UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

THIAGO NASCIMENTO DE LIMA

MÉTODO DE ESTRUTURAÇÃO DOS PASSOS DE REPRODUÇÃO EM BUG REPORTS

CAMPINA GRANDE - PB

2023

THIAGO NASCIMENTO DE LIMA

MÉTODO DE ESTRUTURAÇÃO DOS PASSOS DE REPRODUÇÃO EM BUG REPORTS

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Professor Dr. Franklin de Souza Ramalho.

CAMPINA GRANDE - PB

2023

THIAGO NASCIMENTO DE LIMA

MÉTODO DE ESTRUTURAÇÃO DOS PASSOS DE REPRODUÇÃO EM BUG REPORTS

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Franklin de Souza Ramalho

Orientador – UASC/CEEI/UFCG

Professor Dr. Rohit Gheyi

Examinador – UASC/CEEI/UFCG

Professor Tiago Lima Massoni

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 14 de Fevereiro de 2023.

CAMPINA GRANDE - PB

ABSTRACT

Bug fixing is a natural step in the life cycle of a software, from development to the user interaction phase, problems will arise. Users, in general, report these problems through reports, however it is common that the information in these reports is incomplete or poorly structured. Among this information, the steps for reproduction elucidate the sequence of actions that reproduce the error and are considered as one of the most important information in the report. However, it is common that such information is not present or is present in an unstructured way, making it difficult, for example, for the developer to find the problem and generate automatic tests. Thus, an application is necessary to identify and structure the steps to reproduce a bug by extracting information from the body of the text of the reports, in order to provide data that can be used in the most diverse applications. This project proposes the development of machine learning models based on Natural Language Processing with the ability to identify and structure the steps for reproduction. In which, high efficiency was evidenced in the detection of reproduction steps in a report (F1 = 0.69), but low capacity to promote the structuring of report components, such as actors and actions.

Método de estruturação dos passos de reprodução em Bug Reports

Thiago Nascimento de Lima
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
thiago.lima@ccc.ufcg.edu.br

Franklin de Souza Ramalho
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
franklin@computacao.ufcg.edu.br

Tiago Lima Massoni
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
massoni@computacao.ufcg.edu.br

RESUMO

A resolução de bugs é uma etapa natural no ciclo de vida de um software, desde o desenvolvimento até a fase de interação com o usuário, problemas irão surgir. Os usuários, em geral, relatam estes problemas através de relatórios, entretanto é comum que as informações nestes relatórios estejam incompletas ou mal-estruturadas. Dentre estas informações, os passos para reprodução elucidam a sequência de ações que reproduzem o erro e são considerados como uma das informações mais importantes do relatório. Entretanto, é comum que tal informação não esteja presente ou esteja presente de forma não-estruturada, dificultando, por exemplo, o trabalho de encontrar o problema por parte do desenvolvedor e a geração de testes automáticos. Com isso, uma aplicação se faz necessária para identificar e estruturar os passos de reprodução de um bug através da extração de informações do corpo do texto dos relatórios, a fim de proporcionar dados que possam ser utilizados nas mais diversas aplicações. Este projeto propõe o desenvolvimento de modelos de aprendizagem de máquina baseados em Processamento de Linguagem Natural com capacidade de identificar e estruturar os passos para reprodução. No qual, ficou evidenciado alta eficiência na detecção dos passos de reprodução em um relatório (F1 = 0,69), mas baixa capacidade de promover a estruturação de componentes dos relatórios, tais como os atores e ações.

PALAVRAS-CHAVE

Passos de reprodução, Relatório de Bug, Processamento Linguagem Natural

ABSTRACT

Bug fixing is a natural step in the life cycle of a software, from development to the user interaction phase, problems will arise. Users, in general, report these problems through reports, however it is common that the information in these reports is incomplete or poorly structured. Among this information, the steps for reproduction elucidate the sequence of actions that reproduce the error and are considered as one of the most important information in the report. However, it is common that such information is not present or is present in an unstructured way, making it difficult, for example, for the developer to find the problem and generate automatic tests. Thus, an application is necessary to identify and structure the steps to reproduce a bug by extracting information from the body of the text of the reports, in order to provide data that can be used in the most diverse applications. This project proposes the development of machine learning models based on Natural Language Processing with the ability to identify and structure the steps for reproduction. In which, high efficiency was evidenced in the detection of reproduction steps in a report (F1 = 0.69), but low capacity to promote the structuring of report components, such as actors and actions.

