UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ANDRÉ LUÍS MARTINS BANDEIRA

CLASSIFICAÇÃO AUTOMÁTICA DA PRIORIDADE DE DEFEITOS DE *SOFTWARE* UTILIZANDO SELEÇÃO DE ATRIBUTOS: UM ESTUDO DE CASO NA INDÚSTRIA

DISSERTAÇÃO

CORNÉLIO PROCÓPIO

2019

ANDRÉ LUÍS MARTINS BANDEIRA

CLASSIFICAÇÃO AUTOMÁTICA DA PRIORIDADE DE DEFEITOS DE *SOFTWARE* UTILIZANDO SELEÇÃO DE ATRIBUTOS: UM ESTUDO DE CASO NA INDÚSTRIA

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Tecnológica Federal do Paraná – UTFPR como requisito para a obtenção do título de "Mestre Profissional em Informática".

Orientador: Willian Massami Watanabe

CORNÉLIO PROCÓPIO

2019

Dados Internacionais de Catalogação na Publicação

B214 Bandeira, Andre Luis Martins

Classificação automática da prioridade de defeitos de software utilizando seleção de atributos : um estudo de caso na indústria / André Luís Martins Bandeira. - 2019.
54 f. : il. ; 31 cm.

Orientador: Willian Massami Watanabe.
Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Informática, Cornélio Procópio, 2019.
Bibliografia: p. 53-54.

1. Software - Desenvolvimento. 2. Localização de falhas (Engenharia). 3. Processamento de linguagem natural (Computação). 4. Aprendizado do computador. 5. Informática – Dissertações. I. Watanabe, Willian Massami, orient. II. Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD (22. ed.) 004

Biblioteca da UTFPR - Câmpus Cornélio Procópio
Bibliotecário/Documentalista responsável:
Romeu Righetti de Araujo – CRB-9/1676

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Cornélio Procópio
Programa de Pós-Graduação em Informática

Título da Dissertação Nº 61:

"CLASSIFICAÇÃO AUTOMÁTICA DA PRIORIDADE DE DEFEITOS UTILIZANDO SELEÇÃO DE ATRIBUTOS: UM ESTUDO DE CASO NA INDÚSTRIA".

por

André Luís Martins Bandeira

Orientador: **Prof. Dr. Willian Massami Watanabe**

Esta dissertação foi apresentada como requisito parcial à obtenção do grau de MESTRE EM INFORMÁTICA – Área de Concentração: Computação Aplicada, pelo Programa de Pós-Graduação em Informática – PPGI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Cornélio Procópio, às 14h30 do dia 17 de julho de 2019. O trabalho foi _____ pela Banca Examinadora, composta pelos professores:

Prof. Dr. Willian Massami Watanabe (Presidente – UTFPR-CP)	Prof. Dr. Marco Aurélio Graciotto Silva (UTFPR-CM)
Prof. Dr. Reginaldo Ré (UTFPR-CM)	Prof. Dr. Edson Alves de Oliveira Junior (UEM) Participação à distância via _____

Visto da coordenação:

Danilo Sipoli Sanches
Coordenador do Programa de Pós-Graduação em Informática
UTFPR Câmpus Cornélio Procópio

A Folha de Aprovação assinada encontra-se na Coordenação do Programa.

“Percebe e entende que os melhores amigos
São aqueles que estão em casa, esperando por ti
Acredita nos momentos mais difíceis da vida
Eles sempre estarão por perto, pois só sabem te amar
E se por acaso a dor chegar, ao teu lado vão estar
Pra te acolher e te amparar”

A minha família e a minha esposa dedico este trabalho, meus sonhos e todas as minhas próximas conquistas...

AGRADECIMENTOS

Aos meus pais, Acácio e Aparecida, por todo apoio, incentivo e paciência durante todo esse tempo.

Ao meu irmão Alan, a minha cunhada Gabriela e meus sobrinhos Marcos e Heitor por toda torcida e orações.

Ao meu irmãozinho Acacinho, por dar a oportunidade de dar exemplo. Em momentos difíceis eu me agarrei muito a você, você pode contar comigo sempre.

A todos da minha família que se mantiveram sempre presente ao meu lado, torcendo e vibrando a cada conquista alcançada.

A minha linda e amada esposa Camila pela parceria de longa data, pela força e incentivo desde o ensino médio até o mestrado. Você me ensina que podemos ser melhor a cada dia, mais que isso, dá o exemplo de como podemos mudar nossa trajetória, como podemos sonhar grande e como a educação e o único caminho para sermos melhores. Seu amor e carinho são minha fonte de esperança para que eu conseguisse alcançar esse objetivo que já se tornou nosso. Obrigado por tudo!

À família Silva e Carneiro: Marileide, Edison, Claudemir, Gumercindo, Elisângela, Vinícius, Lucas e Leonardo pelos momentos compartilhados e pela compreensão naqueles em que não pude estar presente.

Ao meu primeiro orientador Dr. Luciano Tadeu Esteves Pansanato pela também parceira de longa data, por ter me orientado durante a graduação e ter me permitido ingressar no programa de mestrado, devo muito a você e serei grato eternamente por tudo que fez por mim.

Ao meu segundo orientador Dr. Reginaldo Re, por ter aceitado me orientar mesmo com o "bonde andando", você me ajudou nesse caminho e não existiria fim sem o meio.

Ao meu terceiro orientador Dr. Willian Massami Watanabe, por ter enxergado possibilidades em mim e em meu trabalho quando eu mesmo já não enxergava. Eu já tinha praticamente desistido do mestrado quando recebi sua ligação, conhecendo você desde os tempos da graduação sabia que seria uma oportunidade única, sua paciência e objetividade me proporcionaram entregar esse trabalho, sem você isso seria ainda apenas um sonho. Agradeço ainda o acolhimento, seus ensinamentos e, principalmente, as valiosas oportunidades que me ofereceu

de crescimento pessoal, profissional e academicamente.

Aos professores, Dr. Marco Aurélio Graciotto Silva e Dr. Edson Oliveira Jr., por aceitarem o convite para participar da banca de defesa desta dissertação. Agradeço a atenção, a leitura cuidadosa do texto e as inúmeras sugestões e contribuições que permitiram a conclusão de um trabalho com maior qualidade.

Ao Dr. Cléber Gimenez por aceitar ser suplente na banca de defesa desta dissertação.

A DSIN Tecnologia da Informação pela compreensão e ajuda durante todo esse processo, sempre me escutaram e fizeram de tudo para possibilitar esse trabalho, cada um de meus companheiros de time tem uma grande parte nisso. Um obrigado mais que especial ao Marcelo, Bruno, Larissa, Jorge, Douglas, Felipe, Guilherme, Andréia e Andressa. Sempre levarei a DSIN comigo.

Ao amigo e parceiro que o mestrado me deu de presente, Fagner Paes. Você tem me ensinado a dar mais valor à vida, aos momentos e, sobretudo, às pessoas. Benditos sejam os amigos que nos fazem alguém melhor!

A todos os meus professores, sou a prova de que a educação pode mudar o destino de qualquer um, espero que um dia vocês sejam valorizados como merecem.

Aos que, embora não citados, me ofereceram apoio e amizade. Aqueles que mesmo distantes permaneceram presentes...

A todos vocês, meu muito obrigado!

É preciso ter esperança, mas ter esperança do verbo esperançar; porque tem gente que tem esperança do verbo esperar. E esperança do verbo esperar não é esperança, é espera.

Esperançar é se levantar, esperançar é ir atrás, esperançar é construir, esperançar é não desistir! Esperançar é levar adiante, esperançar é juntar-se com outros para fazer de outro modo...

Paulo Freire

RESUMO

Bandeira, André. CLASSIFICAÇÃO AUTOMÁTICA DA PRIORIDADE DE DEFEITOS DE *SOFTWARE* UTILIZANDO SELEÇÃO DE ATRIBUTOS: UM ESTUDO DE CASO NA INDÚSTRIA. 54 f. Dissertação – Programa de Pós-graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2019.

Defeitos são inevitáveis em projetos de *software*, portanto, adotar uma política de análise e gerenciamento de defeitos durante o ciclo de desenvolvimento de *software* é vital para a garantia de qualidade do mesmo. O armazenamento de relatórios de defeitos é comum no ciclo de desenvolvimento de *software*, mas as informações contidas nos relatórios são difíceis de entender uma vez que geralmente são escritas em linguagem natural. Nesse sentido, a classificação de defeitos pode ajudar a agilizar o processo de entendimento e gerenciamento desses defeitos. Por serem escritas em linguagem natural, a classificação automática de defeitos se torna difícil e pode ter baixa efetividade, com um analista levando em média de 6 minutos para cada relatório de defeito. Devido a isso, alguns estudos propõem a utilização de abordagens de seleção de atributos para aumentar a precisão da classificação automática de defeitos. Este trabalho apresenta um estudo de caso da indústria com a classificação automática dos defeitos utilizando métodos de seleção de atributos propostos na literatura. Além disso, é proposta uma alteração ao algoritmo de seleção de características USES visando melhorar sua efetividade, o novo algoritmo foi denominado USES+. Como resultado, tem-se que as abordagens automáticas de seleção de atributos resultam em uma maior efetividade da classificação automática de defeito. Por fim, USES+ obteve uma melhor efetividade quando comparado ao USES, inclusive com diferenças significativas. Portanto, de um modo geral, a classificação automática de defeitos utilizando abordagens de seleção de atributos mostrou ser bastante promissora apresentando uma boa efetividade. Dessa forma, a classificação automática permite a redução do custo de priorização dos defeitos, melhorando significativamente o tempo para que as correções essenciais sejam executas.

Palavras-chave: Defeitos de Software, Processamento de Linguagem Natural, Aprendizado de Máquina

ABSTRACT

Bandeira, André. AUTOMATIC SOFTWARE DEFECT PRIORITY CLASSIFICATION USING FEATURE SELECTION: A CASE STUDY IN INDUSTRY. 54 f. Dissertação – Programa de Pós-graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2019.

Defects are inevitable in software projects, so adopting a policy of analyzing and managing defects during the software development cycle is vital for quality assurance. Storing defect reports is commonplace in the software development cycle, but the information contained in the reports is difficult to understand because they are usually written in natural language. In this sense, classifying defects can help streamline the defect management process. As they are written in natural language, automatic defect classification becomes difficult and may have low effectiveness, with an analyst taking an average of 6 minutes for each bug report. Some studies propose using feature selection approaches to increase the accuracy of automatic defect classification. This study presents an industry case study with automatic defect classification using feature selection approaches proposed in the literature. In addition, a change to the USES feature selection algorithm is proposed to improve its effectiveness, the new algorithm was denominated USES+. As a result, we have that the automatic feature selection approaches result in a greater effectiveness of the automatic classification of defect written in natural language. Finally, USES+ obtained a better effectiveness when compared to USES, even with significant differences. Thus, in general, automatic defect classification using feature selection approaches has shown to be promising with good effectiveness. In this way, automatic classification allows reduction of the cost of prioritization of defects, significantly improving the delivery of the essential corrections.

Keywords: Software Defects, Natural Language Processing, Machine Learning

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Revisões, inspeções e testes são tarefas de garantia de qualidade do processo de desenvolvimento de *software* que visam minimizar o número de defeitos encontrados no mesmo. No entanto, apesar de impactar diretamente na qualidade do *software*, os defeitos também carregam muitas informações que podem ser medidas, analisadas e usadas para auxiliar no objetivo de atingir as metas de garantia de qualidade (DENGER; ESP, 2005). Portanto, entender e gerenciar esses defeitos é uma importante tarefa no processo de manutenção de *software* e melhoria contínua da qualidade do *software* (THUNG et al., 2012; XIA et al., 2014; LI et al., 2010; PATIL, 2017).

Entender os defeitos encontrados no *software* pode ajudar os desenvolvedores, testadores e gestores de projetos de *software* a conduzirem uma análise mais profunda de suas causas e tomar as medidas apropriadas para minimizar sua recorrência, como aplicar mais recursos humanos, refatorar código-fonte, treinar desenvolvedores, entre outros (XIA et al., 2014; PATIL, 2017).

A classificação de defeitos usando um modelo de classificação apropriado, como classificação ortogonal de defeitos (ODC) (CHILLAREGE et al., 1992), classificação padrão IEEE para anomalias de *software* (IEEE, 2010) e classificação HP (GRADY, 1992), pode ajudar a agilizar o processo de gerenciamento de defeitos. Além disso, a classificação de defeitos pode trazer outros benefícios, como a possibilidade de identificar padrões nos relatórios de defeitos, fornecer uma análise mais rápida da causa raiz do defeito e restringir o escopo do teste de *software* (PATIL, 2017).

Os relatórios de defeitos são comumente classificados de várias maneiras, como em termos de gravidade, prioridade, módulo de função afetado e *status* de defeito, principalmente para fins de triagem de defeitos e atribuição de tarefas de correção de defeitos.

Uma maneira de realizar essa classificação é por meio de inspeção manual, na qual os analistas precisam ler e compreender a descrição textual de cada defeito relatado e, em seguida, classificá-los de acordo com o modelo utilizado. Essa abordagem, apesar de ter uma

boa efetividade, é um processo humanamente custoso (HUANG et al., 2015; ZIBRAN, 2016; LIMSETTHO et al., 2014). Segundo Herzig et al. (2013), classificar os defeitos utilizando um relatório de defeitos escrito em linguagem natural exige que o analista gaste uma média de 6 minutos e, mesmo com boa precisão, é um processo propenso a falhas humanas uma vez que a perspectiva humana pode afetar o seu resultado.

