UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

BACHARELADO EM ENGENHARIA DE SOFTWARE

JORDÃO MACEDO DE SOUZA

RELACIONANDO REFATORAÇÕES DE CÓDIGO COM ANÁLISE DE SENTIMENTOS EM PROJETOS DE CÓDIGO-FONTE ABERTO

QUIXADÁ – CEARÁ

2018

JORDÃO MACEDO DE SOUZA

RELACIONANDO REFATORAÇÕES DE CÓDIGO COM ANÁLISE DE SENTIMENTOS EM PROJETOS DE CÓDIGO-FONTE ABERTO

Monografia apresentada no curso de Engenharia de Software da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Engenharia de Software. Área de concentração: Computação.

Orientadora: Dra. Ticiana Linhares Coelho da Silva

Coorientador: Dr. Criston Pereira de Souza

QUIXADÁ – CEARÁ

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S238r Souza, Jordão Macedo de.
Relacionando Refatorações de Código com Análise de Sentimentos em Projetos de Código-fonte Aberto / Jordão Macedo de Souza. – 2018.
31 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2018.
Orientação: Profa. Dra. Ticiana Linhares Coelho da Silva.
Coorientação: Prof. Dr. Criston Pereira de Souza.

1. Emoções- Análise. 2. Software- Fatores Humanos. 3. Software- Refatoração. I. Título.
CDD 005.1

JORDÃO MACEDO DE SOUZA

RELACIONANDO REFATORAÇÕES DE CÓDIGO COM ANÁLISE DE SENTIMENTOS EM PROJETOS DE CÓDIGO-FONTE ABERTO

Monografia apresentada no curso de Engenharia de Software da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Engenharia de Software. Área de concentração: Computação.

Aprovada em: _____/_____/_____

BANCA EXAMINADORA

Dra. Ticiana Linhares Coelho da Silva (Orientadora)
Universidade Federal do Ceará – UFC

Dr. Criston Pereira de Souza (Coorientador)
Universidade Federal do Ceará - UFC

Me. Regis Pires Magalhães
Universidade Federal do Ceará - UFC

Este trabalho é dedicado às crianças adultas que, quando pequenas, sonharam em se tornar cientistas.

AGRADECIMENTOS

Os agradecimentos principais são direcionados à minha orientadora Ticiana, pelo suporte, pelas suas correções, pelo seu apoio e incentivos. Ao meu co-orientador Críston, pelo emprenho dedicado à elaboração deste trabalho.

Obrigado à minha família, que nos momentos de minha ausência dedicados ao estudo superior, sempre fizeram entender que o futuro é feito a partir da constante dedicação no presente!

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender. A palavra mestre, nunca fará justiça aos professores dedicados aos quais sem nominar terão os meus eternos agradecimentos. A esta universidade, direção, administração e coordenação que oportunizaram a janela que hoje vislumbro um horizonte superior, eivado pela acendrada confiança no mérito e ética aqui presentes.

Meus agradecimentos aos amigos Jackson, Pereira e Iury, companheiros de trabalhos e irmãos na amizade que fizeram parte da minha formação e que vão continuar presentes em minha vida com certeza.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

"Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito."

(Bíblia Sagrada, Romanos 12, 2)

RESUMO

Este trabalho apresenta um estudo a partir da análise de sentimentos em mensagens expressas pelos desenvolvedores por meio de *commits* em repositórios de código aberto. Foram analisados oito projetos de código aberto e um total de 11,979 mensagens de *commit*. Para o estudo foram classificadas as mensagens dos *commits* em duas categorias: positivo ou negativo. Em seguida, é realizada uma investigação se os sentimentos estão relacionados às atividades de refatoração de código. Como resultados deste estudo, tem-se indícios que quando se trabalha com refatorações a tendência é a expressar menos sentimentos negativos em relação a quando não se refatora código.

Palavras-chave: Análise de sentimentos. Fatores Humanos na Engenharia de Software. Atividades de Refatoração

ABSTRACT

This study presents an analysis of feelings in messages expressed by developers through commits in code repositories. In this work, six open source projects were analyzed, and a total of 12,113 commit messages. In the first phase, commit messages were classified into two categories (positive or negative). As a second phase, the study investigates whether feelings are related to code refactoring activities, and concludes that when working with refactorings, the tendency is to express fewer negative feelings.

Keywords: Sentment analysis. Human factors in Software Engineering. Code refactorings

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Análise de sentimentos é o processo de atribuir um valor de humor quantitativo (positivo ou negativo) a um fragmento de texto (GUZMAN; AZOCAR; LI, 2014). Métodos que se utilizam da análise de sentimentos foram inicialmente desenvolvidos para extrair a polaridade de sentimentos em textos curtos ou comentários em redes sociais onde existe grande interação do público, como em *tweets*. Pesquisas recentes associam fatores humanos, como humor e emoções, com resolução de problemas (GRAZIOTIN; WANG; ABRAHAMSSON, 2014), com linguagens de programação (GUZMAN; AZOCAR; LI, 2014), com resultados do processo de *build* (SOUZA; SILVA, 2017) e com tarefas de refatoração de código (SINGH; SINGH, 2017).

Guzman e Azócar (2014) concluíram que *commits* feitos nas segundas-feiras e projetos desenvolvido na linguagem de programação Java tendem a ter mais sentimentos negativos em relação aos outros dias da semana e a outras linguagens de programação. Singh e Singh (2017) mostraram que, geralmente, os desenvolvedores expressam com mais frequência sentimentos negativos do que positivos enquanto realizam a refatoração de código, o que pode indicar que os desenvolvedores não veem como algo bom para o projeto as refatorações de código. Outras pesquisas mostraram que as emoções afetam a qualidade do produto desenvolvido, a produtividade, a criatividade e a satisfação dos desenvolvedores (CHOUDHURY; COUNTS, 2013). Sabendo disso, este trabalho buscou encontrar a relação de análise de sentimentos com fatores produzidos pelos desenvolvedores, como tarefas de refatoração de código, que tende a adicionar melhorias no código-fonte (FOWLER; BECK, 1999).