KEYWORDS

Steps to reproduce, Bug report, Natural Language Processing

1 INTRODUÇÃO

O processo de resolução de bugs pode ser iniciado a partir da submissão de um relatório de bug encontrado por usuários ou desenvolvedores. Neste contexto, um canal de comunicação é gerado entre aquele que reportou o bug e o desenvolvedor designado para resolução do bug. Porém, devido ao caráter humano presente na comunicação, surgem dificuldades que interferem neste processo e aumentam o custo de esforço para resolução. Por exemplo, certas informações tendem a serem omitidas, repassadas incompletas ou desestruturadas [11].

Através da aplicação de surveys com desenvolvedores de empresas reais, [11] buscou elencar as informações mais importantes na resolução de bugs. Em ordem de importância, são listados: passos de reprodução, stack traces, casos de teste, comportamento observado, screenshots, comportamento esperado, exemplos de código, resumo, versões e relatórios de erro. A falta de estrutura de algumas destas informações, como os passos de reprodução de um bug, mostra-se extremamente prejudicial para sua resolução, visto que estas informações desestruturadas são apontadas como as mais importantes no mesmo survey.

A estruturação dos passos para reprodução traz consigo ao menos dois impactos positivos. Inicialmente, gera uma garantia ao desenvolvedor de que o campo estará presente nos relatórios de bugs em um formato proposto pelos desenvolvedores [15], no qual os passos deixariam de ser uma simples lista para tornar-se um padrão de descrição detalhada e com campos fixos. Com isto é possível a ampla utilização de ferramentas de geração de testes automáticos a partir de passos de reprodução [17, 25].

O objetivo deste trabalho foi avaliar o desempenho da aplicação de modelos de aprendizagem de máquina para identificar os passos de reprodução presentes em um relatório de bug, bem como estruturar os campos existentes nos passos, tais como, os atores presentes na sequência e as ações de cada ator. Estabelecemos as seguintes questões de pesquisa: (**QP1**) Quão precisos são modelos de classificação na identificação dos passos para reprodução em relatórios de bug?; (**QP2**) Quão eficazes são os modelos baseados em processamento de linguagem natural na extração de atores em passos para reprodução?; e (**QP3**) Quão eficazes são as técnicas de

processamento de linguagem natural no reconhecimento de ações em passos de reprodução?

A identificação dos passos de reprodução e a extração dos atores ambos utilizaram dos modelos de classificação SVMC, Naive Bayes e Random Forest [3, 6, 10]. A identificação dos passos demonstrou bons resultados e alcançaram F1 = 0,69, assim como, a extração de atores que alcançou F1 = 0,98, contudo, apresentou forte indicação *overfitting* de seus modelos. Por fim, a extração das ações utilizou da união das técnicas de POS tagging e Lematização, na qual, resultaram em uma baixa taxa de acerto de aproximadamente 0,27 [18, 21].

Este documento está organizado como segue. A Seção 2 introduz alguns conceitos fundamentais para o entendimento do trabalho. A Seção 3 apresenta trabalhos similares na área. A seção 4 formula a metodologia adotada para a execução do trabalho. A seção 5 apresenta os resultados encontrados. A seção 6 conclui o trabalho com um resumo das descobertas.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Relatório de bug

Um relatório de bug consiste na descrição de erros encontrados por usuários ou desenvolvedores durante a utilização de um software. A submissão destes pode ser realizada diretamente através do software em que o usuário se deparou com os problemas, porém também é comum que ela seja realizada em plataformas dedicadas à aglomeração de relatórios de bugs, como Bugzilla [2] e GitHub [5]. A Figura 1 ilustra um exemplo de relatório no Bugzilla.

No processo de escrita de relatórios de bugs, certos campos são necessários para facilitar a resolução dos bugs. O campo passos de reprodução se mostra essencial, uma vez que permite que o desenvolvedor, empenhado na correção, reproduza o erro. A ausência de tal informação torna o processo de investigação do bug mais custoso.

Os passos de reprodução nos relatórios de bugs devem descrever as ações que levaram ao acontecimento do bug, permitindo, assim, que outros usuários possam replicar o erro encontrado. Eles são uma sequência de procedimentos ordenados, em geral, composto por campos como: ator, ação executada pelo ator e ambiente onde o bug ocorreu.

Na Figura 1, os passos de reprodução são retratados no trecho *Steps to Reproduce* que elenca uma sequência de ações realizadas pelo usuário e que resultaram em um comportamento inesperado [11]. Todavia, a disposição dessas informações, em meio ao texto de descrição do bug, dificulta a identificação automática dessas partes, bem como dos atores participantes e das ações realizadas.

Na Tabela 1, é possível verificar parte do relatório da Figura 1 após ter seus passos extraídos e seus campos devidamente estruturados. Na primeira coluna está cada sentença do relatório destrinchado em linhas, na segunda coluna a indicação de quais sentenças foram identificadas como passos de reprodução, por fim, na terceira e quarta colunas os campos extraídos dos passos de reprodução, respectivamente, os atores e suas ações.