Por causa desse fato, muito estudos foram realizados propondo abordagens que possibilitem classificar os defeitos automaticamente a partir do relatório de defeito, sem que haja a necessidade da intervenção humana e mantendo a boa efetividade da classificação (PATIL, 2017; XIA et al., 2014; LIMSETTHO et al., 2014; THUNG et al., 2012; ZIBRAN, 2016; HUANG et al., 2015).

Das abordagens propostas, destacam-se, em termos de efetividade, ou seja, que conseguem classificar de maneira correta um maior número de defeitos, algumas abordagens propostas que com o objetivo de melhorar a precisão da classificação automática de defeitos por meio relatórios escritos em linguagem natural, utilizam técnicas de seleção de atributos com o objetivo de selecionar apenas as palavras mais significativas do relatório na classificação automática do defeito (XIA et al., 2014; HUANG et al., 2015).

1.1 CONTEXTO

Este estudo surge com base no contexto e necessidade da empresa DSIN Tecnologia da Informação[1], que possui atuação focada em desenvolvimento de *software* e tem sua sede na cidade de Marília, interior do estado de São Paulo. A empresa desenvolve sistemas para o gerenciamento das infrações municipais de trânsito, e sua atuação compreende desde o lançamento das infrações pelos agentes atuadores até sua regularização, contemplando seu processamento, notificação e comunicação aos órgãos responsáveis. Atualmente, a empresa possui 34 clientes, distribuıdos nos estados de São Paulo, Rio de Janeiro, Paraná, Mato Grosso, Mato Grosso do Sul e Bahia.

Em janeiro de 2019, a empresa possuía 14 *software* em desenvolvimento/manutenção, com uma média 120 novos defeitos por mês. A fim de gerenciar o ciclo de desenvolvimento do produto, incluindo o uso como repositório de defeitos, o *software* Jira [2] é utilizado.

Com um fluxo mensal alto de defeitos, no processo de planejamento da resolução dos defeitos, os analistas utilizam o atributo prioridade preenchido nos relatórios de defeitos,

[1] http://www.dsin.com.br
[2] https://br.atlassian.com/software/jira

com o objetivo de identificar os defeitos que mais afetam gravemente a operação da empresa e consequentemente terão maior prioridade de resolução.

No Jira, os relatórios de defeito possuem o atributo prioridade, e são disponibilizados os seguintes valores para sua identificação:

- *Blocker*: bloqueia o uso do sistema.
- *Critical*: falhas, perda de dados ou problemas com o uso da memória.
- *Major*: perda de funcionalidade vital do sistema.
- *Minor*: perda de funcionalidade de menor relevância ou de fácil *workaround*.
- *Trivial*: simples de resolver, como uma ortografia ou cor errada.

Por meio dos valores contidos no atributo prioridade dos relatórios, os analistas responsáveis definem quais defeitos terão prioridade de resolução. No entanto, nem sempre o atributo prioridade está preenchido e algumas vezes foi preenchido incorretamente devido a inexperiência de quem esta preenchendo o relatório de defeito.

A falta de garantia do preenchimento correto desse atributo prejudica o processo de priorização dos defeitos, muitas vezes ocasionando em demora de resolução de defeitos que ocorrem em funcionalidades vitais do sistema. Além disso, como a classificação envolve muito esforço manual, o cronograma das equipes de desenvolvimento geralmente não possuem tempo suficiente para a classificação correta dos defeitos (PATIL, 2017).

Devido a essa dificuldade, torna-se necessária a busca de alternativas que possibilitem a classificação automática dos defeitos de acordo com o esperado pelos analistas responsáveis. Nesse contexto, o estado da arte reporta o uso de diferentes técnicas de extração e seleção de atributos visando realizar essa tarefa com maior efetividade.

Dessa forma, este trabalho propõem analisar a efetividade de diferentes técnicas de seleção de atributos na classificação automática da prioridade de defeitos a partir de relatórios de defeitos escritos em português. Além disso, é proposta uma alteração em uma abordagem encontrada na literatura visando melhorar sua efetividade.

1.2 OBJETIVOS

O objetivo geral desse estudo é analisar a efetividade da utilização de técnicas de processamento de linguagem natural visando classificar automaticamente a prioridade de defeitos

a partir de relatórios de defeitos escritos em português. Para isso, espera-se alcançar os seguinte objetivos específicos:

- Coletar, analisar e avaliar as abordagens de automatização da classificação de defeitos existentes na literatura.
- Coletar, analisar e classificar os relatórios de defeitos do repositório. Ao final desse item, tem-se construído um *dataset* de relatórios de defeitos escritos em português.
- Executar, melhorando se possível, as abordagens de classificação automática encontradas na literatura.
- Coletar, avaliar e reportar a efetividade das abordagens executadas no *dataset*.

Por meio desse objetivo espera-se reduzir o custo de priorização dos defeitos, melhorando significativamente o tempo necessário para que as correções essenciais cheguem aos clientes finais do *software* da empresa em questão.

1.3 ESTRUTURA

Esse estudo foi estruturado da seguinte forma: o Capitulo 2 descreve a fundamentação teórica onde é detalhada a revisão da literatura realizada com o objetivo de encontrar e selecionar as principais abordagens propostas na literatura para a classificação automática de defeitos; o Capitulo 3 detalha uma avaliação realizada com as principais abordagens de classificação automática de defeitos selecionadas; por fim, o Capitulo 4 contém as considerações finais desse estudo.

2 ABORDAGENS DE CLASSIFICAÇÃO AUTOMÁTICA DE DEFEITOS

Nesse capítulo é detalhada a revisão sistemática da literatura (SLR do inglês *Systematic Literature Review*) realizada nesse trabalho com o intuito de colher a fundamentação teórica necessária para a execução desse estudo. Como a proposta deste trabalho visa atender uma demanda bem delimitada gerada por uma empresa, possibilitar a classificação automática de prioridade dos defeitos, buscou-se nessa SLR quais as principais abordagens utilizadas para a classificação automática de defeitos, principalmente com o objetivo de estimar a viabilidade de sua aplicação no ambiente real da empresa.

2.1 REVISÃO SISTEMÁTICA DA LITERATURA

De maneira geral, uma SLR é um estudo secundário que utiliza estudos primários de uma determinada área de pesquisa para levantar evidências de pesquisa e construir conhecimento (KITCHENHAM, 2004), fornecendo assim, uma visão geral sobre a área de pesquisa, levantando dados sobre a frequência de publicações, principais pesquisadores, principais locais de pesquisa, tipos de pesquisa, dentre outros (PETERSEN et al., 2008). Tal método de pesquisa mostrou-se capaz de possibilitar uma revisão formal, rigorosa, confiável e passível de auditoria.

Segundo Kitchenham (2004) uma Revisão Sistemática deve seguir as seguintes etapas:

- Planejamento: No planejamento são definidos o objetivo da SLR, as questões de pesquisa que a mesma visa responder, os critérios de seleção dos estudos que farão parte do estudo e as fontes de estudos primários utilizadas;
- Execução: Após o planejamento, é iniciada a etapa de execução da SLR. Nessa etapa são executadas as seguintes tarefas: busca dos estudos primários nas bases selecionadas, seleção dos estudos primários de acordo com os critérios de seleção, avaliação do estudo primário e extração das informações de interesse de acordo com as questões de pesquisa;
- Análise: Por fim, na última etapa é realizada a interpretação e documentação dos resultados da SLR. As informações de interesse são extraídas e é realizada a compilação das

mesmas de uma forma que facilite o acesso a possíveis interessados.

Nas próximas subsecções são detalhadas cada uma dessas etapas da SLR.

2.1.1 PLANEJAMENTO

2.1.1.1 OBJETIVO

Esse estudo teve como objetivo identificar quais as abordagens de classificação automática de defeitos são utilizadas na literatura, destacando suas aplicações e resultados. Além disso, é esperado que os resultados desse estudo auxiliem possíveis estudos futuros nessa área de pesquisa, contribuindo assim, para a evolução dessa área de pesquisa.

2.1.1.2 QUESTÕES DE PESQUISA

As questões de pesquisa servem como base para a definição da *string* de busca que será utilizadas na pesquisa de estudos primários nas fontes definidas. É também de acordo com as questões de pesquisa que são definidos os critérios de inclusão e exclusão dos estudos.

As questões de pesquisa que esse estudo visou responder foram:

- Quais as principais abordagens utilizadas para classificação automática de defeitos ? (Q1)
- Quais os domínios em que a abordagem foi utilizada e quais seus resultados ? (Q2)

2.1.1.3 SELEÇÃO DOS ESTUDOS PRIMÁRIOS

Para a seleção dos estudos primários foi utilizada a seguinte *string* de busca:

(classification OR categorization OR taxonomy) AND (auto OR automatic) AND (defect OR defects OR bugs) AND (software OR softwares).

2.1.1.4 CRITÉRIOS DE INCLUSÃO E EXCLUSÃO

Visando a garantir que somente estudos primários que contribuam para as questões de pesquisa sejam selecionados, foram definidos alguns critérios de inclusão e exclusão.

Os critérios de inclusão foram:

- Estudos que propõem uma abordagem de classificação automática de defeitos de *software*.
- Estudos que utilizam uma abordagem de classificação automática de defeitos de *software*.

Por outro lado, os critérios de exclusão foram:

- Estudos que não mencionem claramente, no resumo, a classificação automática de defeitos em *software*.
- Estudos repetidos.
- Estudos em idioma diferente de inglês.

2.1.1.5 SELEÇÃO DE FONTES

As fontes de estudos foram definidas de acordo com as bases de estudos que possuem acesso disponibilizado aos alunos da Universidade Tecnológica do Paraná (UTFPR) e que possuíam estudos que correspondessem a *string* de busca utilizada. As fontes que possuíam esses dois requisitos foram: *IEEE Xplore* [1], *ACM Digital Library* [2], *Springer Link* [3], *Google Scholar* [4].

2.1.2 EXECUÇÃO DO ESTUDO

Finalizado o planejamento, a execução do estudo sistemático seguiu as seguintes etapas sequenciais:

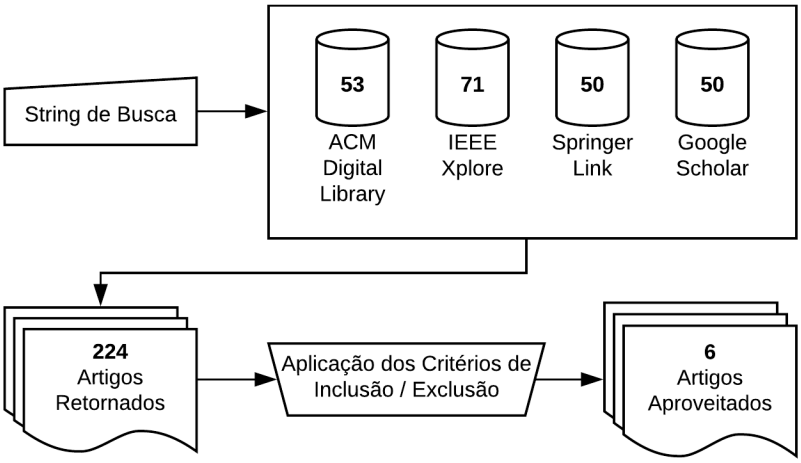
1. Foi executada a busca nas bases de fontes de estudos selecionadas, onde os estudos retornados foram analisados. Nessa etapa, nas fontes *IEEE Xplore*, *ACM Digital Library* e *Springer Link* todos os estudos retornados foram analisados e na fonte *Google Scholar* foram analisados os 50 principais estudos retornados;
2. A partir da leitura do título e do resumo, os estudos foram incluídos ou excluídos de acordo com os critérios definidos anteriormente;

[1] http://ieeexplore.ieee.org/Xplore/home.jsp
[2] http://dl.acm.org
[3] https://link.springer.com
[4] https://scholar.google.com.br

3. Os estudos selecionados foram lidos na íntegra e foram extraídas as informações necessárias para as respostas das questões de pesquisa definidas anteriormente.

A Figura 2.1 apresenta um resumo com o número de estudos retornados em cada base e o número de estudos aproveitados após a aplicação dos critérios de inclusão e exclusão.

Figura 2.1: Processo da SLR



Fonte: Autoria Própria

2.1.3 ANÁLISE DOS RESULTADOS

Visando compreender a popularização da área estudada, os estudos aproveitados foram classificados de acordo com o seu ano de publicação. Os estudos selecionados foram publicados entre os anos de 2011 e 2017 e o ano com a maior quantidade de estudos publicados foi 2014.

Além disso, foram levantadas as abordagens utilizadas por cada estudo. São elas:

- Seleção de Atributos Baseada em Conjunto Fuzzy (XIA et al., 2014);
- Conhecimento de Domínio e Geração de Pseudo Instâncias (HUANG et al., 2015);
- Análise do Código Fonte de Correção Utilizando Árvore Sintática Abstrata (THUNG et al., 2012);
- Atribuição Latente de Dirichlet Rotulada (ZIBRAN, 2016);
- Análise Semântica Explicita (PATIL, 2017);
- Processo de Dirichlet Hierárquico (LIMSETTHO et al., 2014).