Segundo Fowler e Kent (1999), refatoração de código é uma forma de mudar um sistema de software, melhorando a estrutura interna de uma forma que não mude seu comportamento externo. Ainda segundo eles, atividades de refatoração também minimizam os riscos de introdução a *bugs*.

Se as refatoração de código estiverem correlacionados com os sentimentos expressos pelos desenvolvedores, será uma evidência para dizer se atividades de refatoração são vistas por desenvolvedores como algo bom ou ruim para o projeto. Por exemplo, quando o código é refatorado, os desenvolvedores estão mais motivados (ou melhor humorados).

A questão de pesquisa que guia este trabalho é:

Q1) Quando se trabalha com refatoração de código, os sentimentos expressos tendem a ser mais positivos do que quando não se refatora código?

Este trabalho avalia projetos *open source*, a fim de responder esta pergunta. Foram utilizados métodos estatísticos para validar os resultados.

O restante do trabalho é dividido em: Capítulo 2 mostra os objetivos desse trabalho; Capítulo 3 fala sobre análise de sentimentos e refatorações de código; Capítulo 4 fala sobre os estudos que já existem na área; Capítulo 5 mostra como foram realizadas as coletas e análises dos dados; e o Capítulo 6 mostra os resultados encontrados.

2 OBJETIVOS

Neste Capítulo, serão apresentados o objetivo geral e os objetivos específicos deste trabalho.

2.1 Objetivo Geral

Identificar qual a relação entre as emoções expressas pelos desenvolvedores e fatores como aparição de refatorações de código (encontradas por ferramentas de revisão de código).

2.2 Objetivos específicos

- Identificar qual o sentimento mais expresso pelos desenvolvedores quando se refatora código;
- Identificar se as mensagens dos *commits* que tiveram refatoração de código tendem a ter sentimentos positivos.

3 FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo, serão apresentados os principais conceitos presentes neste trabalho.

3.1 Análise de sentimentos

Análise de sentimentos é o processo de atribuir um valor de humor quantitativo (positivo ou negativo) a um fragmento de texto (GUZMAN; AZOCAR; LI, 2014). Pesquisas recentes associam fatores humanos coletados através de métodos que se utilizam da análise de sentimentos (como humor e emoções) a resolução de problemas (GRAZIOTIN; WANG; ABRAHAMSSON, 2014), linguagens de programação (GUZMAN; AZOCAR; LI, 2014), resultados do processo de *build* (SOUZA; SILVA, 2017) e a tarefas de refatoração de código (SINGH; SINGH, 2017).

Existem várias formas de se identificar os sentimentos contidos em texto. Uma delas é o classificador *naive Bayes*, que necessita de um conjunto de mensagens previamente classificadas por um especialista em sentimentos em textos. Porém, nessa abordagem existe o risco de a opinião sobre sentimentos positivos ou negativos do especialista influenciar nos resultados. Neste estudo, foi utilizada a ferramenta SentiStrength (THELWALL et al., 2010), pois não necessita de um conjunto de treinamento a ser feito manualmente pelo especialista.

SentiStrength é uma ferramenta de extração de sentimentos especializada em lidar com textos curtos e de baixa qualidade, já utilizada em trabalhos anteriores (SINHA; LAZAR; SHARIF, 2016), (SINGH; SINGH, 2017), (SOUZA; SILVA, 2017) e (GUZMAN; AZOCAR; LI, 2014). Estudos mostraram que SentiStrength tem uma boa acurácia para mensagens de texto curtas para dados do Twitter (THELWALL; BUCKLEY; PALTOGLOU, 2012).

SentiStrength utiliza um dicionário de *tokens*, onde associa a cada *token* um *score*. Palavras com sentimentos negativos recebem uma pontuação entre -1 e -5, sendo -1 baixo sentimento negativo e -5 palavras com sentimentos extremamente negativos. Já as palavras com sentimentos positivos recebem uma pontuação entre 1 e 5, sendo 1 baixo sentimento positivo e 5 palavras com sentimentos extremamente positivos. Para se analisar uma frase, a ferramenta atribui pontuações às palavras da frase que estão presentes no dicionário, e a pontuação final da frase é um par contendo o maior valor positivo e o maior valor absoluto negativo. Por exemplo, na frase "I love you but hate the current political climate.", a ferramenta associará uma pontuação igual a 3 e -4 respectivamente a "love"e "hate". Logo, o *score* final da frase será o par (3, -4).

Para encontrar o sentimento final de uma mensagem de *commit*, foi feita uma soma da pontuação positiva e da pontuação negativa fornecida pelo *SentiStrength*, assim como é feito em (SINHA; LAZAR; SHARIF, 2016). Dessa forma, uma mensagem de *commit* pode ser classificada como positiva (*score* final maior que 0), negativa (*score* final menor que 0) ou neutra (*score* final igual a 0). A Tabela 2 mostra alguns exemplos de *commits* e seus *scores* finais.