2.2 Aprendizagem de máquina

A categorização de um conjunto de dados em classes realizada através do aprendizado de máquina é denominada de classificação, na qual, através do aprendizado supervisionado é possível predizer as classes de novos dados. Em geral, o propósito de um classificador é gerar uma função de mapeamento dos dados em valores categóricos de saída e com isso, identificar a classe para novos dados [23]. Ao lidar-se com dados textuais é comum combinar técnicas de classificação com técnicas do Processamento de Linguagem Natural [16].

O Processamento de Linguagem Natural (PLN) é uma área de estudo da inteligência artificial focada em possibilitar que computadores processem textos e falas de forma similar a humana. Isto é possível por meio da combinação de técnicas linguísticas com estatística, aprendizagem de máquina e aprendizagem profunda [16]. O restante dessa seção se dedica a descrever técnicas de PLN que serão utilizadas neste trabalho.

A tokenização é uma tarefa presente em PLN responsável por separar os textos em unidades menores, os tokens. Os tokens podem ser caracteres ou palavras. Uma sequência contínua de tokens de um texto é chamado de n-grama, os casos em que essa sequência tem tamanho 1, 2 e 3 são respectivamente chamados unigrama, bigrama e trigrama [19]. Por exemplo, considerando a sentença "O carro andou na rua", os tokens seriam "O", "carro", "andou", "na" e "rua", enquanto os bigramas seriam: "O carro", "carro andou", "andou na" e "na rua".

Em geral, as palavras em um texto apresentam uma grande variação morfológica, como plurais ou conjugações verbais. Muitas vezes é interessante reduzir palavras como "partimos", "partindo" e "partiu" para o infinito "partir". Isto é chamado de lematização, em que as inflexões dos finais das palavras são removidas para que as palavras retornem às suas formas originais presentes no dicionário, os chamados lemas [18].

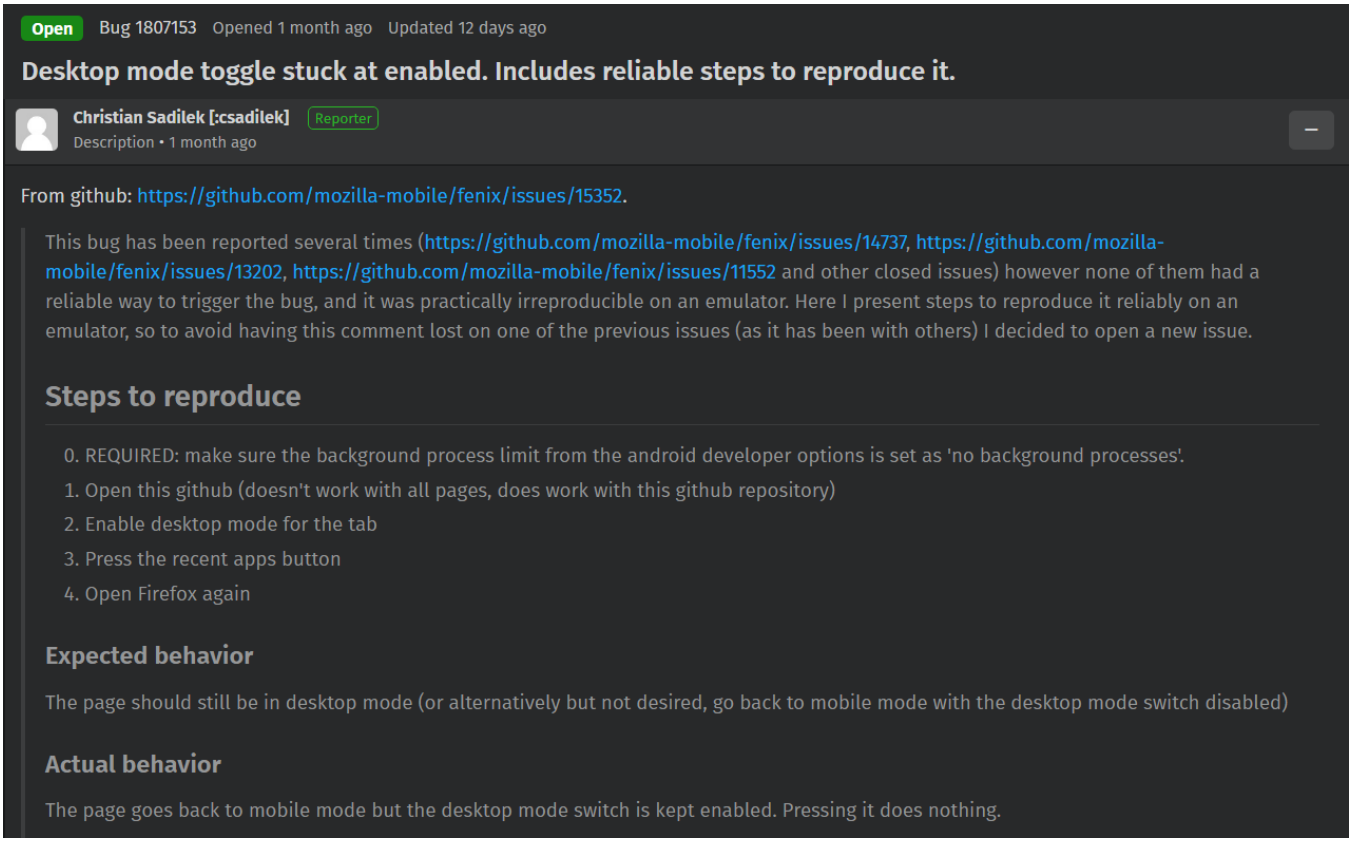
Outra atividade de PLN, é o processo de Part-Of-Speech Tagging (POS Tagging), em que cada palavra ou token de um texto é categorizado em uma classe gramatical baseado em sua definição ou contexto [21]. Por exemplo, na frase "visitei as cidades", o resultado seria ("visitei", verbo), ("as", artigo) e ("cidades", substantivo). Devido este aspecto gramatical, o POS Tagging é muitas vezes utilizado em conjunto com outras técnicas a fim de potencializar seus resultados.

