5	AHF	83.20	5.10
B19-VT-T5	AHT	83.20	5.10
B19-VT-T5	ASF	91.12	13.22
B19-VT-T5	AST	91.12	13.22
B19-VT-T5	RHF	89.30	12.58
B19-VT-T5	RHT	89.30	12.58
B19-VT-T5	RSF	91.24	16.82
B19-VT-T5	RST	91.24	16.82
B19-VT-T5	SHF	78.82	2.36
B19-VT-T5	SHT	78.82	2.36
B19-VT-T5	SSF	74.97	1.97
B19-VT-T5	SST	74.97	1.97
B19-VT-HYBRID-TOK	AHF	89.17	17.11
B19-VT-HYBRID-TOK	AHT	89.17	17.11
B19-VT-HYBRID-TOK	ASF	90.81	17.01
B19-VT-HYBRID-TOK	AST	90.81	17.01
B19-VT-HYBRID-TOK	RHF	90.97	16.87
B19-VT-HYBRID-TOK	RHT	90.97	16.87
B19-VT-HYBRID-TOK	RSF	90.78	18.03
B19-VT-HYBRID-TOK	RST	90.78	18.03
B19-VT-HYBRID-TOK	SHF	88.34	8.68
B19-VT-HYBRID-TOK	SHT	88.34	8.68
B19-VT-HYBRID-TOK	SSF	86.75	9.23
B19-VT-HYBRID-TOK	SST	86.75	9.23

Tabela B.6: AUC-ROC e AUC-PR para os Modelos B19-VT-T5 e B19-VT-HYBRID-TOK.

Model	Config	AUC-ROC	AUC-PR
B19-VT-HYBRID-STO	AHF	88.94	17.74
B19-VT-HYBRID-STO	AHT	88.94	17.74
B19-VT-HYBRID-STO	ASF	91.11	17.39
B19-VT-HYBRID-STO	AST	91.11	17.39
B19-VT-HYBRID-STO	RHF	90.08	18.14
B19-VT-HYBRID-STO	RHT	90.08	18.14
B19-VT-HYBRID-STO	RSF	91.29	18.22
B19-VT-HYBRID-STO	RST	91.29	18.22
B19-VT-HYBRID-STO	SHF	88.22	8.38
B19-VT-HYBRID-STO	SHT	88.22	8.38
B19-VT-HYBRID-STO	SSF	87.12	9.75
B19-VT-HYBRID-STO	SST	87.12	9.75
B19-VT-HYBRID-LEM	AHF	89.90	16.76
B19-VT-HYBRID-LEM	AHT	89.90	16.76
B19-VT-HYBRID-LEM	ASF	91.08	17.00
B19-VT-HYBRID-LEM	AST	91.08	17.00
B19-VT-HYBRID-LEM	RHF	91.07	16.57
B19-VT-HYBRID-LEM	RHT	91.07	16.57
B19-VT-HYBRID-LEM	RSF	90.96	17.83
B19-VT-HYBRID-LEM	RST	90.96	17.83
B19-VT-HYBRID-LEM	SHF	87.53	7.00
B19-VT-HYBRID-LEM	SHT	87.53	7.00
B19-VT-HYBRID-LEM	SSF	88.08	11.08
B19-VT-HYBRID-LEM	SST	88.08	11.08

Tabela B.7: AUC-ROC e AUC-PR para os Modelos B19-VT-HYBRID-STO e B19-VT-HYBRID-LEM.