3.2 Análise de refatorações de código

Refatoração de código é o processo de alterar um sistema de software de maneira que não altere o comportamento externo do código, apenas melhorando sua estrutura interna (FOWLER; BECK, 1999). Um trecho de código é refatorado quando está complexo de se entender, realmente há um erro no seu funcionamento, há a necessidade de melhorar o desempenho do sistema ou outros motivos. Um exemplo de tarefa de refatoração é quando o desenvolvedor move um método de uma classe para outra, com o objetivo de remover dependências excessivas entre essas classes. Essa é uma tarefa de refatoração chamada *Move Method* (FOWLER; BECK, 1999).

Neste estudo, para analisar as refatorações em projetos, foi decidido realizar a coleta de refatorações para cada *commit*, e dessa forma, foi possível classificar os *commits* em "há refatorações"e em "não há refatorações", com a ajuda da ferramenta *RefactoringMiner* (TSANTALIS et al., 2018), que detecta 11 tipos de atividades de refatoração. Em seguida, foi relacionado cada *commit* ao sentimento contido na mensagem. *RefactoringMiner* implementa uma versão leve do algoritmo UMLDiff (XING; STROULIA, 2005) para modelos orientados a objetos. Esse algoritmo é usado para inferir o conjunto de classes, métodos e campos adicionados, excluídos ou movidos entre revisões de código sucessivas. Depois de executar esse algoritmo, um conjunto de regras é usado para identificar diferentes tipos de refatoração. A ferramenta *RefactoringMiner* pode ser utilizada independente de sistema operacional ou IDE. Tsantalis *et al.* (TSANTALIS et al., 2013) identificaram uma precisão de 96,4% no uso da ferramenta, e mostraram que há uma taxa muito baixa de falsos positivos.

3.3 Testes de hipótese

Depois de realizar a coleta das médias de refatorações de código, é necessário verificar se há alguma significância estatística com os sentimentos encontrados nas mensagens de

commit. No caso deste trabalho, os testes de hipótese servem para verificar se existe significância estatística na relação entre sentimentos e refatorações de código. Os testes utilizados foram: *Student1s t-Test* (WINTER, 2013) e *Wilcoxon Test* (GEHAN, 1965).

O teste *Student's t-Test*, em que a hipótese nula é que a média das diferenças da população é igual a zero, ajudou a encontrar qual sentimento os projetos tendem a ter (positivo ou negativo). Já o teste de Wilcoxon, verifica se a distribuição das diferenças é simétrica em torno do valor zero.

Segundo Montgomery et al. (2000), uma hipótese estatística é uma afirmação sobre os parâmetros de uma ou mais populações. No estágio de análise de dados em experimentos comparativos, um dos métodos fundamentais é a estimação de parâmetros com teste de hipóteses estatísticas e com intervalo de confiança.

Teste de hipótese é um procedimento que leva a uma decisão acerca de uma hipótese particular. Nos testes de hipótese, existem as hipóteses nula e alternativa. Hipótese nula é uma interpretação sobre o conjunto observado onde geralmente está acompanhado por um sinal de igualdade, e o que tenta-se fazer é sempre negar a hipótese nula. Já a hipótese alternativa é o contrário da hipótese nula, em que busca-se aceitá-la. Um conjunto de dados pode ter mais de uma hipótese alternativa. Procedimentos de teste de hipóteses se apoiam no uso de informações de uma amostra aleatória proveniente da população de interesse (MONTGOMERY; RUNGER; CALADO, 2000). Uma hipótese é rejeitada se essas informações aleatórias forem inconsistentes. No caso em que a amostra é consistente com a hipótese, não se pode rejeitar a hipótese. Testar uma hipótese consiste em considerar uma amostra aleatória, computar uma estatística de teste a partir dos dados amostrais e então usar a estatística de teste para se tomar uma decisão a respeito da hipótese nula (MONTGOMERY; RUNGER; CALADO, 2000).

4 TRABALHOS RELACIONADOS

Este Capítulo apresenta trabalhos que relacionam análise de sentimentos com fatores na área do desenvolvimento de software, apresentando suas semelhanças e diferenças no que pretende ser desenvolvido.

Guzman *et al.* (2014) fazem uma análise de sentimentos também em mensagens de *commit* de 90 projetos de código-fonte aberto no GitHub. Eles analizam um total de 60425 mensagens de *commit*. As questões propostas pelos autores são diferentes das questões de pesquisa investigadas neste trabalho. O que é proposto no trabalho relacionado é analisar sentimentos e relacionar com a linguagem de programação que está sendo utilizada, com o dia da semana em que o *commit* foi feito, com a distribuição geográfica do time e com a aprovação do projeto. A forma de classificar os sentimentos em uma mensagem nesse trabalho é diferente da que é proposta no trabalho atual. Enquanto Guzman *et al.* consideram as duas pontuações fornecidas pela ferramenta SentiStrength[1], neste trabalho, essas pontuações são calculadas e apenas um sentimento final é associado a cada frase.

Sinha *et al.* (2016) buscam relacionar análise de sentimentos com o dia da semana e com o número de arquivos alterados em um *commit*. Além disso, eles buscam saber qual sentimento (positivo ou negativo) em geral os desenvolvedores expressam. Nesse estudo foram analisadas 2.130.474 mensagens de *commit* de 28.466 projetos. A forma de classificar o sentimento final de uma mensagem de *commit* é semelhante a deste trabalho, que consiste em fazer um cálculo entre a pontuação de sentimentos positivo e de sentimento negativo para chegar no valor de *score* do sentimento final da mensagem de *commit*.

Souza e Silva (2017) relacionam o resultado final do processo de *build* aos sentimentos expressos. Eles buscam saber se os *commits* com sentimentos negativos tendem a resultar na quebra de *build*; se *buids* que quebraram tendem a ter sentimentos negativos associados a ela; e qual sentimento, em geral, é expresso quando os desenvolvedores mencionam o servidor de integração contínua. Eles analisaram 101.6017 mensagens de commit de 1.262 projetos.