Na próxima seção são detalhadas cada uma das abordagens identificadas nessa SLR.

2.2 ABORDAGENS

Nessa seção são detalhadas as diferentes abordagens de classificação automática de defeitos identificadas nessa SLR.

O objetivo dessa seção é responder a Q1:

- Quais as principais abordagens utilizadas para classificação automática de defeitos ?

E fornecer o embasamento necessário para responder a Q2:

- Quais as aplicações em que a abordagem foi utilizada e quais seus resultados ?

As abordagens propostas pelos estudos utilizam *machine learning* e podem ser divididas em duas categorias distintas: aprendizagem supervisionada e aprendizagem não supervisionada. Apesar das diferenças, comumente a todas as abordagens são aplicadas técnicas de preprocessamento de texto para preparar os dados de entrada para cada abordagem. Nas próximas subseções é detalhado o processo de preprocessamento de texto e as abordagens propostas pelos estudos.

2.2.1 PREPROCESSAMENTO DE TEXTO

A etapa de preprocessamento de texto é importante para transformar o texto em algo processável pelo algoritmo que será utilizado nas etapas posteriores da classificação automática. Esse processo é realizado conforme as seguintes etapas:

1. Primeiramente, são extraídos os textos de vários campos do relatório de defeito (por exemplo, título, descrição, resumo e comentários).
2. Em seguida, cada texto extraído na etapa anterior é transformando em *tokens* (ocorre a quebra do texto em palavras, frases, símbolos ou outros elementos significativos), esse processo é conhecido como *Tokenization*.
3. Após, no processo conhecido como *Stop Word Removal*, as palavras que são utilizadas com frequência mas possuem pouco significado são removidas (por exemplo, as palavras eu, e, você, ele, o).

4. Posteriormente, cada uma das palavras é reduzida a sua forma raiz (por exemplo, as palavras “pesca”, “pescador” e “pescada” seriam todas reduzidas para “peixe”), em um processo denominado *Stemming*.

5. Por fim, as palavras restantes podem ser representadas de diversas formas sendo as mais comuns: *Bag of Words* e Espaço vetorial.

    - *Bag of Words*, onde as palavras são agrupadas em uma lista desordenada de palavras, desconsiderando a estrutura gramatical
    - Espaço vetorial, onde as palavras são inseridas em um vetor de atributos do texto, em que cada atributo é uma palavra (termo) e o valor do atributos é um peso de termo. O termo peso pode ser:
        - Um valor binário, com 1 ou 0 indicando ou não a ocorrência do termo.
        - Um valor de frequência, indicando quantas vezes o termo ocorreu no documento.
        - Um valor da frequência do termo–inverso da frequência nos documentos (TF-IDF do inglês *term frequency–inverse document frequency*), indicando a importância de uma palavra para um documento em um corpus, seu valor aumenta proporcionalmente ao número de vezes que uma palavra aparece no documento e é compensada pelo número de documentos no corpus que contém a palavra, o que ajuda a ajustar o fato de que algumas palavras aparecem com mais frequência no geral.

A Tabela 2.1 contém como cada um dos processos de preprocessamento de texto foi utilizada por cada uma das abordagens.

2.2.2 APRENDIZAGEM SUPERVISIONADA

Aprendizagem supervisionada é uma abordagem de *machine learning* em que é necessário um humano para atuar como um guia visando ensinar ao algoritmo quais conclusões ele deve apresentar, este tipo de aprendizado requer que as saídas possíveis do algoritmo já sejam conhecidas e que os dados usados para treinar o algoritmo já estejam rotulados com respostas corretas. A grande desvantagem dessa abordagem é o fato da rotulação dos dados de treinamento ser considerado humanamente custoso e propenso a falhas humanas.

Nessa categoria foram enquadrados as seguintes abordagens:

Tabela 2.1: Preprocessamento de texto em cada uma das abordagens.

Estudo	Campos Utilizados	*Tokenization*	*Stop Word*	*Stemming*	Representação
Patil (2017)	-	Sim	-	-	TF-IDF
Xia et al. (2014)	Resumo e Descrição	Sim	Sim	Sim	*Bag of Words*
Limsettho et al. (2014)	Título, Descrição e Comentários	Sim	Sim	Não	TF-IDF
Thung et al. (2012)	Título e Descrição	Sim	Sim	Sim	*Bag of Words*
Zibran (2016)	Título e Comentários	Sim	Sim	-	-
Huang et al. (2015)	Resumo e Descrição	Sim	Não	Sim	Binário

Fonte: Autoria Própria

- Seleção de Atributos Baseada em Conjunto Fuzzy (XIA et al., 2014).
- Conhecimento de Domínio e Geração de Pseudo Instâncias (HUANG et al., 2015).
- Análise do Código Fonte de Correção Utilizando Árvore Sintática Abstrata (THUNG et al., 2012).
- Atribuição Latente de Dirichlet Rotulada (ZIBRAN, 2016).

Os algoritmos de aprendizagem supervisionada funcionam basicamente conforme as seguintes etapas sequenciais:

1. Primeiramente, os dados utilizados são classificados manualmente de acordo com as categorias predefinidas pelo estudo.
2. Após a classificação manual dos dados, é realizado o preprocessamento do texto conforme descrito na Seção 2.2.1.
3. Com os dados em uma representação compatível, é escolhido um algoritmo de aprendizagem para o treinamento do classificador, por exemplo: máquina de vetores de suporte (SVM, do inglês *Support Vector Machine*), classificação naïve bayes e árvores de decisão.
4. Finalizado o treinamento, o classificador é utilizado para classificar cada registro do conjunto de teste. É atribuído ao registro do conjunto de teste a categoria que o classificador retornou.

Nas próximas subseções são detalhadas as diferenças entre as abordagens dessa categoria.

2.2.2.1 SELEÇÃO DE ATRIBUTOS BASEADA EM CONJUNTO FUZZY

Xia et al. (2014) propõem classificar automaticamente um defeito com base nas condições de gatilho para a ocorrência do mesmo, para isso foram definidas duas categorias: *Bohrbug* que refere-se a um defeito que pode ser facilmente isolado e sua ativação e propagação de erros são simples, e, *Mandelbug* que identifica um defeito cuja sua ativação e propagação são complexas, ou seja, um defeito de difícil reprodução.

Ao final do preprocessamento dos dados é retornado um número excessivo de *tokens* e isso pode causar problemas na previsão da categoria, por isso, é proposta uma abordagem de seleção de atributos denominada USES (do inglês *fUzzy Set based fEature Selection algorithm*) com o objetivo de selecionar o subconjunto de atributos textuais mais relevantes visando melhorar o desempenho de previsão da categoria.

O USES primeiramente calcula o *score* de afinidade entre uma palavra e uma categoria, que é calculado conforme o aparecimento ou não da palavra nos registros previamente classificados. Após o cálculo do *score* de afinidade, o *score* de seleção de atributos é calculado pela diferença entre o *score* de afinidade da palavra na categoria A e o *score* de afinidade da palavra na categoria B. Com o *score* de seleção de atributos finalizado, são selecionados *n* atributos com os maiores *scores* de seleção positivos e os *n* atributos com os maiores *scores* de seleção negativos.

A partir disso, o algoritmo itera muitas vezes e em cada iteração, seleciona aleatoriamente um subconjunto de *n* atributos do total de atributos, e um novo classificador é construído com os *n* atributos selecionado. Nessa etapa, o conjunto de dados de treinamento é dividido em dois subconjuntos: um é usado para selecionar *n* atributos e outro é usado para avaliar o desempenho dos atributos selecionados. Ao fim, os atributos com o melhor desempenho são selecionados.

2.2.2.2 CONHECIMENTO DE DOMÍNIO E GERAÇÃO DE PSEUDO INSTÂNCIAS

Huang et al. (2015) propõem a classificação automática do atributo “impacto” do ODC, para isso são utilizadas seis categorias: capacidade, segurança, desempenho, confiabilidade, requisitos e usabilidade.

Após a etapa de preprocessamento, são construídos os classificadores utilizando SVM como o algoritmo de aprendizagem para o treinamento. Neste estudo, cada classificador construído representou uma classe do ODC, por exemplo, foi treinado um classificador para determinar se um registro pertence a classe “confiabilidade”, outro classificador para determinar se

um registro pertence a classe “usabilidade”, e assim por diante. Sendo assim, o número final de classificadores é igual ao número de categorias de defeitos.

Após o treinamento, os classificadores resultantes são utilizados para classificar cada registro do conjunto de teste. É atribuído ao registro do conjunto de teste a classe cujo classificador retorna o maior valor absoluto entre o conjunto de valores retornados por todos os classificadores.

Com o objetivo de melhorar a precisão da classificação, os autores adicionaram ao sistema de classificação a estrutura de anotações relevantes que é composta 3 extensões. Uma anotação relevante pode ser uma palavra ou frase relevante na descrição do defeito que fornece dicas para que os analistas decidam a qual categoria ODC pertence determinado defeito.

A primeira extensão envolve o aumento do conjunto de treinamento com instâncias de treinamento mais negativas geradas a partir das anotações e modificação do procedimento de aprendizagem para explorar essas instâncias de treinamento adicionais como pseudo instâncias.

Na segunda extensão são exploradas as anotações relevantes para gerar atributos adicionais para treinar o classificador, essa extensão é realizada da seguinte maneira: primeiramente, foram coletadas todas as anotações relevantes dos registros de defeitos no conjunto de treinamento. Em seguida, foi criado um atributo de cada anotação relevante e foi aumentado o peso desse atributo.

A última extensão, que é uma maneira de combater a dispersão dos dados, é a aplicação do conhecimento de domínio para identificar as anotações relevantes que são sinônimas. Para isso, os autores coletaram todas as anotações relevantes dos registros de treinamento e pediram a um analista que os dividissem de modo em que cada agrupamento contenha apenas anotações relevantes sinônimas, atribuindo um *id* exclusivo a cada agrupamento. Os autores exploraram o conhecimento de domínio da seguinte forma: verificaram se uma anotação relevante está presente no registro de defeito correspondente, em caso afirmativo, é adicionado um atributo que corresponde ao id do agrupamento que contém a anotação relevante.

2.2.2.3 ANÁLISE DO CÓDIGO FONTE DE CORREÇÃO UTILIZANDO ÁRVORE SINTÁTICA ABSTRATA

Thung et al. (2012) realizaram um estudo com o objetivo de classificar automaticamente um defeito com base no conteúdo do relatório do defeito e as alterações feitas no código fonte de correção do mesmo. As categorias foram definidas com base no atributo “tipo” do ODC, no entanto, as categorias propostas inicialmente pelo ODC foram reagrupadas a fim de

diminuir a quantidade de categorias utilizadas na classificação, ao final foram definidas as seguintes categorias: controle e fluxo de dados, estrutural e não funcional.

Visando converter os dados em uma representação adequada, os autores realizaram duas etapas de preprocessamento: preprocessamento do texto, realizado de acordo com o descrito na Subseção 2.2.1. e preprocessamento do código.

O preprocessamento de código é iniciado pela extração do código de correção do defeito, após isso, o código é convertido em árvores sintáticas abstratas (ASTs do inglês *Abstract Syntax Tree*), que são uma representação simplificada da estrutura semântica do código fonte. Sendo assim, existem duas versões do código, uma antes da correção do defeito, e outra depois da correção.

Primeiramente, identifica-se as linhas alteradas nas duas versões do código executando um diff [5] que fornece as mudanças de código como um conjunto de linhas de código adicionadas e o conjunto de linhas de código excluídas. Após isso, são analisadas as duas versões do código antes e depois da correção para formar duas ASTs e, em seguida, identifica-se os nós em ambas ASTs que correspondem às linhas de código adicionadas e excluídas, e remove as árvores que os nós não estão diretamente relacionados às linhas de código adicionadas ou excluídas.

Finalizado o preprocessamento, o próximo passo é a extração de atributos que capturam as informações de vários elementos do código adicionado e do código excluído. As informações dos elementos que foram consideradas incluem atribuições, comentários, estrutura de *loop*, invocações de métodos, declaração de chamada, etc.

Após a extração das atributos do texto e do código, é construído o modelo de aprendizagem. Nesse estudo, foi utilizado o algoritmo de aprendizagem SVM [6].

2.2.2.4 ATRIBUIÇÃO LATENTE DE DIRICHLET ROTULADA

Zibran (2016) investigaram a viabilidade da aplicação da atribuição latente de Dirichlet rotulada (LLDA, do inglês *Labeled Latent Dirichlet Allocation*) na classificação automática de relatórios de defeitos em um conjunto fechado de 22 categorias. A LLDA é um modelador de tópico que restringe a atribuição latente de Dirichlet definindo uma correspondência entre os tópicos gerados e as categorias predefinidas (RAMAGE et al., 2009).

A primeira etapa do estudo, foi a construção do oráculo, utilizando um conjunto de dados do estudo que reuniu 428 defeitos de três projetos diferentes, Eclipse, GNOME e Python.

[5] http://www.gnu.org/software/diffutils/

[6] http://svmlight.joachims.org/svm$_{m}ulticlass.html$

Nesta etapa os defeitos foram rotulados manualmente com uma ou mais das 22 categorias. O conjunto de categorias utilizado foi o proposto por Giger et al. (2010) no seu estudo sobre defeitos de usabilidade de API.

