Classificação de Issues do Github com Relação a Segurança

1st Bruno Gonçalves de Oliveira
Universidade Federal do Paraná (UFPR)
Curitiba– PR – Brasil
bruno.mphx2@gmail.com

2nd Diogo Cezar Teixeira Batista
Universidade Federal do Paraná (UFPR)
Curitiba– PR – Brasil
diogocezar@utfpr.br

***Abstract*—As *issues* do GitHub representam grande parte da evolução dos projetos *OpenSource*. Este trabalho propõe uma solução para classificar *issues* que podem ou não estar relacionadas a segurança da informação. As mensagens serão analisadas utilizando as palavras frequentes em textos referentes à segurança. Utiliza-se a técnica de *Bag-of-Words* em conjunto com *TF-IDF* para a criação dos vetores de representação. Na sequência, múltiplos classificadores foram utilizados, entre eles: svm, knn, naive_bayes, lda, logistic_regression, perceptron, tree e mlp. Os resultados obtidos apresentam uma acurácia superior a 80%.**

***Index Terms*—Issues, GitHub, Segurança da Informação, Classificadores, Aprendizagem de Máquina**

I. Introdução

O controle, gerenciamento e manutenção de arquivos, especialmente no âmbito do desenvolvimento de *software*, sempre foi um desafio. Problemas recorrentes como: backups não realizados, sobrescritas de arquivos, dificil manutenabilidade de projetos desenvolvidos em equipes, são motivações para a utilização de algum sistema de versionamento de arquivos. [1]

Dentre as várias ferramentas existenstes no mercado, como por exemplo: CVS, Subversion, TFS, Mercurial, o Git se destaca por ter sido amplamente utilizado pela comunidade de desenvolvimento, com o advento da popularização dos projetos OpenSource. Estes projetos se consolidaram em com ferramentas como o GitHub que é uma plataforma para versionamento, gerenciamento e colaboração de projetos, que utiliza o Git como base.

São várias as possibilidades que essas ferramentas proporcionam para verionamento dos projetos, no GitHub, por exemplo, existe uma sessão de *issues* (problemas) na qual, os colaboradores de um projeto *Open Source* podem cadastrar possíveis *bugs*, melhorias, ou novas features para os projetos compartilhados entre os usuários colaboradores.

Na descrição dessas *issues* feita através do preenchimento de um campo de texto, é possível identificar qual é o contexto em que se aplica determinada correção.

Eventualmente, dentre as correções realizadas nestes projetos, são identificados ajustes relacionados a segurança. Essa issues, quando consideradas críticas, podem ser analisadas por outros especialistas para uma arguição mais detalhada e um possível aprimoramento.

Mas como identificar quais são os ajustes de um projeto que estão relacionados com segurança? Como separar estes ajustes para que especialistas possam analisar os cógidos? Essas perguntas guiam a motivação para o desenvolvimento deste trabalho.

Propõe-se a criação de uma ferramenta que utilize técnicas de aprendizagem de máquina para a criação de um oráculo classificador que consiga analisar as palavras contidas em uma mensagem de issue, e classificar se esta issue está ou não relacionada à uma implementação de segurança.

Para criação de uma base de dados de treinamento e testes, duzentas *issues* foram extraídas e classificadas dos projetos *OKHTTP*, *jgit* e *couchbase*.

Para a automatização dos experimentos, foi criado um sistema *orquestrador* no qual é possível definir as instruções para a execução e coleta das evidências de todo o fluxo para os diferentes classificadores, os utilizados neste experimento foram: svm, knn, naive_bayes, lda, logistic_regression, perceptron, tree e mlp. Para cada um dos testes, anotou-se a acurácia, f1score, tempo de execução e sua matriz de confusão.

Por fim, discute-se os resultados obtidos dos experimentos e possíveis trabalhos futuros.

II. Obtenção dos Dados

Projetos *OpenSource* são, por essência, públicos na Internet, e por este motivo, suas *issues* também estão dispostas de forma pública.

Para a geração dos dados de treinamento e testes, foram coletados e classificados (de forma manual) algumas *issues* aleatórias dos projetos: *https://github.com/square/okhttp/*, *https://github.com/eclipse/jgit* e *https://github.com/couchbase*

Estes dados foram organizados em arquivos no formato CSV que possuem basicamente 2 colunas. A primeira coluna indica se a *issue* é um tópico relacionado a segurança ou não, e a segunda coluna representa de fato o texto coletado.

Algumas linhas dos arquivos extraídos podem ser vistas no Código 1

```
1  security,We are able to access the SSLSocketFactory
       from the OkHttpClient...
2  security,PushObserver can be used to push
       serverinitiated HTTP/2 requests into an
       OkResponseCache...
3  not,Handle LOCKED in conversions.Motivation...
```

Código 1. CSV Exemplo com Base de Dados

Para a fonte de dados de treinamento foram inseridas 199 entradas, enquanto que para a base de teste foram utilizadas 211 entradas.