Model	Config	FNR	TNR	FPR	TPR	Precision	F1-Score
UNB-VT-TFIDF-TOK	AHF	58.02	99.45	0.55	41.98	33.56	37.30
UNB-VT-TFIDF-TOK	AHT	58.02	99.45	0.55	41.98	33.56	37.30
UNB-VT-TFIDF-TOK	ASF	73.47	99.88	0.12	26.53	59.25	36.65
UNB-VT-TFIDF-TOK	AST	74.03	99.88	0.12	25.97	59.26	36.11
UNB-VT-TFIDF-TOK	RHF	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-TFIDF-TOK	RHT	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-TFIDF-TOK	RSF	95.70	99.97	0.03	4.30	49.92	7.92
UNB-VT-TFIDF-TOK	RST	90.95	99.93	0.07	9.05	44.74	15.05
UNB-VT-TFIDF-TOK	SHF	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-TFIDF-TOK	SHT	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-TFIDF-TOK	SSF	79.81	99.76	0.24	20.19	35.69	25.80
UNB-VT-TFIDF-TOK	SST	80.08	99.77	0.23	19.92	36.11	25.67
UNB-VT-TFIDF-STO	AHF	66.88	99.90	0.10	33.12	68.49	44.65
UNB-VT-TFIDF-STO	AHT	66.88	99.90	0.10	33.12	68.49	44.65
UNB-VT-TFIDF-STO	ASF	70.27	99.87	0.13	29.73	59.40	39.63
UNB-VT-TFIDF-STO	AST	69.76	99.87	0.13	30.24	59.81	40.17
UNB-VT-TFIDF-STO	RHF	99.07	100.00	0.00	0.93	67.68	1.84
UNB-VT-TFIDF-STO	RHT	99.07	100.00	0.00	0.93	67.68	1.84
UNB-VT-TFIDF-STO	RSF	95.32	99.97	0.03	4.68	52.09	8.58
UNB-VT-TFIDF-STO	RST	91.80	99.94	0.06	8.20	46.56	13.94
UNB-VT-TFIDF-STO	SHF	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-TFIDF-STO	SHT	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-TFIDF-STO	SSF	72.15	99.52	0.48	27.85	27.65	27.75
UNB-VT-TFIDF-STO	SST	72.97	99.55	0.45	27.03	28.37	27.68

Tabela B.8: FNR, TNR, FPR, TPR, Precision e F1-Score para os Modelos UNB-VT-TFIDF-TOK e UNB-VT-TFIDF-STO.

Model	Config	FNR	TNR	FPR	TPR	Precision	F1-Score
UNB-VT-TFIDF-LEM	AHF	69.77	99.90	0.10	30.23	66.02	41.47
UNB-VT-TFIDF-LEM	AHT	69.77	99.90	0.10	30.23	66.02	41.47
UNB-VT-TFIDF-LEM	ASF	72.46	99.89	0.11	27.54	63.33	38.39
UNB-VT-TFIDF-LEM	AST	69.39	99.85	0.15	30.61	57.61	39.97
UNB-VT-TFIDF-LEM	RHF	98.76	100.00	0.00	1.24	65.44	2.43
UNB-VT-TFIDF-LEM	RHT	98.76	100.00	0.00	1.24	65.44	2.43
UNB-VT-TFIDF-LEM	RSF	90.37	99.92	0.08	9.63	44.02	15.80
UNB-VT-TFIDF-LEM	RST	88.11	99.90	0.10	11.89	43.37	18.66
UNB-VT-TFIDF-LEM	SHF	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-TFIDF-LEM	SHT	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-TFIDF-LEM	SSF	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-TFIDF-LEM	SST	74.99	99.62	0.38	25.01	30.20	27.36
UNB-VT-T5	AHF	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-T5	AHT	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-T5	ASF	84.72	99.89	0.11	15.28	47.78	23.16
UNB-VT-T5	AST	84.86	99.93	0.07	15.14	57.78	24.00
UNB-VT-T5	RHF	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-T5	RHT	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-T5	RSF	63.20	98.98	1.02	36.80	19.16	25.20
UNB-VT-T5	RST	65.37	99.27	0.73	34.63	23.91	28.29
UNB-VT-T5	SHF	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-T5	SHT	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-T5	SSF	99.87	100.00	0.00	0.13	81.82	0.25
UNB-VT-T5	SST	99.87	100.00	0.00	0.13	81.82	0.25

Tabela B.9: FNR, TNR, FPR, TPR, Precision e F1-Score para os Modelos UNB-VT-TFIDF-LEM e UNB-VT-T5.