Após a finalização da construção do oráculo, os autores iniciaram o processamento de linguagem natural utilizando os relatórios de defeitos. Para isso, aplicaram a LLDA, usando a *Stanford Topic Modeling Toolbox* [7], que é uma *toolbox* com os principais algoritmos de NLP, incluindo LLDA.

O preprocessamento do texto foi executado conforme descrito na Subseção 2.2.1. Após o preprocessamento é realizada a aplicação LLDA que consiste em duas etapas: aprendizagem e inferência (RAMAGE et al., 2009). No estágio de aprendizagem, a LLDA é aplicada em um conjunto de treinamento para a construção do modelo, que então é utilizado na fase de inferência para inferir a categoria de outro documento.

2.2.3 APRENDIZAGEM NÃO SUPERVISIONADA

Aprendizagem não supervisionada, também conhecida como agrupamento de documentos, é uma abordagem de *machine learning* em que a classificação é realizada sem referência a informações externas, ou seja, sem a necessidade de predefinir as categorias e categorizar o conjunto de treinamento, esse tipo de abordagem é considerado bem menos humanamente custoso que o anterior, porém os seus resultados são de mais difícil interpretação.

Nessa categoria foram enquadradas as seguintes abordagens:

- Análise Semântica Explícita (PATIL, 2017).
- Processo de Dirichlet Hierárquico (LIMSETTHO et al., 2014).

Nas próximas subseções são detalhadas as abordagens enquadradas nessa categoria.

2.2.3.1 ANÁLISE SEMÂNTICA EXPLÍCITA

Patil (2017) propõem uma abordagem que visa classificar automaticamente um defeito com base na similaridade entre a categoria do defeito e a descrição do defeito utilizando a Wikipédia como espaço conceitual. As categorias utilizadas foram: controle e fluxo de dados, estrutural e não funcional.

[7] https://nlp.stanford.edu/software/tmt

A análise semântica explícita (ESA do inglês *Explicit Semantic Analysis*) baseia-se na hipótese de que os humanos julgam a relação semântica de duas palavras ou dois documentos a nível conceitual (GABRILOVICH; MARKOVITCH, 2007). Na abordagem cada artigo da Wikipédia foi considerado a um conceito interpretável pelo homem, sendo assim, o espaço vetorial abrangido por todos os artigos na Wikipédia foi definido como o espaço conceitual da abordagem. A partir disso, utilizando a ESA, projetou-se os relatórios de defeitos neste espaço conceitual.

Cada relatório de defeito foi representado por meio de um vetor de artigos da Wikipédia utilizando ESA enquanto cada categoria de defeito também foi representada usando um vetor ESA. Com isso, um relatório de defeito é atribuído a categoria de defeito cuja a representação ESA é mais próxima da representação ESA do relatório de defeito.

2.2.3.2 PROCESSO DE DIRICHLET HIERÁRQUICO

Limsettho et al. (2014) propõem uma abordagem composta por três fases principais; Modelagem de Tópicos, Agrupamento e Rotulação de Grupos.

O processo de Dirichlet hierárquico (HDP, do inglês *Hierarchical Dirichlet Process*) foi escolhido como modelador de tópicos, devido à sua capacidade de inferir o número de tópicos automaticamente.

Na fase de Agrupamento os relatórios de defeitos no espaço vetorial do tópico são agrupados com outros semelhantes, os métodos de agrupamentos utilizados nesta etapa foram o algoritmo de Maximização de Expectativa (EM) e X-Means (PELLEG et al., 2000).

A fase de rotulação de grupos é divida em 5 etapas: Cálculo da Proporção Média do Tópico, Ranqueamento de Tópicos por sua Relevância e Seleção de Tópicos, Criação de Lista de Palavras Relevantes, *Chunking* e Criação de Lista de Rótulos.

Na etapa do Cálculo da Proporção Média do Tópico, as proporções médias de cada tópico são calculadas para cada grupo e também para todo o corpus.

A etapa de Ranqueamento de Tópicos por sua Relevância e Seleção de Tópicos visa classificar e selecionar tópicos relevantes baseados em sua classificação. O primeiro passo é encontrar a relação de cada tópico em cada grupo. Essa relação é calculada dividindo a proporção média do tópico de cada grupo com uma proporção média de corpus de documento inteiro. Após isso é realizado o ranqueamento de tópicos por sua relevância onde o objetivo é classificar o tópico com base em sua relevância. A relevância de cada tópico no grupo é determinada pela relação entre a ocorrência no grupo em comparação com a ocorrência no

corpus. Para finalizar essa etapa é realizada a seleção dos principais tópicos relevantes, onde foram selecionados os 20% maiores como tópicos relevantes para cada grupo. O resultado para cada grupo é um conjunto de tópicos considerados relevantes para esse grupo.

Na etapa de Criação de Lista de Palavras Relevantes é utilizada a lista de palavras mais importantes e os principais tópicos relevantes selecionados na etapa anterior. A pontuação da palavra *k* no grupo *j* é igual à proporção do tópico *i* no grupo *j* multiplicado pela frequência da palavra *k* no tópico *i* e dividido pela frequência total de todas as palavras principais no tópico i. Se a mesma palavra for encontrada em vários tópicos relevantes, a pontuação dessa palavra é a soma da pontuação de todos os tópicos relevantes. A ideia é que, se a palavra *k* estiver ocorrendo muito mais no grupo específico *j* do que em todo o corpus, então essa palavra *k* provavelmente será um bom candidato para uma palavra em uma frase que represente o conjunto

Na etapa de *chunking*, os autores utilizaram um *chunker* de processamento de linguagem natural para extrair frases significativas dos relatórios de cada grupo nesta fase. O processo de *chunking* neste estudo foi realizado utilizando o OpenNLP [8].

Por fim, na última etapa foi realizada Criação da Lista de rótulos, para isso foram utilizadas a lista de palavras relevantes e a lista de frases. Cada grupo é rotulado com cinco rótulos com a maior pontuação naquele grupo. Os rótulos são uma lista e a pontuação de cada rótulo é calculada depois de selecionar uma combinação de duas palavras: palavra k_1 e palavra k_2, da lista de palavras relevantes. Cada frase na lista que contém essas duas palavras é movida para a lista de rotulagem e sua pontuação é calculada. A pontuação da frase m no grupo *j* é igual à frequência da frase m no grupo *j* multiplicado pela pontuação média da palavra k_1 e palavra k_2 e dividida pelo comprimento da frase ao quadrado. Depois de verificar e calcular a pontuação para cada item na lista, a combinação de novas palavras é selecionada. Este processo é repetido até que cada combinação seja usada. Caso essa frase contenha mais de uma combinação, sua pontuação é a soma de toda a combinação existente nessa frase.

A ideia deste método de pontuação é que a frase que ocorre regularmente e consiste em palavras relevantes de pontuação alta deve ser um bom rótulo para o grupo.

2.3 COMPARAÇÃO DOS RESULTADOS

Por meio dessa SLR, espera-se mapear as diferentes abordagens com o objetivo de auxiliar trabalhos futuros nessa área de pesquisa e também pesquisas com a comparação das diferentes abordagens. Para isso, os estudos foram classificados de acordo a abordagem proposta

[8] https://opennlp.apache.org/

para a classificação automática de defeitos, o tamanho do *dataset* utilizado, as entradas utilizados, a quantidade de categorias e as métricas de avaliação, conforme exibido na Tabela 2.2 .

As abordagens de classificação automática de defeitos identificadas (questão de pesquisa Q1) foram:

- Seleção de Atributos Baseada em Conjunto Fuzzy (XIA et al., 2014).
- Conhecimento de Domínio e Geração de Pseudo Instâncias (HUANG et al., 2015).
- Análise do Código Fonte de Correção Utilizando Árvore Sintática Abstrata (THUNG et al., 2012).
- Atribuição Latente de Dirichlet Rotulada (ZIBRAN, 2016).
- Análise Semântica Explicita (PATIL, 2017).
- Processo de Dirichlet Hierárquico (LIMSETTHO et al., 2014).

Para responder a questão de pesquisa Q2, foram levantadas informações sobre a aplicação da abordagem utilizada pelos autores e seus resultados.

Com relação aos sistemas utilizados a grande maioria das abordagens utilizaram sistemas *open source* para a avaliação das mesmas sendo o Lucene o principal projeto utilizado, seguindo de perto pelo Mahout e OpenNLP, somente Huang et al. (2015) avaliou sua abordagem com um projeto de código proprietário cujo sua privacidade foi protegida pelo autor.

Huang et al. (2015) utilizou o menor *dataset* dentre as abordagens com 403 defeitos enquanto Limsettho et al. (2014) utilizou o maior com 2719 defeitos, a mediana do tamanho dos *datasets* foi de 655 defeitos e a moda 500 defeitos.

Como entrada para a abordagem, quase todos os estudos utilizaram o somente o relatório de defeito, somente Thung et al. (2012) utilizou, além do relatório de defeitos, o código fonte de correção do defeito.

Dos projetos que utilizaram aprendizagem supervisionada, a menor quantidade de categorias utilizadas foram 2 e a maior 22 categorias, tendo como mediana 4 categorias. Essa informação é relevante pois, segundo Thung et al. (2012) a classificação automática fica mais difícil à medida que o número de categorias aumenta.

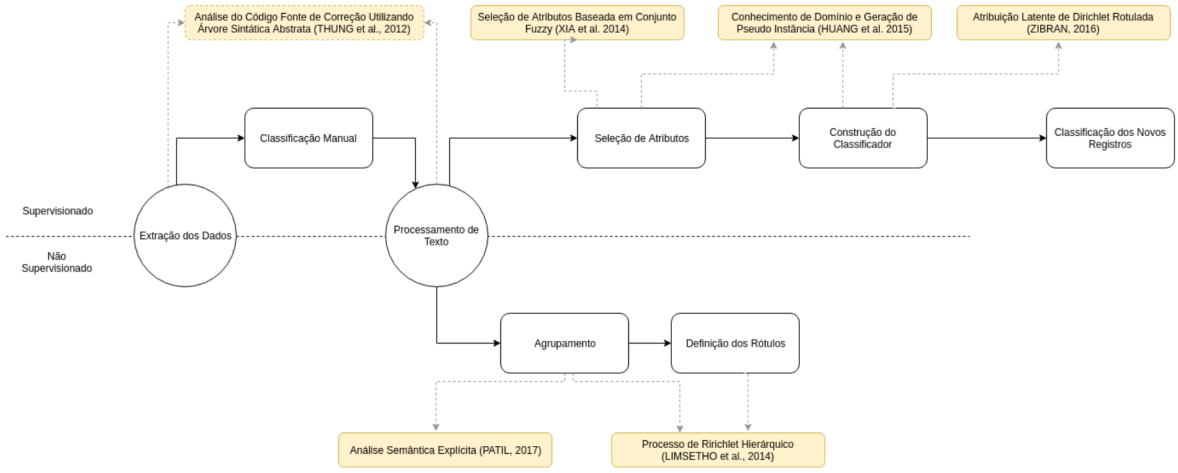
Tabela 2.2: Revisão sobre Classificação Automática de Defeitos.

Estudo	Sistemas Utilizados	Tamanho do *Dataset*	Entradas Utilizadas	Quantidade de Categorias	Precisão	*Recall*	*F-Measure*
Patil (2017)	Mahout, Lucene e OpenNLP	500	Relatório de Defeito	3 (Adaptação do atributo "tipo" do ODC)	0,487	0,489	0,487
Thung et al. (2012)	Mahout, Lucene e OpenNLP	500	Relatório de Defeito e Código Fonte	3 (Adaptação do atributo "tipo" do ODC)	0,690	0,701	0,690
Xia et al. (2014)	Linux, Mysql, Apache HTTPD e AXIS	809	Relatório de Defeito	2 (Bohrbug e Mandelbug)	0,721	-	0,617
Limsettho et al. (2014)	Lucene, JCR e HTTPClient	2719	Relatório de Defeito	Indefinida	0,628	-	0,585
Zibran (2016)	GNOME, Eclipse e Python	1138	Relatório de Defeito	22 (Defeitos de usabilidade da APIs)	0,344	0,449	0,356
Huang et al. (2015)	Empresa P	403	Relatório de Defeito	6 (Atributo "impacto" do ODC)	0,716	0,514	0,565

Fonte: Autoria Própria

Com relação as abordagens propostas, temos que 4 utilizaram aprendizagem de máquina supervisionado e 2 utilizaram aprendizagem não supervisionado. As abordagens foram realizadas em diferentes etapas do processo de aprendizagem de máquina. Das abordagens de aprendizagem supervisionada, Thung et al. (2012) focou nas etapas de extração de dados e processamento de texto, Xia et al. (2014) propôs um algoritmo da etapa de seleção de atributos, enquanto Huang et al. (2015) teve como objetivo otimizar as etapas de seleção de atributos e construção do classificador, por fim Zibran (2016) teve sua abordagem na etapa de construção do classificador. Das abordagens de aprendizagem não supervisionada, Patil (2017) focou na etapa de agrupamento e Limsettho et al. (2014) trabalhou nas etapas de agrupamento e definição dos rótulos. A Figura 2.2 contém um resumo das etapas de cada tipo de aprendizagem e a forma que as abordagens se relacionam com elas.

Figura 2.2: Resumo das Abordagens.