Singh e Singh (2017) , assim como proposto neste trabalho, buscam relacionar atividades de refatoração de código com análise de sentimentos. Eles analisaram 3171 mensagens de *commit* com refatorações e concluíram que, quando se trabalha com refatorações de código, os desenvolvedores expressam mais sentimentos negativos do que positivos. A diferença para

[1] <http://sentistrength.wlv.ac.uk>

Quadro 1 – Comparativo de trabalhos

Trabalho	**Classificação única para cada commit**	**Testes de hipótese**	**Fatores analisados**
Guzman et al. (2014)	Não	Wilcoxon test	Linguagem de programação; dia da semana; distribuição geográfica; aprovação do projeto
Sinha et al. (2016)	Sim	Pearson's correlation test	Dia da semana; quantidade de arquivos alterados
Souza e Silva (2017)	Não	Mann-Whitney U test; chi-squared test	Quebra de build; Menção do servidor de integração contínua;
Singh e Singh (2017)	Sim	Mann-Whitney-Wilcoxon	Refatorações de código
Este trabalho	Sim	Wilcoxon Test; Student's t-Test	Refatorações de código (todo o conjunto)

Fonte – Elaborado pelo autor

este trabalho é que, ao invés de analisar todos os *commits* com refatorações, (SINGH; SINGH, 2017) criam uma ferramenta própria para tentar prever quais *commits* tem refatorações, para só depois disso, usar a ferramenta *Refactoring Minner*. Isso pode acarretar em uma ameaça à validade do trabalho, visto que alguns dados estão sendo dispensados. Neste trabalho, são analisados todos os *commits* com refatorações.

O Quadro 1 mostra a diferença entre os trabalhos relacionados a este trabalho.

5 PROCEDIMENTOS METODOLÓGICOS

Neste Capítulo é descrito como os projetos foram selecionados, apresentando os critérios de inclusão e exclusão; como foi realizada a coleta das refatorações a partir dos projetos selecionados; e como foi feita a análise de sentimentos a partir das mensagens dos *commits*.

5.1 Coleta dos dados dos projetos

Para iniciar o estudo, foram selecionados oito projetos do GitHub: *Dropwizard*[1], *Guava*[2], *Kafka*[3], *Mockito*[4], *RxJava*[5], *Tutorials*[6], *Netty*[7] e *Java Tron*[8]. Uma primeira coleta foi realizada em Abril de 2018, para o início do estudo, e uma outra foi realizada em Julho de 2018 para explorar mais projetos. Os projetos foram selecionados com base em 4 critérios, que são mostrados e explicados na Seção 5.1.1. Foram clonados os repositórios dos projetos no *Github*, para que caso seja necessário refazer alguma das análise, novas mensagens de *commit* não influenciem nos resultados. Os projetos foram escolhidos com base na quantidade de *commits* que continham com o intuito de analisar projetos com quantidades de *commit* equivalentes. Foram selecionados para este estudo, projetos considerados pequenos em relação à quantidade de *commits*.

A quantidade total de *commits* em cada projeto é mostrada na Tabela 1. A Tabela 1 mostra também a quantidade de *commits* válidos para cada projeto. Nem todos os *commits* dos projetos foram analisados. Foram definido alguns critérios, para um *commit* poder entrar na análise:

- Deve ter algum sentimento expressado na mensagem (não pode ter pontuação positiva igual a 1 e ao mesmo tempo pontuação negativa igual a -1, pois de acordo com a ferramenta *SentiStrength*, isso indica que não há nenhum sentimento expresso na mensagem)
- Não pode ser *commit* de *merge* (*commits* de merge normalmente não fazem nenhuma modificação no código, apenas é responsável pela junção de outros *commits*. Além disso, as mensagens nesses tipo de *commit* seguem um padrão, e não expressam nenhum

[1] <https://github.com/dropwizard/dropwizard>
[2] <https://github.com/google/guava>
[3] <https://github.com/apache/kafka>
[4] <https://github.com/mockito/mockito>
[5] <https://github.com/ReactiveX/RxJava>
[6] <https://github.com/eugenp/tutorials>
[7] <https://github.com/netty/netty>
[8] <https://github.com/tronprotocol/java-tron>

Tabela 1 – Quantidade total de *commits* por projeto

Projeto	**Número de commits**	**Número de commits válidos**
Dropwizard	4482	637
Guava	4676	1478
Java Tron	6076	515
Kafka	5298	2123
Mockito	4653	2193
Netty	8880	3069
RxJava	5330	1360
Tutorials	7817	604
Total	47212	11979

Fonte – Elaborado pelo autor.

sentimento)

- Foi encontrado um caso específico de um *commit*[9] que desviava dos demais, e acabava influenciando nos resultados. Foi feita uma análise nesse *commit* em específico e notou-se que o desenvolvedor havia adicionado todos os testes do projeto em um só *commit*. Como medida preventiva, para que esse caso não influenciasse nos resultados, foi decidido remover este commit da análise.

5.1.1 Critérios da escolha dos projetos

A seleção dos projetos para as análises se baseou em 4 critérios:

1. Mensagens de *commit* escritas em inglês;
2. A linguagem de programação do projeto ser Java;
3. O projeto estar no *Trending* do GitHub na data da análise;
4. Quantidade de *commits* entre 4.400 e 8.000.