Model	Config	FNR	TNR	FPR	TPR	Precision	F1-Score
UNB-VT-HYBRID-TOK	AHF	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-HYBRID-TOK	AHT	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-HYBRID-TOK	ASF	68.62	99.84	0.16	31.38	56.59	40.38
UNB-VT-HYBRID-TOK	AST	71.43	99.88	0.12	28.57	61.43	39.00
UNB-VT-HYBRID-TOK	RHF	88.88	99.94	0.06	11.12	54.10	18.45
UNB-VT-HYBRID-TOK	RHT	96.81	99.98	0.02	3.19	50.00	5.99
UNB-VT-HYBRID-TOK	RSF	48.35	97.45	2.55	51.65	11.79	19.19
UNB-VT-HYBRID-TOK	RST	48.35	97.45	2.55	51.65	11.79	19.19
UNB-VT-HYBRID-TOK	SHF	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-HYBRID-TOK	SHT	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-HYBRID-TOK	SSF	93.79	99.96	0.04	6.21	50.34	11.05
UNB-VT-HYBRID-TOK	SST	93.79	99.96	0.04	6.21	50.34	11.05
UNB-VT-HYBRID-STO	AHF	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-HYBRID-STO	AHT	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-HYBRID-STO	ASF	70.03	99.88	0.12	29.97	61.66	40.33
UNB-VT-HYBRID-STO	AST	70.03	99.88	0.12	29.97	61.66	40.33
UNB-VT-HYBRID-STO	RHF	96.56	99.98	0.02	3.44	49.01	6.42
UNB-VT-HYBRID-STO	RHT	96.48	99.98	0.02	3.52	52.06	6.60
UNB-VT-HYBRID-STO	RSF	46.97	96.85	3.15	53.03	10.01	16.84
UNB-VT-HYBRID-STO	RST	51.80	97.68	2.32	48.20	12.06	19.29
UNB-VT-HYBRID-STO	SHF	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-HYBRID-STO	SHT	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-HYBRID-STO	SSF	76.48	99.60	0.40	23.52	28.18	25.64
UNB-VT-HYBRID-STO	SST	76.37	99.60	0.40	23.63	28.08	25.67

Tabela B.10: FNR, TNR, FPR, TPR, Precision e F1-Score para os Modelos UNB-VT-HYBRID-TOK e UNB-VT-HYBRID-STO.

Model	Config	FNR	TNR	FPR	TPR	Precision	F1-Score
UNB-VT-HYBRID-LEM	AHF	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-HYBRID-LEM	AHT	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-HYBRID-LEM	ASF	68.50	99.82	0.18	31.50	53.87	39.75
UNB-VT-HYBRID-LEM	AST	68.50	99.82	0.18	31.50	53.87	39.75
UNB-VT-HYBRID-LEM	RHF	99.01	100.00	0.00	0.99	69.61	1.95
UNB-VT-HYBRID-LEM	RHT	99.01	100.00	0.00	0.99	69.61	1.95
UNB-VT-HYBRID-LEM	RSF	51.44	97.60	2.40	48.56	11.79	18.97
UNB-VT-HYBRID-LEM	RST	57.98	98.34	1.66	42.02	14.27	21.31
UNB-VT-HYBRID-LEM	SHF	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-HYBRID-LEM	SHT	100.00	100.00	0.00	0.00	NaN	NaN
UNB-VT-HYBRID-LEM	SSF	86.36	99.87	0.13	13.64	40.61	20.42
UNB-VT-HYBRID-LEM	SST	79.07	99.68	0.32	20.93	30.24	24.74
B19-VT-TFIDF-TOK	AHF	58.34	98.55	1.45	41.66	15.97	23.09
B19-VT-TFIDF-TOK	AHT	58.34	98.55	1.45	41.66	15.97	23.09
B19-VT-TFIDF-TOK	ASF	36.59	95.80	4.20	63.41	09.05	15.84
B19-VT-TFIDF-TOK	AST	36.80	95.84	4.16	63.20	9.10	15.91
B19-VT-TFIDF-TOK	RHF	41.75	96.63	3.37	58.25	10.23	17.41
B19-VT-TFIDF-TOK	RHT	41.75	96.63	3.37	58.25	10.23	17.41
B19-VT-TFIDF-TOK	RSF	36.58	96.13	3.87	63.42	9.75	16.89
B19-VT-TFIDF-TOK	RST	36.58	96.13	3.87	63.42	9.75	16.89
B19-VT-TFIDF-TOK	SHF	76.73	99.12	0.88	23.27	14.83	18.11
B19-VT-TFIDF-TOK	SHT	76.73	99.12	0.88	23.27	14.83	18.11
B19-VT-TFIDF-TOK	SSF	100.00	100.00	0.00	0.00	NaN	NaN
B19-VT-TFIDF-TOK	SST	100.00	100.00	0.00	0.00	NaN	NaN