Fonte: Autoria Própria

Analisando os resultados, dentre as abordagens que utilizaram aprendizado supervisionado, duas abordagens propuseram melhorias no método de seleção de atributos para a *machine learning*, no caso, a seleção das palavras mais significativas do relatório de defeito. Nessas abordagens a seleção de atributos aumentou a eficiência da classificação, mostrando ser um recurso importante na classificação automática de relatórios escritos em linguagem natural.

Por outro lado, a comparação da efetividade entre as abordagens não pode ser realizada pois o resultado é dependente de diversos fatores como, por exemplo, definição das categorias e do *dataset* escolhido. Nesse sentido, apenas Patil (2017) e Thung et al. (2012) usaram o mesmo *dataset* e categorias e conforme já comparado por Patil (2017), embora a precisão da análise semântica explicita seja um menor a abordagem é promissora devido não precisar de dados previamente categorizados.

2.4 AMEAÇAS A VALIDADE

Apesar do planejamento para execução da SLR é possível que existam estudos primários que deveriam ser selecionados mas não foram selecionados, seja por falhas na avaliação dos autores ou por estarem presentes em outras bases de artigos diferentes das utilizadas utilizadas.

Para tentar minimizar as possíveis falhas na avaliação todo o processo da SLR foi realizado por dois pesquisadores e para minimizar a falta de algum estudo de outra fonte foi utilizado o *google scholar* que tem como característica reunir estudos de diversas fontes.

Além disso, como trabalhos futuros, seria importante que essa SLR seja complementada por uma pesquisa utilizando *snowballing* que poderá trazer novos estudos que contribuam com o estado da arte atual da área de pesquisa. No entanto, apesar dessa possibilidade, a inclusão de novos estudos de forma alguma invalidaria os resultados relatados por essa SLR.

3 CLASSIFICAÇÃO AUTOMÁTICA DE DEFEITOS UTILIZANDO SELEÇÃO DE ATRIBUTOS

De acordo com o resultado da revisão sistemática detalhada no Capítulo 2, as abordagens com melhores resultados foram as que propuseram melhorias no método de seleção de atributos para a *machine learning*, no caso, a seleção das palavras mais significativas do relatório de defeito. Dessa forma, com o objetivo de possibilitar a classificação automática da prioridade dos defeitos com boa efetividade, as abordagens de seleção de atributos foram implementadas e avaliadas utilizando um dataset real da indústria, além disso, foi proposta uma melhoria do algoritmo de seleção de atributos USES.

Neste capítulo são detalhadas as abordagens utilizadas, a avaliação realizada, seus resultados e a discussão comparando as abordagens e a efetividade da utilização das mesmas na classificação automática de defeitos.

3.1 USES

Nessa abordagem, primeiramente é calculado o *score* de afinidade entre categoria e palavra, que é calculado de acordo com a existência ou não da palavra nos registros previamente classificados. A Equação 3.1 é utilizada para calcular o *score* de afinidade, onde $n_{c,w}$ representa o número de defeitos na categoria c cuja a sua descrição contém a palavra w, n_c representa o número de defeitos da categoria c e n_w é o número de defeitos que contém a palavra w.

$$Aff(c,w) = \frac{n_{c,w}}{n_c + n_w - n_{c,w}} \tag{3.1}$$

Após o cálculo do *score* de afinidade, o *score* de seleção de atributos é calculado pela diferença entre o *score* de afinidade da palavra na categoria A e o *score* de afinidade da palavra na categoria B. A Equação 3.2 demonstra esse cálculo. Com o *score* de seleção de atributos finalizado, os n atributos com maior pontuação positiva e os n atributos com maior pontuação negativa são selecionados como um subconjunto de atributos.

$$Atributo(w) = Aff(A,w) - Aff(B,w) \tag{3.2}$$

Após isso, o algoritmo itera várias vezes e em cada iteração seleciona aleatoriamente um subconjunto de n atributos do subconjunto de atributos. Nesta etapa, o conjunto de dados é dividido em dois subconjuntos: um é usado para construir o classificador e o outro para avaliar o desempenho com os atributos selecionados. No final do número de iterações predeterminadas ao algoritmo, os atributos com o melhor desempenho são selecionados. O Algoritmo 1 contém o algoritmo completo utilizado por essa abordagem.

Algorithm 1 Abordagem USES

Entrada:
D: Conjunto de relatórios de defeitos
p%: Porcentagem de atributos a serem selecionados
sp%: Porcentagem de relatórios de defeitos utilizada para a construção do classificador
ITER: Número de Iterações
Saída:
selectedFeatures: Atributos selecionados (ou seja, as palavras que irão compor o dicionário)
Método:
Let N: Número total atributos (ou seja, palavras únicas) em D
Let l = N x p, o número de atributos que serão selecionados%;
Calcular o *score* de seleção de atributos para cada palavra em D;
candPos = l palavras com os maiores *score* de seleção de atributos positivos de D;
candNeg = l palavras com os maiores *score* de seleção de atributos positivos de D;
Unir candPos e candNeg em um único conjunto;
Dividir D em dois subconjuntos D_{build} e $D_{validate}$;
Let bestFMeasure = 0;
Let iter = 0;
Let selectedFeatures = { };
while iter $<$ ITER **do**
    Selecionar aleatoriamente um subconjunto de palavras l (T_{tmp}) do conjunto composto por candPos e candNeg;
    Construir um classificador de D_{build} com base nas l palavras selecionadas;
    Avaliar a F-Measure F_{tmp} do classificador usando $D_{validate}$.
    if Ftmp $>$ bestFMeasure **then**
        bestFMeasure = F_{tmp};
        selectedFeatures = T_{tmp};
    end if
    iter = iter + 1;
end while
return selectedFeatures;

Xia et al. (2014) avaliaram abordagem através de um experimento com 4 conjuntos de dados: Linux, MySQL, Apache HTTPD e AXIS, contendo um total de 809 relatórios de defeitos. A abordagem resultou em precisão média de 0,721 e F-Measure média de 0,617.

3.2 USES+

A fim de melhorar a efetividade do USES, foi proposta uma alteração no seu algoritmo e essa alteração também foi utilizada na avaliação. A principal intenção dessas mudanças é realizar a seleção das palavras mais relevantes de cada conjunto de atributos, evitando a aleatoriedade na formação do conjunto de atributos candidatos.

Para isso, foi alterada a equação que calcula o *score* de afinidade entre uma categoria e uma palavra. No USES+, o *score* é simplesmente a relação entre o número de atributos de uma categoria com uma determinada palavra e a quantidade total de atributos com essa palavra. Essa alteração no USES foi motivada por dois preceitos que não eram contemplados na equação original:

- Uma palavra que aparece apenas em atributos da mesma categoria é uma palavra com *score* de afinidade entre categoria e palavra alta.
- Uma palavra que aparece em todos os atributos, independentemente da categoria, é uma palavra com *score* de afinidade entre categoria e palavra baixa.

A Equação 3.3 contém a equação resultante das alterações realizadas, onde $n_{c,w}$ representa o número de defeitos na categoria c cuja sua descrição contém a palavra w e n_w é o número de defeitos que contém a palavra w.

$$Aff(c,w) = \frac{n_{c,w}}{n_w} \tag{3.3}$$

Depois de ordenar as palavras por sua *score* de afinidade, o algoritmo itera várias vezes e, a cada iteração, seleciona uma certa quantidade de palavras de cada registro de defeitos. Por exemplo, na iteração 1, uma palavra é selecionada de cada registro de defeito e o subconjunto de atributos candidatos é construído pelo número total de palavras selecionadas, na iteração 2, duas palavras são selecionadas e assim por diante.

Com o conjunto de atributos candidatas construído, ele é separado em dois subconjuntos: um é usado para construir o classificador e o outro para avaliar seu desempenho. No final do número de iterações, os atributos com o melhor desempenho são selecionados. O Algoritmo 2 contém o algoritmo completo utilizado por essa abordagem.

Algorithm 2 USES+

1: **Entrada:**
2: D: Conjunto de relatórios de defeitos
3: sp%: Porcentagem de relatórios de defeitos utilizada para a construção do classificador
4: maxWords: Número máximo de palavras selecionadas de cada registro
5: **Saída:**
6: selectedFeatures: Atributos selecionados (ou seja, as palavras que irão compor o dicionário)
7: **Método:**
8: Calcular o *score* de seleção de atributos para cada palavra em D;
9: Dividir D em dois subconjuntos D_{build} e $D_{validate}$;
10: Let bestFMeasure = 0;
11: Let numberWords = 1;
12: Let selectedFeatures = { };
13: **while** numberWords $<=$ maxWords **do**
14: Selecionar as "numberWords" palavras com maior *score* de seleção de atributos
15: Construir um classificador de D_{build} com base nas palavras selecionadas;
16: Avaliar a F-Measure F_{tmp} do classificador usando $D_{validate}$.
17: **if** Ftmp $>$ bestFMeasure **then**
18: bestFMeasure = F_{tmp};
19: selectedFeatures = T_{tmp};
20: **end if**
21: numberWords = numberWords + 1;
22: **end while**
23: **return** selectedFeatures;

3.3 AVALIAÇÃO

Com o objetivo de comparar a efetividade das abordagens descritas na Subseção 3.1 e na Subseção 3.2 em um *dataset* da indústria, um experimento foi realizado visando analisar a efetividade de cada abordagem na classificação automática de defeitos.

O objetivo do experimento foi formalizado utilizando o modelo GQM proposto por Basili e Weiss (1984): **Analisar** o uso de abordagens classificação automática de defeitos utilizando seleção de atributos, **com a finalidade de** avaliar se a utilização de uma abordagem de classificação automática pode substituir classificação manual, **com respeito** à eficiência, medida por meio da métrica F-Measure que leva em consideração o número de falsos positivos e falsos negativos detectados na classificação, **do ponto de vista de** analistas de *software*, **no contexto de** uma empresa de desenvolvimento de *software*.

Nesse experimento, além das abordagens de seleção automática de atributos, foi realizada a seleção de atributos por meio de inspeção manual igualmente utilizada por Huang et al. (2015) e que combinada com outros processos, como por exemplo a geração de pseudo instâncias, obteve precisão média de 0,716 e F-Measure média de 0,565.

Desta forma, os resultados com as abordagens de seleção automática de atributos, com a seleção manual de atributos e a classificação sem seleção de atributos foram comparados e analisados estatisticamente. As questões de pesquisa que guiaram o experimento foram:

- Q1) As abordagens automáticas de seleção de atributos são tão efetivas quanto a seleção manual de atributos na classificação automática de defeitos ?
- Q2) A alteração proposta no algoritmo do USES aumenta a efetividade da classificação automática de defeitos ?

Para avaliar estas questões, foi utilizada a métrica F-Measure obtida na classificação automática como cada abordagem. Com o objetivo e a métrica definida, foram levantadas as seguintes hipóteses:

- H1.0: A abordagem com seleção manual de atributos tem maior efetividade que abordagens automáticas de seleção de atributos na classificação automática de defeitos.
- H1: As abordagens automáticas de seleção de atributos são tão efetivas quanto a seleção manual de atributos na classificação automática de defeitos.

- H2.0: A alteração proposta no algoritmo do USES não aumenta a efetividade da classificação automática de defeitos.
- H2: A alteração proposta no algoritmo do USES aumenta a efetividade da classificação automática de defeitos.

A metodologia utilizada neste experimento é composta pelas seguintes 4 etapas sequenciais:

1. Aquisição de dados: Nesta etapa, é realizada a extração dos relatórios de defeitos armazenados no repositório de defeitos.
2. Classificação manual: Após a aquisição dos dados, todos os defeitos são analisados individualmente e são classificados de acordo com as categorias predefinidas.
3. Preprocessamento de texto: O preprocessamento de texto é realizado com o objetivo de extrair as palavras do relatório e gerar uma representação processável das palavras dos relatórios.
4. Execução e análise de dados: Finalmente, cada abordagem é avaliada utilizando o *10-fold cross-validation*, após isso, as métricas de efetividade são coletadas e estatisticamente analisadas.

Por meio da análise de dados, tem-se evidências para suportar ou não a hipótese H1.

- Se o resultado do *10-fold cross-validation* das abordagens de seleção automática forem equivalentes à abordagem manual, a evidência suportará H1.
- Por outro lado, se as abordagens de seleção automática de atributos obtiverem um resultado menor do que a abordagem manual, a evidência rejeitará H1.

De forma similar, o suporte ou não da hipótese H2 se dará por:

- Se o resultado do *10-fold cross-validation* com a abordagem USES+ for superior ao resultado com a abordagem USES, a evidência suportará H2.
- Por outro lado, se a abordagem USES+ obtiver um resultado menor ou igual a abordagem USES, a evidência rejeitará H2.

A próxima subseção fornece mais detalhes sobre como cada etapa da metodologia foi realizada neste experimento.

3.3.1 METODOLOGIA

Conforme descrito anteriormente, os relatórios de defeito possuem o atributo prioridade, e são disponibilizados os seguintes valores para sua identificação:

- *Blocker*: bloqueia o uso do sistema.
- *Critical*: falhas, perda de dados ou problemas com o uso da memória.
- *Major*: perda de funcionalidade vital do sistema.
- *Minor*:perda de funcionalidade de menor relevância ou de fácil *workaround*.
- *Trivial*: simples de resolver, como uma ortografia ou cor errada.