3 TRABALHOS RELACIONADOS

Os passos de reprodução foram citados em Bettenburg et al. [12] como informação relevante para reprodução e resolução de um bug. Tal fato foi reforçado em carta aberta ao GitHub no ano de 2016 pelo Dear-Github [15], no qual desenvolvedores de mais de mil projetos open-source solicitaram a garantia do preenchimento dos passos de reprodução em *issues*. Contudo, devido a seu formato ser apresentado em linguagem natural é necessário, para que seja mais facilmente analisado, a detecção e extração dos passos de reprodução para um formato estruturado, através da separação em campos bem definidos por exemplo.

Diversos trabalhos já foram feitos na área de análise de passos de reprodução com variados enfoques, tais como detecção da ausência dos passos em relatórios [14, 22] e utilização dos passos para estimar a qualidade geral dos relatórios [13].

Figura 1: Exemplo de relatório de bug com passos de reprodução



Fonte: <https://bugzilla.mozilla.org/show_bug.cgi?id=1807153>

Tabela 1: Exemplo de relatório de bug estruturado

Sentença	S2R	Ator	Ação
...	...	...	...
Steps to reproduce	False	-	-
REQUIRED: make sure the background process limit from the android developer options is set as 'no background processes'.	True	user	make
Open this github (doesn't work with all pages, does work with this github repository)	True	user	open
Enable desktop mode for the tab	True	user	enable
...	...	...	...

Em Song and Chaparro [22], o trabalho utilizou o modelo de classificação Support Vector Machine Classifier (SVMC) para identificar as sentenças dos relatórios como passo para reprodução, utilizando a informação individual de cada sentença do relatório. O modelo apresentou bons resultados com uma precisão de 93%, uma cobertura de 70% e uma acurácia de 84%.

Em Chaparro et al. [14], os autores utilizaram três formas para detectar a ausência dos passos de reprodução, através de: expressões regulares, heurística com PLN e aprendizado de máquina utilizando SVMC. Observou-se que a implementação com aprendizado de máquina foi a melhor na métrica F1-Score com resultado de 74%.

Por sua vez, Chaparro et al. [13] utilizou da técnica de aprendizado profundo com uma camada convolucional e uma camada de inferência, utilizando Conditional Random Field, para evitar classificar cada sentença independentemente, alcançando uma alta acurácia de 98%.

Entretanto, pouco foi feito em relação à estruturação dos passos a fim de extrair os campos que possam ser úteis para finalidades como geração de testes automáticos, por exemplo o ator e a ação.

4 METODOLOGIA

Esta seção descreve o procedimento que foi utilizado para o desenvolvimento de um modelo capaz de extrair campos, a partir

dos passos para reprodução presentes em relatório de bugs. Inicialmente, descrevemos o processo de obtenção e pré-processamento dos dados, em seguida tratamos do método para identificar os passos para reprodução dentro do texto completo do relatório via modelos de classificação, então descrevemos as técnicas que serão utilizadas na extração dos campos. Por fim, detalhamos como foi realizada a avaliação dos resultados [1].