Tabela B.11: FNR, TNR, FPR, TPR, Precision e F1-Score para os Modelos UNB-VT-HYBRID-LEM e B19-VT-TFIDF-TOK.

Model	Config	FNR	TNR	FPR	TPR	Precision	F1-Score
B19-VT-TFIDF-STO	AHF	57.62	98.54	1.46	42.38	16.09	23.33
B19-VT-TFIDF-STO	AHT	57.62	98.54	1.46	42.38	16.09	23.33
B19-VT-TFIDF-STO	ASF	36.49	95.72	4.28	63.51	8.92	15.64
B19-VT-TFIDF-STO	AST	36.49	95.72	4.28	63.51	8.92	15.64
B19-VT-TFIDF-STO	RHF	38.69	96.22	3.78	61.31	9.66	16.70
B19-VT-TFIDF-STO	RHT	38.69	96.22	3.78	61.31	9.66	16.70
B19-VT-TFIDF-STO	RSF	38.19	95.93	4.07	61.81	9.10	15.86
B19-VT-TFIDF-STO	RST	38.19	95.93	4.07	61.81	9.10	15.86
B19-VT-TFIDF-STO	SHF	79.36	99.28	0.72	20.64	15.84	17.93
B19-VT-TFIDF-STO	SHT	79.36	99.28	0.72	20.64	15.84	17.93
B19-VT-TFIDF-STO	SSF	63.98	98.66	1.34	36.02	15.06	21.24
B19-VT-TFIDF-STO	SST	63.98	98.66	1.34	36.02	15.06	21.24
B19-VT-TFIDF-LEM	AHF	48.60	97.40	2.60	51.40	11.53	18.83
B19-VT-TFIDF-LEM	AHT	48.60	97.40	2.60	51.40	11.53	18.83
B19-VT-TFIDF-LEM	ASF	36.10	95.78	4.22	63.90	9.08	15.89
B19-VT-TFIDF-LEM	AST	35.52	95.74	4.26	64.48	9.08	15.91
B19-VT-TFIDF-LEM	RHF	35.85	95.94	4.06	64.15	9.43	16.44
B19-VT-TFIDF-LEM	RHT	35.85	95.94	4.06	64.15	9.43	16.44
B19-VT-TFIDF-LEM	RSF	37.29	95.71	4.29	62.71	8.79	15.41
B19-VT-TFIDF-LEM	RST	37.29	95.71	4.29	62.71	8.79	15.41
B19-VT-TFIDF-LEM	SHF	79.83	99.29	0.71	20.17	15.79	17.71
B19-VT-TFIDF-LEM	SHT	79.83	99.29	0.71	20.17	15.79	17.71
B19-VT-TFIDF-LEM	SSF	71.91	99.17	0.83	28.09	18.30	22.16
B19-VT-TFIDF-LEM	SST	71.82	99.17	0.83	28.18	18.24	22.15

Tabela B.12: FNR, TNR, FPR, TPR, Precision e F1-Score para os Modelos B19-VT-TFIDF-STO e B19-VT-TFIDF-LEM.