O primeiro critério, que é a seleção de projetos onde as mensagens de *commit* são escritas em inglês é devido a ferramenta SentiStrength analisar melhor mensagens nesse idioma. O dicionário da ferramenta em inglês é melhor e mais rico que em outros idiomas. O segundo critério está presente pelo fato de que a ferramenta utilizada para a coleta de refatorações (*RefactoringMinner* só analisa projetos em Java. A busca de projetos apenas do *Trending* do GitHub permite responder a questão de pesquisa para o contexto atual de desenvolvimento, para projetos recentes, já que no *Trending* estão os projetos que são tendências no momento. O intervalo de 4,400 a 8,000 referente ao 4º critério foi devido a incapacidade de uma das ferramentas (*RefactoringMiner*) de analisar grandes quantidades de *commits* e também porque

[9] <https://github.com/jordaos/guava/commit/b49c1296eb569afcaee5b521ad2d0c7afd921d8f>

era uma quantidade razoável de se trabalhar e analisar os dados.

5.2 Coleta de refatorações

Para analisar as refatorações nos projetos escolhidos, foram coletadas as refatorações de cada *commit*. Foi feita uma classificação dos *commits* em "há refatorações"e em "não há refatorações". Isso foi feito com a ajuda da ferramenta *RefactoringMiner* (TSANTALIS et al., 2018), que detecta onze tipos de atividades de refatoração. A ferramenta implementa uma versão otimizada do algoritmo UMLDiff (XING; STROULIA, 2005) para modelos orientados a objetos. Esse algoritmo é usado para inferir o conjunto de classes, métodos e campos adicionados, excluídos ou movidos entre revisões de código sucessivas.

Depois de executar esse algoritmo, um conjunto de regras é usado pela ferramenta *RefactoringMiner* para identificar diferentes tipos de refatoração. *RefactoringMiner* pode ser utilizada independente de sistema operacional ou IDE. Tsantalis *et al.* (TSANTALIS et al., 2013) identificaram uma precisão de 96,4% no uso da ferramenta, e mostraram que há uma taxa muito baixa de falsos positivos. Já Silva *et al.* (SILVA; TSANTALIS; VALENTE, 2016) identificaram um *recall* de 93% no uso da ferramenta.

5.3 Análise de sentimentos

Para analizar os sentimentos das mensagens de *commit*, foi utilizada a versão Java da ferramenta SentiStrength (THELWALL et al., 2010), uma ferramenta de extração de sentimentos especializada em lidar com textos curtos e de baixa qualidade, também usada em trabalhos anteriores (SINHA; LAZAR; SHARIF, 2016; SINGH; SINGH, 2017; SOUZA; SILVA, 2017; GUZMAN; AZOCAR; LI, 2014). Estudos mostraram que SentiStrength tem uma boa acurácia para mensagens de texto curtas para dados do Twitter (THELWALL; BUCKLEY; PALTOGLOU, 2012).

Em um estudo exploratório, foi investigada a melhor forma de se analisar sentimentos em mensagens curtas. Foram comparadas três formas de se classificar as mensagens: o uso da ferramenta SentiStrength com o dicionário original; o uso da ferramenta SentiStrength com o dicionário modificado com palavras do contexto de engenharia de software; e o uso do classificador *Naive Bayes*, que é o modelo de classificação mais simples, pois assume que todos os atributos de treinamento são independentes uns dos outros (LEWIS, 1998).

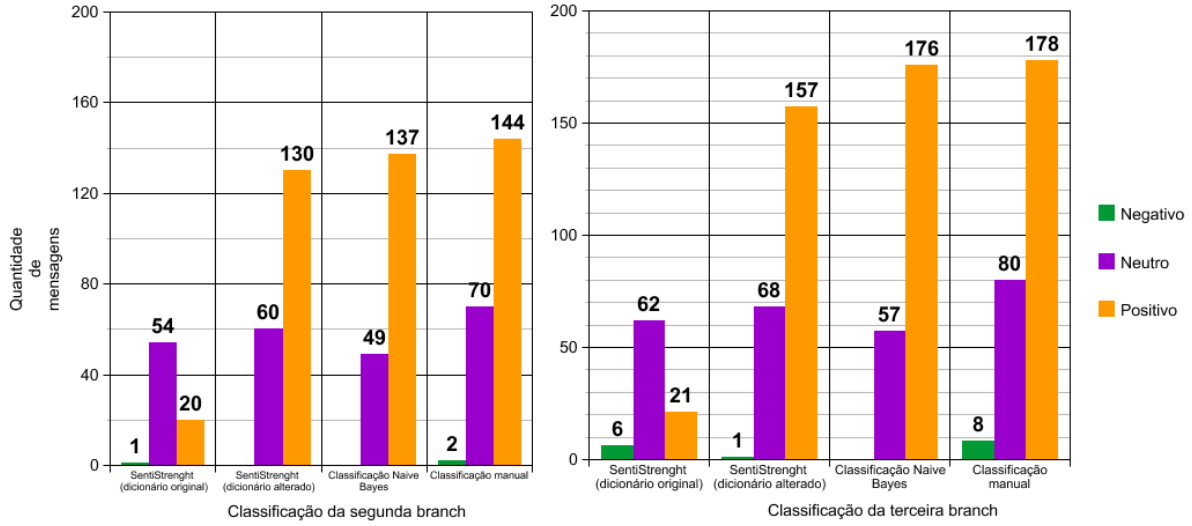
Tabela 2 – Exemplos de mensagens de *commit*

Score	**Mensagem**	***Score* Final**
{4, -1}	Follow @fleaflicker's excellent[4]advice .	3
{3, -2}	Relax[3]the constraints[-2]on ConfiguredBundle .	1
{1, -5}	Definitely hating [-4][-1 booster word]#320 .	-4
{2, -2}	Correct default[-2]value[2] of rootPath	0

Fonte – Elaborado pelo autor.