4.1 Obtenção e pré-processamento dos dados

Os dados utilizados neste trabalho foram coletados do Bugzilla [2], um *issue tracker* que agrega milhares de relatórios de bugs de diversos produtos e áreas, sendo assim, ideal para uma aplicação de caráter geral e por isto foi o escolhido para o trabalho. Ao total, foram capturados 64325 bugs de projetos variados, no período de 2019 a 2021, para termos relatórios atuais, mas com certa abrangência de tópicos.

Após a coleta dos dados, estes foram submetidos a um pré-processamento para que o texto bruto passe para um formato válido a ser utilizado no treinamento de classificadores. O processo está descrito abaixo:

(1) Cada relatório foi dividido em sentenças;
(2) Para cada sentença foi aplicada lematização para reduzir o vocabulário possível;
(3) Para cada sentença foi gerado um conjunto de tokens, considerando unigramas, bigramas e trigramas.

4.2 Identificação dos passos para reprodução

As técnicas de classificação utilizadas neste trabalho são técnicas de aprendizado supervisionado, dessa forma é preciso declarar quais sentenças dos relatórios pertencem aos passos de reprodução ou não.

Figura 2: Pseudocódigo do algorítmo baseaso em heurísticas

Heurística 1 Classificação de sentenças de um Bug Report em S2R

```
1:  S2R ← dict{}
2:  S2R_Block ← FALSE
3:  for each s ∈ S do
4:      if S2R_Block then
5:          if s é uma sentença vazia then
6:              S2R_Block ← FALSE
7:          else
8:              S2R[s] ← TRUE
9:          end if
10:     else
11:         if 'step' ∈ s ∨ 'reproduce' ∈ s then
12:             S2R_Block ← TRUE
13:             S2R[s] ← FALSE
14:         end if
15:         if s começa com dígitos seguido por '-' or '.' then
16:             S2R[s] ← TRUE
17:         end if
18:         S2R[s] ← FALSE
19:     end if
20: end for
```

Essa rotulação foi realizada em duas etapas; na primeira os dados foram automaticamente rotulados por um algoritmo, elaborado pelos autores, baseado em heurísticas, em seguida ocorreu uma validação manual dos resultados encontrados pelo algoritmo, com intuito de garantir exatidão do rótulos. Esta última etapa também incluiu a identificação dos atores e ações de cada passo identificado.

O algoritmo de rotulação baseado em heurística, demonstrado na Figura 2, foi elaborado com base na observação dos dados, uma vez que, comumente os passos são encapsulados em blocos entitulados. Propusemos a criação do identificador *S2R_Block*, no qual, marca todas sentenças do bloco como possíveis passos. Além disso, os passos de reprodução costumam ser listagens numeradas e para tais casos existe a expressão condicional presente na linha 15.

Foram rotulados e formatados 640 relatórios de bugs (aproximadamente 1% do total de relatórios de bugs capturados), escolhidos de forma aleatória, e foi iniciada a etapa de classificação dos dados, na qual, diferentes técnicas de classificação foram utilizadas a fim de comparar qual modelo apresentou os melhores resultados. A escolha destes modelos foi feita considerando a diversidade destes, bem como os resultados previamente relatados em trabalhos anteriores. Foram utilizadas as implementações disponíveis na ScikitLearn [20], a escolha dos modelos ocorreu pela distinção de metodologias que empregam. A seguir descrevemos os modelos avaliados:

- *Support Vector Machine Classifier* (SVMC): modelo de classificação que utiliza da separação de pontos por uma função linear como classificador. Esta técnica é amplamente utilizada na área devido sua facilidade em lidar com matrizes esparsas, caso comum de relatórios de bug, uma vez que o relatório apresenta apenas um pequeno subconjunto de todo o universo de palavras conhecido [10, 22, 24];
- *Naive Bayes*: modelo de classificação baseado na aplicação do Teorema de Bayes e que assume a independência condicional entre os atributos. Este modelo apesar de simples obteve bons resultados na classificação de documentos [6];
- *Random Forest*: modelo de classificação que utiliza o paradigma de *ensemble*, no qual múltiplos modelos são combinados para classificar a entrada. Neste caso, os modelos combinados são árvores de decisão. Cada árvore é treinada com uma amostra dos dados de entrada, obtida via amostra aleatória simples com reposição. Além disso, durante a construção das árvores, os atributos a serem considerados são aleatoriamente escolhidos [3].

4.3 Extração dos campos

Utilizando as sentenças identificadas como passos para reprodução na etapa de classificação, prosseguimos para a etapa de extração dos campos. O foco foi em extrair dois campos: os atores e suas ações.