Model	Config	FNR	TNR	FPR	TPR	Precision	F1-Score
B19-VT-T5	AHF	83.77	98.79	1.21	16.23	8.12	10.82
B19-VT-T5	AHT	83.77	98.79	1.21	16.23	8.12	10.82
B19-VT-T5	ASF	30.84	91.91	8.09	69.16	5.34	9.91
B19-VT-T5	AST	30.84	91.91	8.09	69.16	5.34	9.91
B19-VT-T5	RHF	74.00	99.20	0.80	26.00	17.57	20.97
B19-VT-T5	RHT	74.00	99.20	0.80	26.00	17.57	20.97
B19-VT-T5	RSF	61.31	98.67	1.33	38.69	16.09	22.72
B19-VT-T5	RST	61.31	98.67	1.33	38.69	16.09	22.72
B19-VT-T5	SHF	100.00	100.00	0.00	0.00	NaN	NaN
B19-VT-T5	SHT	100.00	100.00	0.00	0.00	NaN	NaN
B19-VT-T5	SSF	100.00	100.00	0.00	0.00	NaN	NaN
B19-VT-T5	SST	100.00	100.00	0.00	0.00	NaN	NaN
B19-VT-HYBRID-TOK	AHF	47.40	97.74	2.26	52.60	13.29	21.22
B19-VT-HYBRID-TOK	AHT	47.40	97.74	2.26	52.60	13.29	21.22
B19-VT-HYBRID-TOK	ASF	38.11	96.10	3.90	61.89	9.48	16.44
B19-VT-HYBRID-TOK	AST	38.11	96.10	3.90	61.89	9.48	16.44
B19-VT-HYBRID-TOK	RHF	42.06	96.67	3.33	57.94	10.30	17.49
B19-VT-HYBRID-TOK	RHT	42.06	96.67	3.33	57.94	10.30	17.49
B19-VT-HYBRID-TOK	RSF	35.16	95.90	4.10	64.84	9.45	16.49
B19-VT-HYBRID-TOK	RST	35.16	95.90	4.10	64.84	9.45	16.49
B19-VT-HYBRID-TOK	SHF	83.17	99.34	0.66	16.83	14.43	15.54
B19-VT-HYBRID-TOK	SHT	83.17	99.34	0.66	16.83	14.43	15.54
B19-VT-HYBRID-TOK	SSF	61.38	96.84	3.16	38.62	7.45	12.49
B19-VT-HYBRID-TOK	SST	61.38	96.84	3.16	38.62	7.45	12.49

Tabela B.13: FNR, TNR, FPR, TPR, Precision e F1-Score para os Modelos B19-VT-T5 e B19-VT-HYBRID-TOK.

Model	Config	FNR	TNR	FPR	TPR	Precision	F1-Score
B19-VT-HYBRID-STO	AHF	47.06	97.84	2.16	52.94	13.94	22.06
B19-VT-HYBRID-STO	AHT	47.06	97.84	2.16	52.94	13.94	22.06
B19-VT-HYBRID-STO	ASF	38.69	96.11	3.89	61.31	9.42	16.32
B19-VT-HYBRID-STO	AST	38.69	96.11	3.89	61.31	9.42	16.32
B19-VT-HYBRID-STO	RHF	44.22	97.57	2.43	55.78	13.15	21.29
B19-VT-HYBRID-STO	RHT	44.22	97.57	2.43	55.78	13.15	21.29
B19-VT-HYBRID-STO	RSF	36.45	96.12	3.88	63.55	9.74	16.90
B19-VT-HYBRID-STO	RST	36.45	96.12	3.88	63.55	9.74	16.90
B19-VT-HYBRID-STO	SHF	83.88	99.39	0.61	16.12	14.87	15.47
B19-VT-HYBRID-STO	SHT	83.88	99.39	0.61	16.12	14.87	15.47
B19-VT-HYBRID-STO	SSF	59.62	96.72	3.28	40.38	7.51	12.67
B19-VT-HYBRID-STO	SST	59.62	96.72	3.28	40.38	7.51	12.67
B19-VT-HYBRID-LEM	AHF	45.09	97.53	2.47	54.91	12.79	20.75
B19-VT-HYBRID-LEM	AHT	45.09	97.53	2.47	54.91	12.79	20.75
B19-VT-HYBRID-LEM	ASF	38.93	96.07	3.93	61.07	9.31	16.15
B19-VT-HYBRID-LEM	AST	38.93	96.07	3.93	61.07	9.31	16.15
B19-VT-HYBRID-LEM	RHF	46.64	97.22	2.78	53.36	11.23	18.56
B19-VT-HYBRID-LEM	RHT	46.64	97.22	2.78	53.36	11.23	18.56
B19-VT-HYBRID-LEM	RSF	34.86	95.93	4.07	65.14	9.54	16.65
B19-VT-HYBRID-LEM	RST	34.86	95.93	4.07	65.14	9.54	16.65
B19-VT-HYBRID-LEM	SHF	87.74	99.51	0.49	12.26	14.12	13.13
B19-VT-HYBRID-LEM	SHT	87.74	99.51	0.49	12.26	14.12	13.13
B19-VT-HYBRID-LEM	SSF	53.50	96.12	3.88	46.50	7.33	12.67
B19-VT-HYBRID-LEM	SST	53.50	96.12	3.88	46.50	7.33	12.67