Com base nesses valores de prioridade disponibilizados pelo Jira, as prioridade foram reagrupadas em duas categorias:

- Severo: *Blocker*, *Critical* e *Major*
- Não Severo: *Minor* e *Trivial*.

Esse reagrupamento foi realizado por dois motivos:

- Por meio de conversas com os analistas da empresa, acredita-se que essas duas categorias são suficientes para auxiliar na priorização correta da resolução de defeitos.
- Com os defeitos classificados nessas duas categorias, poderiam ser priorizados aqueles defeitos com maior impacto na operação da empresa, melhorando assim a qualidade do *software*.

A aquisição de dados foi realizada diretamente do banco de dados utilizado pelo Jira, e, ao fim da etapa de aquisição foram retornados 1.850 defeitos. Essa etapa de aquisição foi realizada no período de jan/2018 e jul/2018.

Finalizada a aquisição dos dados, foi realizada a classificação manual dos defeitos. Nesta fase, os 1.850 defeitos foram classificados por 2 analistas que fazem parte do processo de desenvolvimento da empresa há mais de 3 anos. Ao final da classificação dos defeitos, as respostas dos analistas foram comparadas e utilizadas da seguinte forma:

- Se a categoria dos dois analistas fosse igual, a prioridade do defeito é definida de acordo com a classificação dos analistas.
- Se houver divergência entre os analistas, eles tentam chegar a um consenso, caso não haja consenso o defeito é descartado.

Ao final deste processo, a categoria Severo apresentou 933 defeitos e, na categoria "Não Severo", 866 defeitos. Após a etapa de classificação manual, o conjunto de dados está pronto para ser utilizado no treinamento do modelo que realizará a classificação automática de defeitos. Com isso, o preprocessamento de texto de cada relatório de defeito foi executado de forma similar aos trabalhos relacionados (THUNG et al., 2012; XIA et al., 2014; LI et al., 2010; PATIL, 2017).

Após o preprocessamento do texto, os atributos foram selecionados de acordo com cada abordagem presente no experimento. Após a seleção, os atributos remanescentes foram representados como TF-IDF e um classificador foi treinado usando SVM. Esse algoritmo foi escolhido porque é comumente usado em outros estudos relacionados ao processamento de linguagem natural, por exemplo Thung et al. (2012) e Xia et al. (2014).

A fim de reduzir a chance de comportamentos aleatórios nos resultados, o *10-fold cross-validation* foi utilizado. Os *scripts* necessários para executar o experimento foram implementados na linguagem *Python* e foram disponibilizados [1].

A efetividade da classificação foi medida pela Precisão, Recall e F-Measure de cada abordagem. A próxima subseção apresenta os resultados do experimento.

3.3.2 RESULTADOS

A Tabela 3.1 apresenta valor médio da Precisão, Recall e F-Measure de cada abordagem analisada neste estudo, além disso, o resultado da classificação sem qualquer abordagem de seleção de atributos foi coletado para fins de comparação.

Tabela 3.1: Valor Médio da Precisão, Recall e F-Measure (*Dataset* Industrial).

Abordagem	Precisão	Recall	F-Measure
Sem Seleção	0,790	0,789	0,789
Seleção Manual	0,798	0,797	0,797
USES	0,800	0,800	0,800
USES+	0,838	0,836	0,836

Fonte: Autoria Própria

[1] https://github.com/andrebandeira/nlp-severity

A primeira métrica analisada foi a precisão média que variou entre 0,790, para a classificação sem qualquer abordagem de seleção de atributos, e 0,838 para a classificação utilizando a abordagem USES+. Nessa métrica, valores altos significam uma redução no número de falsos positivos detectados.

Em relação ao recall médio, os valores variaram entre 0,789, novamente para classificação sem qualquer abordagem de seleção de atributos, e 0,836, novamente para classificação usando a abordagem USES+. Nessa métrica, valores altos significam uma redução no número de falsos negativos detectados.

Finalmente, a F-Measure, que representa a combinação de precisão e recall, variou entre 0,789, na classificação sem qualquer abordagem de seleção de atributos, e 0,836, na classificação usando a abordagem USES+.

Em resumo, a abordagem que apresentou os melhores resultados médios nesta avaliação foi a USES+.

Após essa análise, foi realizada a análise estatística da F-Measure. A Figura 3.1 contém o histograma de densidade de cada abordagem com os resultados do *10-Fold Cross-Validation*.

Primeiramente, o teste de normalidade de *Shapiro Wilk* foi executado com base nos resultados de cada abordagem e, de acordo com a Tabela 3.2, todos os resultados das abordagens seguiram distribuições normais (considerando um intervalo de confiança de 0,95).

Tabela 3.2: Teste de Normalidade de *Shapiro Wilk* (*Dataset* Industrial).

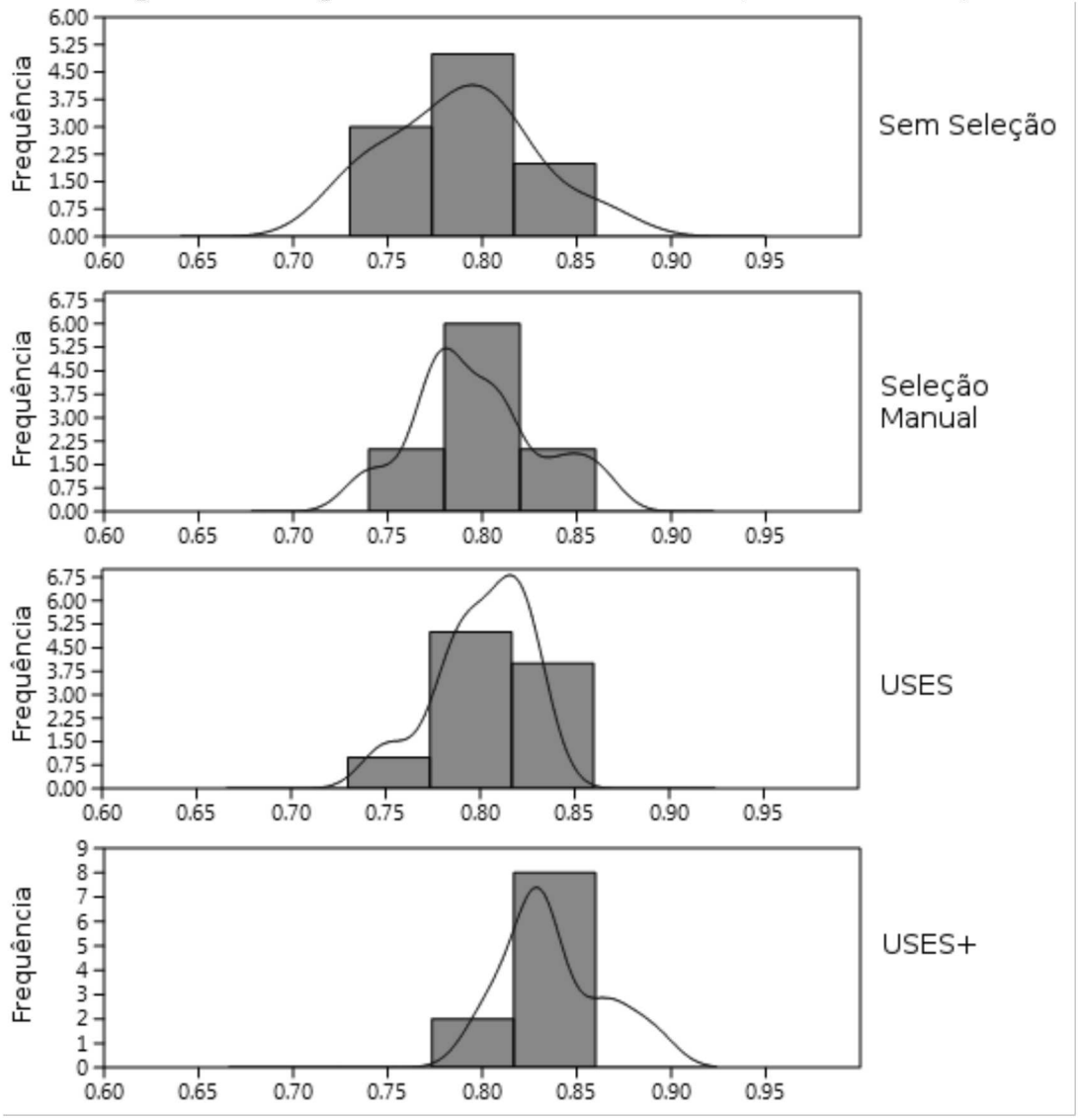
Abordagem	W	p-valor
Sem Seleção	0,973	0,918
Seleção Manual	0,959	0,781
USES	0,915	0,316
USES+	0,895	0,195

Fonte: Autoria Própria

Considerando que a distribuição dos resultados é normal, o teste ANOVA (unidirecional) foi realizado com os resultados visando identificar se houveram diferenças significativas entre resultados das abordagens. O resultado do teste ANOVA, com F = 4,765 e p-valor = 0,006, indica que a F-Measure foi significativamente diferente entre as abordagens (novamente considerando um intervalo de confiança de 0,95).

Com esses dados, visando descobrir quais abordagens possuíram diferenças significativas, uma análise *post-hoc* foi realizada utilizando o *Tukey's Pairwise*, que revelou diferenças significativas (novamente considerando um intervalo de confiança de 0,95) entre:

Figura 3.1: Histograma de Densidade - F-Measure (*Dataset* Industrial).



Fonte: Autoria Própria

- USES+ e Sem Seleção, com p-valor = 0,006;
- USES+ e Seleção Manual, com p-valor = 0,032.

A Tabela 3.3 mostra todos os resultados de p-valor retornados pelo *Tukey's Pairwise*.

Tabela 3.3: *Tukey's Pairwise* p-valor (*Dataset* Industrial).

Abordagem	Sem Seleção	Seleção Manual	USES	USES+
Sem Seleção	-	0,918	0,798	**0,006**
Seleção Manual	0,892	-	0,992	**0,032**
USES	1,289	0,396	-	0,062
USES+	4,957	4,065	3,669	-

Fonte: Autoria Própria

Os resultados da Precisão, Recall e F-Measure desta avaliação representam evidências que suportam a hipótese H1, isto é, abordagens automáticas de seleção de atributos são tão efetivos quanto a seleção manual de atributos na classificação automática de defeitos. Além disso, a abordagem que apresentou os melhores resultados gerais foi a abordagem USES+, porém estatisticamente não houveram diferenças significativas entre o USES+ e o USES, fornecendo evidencias que rejeitam a H2.

3.3.3 AVALIAÇÃO COM OUTROS DATASETS

Com o objetivo de validar os resultados obtidos na seção anterior e minimizar uma possível influência do *dataset* utilizado, foi executada a mesma avaliação com *datasets* de *softwares open source*, utilizados e disponibilizados por Xia et al. (2014).

Os defeitos dos *datasets* foram classificados, de acordo com as condições de gatilho para a ocorrência de determinado defeito, em duas categorias: *Bohrbug* que refere-se a um defeito que pode ser facilmente isolado e sua ativação e propagação de erros são simples, e, *Mandelbug* que identifica um defeito cuja sua ativação e propagação são complexas, ou seja, um defeito de difícil reprodução.

Os datasets são de três *softwares open source* distintos, a Tabela 3.4 contém as principais características de cada *dataset*:

Tabela 3.4: *Datasets* de *Software Open Source*.

Projeto	Quantidade de Defeitos	Repositório de Defeitos
Apache	143 (116 *Bohrbugs* e 27 *Mandelbugs*)	https://bz.apache.org/bugzilla
Linux	267 (122 *Bohrbugs* e 145 *Mandelbugs*)	https://bugzilla.kernel.org
MySQL	209 (84 *Bohrbugs* e 125 *Mandelbugs*)	https://bugs.mysql.com

Fonte: Autoria Própria

Nas próximas subseções são relatados os resultados, utilizando a métrica F-Measure, da avaliação com cada um desses datasets, onde cada abordagem foi avaliada utilizando *10-fold cross-validation*.

3.3.4 APACHE

Assim com na subseção anterior, a métrica analisada foi a F-Measure que teve variação entre 0,713, na classificação sem qualquer abordagem de seleção de atributos, e 0,809, na classificação usando a abordagem USES+, conforme mostra a Tabela 3.5. Nessa métrica, a abordagem que apresentou o melhor resultado médio foi a USES+.