Devido ao caráter imperativo presente nas sentenças de passos de reprodução, a melhor abordagem neste cenário para extração de atores foi através de classificação entre passos realizados por usuário ou pelo sistema, utilizando as mesmas técnicas citadas anteriormente para a identificação dos passos de reprodução. Para a extração das ações utilizamos a técnica de POS Tagging com a transformação dos verbos para o seu infinitivo.

[1] O arquivo de estudo de estruturação dos passos de reprodução: link

4.4 Avaliação dos resultados

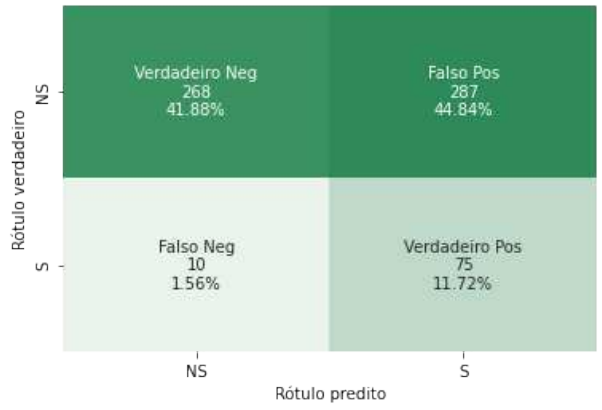
Para determinar o melhor modelo na identificação dos passos de reprodução (**QP1**) realizada através da classificação, foram calculadas as métricas Acurácia [1], Precisão [8], Cobertura [9] e F1-Score [4], considerando os rótulos manuais. Tais métricas foram calculadas para cada um dos modelos treinados, bem como para a heurística de rotulagem. Em seguida, foi realizada a comparação para identificar qual teve o melhor desempenho. Por sua vez, a efetividade da identificação dos atores utilizando classificação, medido também utilizando as métricas citadas acima, (**QP2**) e da identificação de ações utilizando POS Tagging (**QP3**), ambos foram feitas comparando os resultados obtidos com os determinados manualmente.

5 RESULTADOS E DISCUSSÕES

A primeira etapa deste trabalho consistiu na elaboração e implementação de um algoritmo, ver na Figura 2, que utiliza de heurísticas para identificar a presença de passos para reprodução em um relatório de bug. O objetivo foi entender se o problema poderia ser resolvido por um algoritmo simples e intuitivo.

Para fins de simplificação, chamaremos a classe com passos de reprodução de classe S, enquanto a classe sem passos de reprodução classe NS. A Figura 3 apresenta a matriz de confusão para a aplicação deste algoritmo sobre os 640 relatórios de bugs, escolhidos de forma aleatória, do dado de entrada. O algoritmo teve baixa precisão (p = 0,21) e alta cobertura (r = 0,88) na classe S, de modo que dentre os 362 relatórios identificados como possuidores de passos para reprodução apenas 76 realmente o tinham. Já na classe NS, existiu uma inversão da magnitude dos valores, tendo uma alta precisão na classe (p = 0,96) e baixa cobertura (r = 0,48).

Figura 3: Matriz de confusão dos resultados do algoritmo heurístico



A realização do treinamento de modelos para identificação dos passos de reprodução exige que o texto original seja dividido em sentenças, cada sentença pode ser ou não um passo de reprodução. Com isso, é comum que para cada sentença classificada como passo de reprodução existam centenas de sentenças que não são passos, o que torna os dados bastante desbalanceados.

Tabela 2: Métricas para modelos avaliados na classificação dos passos de reprodução

Modelo	Acurácia	Precisão	Cobertura	F1
Random Forest	0,992	0,816	0,606	0,696
SMVC	0,992	0,830	0,591	0,690
Naive Bayes	0,933	0,127	0,606	0,210

Levando esse fato em consideração, existem duas estratégias principais para mitigar essa desproporção de dados, sendo elas *undersampling* e *oversampling* [7]. As estratégias buscam alterar os dados de forma que duas classes com quantidade de dados distintos se aproximem. O *undersampling* elimina observações da classe predominante para a aproximar da classe com menos valores, enquanto o *oversampling* replica observações da classe minoritária para equiparar as observações.

5.1 Identificação dos passos de reprodução

Nos dados utilizados neste estudo, para responder **QP1**, a estratégia que se mostrou mais eficiente para combater o desbalanceamentos de dados foi a utilização de ambas as abordagens, *undersampling* e *oversampling*, ou seja, diminuir as observações da classe dominante e aumentar as da classe minoritária.

A Tabela 2 apresenta as métricas dos modelos de classificação de sentenças com passos de reprodução utilizados no estudo.