III. Pré-processamento

Após a obtenção dos dados, é importante a realização de algumas etapas de pré-processamento do texto, estes tratamentos procuram maximizar a representatividade das *issues* em questão de acordo com o seu significado.

Para isso, é necessário remover palavras que não representam o contexto das frases. Com esse objetivo, aplicou-se algumas técnicas para cada uma das frases da base de treinamento e testes.

As regras aplicadas foram:

1) Transformar todo texto em minúsculo;
2) Ignorar pontuações;
3) Corrigir palavras com ortografia incorreta;
4) Remover as chamadas *stop words* que não acrescentam informação aos textos, por exemplo: *of, a, in, on*.

Para a aplicação das regras criou-se uma função de preparação do *dataset* que foi aplicada tanto na base de testes quanto na base de treinamento.

IV. Extração de Características

Para a extração de características, utiliza-se uma técnica conhecida como *Bag-of-Words*. Nesta técnica, as sentenças são representadas através da identificação de suas palavras e a quantidade em que aparecem no texto. Por exemplo, considerando o seguinte text, escrito por *Charles Dickens*:

> It was the best of times,
> It was the wost of times,
> It was the age of wisdom,
> It was the age of follishness.

Deve-se considerar palavras únicas, neste caso teríamos um vocabulário com 10 palavras: [it, was, the, best, of, times, worst, age, wisdom, foolishness]

Na sequência deve-se criar os vetores que representam a quantidade de aparições de uma palavra no texto. Por exemplo, no documento "It was the age of wisdom" pode-se representar através de um vetor dado por: $[1, 1, 1, 0, 1, 0, 0, 1, 1, 0]$. Se novas frases, com palavras fora do vocabulário definido forem adicionadas, então essas serão descartadas.

Com os vetores de palavras formados com a identificação e a quantidade, é possível aplicar a técnica de *TF-IDF (term frequency-inverce document frequency)*. Nesta abordagem, valoriza-se o quão importante uma palavra é ao documento. O valor de *TF-IDF* aumenta proporcionalmente conforme o número de vezes a palavra aparece no texto e é compensado pelo número de textos na base de dados, ajustando o número da frequência das palavras.

Basicamente para cada uma das *issues* já sanitizadas aplica-se uma contagem das palavras mais relevantes para o problema em questão. Essas palavras foram obtidas a partir de uma contagem das palavras que mais apareciam nas *issues* que eram relacionadas à segurança.

Obteve-se as seguintes palavras: ['security', 'secure', 'vulnerable', 'leak', 'exception', 'crash', 'malicious', 'sensitive', 'user', 'authentication', 'protect', 'vulnerability', 'authenticator', 'auth', 'npe']

Após a aplicação destas técnicas, obtém-se 4 vetores que serão utilizados nas próximas fases do experimento. São eles: (x_train, y_train, x_test, y_test)

V. Orquestrador de Classificadores

Essencialmente em problemas que podem envolver diferentes classificadores, é bastante comum a realização dos experimentos em diferentes abordagens, utilizando classificadores diferentes, e para cada um dos classificadores, parâmetros diferentes.

Com o intúito de automatizar o processo de variação entre os experimentos, desenvolveu-se um sistmea orquestrador de classificadores.

Este sistema utiliza um arquivo no forma *JSON* para definir 2 principais blocos: configurações e experimentos.

No bloco de configurações, é possível definir quais serão os arquivos de entrada e saída para a realização dos experimentos. Já no bloco de experimentos define-se um *array* com todos os experimentos a serem realizados.

Um exemplo do arquivo de orquestração pode ser visto no Código 2.

```
1  {
2      "configs": {
3          "train": "data/train.csv",
4          "test": "data/test.csv",
5          "result_classifiers": "results/classifiers/
               tabulation_{timestamp}.csv",
6          "result_conf_mat": "results/conf_mat/"
7      },
8      "experiments": [
9          {
10         "classifier": "svm",
11         "parameters": {}
12         },
13         {
14         "classifier": "knn",
15         "parameters": {
16             "n_neighbors": 7
17         }
18         ...
19     ]
20 }
```

Código 2. JSON do Orquestrador

O sistema desenvolvido também armazena automaticamente em um arquivo no formato CSV os resultados para cada um dos experimentos. Os campos salvos são: Classifier, F1Score, Accuracy e Execution Time (s). Além disso, para cada execução são armazenadas as matrizes de confusão como imagens e também como arquivos CSV.

VI. Resultados Obtidos

VII. Conclusão

VIII. Trabalhos Futuros

Para aumentar a base de treinamento pode-se desenvolver um programa que extrai automaticamente informações de

issues de repositórios do GitHub marcados com uma label que contenha informações relacionadas a segurança.

Também é possível a criação de uma API que retorne se uma mensagem é ou não referente a segurança, que poderia ser utilizada em outros contextos.

IX. Código Fonte

O código fonte dos experimentos pode ser acessado em: https://github.com/bmphx2/aprendizagem-de-maquina

References

[1] S. Chacon and B. Straub, *Pro Git*. Apress, 2020.