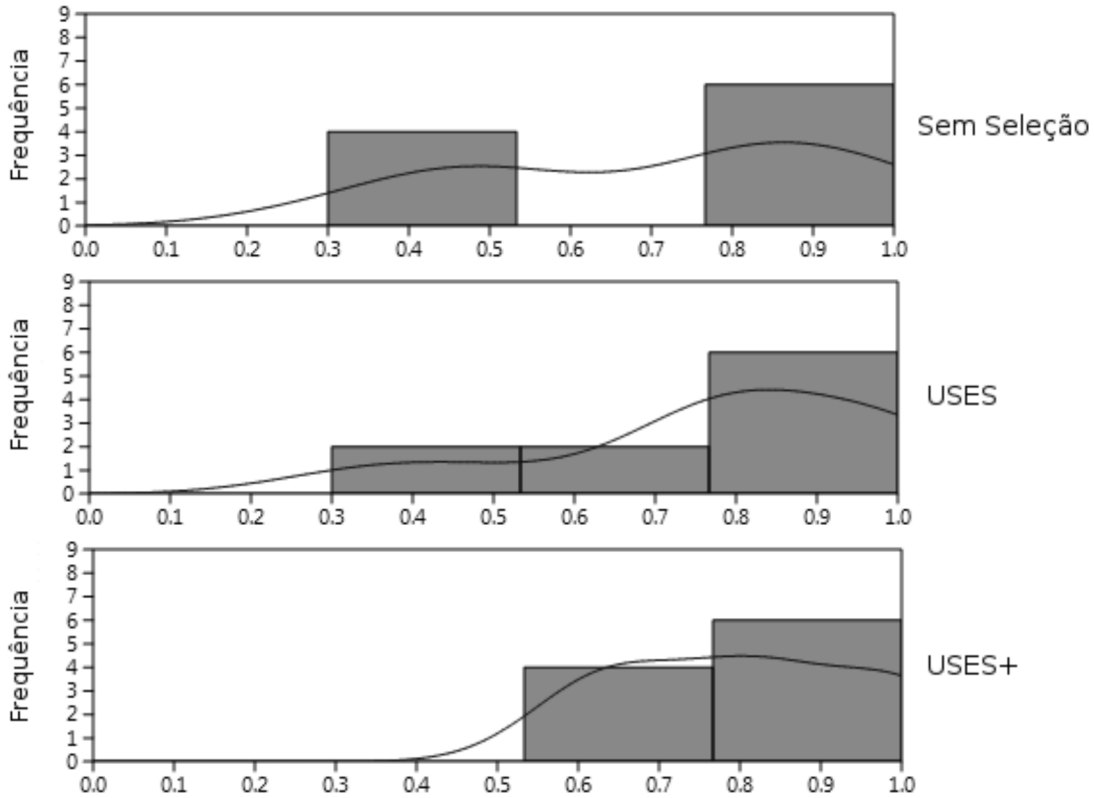
Tabela 3.5: Valor Médio da F-Measure (Apache).

Abordagem	F-Measure
Sem Seleção	0,713
USES	0,777
USES+	0,809

Fonte: Autoria Própria

Após essa análise, foi realizada a análise estatística da F-Measure. A Figura 3.2 contém o histograma de densidade de cada abordagem com os resultados do *10-Fold Cross-Validation*.

Figura 3.2: Histograma de Densidade - F-Measure (Apache).



Fonte: Autoria Própria

Primeiramente, o teste de normalidade de *Shapiro Wilk* foi executado com base nos resultados e, de acordo com a Tabela 3.6, todos os resultados seguiram distribuições normais (considerando um intervalo de confiança de 0,95).

Tabela 3.6: Teste de Normalidade de *Shapiro Wilk* (Apache).

Abordagem	W	p-valor
Sem Seleção	0,87	0,099
USES	0,87	0,099
USES+	0,85	0,060

Fonte: Autoria Própria

Considerando que a distribuição dos resultados é normal, o teste ANOVA (unidireci-

onal) foi realizado com os resultados visando identificar se houveram diferenças significativas entre resultados das abordagens. O resultado do teste ANOVA, com F = 0,55 e p-valor = 0,58, indica que a F-Measure não foi significativamente diferente entre as abordagens (novamente considerando um intervalo de confiança de 0,95).

Os resultados da F-Measure desta avaliação não corroboram com os resultados das avaliações anteriores, uma vez que nessa avaliação, diferentemente das outras avaliações, não houveram diferenças significativas nos resultados com as diferentes abordagens.

3.3.5 LINUX

Nessa avaliação, a métrica F-Measure teve variação entre 0,575, na classificação sem qualquer abordagem de seleção de atributos, e 0,785, na classificação usando a abordagem USES+, conforme mostra a Tabela 3.7. Nessa métrica, a abordagem que apresentou o melhor resultado médio foi a USES+.

Tabela 3.7: Valor Médio da F-Measure (Linux).

Abordagem	F-Measure
Sem Seleção	0,575
USES	0,656
USES+	0,785

Fonte: Autoria Própria

Posteriormente, foi realizada a análise estatística da F-Measure. A Figura 3.3 mostrar o histograma de densidade de cada abordagem proposta com os resultados do *10-Fold Cross-Validation*.

Na análise estatística, primeiramente foi executado o teste de normalidade de *Shapiro Wilk* e, de acordo com a Tabela 3.8, todos os resultados seguiram distribuições normais (considerando um intervalo de confiança de 0,95).

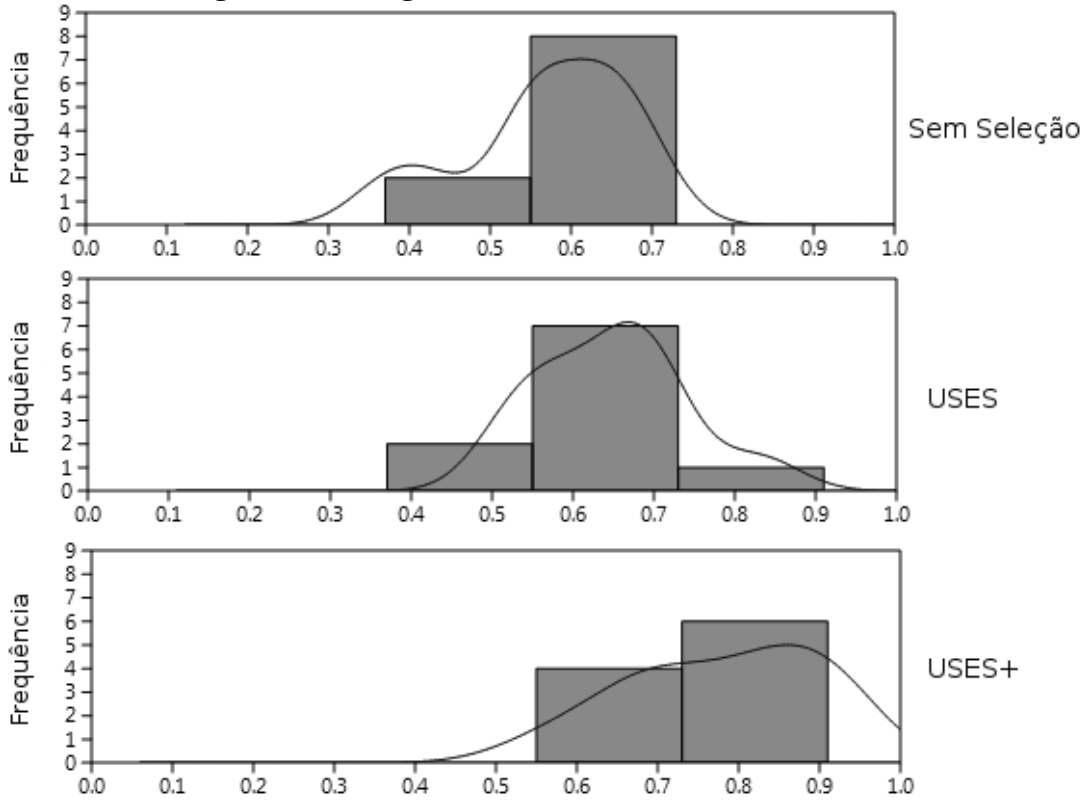
Tabela 3.8: Teste de Normalidade de *Shapiro Wilk* (Linux).

Abordagem	W	p-valor
Sem Seleção	0,915	0,322
USES	0,932	0,469
USES+	0,914	0,315

Fonte: Autoria Própria

Considerando a normalidade da distribuição dos resultados, o teste ANOVA (unidirecional) foi realizado visando identificar se houveram diferenças significativas entre resultados das abordagens. O resultado do teste ANOVA, com F = 10,1 e p-valor = 0,0005, indica que

Figura 3.3: Histograma de Densidade - F-Measure (Linux)



Fonte: Autoria Própria

a F-Measure foi significativamente diferente entre as abordagens (novamente considerando um intervalo de confiança de 0,95).

Para descobrir quais abordagens possuíram diferenças significativas, uma análise *post-hoc* foi realizada utilizando o *Tukey's Pairwise*, que revelou diferenças significativas (novamente considerando um intervalo de confiança de 0,95) entre:

- USES+ e Sem Seleção, com p-valor = 0,0003;
- USES+ e USES, com p-valor = 0,02;

A Tabela 3.9 mostra todos os resultados de p-valor retornados pelo *Tukey's Pairwise*.

Tabela 3.9: *Tukey's Pairwise* p-valor (Linux).

Abordagem	Sem Seleção	USES	USES+
Sem Seleção	-	0,242	**0,0003**
USES	2,335	-	**0,024**
USES+	6,286	3,951	-

Fonte: Autoria Própria

Os resultados desta avaliação representam evidências que suportam os resultados da avaliação principal. Além disso, novamente, a abordagem que apresentou os melhores resultados foi a abordagem USES+, apresentando F-Measure significativamente maiores que as outras abordagens, e, dessa vez, a diferença entre os resultados foi estatisticamente significativa.

3.3.6 MYSQL

Na última avaliação, a métrica F-Measure teve variação entre 0,610, na classificação sem qualquer abordagem de seleção de atributos, e 0,791, na classificação usando a abordagem USES+, conforme mostra a Tabela 3.10. Nessa métrica, a abordagem que apresentou o melhor resultado médio foi a USES+.

Tabela 3.10: Valor Médio da F-Measure (MySQL).

Abordagem	F-Measure
Sem Seleção	0,610
USES	0,648
USES+	0,791

Fonte: Autoria Própria

Após essa análise, foi realizada a análise estatística da F-Measure. A Figura 3.4 contém o histograma de densidade de cada abordagem com os resultados do *10-Fold Cross-Validation*.

Primeiramente, o teste de normalidade de *Shapiro Wilk* foi executado com base nos resultados de cada abordagem e, de acordo com a Tabela 3.11, todos os resultados das abordagens seguiram distribuições normais (considerando um intervalo de confiança de 0,95).

Tabela 3.11: Teste de Normalidade de *Shapiro Wilk* (MySQL).

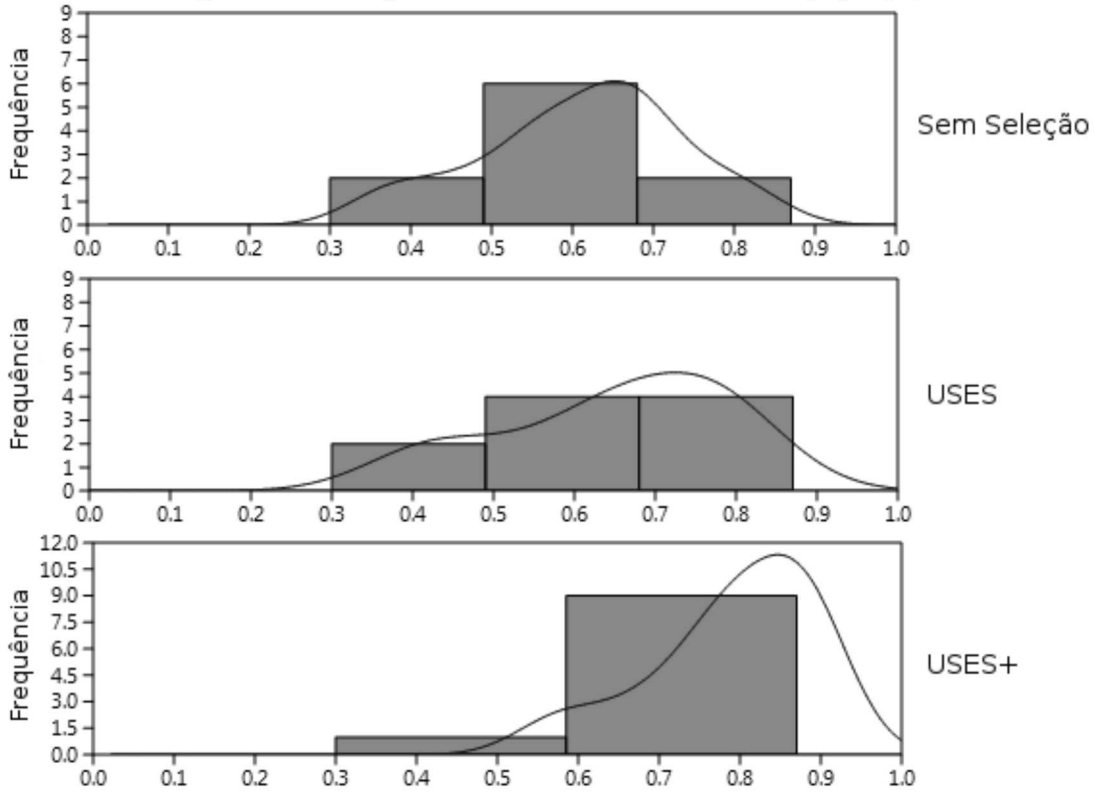
Abordagem	W	p-valor
Sem Seleção	0,978	0,953
USES	0,905	0,248
USES+	0,876	0,125

Fonte: Autoria Própria

Considerando que a distribuição dos resultados é normal, o teste ANOVA (unidirecional) foi realizado com os resultados visando identificar se houveram diferenças significativas entre resultados das abordagens. O resultado do teste ANOVA, com F = 5,87 e p-valor = 0,007, indica que a F-Measure foi significativamente diferente entre as abordagens (novamente considerando um intervalo de confiança de 0,95).