Tabela B.14: FNR, TNR, FPR, TPR, Precision e F1-Score para os Modelos B19-VT-HYBRID-STO e B19-VT-HYBRID-LEM.

Model	Config	N	Precision	Recall
UNB-VT-TFIDF-TOK	AHF	1	53.79	28.51
UNB-VT-TFIDF-TOK	AHF	2	49.23	35.09
UNB-VT-TFIDF-TOK	AHF	3	47.42	38.43
UNB-VT-TFIDF-TOK	AHF	4	46.06	39.73
UNB-VT-TFIDF-TOK	AHF	5	43.73	39.36
UNB-VT-TFIDF-TOK	AHF	6	42.02	39.19
UNB-VT-TFIDF-TOK	AHF	7	40.93	38.45
UNB-VT-TFIDF-TOK	AHF	8	36.11	33.86
UNB-VT-TFIDF-TOK	AHF	9	34.92	33.49
UNB-VT-TFIDF-TOK	AHF	10	30.00	30.00
UNB-VT-TFIDF-STO	AHF	1	50.34	26.26
UNB-VT-TFIDF-STO	AHF	2	43.91	30.74
UNB-VT-TFIDF-STO	AHF	3	40.91	32.79
UNB-VT-TFIDF-STO	AHF	4	38.83	33.29
UNB-VT-TFIDF-STO	AHF	5	36.85	33.05
UNB-VT-TFIDF-STO	AHF	6	35.05	32.47
UNB-VT-TFIDF-STO	AHF	7	35.16	32.84
UNB-VT-TFIDF-STO	AHF	8	31.94	29.61
UNB-VT-TFIDF-STO	AHF	9	36.51	35.08
UNB-VT-TFIDF-STO	AHF	10	30.00	30.00
UNB-VT-TFIDF-LEM	AST	1	47.24	24.11
UNB-VT-TFIDF-LEM	AST	2	40.85	28.60
UNB-VT-TFIDF-LEM	AST	3	38.06	30.43
UNB-VT-TFIDF-LEM	AST	4	36.43	31.03
UNB-VT-TFIDF-LEM	AST	5	35.79	32.05
UNB-VT-TFIDF-LEM	AST	6	34.75	32.27
UNB-VT-TFIDF-LEM	AST	7	33.52	31.23
UNB-VT-TFIDF-LEM	AST	8	31.94	29.91
UNB-VT-TFIDF-LEM	AST	9	31.75	30.48
UNB-VT-TFIDF-LEM	AST	10	26.67	26.67

Tabela B.15: Precision@n e Recall@n para os Modelos UNB-VT-TFIDF-TOK, UNB-VT-TFIDF-STO e UNB-VT-TFIDF-LEM.