Para se chegar a uma conclusão, os resultados das análises para cada método foram comparados com a classificação manual de duas *releases* iniciais do projeto *Hadoop*[10] feitas por um especialista. Em resumo, os resultados mostraram que o classificador *Naive Bayes* tem precisão maior para detectar sentimentos positivos nas mensagens, enquanto que para sentimentos negativos, a ferramenta SentiStrength com o dicionário modificado (adaptado ao contexto de engenharia de software, excluindo e adicionando palavras específicas da área) apresenta precisão melhor. A ferramenta SentiStrength com o dicionário original teve resultados piores comparado aos demais métodos de classificação. Os resultados posem ser vistos na Figura 1.

Figura 1 – Resultados das classificações para duas releases do projeto Hadoop



Fonte – Elaborado pelo autor.

Dessa forma, este trabalho optou por utilizar a ferramenta SentiStrength com uma

[10] <https://github.com/apache/hadoop>

modificação no dicionário de classificação (SOUZA; SILVA, 2017). A utilização dessa ferramenta não torna necessária a rotulação manual dos *commits* para montar um conjunto de treino, assim como é feito quando se utilizam métodos de classificação como o classificador *Naive Bayes*, onde corre o risco de a opinião sobre sentimentos positivos ou negativos do especialista influencie nos resultados, caso haja um grupo pequeno de especialistas para fazerem essas classificações.

Para encontrar o sentimento final de uma mensagem de *commit*, foi feita uma soma da pontuação positiva e da pontuação negativa fornecida pelo *SentiStrength*, assim como é feito em (SINHA; LAZAR; SHARIF, 2016). Dessa forma, uma mensagem de *commit* pode ser classificada como positiva (*score* final maior que 0), negativa (*score* final menor que 0) ou neutra (*score* final igual a 0). Na Tabela 2 é possível ver o *score* final de algumas mensagens de *commit*.

6 RESULTADOS

Neste Capítulo serão apresentados os resultados encontrados a partir do estudo.

6.1 Análise de Sentimentos e Refatoração de Código

O estudo foi realizado coletando as refatorações de oito projetos *Open Source* e analisando os sentimentos contidos nas mensagens dos *commits* desses projetos. A quantidade de *commits* analisada para cada um dos seis projetos é mostrada na Tabela 4, com suas respectivas porcentagens. Para responder a pergunta 1 (A aparição de anomalias de código causa o aumento de sentimentos negativos?) os commits foram divididos em dois grupos: "*Commits* com refatorações"e "*Commits* sem refatorações".

Tabela 3 – Sentimento através dos commits

Sentimento	**Score final de sentimento**	**Com refatorações**		**Sem refatorações**	
		Número de Commits	**Porcentagem do Sentimento**	**Número de Commits**	**Porcentagem do Sentimento**
Negativo	-4	0	44,6%	3	53,67%
	-3	11		40	
	-2	125		541	
	-1	877		4627	
Neutro	0	340	14,97%	1027	10,58%
Positivo	1	864	40,42%	3252	35,75%
	2	54		215	
	3	0		3	
	4	0		1	
Total	-	2271	-	9709	-

Fonte – Elaborado pelo autor.

De modo similar ao procedimento de Sinha et al. (2016) , os *commits* com o par de scores {-1, 1} foram descartados, pois segundo a ferramenta SentiStrength, quando acontece isso, significa que na mensagem não há nenhum sentimento positivo nem negativo. Desta forma, foram considerados apenas os commits que tinha valor positivo de no mínimo 2, ou valor negativo no máximo -2. Na Tabela 3, é apresentada a porcentagem encontrada para os sentimentos negativos, positivos e neutros. Quando se trata de *commits* com refatorações há um balanceamento entre sentimentos positivos e negativos, com uma porcentagem de 40,42% e 44,6% respectivamente. Já no restante dos *commits* (sem refatorações), o sentimento negativo domina, aparecendo em 53,67% dos *commits*, enquanto os sentimentos positivos apareceram em 35,75% das vezes.

Tabela 4 – Número de commits com e sem refatorações

Projeto	**Commits com refatorações**	**Commits sem refatorações**
Dropwizard	73 (11%)	564 (89%)
Guava	290 (19,6%)	1189 (80,4%)
Java-tron	42 (8,2%)	473 (91,8%)
Kafka	319 (15%)	1804 (85%)
Mockito	531 (24%)	1662 (76%)
Netty	672 (21,9%)	2397 (78,1%)
RxJava	257 (18,9%)	1103 (81,1%)
Tutorials	87 (14,4%)	517 (85,6%)
Total	2271 (19%)	9709 (81%)

Fonte – Elaborado pelo autor.

De fato, o Wilcoxon *test* rejeitou a hipótese de que a distribuição dos *scores* dos *commits* sem refatoração é simétrica em relação ao zero (na Tabela 3 *p-value* < 2,2e-16), embora também rejeite esta simetria para os *commits* com refatoração (*p-value* < 0,0002) se considerarmos 95% de significância. Além disso, foi aplicado o *Student's t-Test* para testar se o score médio dos *commits* positivos é igual ao score médio dos *commits* negativos em módulo. E com significância de 95%, esta hipótese foi rejeitada nos dois casos (com e sem refatoração).

Em resumo, foram encontradas evidências de que, em geral, os sentimentos dos projetos tendem a ser negativos, mas quando se trabalha com tarefas de refatorações, os *commits* tendem a ser menos negativos. Porém, foram encontrados casos de projetos onde a média dos sentimentos é positiva e quando se trabalhava com atividades de refatorações as médias são mais positivas ainda. A partir desses resultados, foram encontradas evidências de que tarefas de refatoração de código estão associadas a um bem estar nos desenvolvedores, fazendo com que eles expressem menos sentimentos negativos ou mais sentimentos positivos, o que é um passo inicial para responder a primeira questão de pesquisa deste trabalho.