Com esses dados, visando descobrir quais abordagens possuíram diferenças significativas, uma análise *post-hoc* foi realizada utilizando o *Tukey's Pairwise*, que revelou diferenças significativas (novamente considerando um intervalo de confiança de 0,95) entre:

Figura 3.4: Histograma de Densidade - F-Measure (MySQL).



Fonte: Autoria Própria

- USES+ e Sem Seleção, com p-valor = 0,008;
- USES+ e USES, com p-valor = 0,041;

A Tabela 3.12 mostra todos os resultados de p-valor retornados pelo *Tukey's Pairwise*.

Tabela 3.12: *Tukey's Pairwise* p-valor (MySQL).

Abordagem	Sem Seleção	USES	USES+
Sem Seleção	-	0,777	**0,008**
USES	0,959	-	**0,041**
USES+	4,594	3,635	-

Fonte: Autoria Própria

Assim como na avaliação com o dataset do *Linux*, os resultados desta avaliação representam evidências que suportam os resultados da avaliação principal, com novamente a abordagem USES+ apresentando os melhores resultados, apresentando a F-Measure significativamente maior que das outras abordagens.

3.4 DISCUSSÃO

Este trabalho relata um estudo de caso executado com o objetivo de realizar a classificação automática da prioridade dos defeitos utilizando abordagens de seleção de atributos. A execução desse estudo foi realizada utilizando um conjunto de dados real da indústria.

Foram executadas diferentes abordagens de seleção de atributos com o objetivo de alcançar uma melhoria da efetividade da classificação automática de prioridade de defeitos. As seguintes abordagens foram utilizadas: sem seleção, seleção manual, USES e USES+.

A abordagem USES+ foi proposta com base no algorítimo proposto no USES, com duas alterações:

- Alteração da equação que calcula o *score* de afinidade entre categoria e palavra com o objetivo de privilegiar as palavras que aparecem mais recorrentemente em uma determinada categoria.
- Alteração no algoritmo para evitar a aleatoriedade na geração do conjunto de atributos candidatos, desta forma, apenas os atributos com o maior *score* de afinidade de entre categoria e palavra são selecionados.

Depois de realizar a seleção de atributos, um classificador foi treinado usando o algoritmo de aprendizado SVM e, para reduzir a chance de comportamentos aleatórios nos resultados, foi utilizado o *10-folds cross-validation*. O melhor desempenho médio do classificador foi obtido pela abordagem USES+ com F-Measure de 0,836, o USES teve F-Measure de 0,800, a seleção manual teve F-Measure de 0,797 e sem seleção teve F-Measure 0,789.

Com os resultados, o experimento mostrou evidências que suportam a hipótese H1 de que as abordagens automáticas de seleção de atributos são tão efetivas quanto a abordagem manual. No caso do *dataset* utilizado na avaliação, as abordagens automáticas tiveram maior efetividade do que as manuais. Além disso, neste primeiro *dataset* não houveram melhoras significativas com as alterações propostas ao algoritmo do USES.

Visando generalizar esse resultado foi realizada uma nova avaliação com datasets públicos que mostrou evidências de que as abordagens podem ser generalizadas e utilizadas em outras aplicações para melhorar o processo de classificação de prioridade automática de defeitos com base no relatório de defeitos escrito em linguagem natural. Em dois dos três *datasets* públicos utilizados (Linux e MySQL), os resultados mostraram que as alterações no algoritmo USES proposto por Xia et al. (2014), melhoraram sua efetividade com o conjunto de dados

utilizado. A Tabela 3.13 apresenta um resumo com as principais informações das avaliações realizadas.

Tabela 3.13: Resumo dos Resultados das Avaliações.

Dataset	Melhor *F-Measure* Médio	Diferença Estatística	H1	H2
Industrial	USES+	USES+ e Sem Seleção / USES+ e Seleção Manual	Suporta	Rejeita
Apache	USES+	Não	-	Rejeita
Linux	USES+	USES+ e Sem Seleção / USES+ e USES	-	Suporta
MySQL	USES+	USES+ e Sem Seleção / USES+ e USES	-	Suporta

Fonte: Autoria Própria

Este estudo em comparação com o estudo de Huang et al. (2015), fornece evidências de que os resultados da abordagem de seleção automática de atributos podem ser tão efetivos quanto a abordagem de seleção manual de atributos. Isso possibilitaria automatizar o processo de seleção de recursos e executar todo o processo de classificação automática de defeitos com maior facilidade, possibilitando a redução de custos na triagem de defeitos e, assim, melhorando o tempo de entrega da correção de defeitos com maior prioridade.

Como esse estudo não utiliza análise estática do código fonte de correção do defeito, não se pode comparar os resultados com o estudo de Thung et al. (2012). Apesar disso, a possibilidade de uma abordagem combinando a seleção de atributos com análise estática de código fonte é um importante trabalho futuro.

Neste mesmo sentido, como trabalho futuro, as abordagens de aprendizagem não supervisionada, como Patil (2017), poderiam ser executadas com a seleção de atributos como preparação dos dados e comparada sua efetividade com a classificação sem a seleção de atributos.

3.5 AMEAÇAS À VALIDADE

Algumas possíveis ameaças à confiabilidade dos resultados deste experimento foram levantadas e para cada ameaça levantada foram adotadas providências visando minimizar seus efeitos. As seguintes ameaças foram levantadas:

- Classificação manual de defeitos incorreta. Para minimizá-lo, a classificação foi realizada por 2 analistas, seu resultado foi comparado e o relatório de defeito foi selecionado apenas para o conjunto de dados em caso de concordância entre os analistas.

- Influência do algoritmo de aprendizagem. Para minimizar isso, a classificação foi realizada com outros dois algoritmos de aprendizado: Árvore de Decisão e Naïve Bayes. Não foram encontradas diferenças significativas entre os resultados da SVM (relatados na seção de resultados) e os resultados dos outros algoritmos. Mesmo assim, mais estudos poderiam ser realizados para verificar como os outros modelos de classificação se comportam. No entanto, vale ressaltar que este estudo teve como objetivo analisar se as abordagens poderiam ser utilizadas para priorizar os relatórios de defeitos, portanto, o resultado deste trabalho, utilizando um único modelo de classificação, fornece evidências que sustentam esse objetivo.

- Conjunto de dados utilizado. Algoritmos de *machine learning* podem ser influenciados pelo conjunto de dados que é utilizado no experimento. A fim de reduzir a chance disso ocorrer, utilizamos o *10-Fold Cross-Validation*, pois ter uma configuração de conjunto de dados diferente no procedimento de avaliação reduz a chance de influencia neste contexto. Além disso, a avaliação foi executada com três *datasets* de *software open source* com objetivo de validar os resultados iniciais.

4 CONSIDERAÇÕES FINAIS

Esse trabalho apresentou um estudo de caso sobre a efetividade de abordagens de seleção de atributos na classificação automática de defeitos em *software* por meio de relatórios de defeitos escritos em linguagem natural. Neste estudo de caso, um *dataset* foi construído utilizando relatórios de defeitos reais escritos em português de uma empresa com atuação focada em desenvolvimento de *software*.

As abordagens de seleção de atributos avaliadas foram: Manual, USES, USES+. Além dessas, uma classificação automática dos defeitos foi realizada sem qualquer abordagem de seleção de atributos visando comparar com as demais. Para cada abordagem foi construído um modelo de classificação utilizando o algoritmo de aprendizagem SVM. Os resultados médios da F-Measure ficaram entre 0,789 (para a abordagem sem seleção de atributos) e 0,836 (para o USES+).

Esses resultados fornecem evidências que apoiam a hipótese de que as abordagens de seleção automática de atributos são tão efetivas quanto a seleção manual de recursos. Além disso, os resultados indicam que as abordagens de seleção automática de atributos podem ter resultados superiores a seleção manual de atributos.

Além disso, a alteração proposta no algoritmo USES se mostrou promissora, uma vez que, em todos os *datasets* avaliados teve uma efetividade media maior que as demais abordagens avaliadas, e, em dois dos quatro *datasets* utilizados essa diferença foi estatisticamente significante.

A classificação automática da prioridade dos defeitos utilizando abordagens de seleção automática de atributos mostrou ser bastante promissora tendo apresentado uma boa efetividade geral nos resultados. Dessa forma, a classificação poderia reduzir o custo de priorização dos defeitos, melhorando significativamente o tempo necessário para que as correções essenciais cheguem os clientes finais do *software*.

Em consequência, com a possibilidade da execução da classificação da prioridade dos defeitos de forma automática, os analistas podem dispender o tempo que atualmente disponibili-

zam em tal atividade para outras atividades que tenham maior impacto na qualidade do *software* melhorando assim a satisfação geral de seus clientes.

Visando garantir que as abordagens possam ser aplicadas em outros domínios, foi realizada uma avaliação utilizando três datasets de *software open source* disponibilizado por Xia et al. (2014). Os resultados dessa nova avaliação corroboraram com os resultados obtidos na primeira, uma vez que em dois dos três datasets a abordagem de seleção automática USES+ teve um resultado significativamente melhor que os demais.

Como trabalho futuro, seria importante analisar como a seleção de atributos contribui quando combinada com outras abordagens encontradas na literatura, como por exemplo a analise do código fonte (THUNG et al., 2012) e a geração de pseudo-instâncias (HUANG et al., 2015).

Outro trabalho futuro possível é a aplicação do classificador construído nesse estudo para classificar a prioridade de defeitos de outros projetos a partir de relatórios de defeitos também escritos em português. Com isso, pode-se verificar se os padrões linguísticos se repetem em projetos de diferentes analistas, áreas e empresas.

REFERÊNCIAS

BASILI, V. R.; WEISS, D. M. A methodology for collecting valid software engineering data. **IEEE Transactions on software engineering**, IEEE, n. 6, p. 728–738, 1984.

CHILLAREGE, R. et al. Orthogonal Defect Classification - A Concept for In-Process Measurements. **IEEE Transactions on Software Engineering**, v. 18, n. 11, p. 943–956, 1992. ISSN 00985589.

DENGER, C.; ESP, B. G.-e. An Industrial Case Study of Implementing and Validating Defect Classification for Process Improvement and Quality Management Bernd Freimut. **Software Metrics, 2005. 11th IEEE International Symposium**, n. Metrics, p. 10–19, 2005. ISSN 1530-1435.

GABRILOVICH, E.; MARKOVITCH, S. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: **IJcAI**. [S.l.: s.n.], 2007. v. 7, p. 1606–1611.

GIGER, E.; PINZGER, M.; GALL, H. Predicting the fix time of bugs. In: ACM. **Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering**. [S.l.], 2010. p. 52–56.

GRADY, R. B. **Practical software metrics for project management and process improvement**. [S.l.]: Prentice-Hall, Inc., 1992.

HERZIG, K.; JUST, S.; ZELLER, A. It's not a bug, it's a feature: how misclassification impacts bug prediction. In: IEEE PRESS. **Proceedings of the 2013 International Conference on Software Engineering**. [S.l.], 2013. p. 392–401.

HUANG, L. et al. Autoodc: Automated generation of orthogonal defect classifications. **Automated Software Engineering**, Springer, v. 22, n. 1, p. 3–46, 2015.

IEEE. Ieee standard classification for software anomalies. **IEEE**, v. 1044-2009, 2010.

KITCHENHAM, B. Procedures for performing systematic reviews. **Keele, UK, Keele University**, v. 33, n. TR/SE-0401, p. 28, 2004. ISSN 13537776.

LI, N.; LI, Z.; SUN, X. Classification of software defect detected by black-box testing: An empirical study. In: IEEE. **Software Engineering (WCSE), 2010 Second World Congress on**. [S.l.], 2010. v. 2, p. 234–240.

LIMSETTHO, N. et al. Automatic unsupervised bug report categorization. In: IEEE. **Empirical Software Engineering in Practice (IWESEP), 2014 6th International Workshop on**. [S.l.], 2014. p. 7–12.

PATIL, S. Concept-based classification of software defect reports. In: IEEE PRESS. **Proceedings of the 14th International Conference on Mining Software Repositories**. [S.l.], 2017. p. 182–186.

PELLEG, D.; MOORE, A. W. et al. X-means: Extending k-means with efficient estimation of the number of clusters. In: **Icml**. [S.l.: s.n.], 2000. v. 1, p. 727–734.

PETERSEN, K. et al. Systematic mapping studies in software engineering. **EASE'08 Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering**, p. 68–77, 2008. ISSN 02181940.

RAMAGE, D. et al. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1**. [S.l.], 2009. p. 248–256.

THUNG, F.; LO, D.; JIANG, L. Automatic defect categorization. **Proceedings - Working Conference on Reverse Engineering, WCRE**, n. October, p. 205–214, 2012. ISSN 10951350.

XIA, X. et al. Automatic defect categorization based on fault triggering conditions. In: IEEE. **Engineering of Complex Computer Systems (ICECCS), 2014 19th International Conference on**. [S.l.], 2014. p. 39–48.

ZIBRAN, M. F. On the effectiveness of labeled latent dirichlet allocation in automatic bug-report categorization. In: ACM. **Proceedings of the 38th International Conference on Software Engineering Companion**. [S.l.], 2016. p. 713–715.