Model	Config	N	Precision	Recall
UNB-VT-T5	RSF	1	37.32	19.63
UNB-VT-T5	RSF	2	35.46	25.33
UNB-VT-T5	RSF	3	35.71	29.06
UNB-VT-T5	RSF	4	35.65	30.91
UNB-VT-T5	RSF	5	34.77	31.21
UNB-VT-T5	RSF	6	34.95	32.59
UNB-VT-T5	RSF	7	33.24	30.79
UNB-VT-T5	RSF	8	36.81	34.44
UNB-VT-T5	RSF	9	33.33	31.43
UNB-VT-T5	RSF	10	40.00	40.00
UNB-VT-HYBRID-TOK	RSF	1	41.02	21.73
UNB-VT-HYBRID-TOK	RSF	2	38.63	27.76
UNB-VT-HYBRID-TOK	RSF	3	38.63	31.50
UNB-VT-HYBRID-TOK	RSF	4	38.88	33.60
UNB-VT-HYBRID-TOK	RSF	5	39.32	35.41
UNB-VT-HYBRID-TOK	RSF	6	40.61	38.08
UNB-VT-HYBRID-TOK	RSF	7	39.01	36.77
UNB-VT-HYBRID-TOK	RSF	8	33.33	31.16
UNB-VT-HYBRID-TOK	RSF	9	36.51	34.92
UNB-VT-HYBRID-TOK	RSF	10	33.33	33.33
UNB-VT-HYBRID-STO	RSF	1	41.48	21.69
UNB-VT-HYBRID-STO	RSF	2	38.80	27.68
UNB-VT-HYBRID-STO	RSF	3	38.57	31.40
UNB-VT-HYBRID-STO	RSF	4	39.01	33.67
UNB-VT-HYBRID-STO	RSF	5	39.18	35.25
UNB-VT-HYBRID-STO	RSF	6	38.99	36.40
UNB-VT-HYBRID-STO	RSF	7	36.54	34.15
UNB-VT-HYBRID-STO	RSF	8	34.03	31.79
UNB-VT-HYBRID-STO	RSF	9	36.51	34.92
UNB-VT-HYBRID-STO	RSF	10	43.33	43.33

Tabela B.16: Precision@n e Recall@n para os Modelos UNB-VT-T5, UNB-VT-HYBRID-TOK e UNB-VT-HYBRID-STO.

Model	Config	N	Precision	Recall
UNB-VT-HYBRID-LEM	RSF	1	40.34	21.21
UNB-VT-HYBRID-LEM	RSF	2	37.11	26.39
UNB-VT-HYBRID-LEM	RSF	3	37.31	30.33
UNB-VT-HYBRID-LEM	RSF	4	37.68	32.58
UNB-VT-HYBRID-LEM	RSF	5	37.87	34.16
UNB-VT-HYBRID-LEM	RSF	6	38.28	35.76
UNB-VT-HYBRID-LEM	RSF	7	36.26	33.77
UNB-VT-HYBRID-LEM	RSF	8	36.81	34.57
UNB-VT-HYBRID-LEM	RSF	9	36.51	34.76
UNB-VT-HYBRID-LEM	RSF	10	36.67	36.67
B19-VT-TFIDF-TOK	AST	1	36.90	19.74
B19-VT-TFIDF-TOK	AST	2	36.00	25.99
B19-VT-TFIDF-TOK	AST	3	37.54	30.68
B19-VT-TFIDF-TOK	AST	4	38.94	33.80
B19-VT-TFIDF-TOK	AST	5	39.95	36.00
B19-VT-TFIDF-TOK	AST	6	40.71	37.86
B19-VT-TFIDF-TOK	AST	7	42.03	39.09
B19-VT-TFIDF-TOK	AST	8	41.67	38.78
B19-VT-TFIDF-TOK	AST	9	42.86	40.79
B19-VT-TFIDF-TOK	AST	10	43.33	43.33
B19-VT-TFIDF-STO	RHF	1	35.44	19.09
B19-VT-TFIDF-STO	RHF	2	36.33	26.25
B19-VT-TFIDF-STO	RHF	3	38.09	31.10
B19-VT-TFIDF-STO	RHF	4	39.54	34.23
B19-VT-TFIDF-STO	RHF	5	41.36	37.43
B19-VT-TFIDF-STO	RHF	6	41.92	39.30
B19-VT-TFIDF-STO	RHF	7	40.38	37.89
B19-VT-TFIDF-STO	RHF	8	38.89	36.44
B19-VT-TFIDF-STO	RHF	9	39.68	37.78
B19-VT-TFIDF-STO	RHF	10	40.00	40.00

Tabela B.17: Precision@n e Recall@n para os Modelos UNB-VT-HYBRID-LEM e B19-VT-TFIDF-TOK, B19-VT-TFIDF-STO.