6.1.1 Resultados por projeto

A análise foi realizada também para cada projeto separadamente, utilizando-se dos mesmos testes de hipótese e mesmas questões. Os resultados encontrados para alguns projetos não foram os mesmos encontrados no geral. A Tabela 5 mostra os resultados de *p-value* encontrados nos testes e as médias para cada um dos projetos separadamente. Quando o *p-value* aparece em negrito, significa que a hipótese nula foi rejeitada (*p-value* <0.05).

Em relação aos *commits* com refatorações, nos projetos *Dropwizard* e *Java-tron* não

Tabela 5 – Testes por projetos nos *commits* com e sem Refatorações

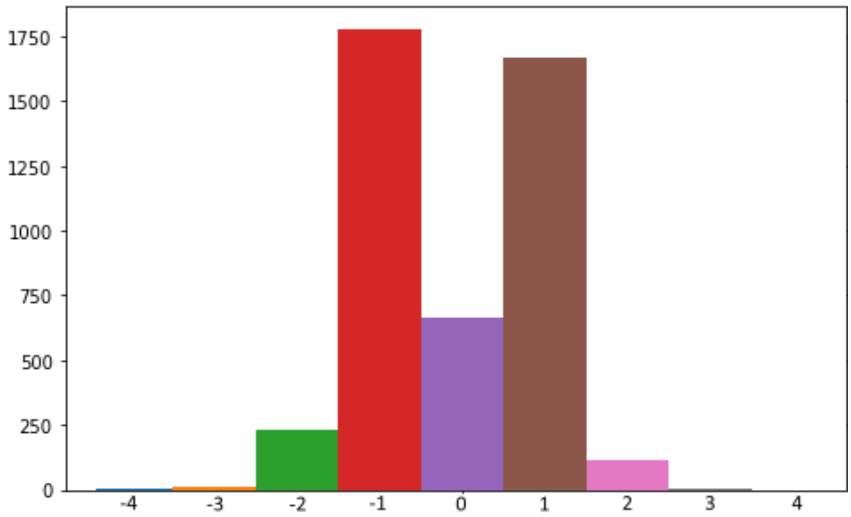
	Com refatorações			Sem refatorações		
Projetos	**Média de sentimento**	**Wilcoxon signed-rank test (p-value)**	**Student's T-Test (p-value)**	**Média de sentimento**	**Wilcoxon signed-Rank test (p-value)**	**Student's T-Test (p-value)**
Dropwizard	-0.19	0.16	0.13	-0.37	**1.291e-13**	**1.407e-14**
Guava	-0.33	**4.646e-07**	**2.839e-07**	-0.34	**<2.2e-16**	**<2.2e-16**
Kafka	-0.41	**6.102e-12**	**6.177e-13**	-0.51	**<2.2e-16**	**<2.2e-16**
Java-tron	-0.26	0.12	0.12	-0.34	**7.723e-11**	**9.436e-11**
Mockito	0.33	**9.68e-14**	**1.405e-14**	0.37	**<2.2e-16**	**<2.2e-16**
Netty	-0.28	**1.161e-10**	**2.008e-11**	-0.43	**<2.2e-16**	**<2.2e-16**
RxJava	0.4	**4.637e-12**	**3.983e-13**	0.11	0.0002	0.0002
Tutorials	-0.29	**0.01**	**0.01**	-0.29	**4.4e-09**	**2.118e-09**
Total	-0.08	**0.0002**	**0.0002**	-0.22	**<2.2e-16**	**<2.2e-16**

Fonte – Elaborado pelo autor.

foi possível rejeitar as hipóteses nulas em ambos os testes. Nos outros casos as hipóteses nulas foram refutadas, porém nos projetos *Mockito* e *RxJava*, as médias são positivas, o que indica que nesses casos, os *commits* quando há refatorações tendem a ter sentimentos positivos. Ainda nesses dois projetos, quando se trata de commits sem refatorações, as médias são positivas, o que indica que nesses projetos os desenvolvedores expressam sentimentos positivos independente de se trabalhar ou não com refatorações de código. Para os demais casos, os resultados acompanham o resultado encontrado no geral.

6.2 Análise de sentimentos e Anomalias de código

Figura 2 – Divisão de *commits* com anomalias de código de acordo com o sentimento



Fonte – Elaborado pelo autor.

Além do estudo da relação entre refatorações de código e os sentimentos expressos

pelos desenvolvedores, também foi feita uma comparação entre os sentimentos e anomalias de código, que são trechos no código-fonte que fogem dos padrões de programação para a linguagem especificada (AYEWAH et al., 2008). Para realizar esta análise, os *commits* que adicionavam anomalias no código foram classificados de acordo com seu *score* final de sentimento. A Figura 2 mostra como ficou essa divisão. A partir desses dados, não foi possível chegar a nenhuma conclusão sobre qual o sentimento mais expresso pelos desenvolvedores quando são introduzidas anomalias de código no projeto, já que o gráfico ficou de forma geral, simétrico, e isso implica que, não importa o sentimento expresso (positivo ou negativo), o usuário sempre adiciona anomalias no código.