O modelo com o melhor desempenho foi o Random Forest que apresentou o maior F1 (0,696), portanto tem o mais alto balanceamento entre precisão e cobertura. Em seguida, temos o SVMC que apresentou F1 (0,690) bastante próximo, além disso, apresentaram acurácias idênticas de 0,992. Este resultado pode ser explicado devido ao modelo Random Forest utilizar de uma estratégia de *ensemble* o que propiciona naturalmente resistência a efeito de alta taxa de ajuste aos dados, *overfitting*.

Como é possível notar ainda na Tabela 2, o Naive Bayes apresenta os piores resultados em todas as métricas, isto possivelmente se deve a sua característica de independência entre os atributos. Fato este, que costuma não ser verdadeiro em textos, pelo contrário, existe uma depedência natural na progessão da sentença.

5.2 Classificação dos atores

Com intuito de responder à **QP2**, foram selecionadas apenas as observações com passos de reprodução. Os atores destas observações foram rotulados entre usuário e sistema. Contudo, existe uma grande desproporção na quantidade dessas entidades: 302 ações feitas por usuários para 8 feitas pelo sistema, por isso foi utilizado a técnica de *oversampling* nas observações.

Todos os modelos apresentaram ótimos valores alcançados para as métricas, como é possível notar na Tabela 3, com os modelos Random Forest e Naive Bayes apresentando os melhores resultados, sendo estas idênticas, com F1 de 0,98 e acurácia de 0,961.

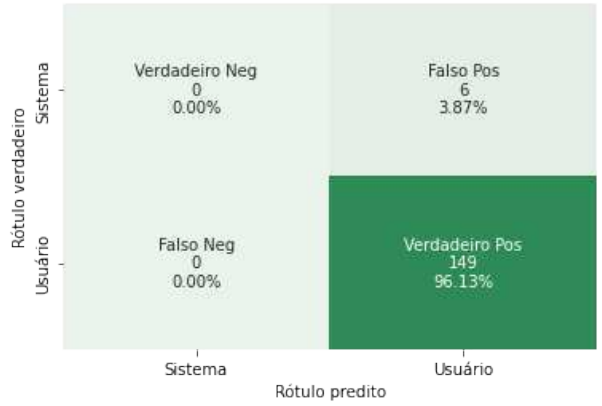
O modelo SVMC, temos a matriz de confusão na Figura 4, na qual deixa evidenciado que o modelo predice todas as observações como "usuário" e ainda foi capaz de gerar boas métricas, tais como F1 = 0,967.

Tabela 3: Métricas para modelos avaliados na classificação dos atores

Modelo	Acurácia	Precisão	Cobertura	F1
Random Forest	0,961	0,967	0,993	0,980
Naive Bayes	0,961	0,967	0,993	0,980
SMVC	0,935	0,935	1,000	0,967

Sendo assim, fica evidenciado que mesmo com a utilização das técnicas de *sampling*, os modelos testados, com bons resultados, não garantem ter generalização para grandes quantidades de dados.

Figura 4: Matriz de confusão dos resultados da identificação dos atores - SVMC



5.3 Extração das ações

Por fim, para a avaliação da **QP3** foi utilizada a técnica de POS Tagging em união com lematização nas sentenças com passos de reprodução. Entretanto, essa técnica mostrou-se ineficaz, uma vez que, ao realizar a comparação com os dados rotulados houve 26 acertos das 95 sentenças.

Grande parte dos erros advém da lematização ser incapaz de identificar propriamente ambiguidades semânticas, tais como a palavra "open" que pode ser interpretada como adjetivo ou verbo.

6 CONCLUSÕES

Em relação à **QP1**, como demonstrado nos resultados, o modelo Random Forest se mostrou levemente superior ao SVMC, contudo o modelo SVMC é o modelo mais utilizado atualmente para detecção de passos de reprodução, o que indica idealmente um trabalho futuro para melhor comparar tais modelos, especialmente com maior número de dados.

Por sua vez, **QP2** apresentou todos os modelos tiveram bons resultados, contudo tais resultados não podem ser totalmente levados em consideração, uma vez, existe um enorme desbalanceamento desleal dos dados, tornando os resultados quase inconclusivos.

Na **QP3** deixou evidenciado que apenas o uso de POS tagging e lematização não são suficientes para identificar as ações realizadas um passo de reprodução, especialmente devido ao seu caráter de baixa contextualização presente em sentenças.

Por fim, um grande limitador do trabalho foi a não existência de uma extensa fonte de dados já rotulados, devido a isto, a necessidade de rotulagem manual limita o número de dados. Fator este, que se mostrou um grande catalisador dos resultados de **QP2** e **QP3**.