Model	Config	N	Precision	Recall
B19-VT-TFIDF-LEM	RHF	1	35.62	19.31
B19-VT-TFIDF-LEM	RHF	2	36.66	26.53
B19-VT-TFIDF-LEM	RHF	3	38.86	31.82
B19-VT-TFIDF-LEM	RHF	4	40.61	35.13
B19-VT-TFIDF-LEM	RHF	5	42.86	38.73
B19-VT-TFIDF-LEM	RHF	6	43.74	40.87
B19-VT-TFIDF-LEM	RHF	7	43.41	40.78
B19-VT-TFIDF-LEM	RHF	8	39.58	37.35
B19-VT-TFIDF-LEM	RHF	9	36.51	34.76
B19-VT-TFIDF-LEM	RHF	10	40.00	40.00
B19-VT-T5	RSF	1	33.74	17.92
B19-VT-T5	RSF	2	32.77	23.51
B19-VT-T5	RSF	3	33.73	27.30
B19-VT-T5	RSF	4	34.55	29.73
B19-VT-T5	RSF	5	35.88	32.31
B19-VT-T5	RSF	6	36.26	33.90
B19-VT-T5	RSF	7	34.34	31.95
B19-VT-T5	RSF	8	34.72	32.35
B19-VT-T5	RSF	9	36.51	34.60
B19-VT-T5	RSF	10	43.33	43.33
B19-VT-HYBRID-TOK	RSF	1	35.55	19.31
B19-VT-HYBRID-TOK	RSF	2	35.96	26.06
B19-VT-HYBRID-TOK	RSF	3	37.38	30.38
B19-VT-HYBRID-TOK	RSF	4	39.90	34.50
B19-VT-HYBRID-TOK	RSF	5	40.24	36.20
B19-VT-HYBRID-TOK	RSF	6	42.12	39.26
B19-VT-HYBRID-TOK	RSF	7	44.51	41.75
B19-VT-HYBRID-TOK	RSF	8	43.06	40.80
B19-VT-HYBRID-TOK	RSF	9	36.51	35.08
B19-VT-HYBRID-TOK	RSF	10	33.33	33.33

Tabela B.18: Precision@n e Recall@n para os Modelos B19-VT-TFIDF-LEM, B19-VT-T5 e B19-VT-HYBRID-TOK.

Model	Config	N	Precision	Recall
B19-VT-HYBRID-STO	RSF	1	35.77	19.37
B19-VT-HYBRID-STO	RSF	2	36.93	26.69
B19-VT-HYBRID-STO	RSF	3	38.30	31.21
B19-VT-HYBRID-STO	RSF	4	40.16	34.62
B19-VT-HYBRID-STO	RSF	5	42.03	37.93
B19-VT-HYBRID-STO	RSF	6	42.32	39.40
B19-VT-HYBRID-STO	RSF	7	43.68	40.96
B19-VT-HYBRID-STO	RSF	8	44.44	41.91
B19-VT-HYBRID-STO	RSF	9	44.44	42.38
B19-VT-HYBRID-STO	RSF	10	46.67	46.67
B19-VT-HYBRID-LEM	RSF	1	36.00	19.48
B19-VT-HYBRID-LEM	RSF	2	36.41	26.39
B19-VT-HYBRID-LEM	RSF	3	37.73	30.69
B19-VT-HYBRID-LEM	RSF	4	40.29	34.75
B19-VT-HYBRID-LEM	RSF	5	41.89	37.80
B19-VT-HYBRID-LEM	RSF	6	43.23	40.33
B19-VT-HYBRID-LEM	RSF	7	43.13	40.64
B19-VT-HYBRID-LEM	RSF	8	38.89	36.64
B19-VT-HYBRID-LEM	RSF	9	38.10	36.67
B19-VT-HYBRID-LEM	RSF	10	33.33	33.33

Tabela B.19: Precision@n e Recall@n para os Modelos B19-VT-HYBRID-STO e B19-VT-HYBRID-LEM.