De forma similar ao estudo acima, foram feitas outras pesquisas a fim de encontrar a relação do sentimento com outros fatores. No entanto, não foi possível encontrar a relação de nenhum desses outros fatores com os sentimentos expressos pelos desenvolvedores, os resultados são equivalentes aos da pesquisa do parágrafo anterior. Os fatores estudados foram:

- Linhas de código alteradas
- Anomalias de código resolvidas
- Anomalias de código não resolvidas

7 CONSIDERAÇÕES FINAIS

Neste trabalho foram analisados os sentimentos de oito projetos *open-source*, os quais tinham seus códigos-fonte disponíveis no Github. Os sentimentos encontrados foram cruzados com as refatorações de código que haviam sido feitas no projeto. Foram encontradas evidências de que quando o desenvolvedor refatora código, o sentiment negativo (que na maioria dos casos é predominante nos projetos tende a diminuir. Isso pode ser uma evidência de que quado o desenvolvedor refatora código, causa um bem-estar nele, já que está contribuindo positivamente para o projeto.

Como trabalhos futuros, pretende-se fazer esta mesma análise para um conjunto maior de projetos. Além disso, pretende-se fazer análises mais profundas, a fim de encontrar alguma correlação entre as anomalias do código e o sentimento expresso.

REFERÊNCIAS

AYEWAH, N.; HOVEMEYER, D.; MORGENTHALER, J. D.; PENIX, J.; PUGH, W. Using static analysis to find bugs. **IEEE software**, IEEE, v. 25, n. 5, 2008.

CHOUDHURY, M. D.; COUNTS, S. Understanding affect in the workplace via social media. In: **Proceedings of the 2013 Conference on Computer Supported Cooperative Work**. New York, NY, USA: ACM, 2013. (CSCW '13), p. 303–316. ISBN 978-1-4503-1331-5. Disponível em: <http://doi.acm.org/10.1145/2441776.2441812>. Acesso em: 14/05/2018.

FOWLER, M.; BECK, K. **Refactoring**: improving the design of existing code. [S.l.]: Addison-Wesley Professional, 1999.

GEHAN, E. A. A generalized wilcoxon test for comparing arbitrarily singly-censored samples. **Biometrika**, Oxford University Press, v. 52, n. 1-2, p. 203–224, 1965.

GRAZIOTIN, D.; WANG, X.; ABRAHAMSSON, P. Happy software developers solve problems better: psychological measurements in empirical software engineering. **PeerJ**, v. 2, p. e289, mar. 2014. ISSN 2167-8359. Disponível em: <https://doi.org/10.7717/peerj.289>. Acesso em: 14/05/2018.

GUZMAN, E.; AZOCAR, D.; LI, Y. Sentiment analysis of commit comments in github: An empirical study. In: **Proceedings of the 11th Working Conference on Mining Software Repositories**. New York, NY, USA: ACM, 2014. (MSR 2014), p. 352–355. ISBN 978-1-4503-2863-0. Disponível em: <http://doi.acm.org/10.1145/2597073.2597118>. Acesso em: 14/05/2018.

LEWIS, D. D. Naive (bayes) at forty: The independence assumption in information retrieval. In: NÉDELLEC, C.; ROUVEIROL, C. (Ed.). **Machine Learning: ECML-98**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 4–15. ISBN 978-3-540-69781-7.

MONTGOMERY, D. C.; RUNGER, G. C.; CALADO, V. **Estatística Aplicada E Probabilidade Para Engenheiros .** [S.l.]: Grupo Gen-LTC, 2000.

SILVA, D.; TSANTALIS, N.; VALENTE, M. T. Why we refactor? confessions of github contributors. In: ACM. **Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering**. [S.l.], 2016. p. 858–870.

SINGH, N.; SINGH, P. How do code refactoring activities impact software developers. In: IEEE. **2017 24th Asia-Pacific Software Engineering Conference (APSEC)**. [S.l.], 2017. p. 648–653.

SINHA, V.; LAZAR, A.; SHARIF, B. Analyzing developer sentiment in commit logs. In: ACM. **Proceedings of the 13th International Conference on Mining Software Repositories**. New York, NY, USA: ACM, 2016. p. 520–523.

SOUZA, R.; SILVA, B. Sentiment analysis of travis ci builds. In: **2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)**. [S.l.: s.n.], 2017. p. 459–462.

THELWALL, M.; BUCKLEY, K.; PALTOGLOU, G. Sentiment strength detection for the social web. **Journal of the Association for Information Science and Technology**, Wiley Online Library, v. 63, n. 1, p. 163–173, 2012.

THELWALL, M.; BUCKLEY, K.; PALTOGLOU, G.; CAI, D.; KAPPAS, A. Sentiment strength detection in short informal text. **Journal of the Association for Information Science and Technology**, Wiley Online Library, v. 61, n. 12, p. 2544–2558, 2010.

TSANTALIS, N.; GUANA, V.; STROULIA, E.; HINDLE, A. A multidimensional empirical study on refactoring activity. In: IBM CORP. **Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research**. Riverton, NJ, USA: IBM Corp., 2013. p. 132–146.

TSANTALIS, N.; MANSOURI, M.; ESHKEVARI, L. M.; MAZINANIAN, D.; DIG, D. Accurate and efficient refactoring detection in commit history. In: **Proceedings of the 40th International Conference on Software Engineering**. New York, NY, USA: ACM, 2018. (ICSE '18), p. 483–494. ISBN 978-1-4503-5638-1. Disponível em: <http://doi.acm.org/10.1145/3180155.3180206>. Acesso em: 14/05/2018.

WINTER, J. C. D. Using the student's t-test with extremely small sample sizes. **Practical Assessment, Research & Evaluation**, v. 18, n. 10, 2013.

XING, Z.; STROULIA, E. Umldiff: an algorithm for object-oriented design differencing. In: ACM. **Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering**. New York, NY, USA: ACM, 2005. p. 54–65. 594045.