Tal fator, pode ser potencialmente apaziguado no futuro, uma vez que, este trabalho gerou um *dataset* [2]com valores rotulados podendo facilmente ser expandido para obtenção de melhores resultados.

AGRADECIMENTOS

Eu gostaria de agradecer aos meus orientadores, Franklin Ramalho e Tiago Massoni, por sua disposição contínua e ótimos aconselhamentos, além de me inspirarem a buscar o melhor do academicismo. Ademais, gostaria de demonstrar minha gratidão por todos os amigos e colegas que acreditaram em mim mesmo quando deixei de fazê-lo. Em especial, a minha musa inspiradora, Sia, que mesmo em seu tempo reclusa ainda foi capaz de encontrar meios para me alcançar.

REFERÊNCIAS

[1] [n. d.]. Accuracy Score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html
[2] [n. d.]. Bugzilla. https://bugzilla.mozilla.org
[3] [n. d.]. Ensemble methods - Forests of randomized trees. https://scikit-learn.org/stable/modules/ensemble.html#forests-of-randomized-trees
[4] [n. d.]. F1 Score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score
[5] [n. d.]. Github. https://github.com/
[6] [n. d.]. Naive Bayes. https://scikit-learn.org/stable/modules/naive_bayes.html
[7] [n. d.]. Oversampling and Undersampling. https://towardsdatascience.com/oversampling-and-undersampling-5e2bbaf56dcf
[8] [n. d.]. Precision Score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html#sklearn.metrics.precision_score
[9] [n. d.]. Recall Score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html#sklearn.metrics.recall_score
[10] [n. d.]. Support Vector Machines. https://scikit-learn.org/stable/modules/svm.html
[11] Nicolas Bettenburg, Sascha Just, Adrian Schröter, Cathrin Weiss, Rahul Premraj, and Thomas Zimmermann. 2008. What Makes a Good Bug Report? (2008), 308–318. https://doi.org/10.1145/1453101.1453146
[12] Nicolas Bettenburg, Rahul Premraj, Thomas Zimmermann, and Sunghun Kim. 2008. Extracting Structural Information from Bug Reports. (2008), 27–30. https://doi.org/10.1145/1370750.1370757
[13] Oscar Chaparro, Carlos Bernal-Cárdenas, Jing Lu, Kevin Moran, Andrian Marcus, Massimiliano Di Penta, Denys Poshyvanyk, and Vincent Ng. 2019. Assessing the Quality of the Steps to Reproduce in Bug Reports. (2019), 86–96. https://doi.org/10.1145/3338906.3338947
[14] Oscar Chaparro, Jing Lu, Fiorella Zampetti, Laura Moreno, Massimiliano Di Penta, Andrian Marcus, Gabriele Bavota, and Vincent Ng. 2017. Detecting Missing Information in Bug Descriptions. (2017), 396–407. https://doi.org/10.1145/3106237.3106285
[15] Dear-Github. 2016. An open letter to github from the maintainers of open source projects. https://github.com/dear-github/dear-github
[16] IBM Cloud Education. 2020. What is Natural Language Processing? https://www.ibm.com/cloud/learn/natural-language-processing
[17] Mattia Fazzini, Martin Prammer, Marcelo d'Amorim, and Alessandro Orso. 2018. Automatically Translating Bug Reports into Test Cases for Mobile Apps. (2018), 141–152. https://doi.org/10.1145/3213846.3213869
[18] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Introduction to Information Retrieval. *Cambridge University Press* (2008).
[19] Aravindpai Pai. 2020. What is Tokenization in NLP? Here's All You Need To Know. https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/
[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[21] Kurtis Pykes. 2020. Part Of Speech Tagging for Beginners. https://medium.com/mysuperai/what-is-named-entity-recognition-ner-and-how-can-i-use-it-2b68cf6f545d

[2]Dataset com os valores rotulados: link

[22] Yang Song and Oscar Chaparro. 2020. BEE: A Tool for Structuring and Analyzing Bug Reports. (2020), 1551–1555. https://doi.org/10.1145/3368089.3417928
[23] Mohammad Waseem. 2022. How To Implement Classification In Machine Learning? https://www.edureka.co/blog/classification-in-machine-learning/#classification
[24] Yu Zhao, Kye Miller, Tingting Yu, Wei Zheng, and Minchao Pu. 2019. Automatically Extracting Bug Reproducing Steps from Android Bug Reports. (2019), 100–111.
[25] Yu Zhao, Tingting Yu, Ting Su, Yang Liu, Wei Zheng, Jingzhi Zhang, and William G.J. Halfond. 2019. ReCDroid: Automatically Reproducing Android Application Crashes from Bug Reports. (2019), 128–139. https://doi.org/10.1109/ICSE.2019.00030