Prediction of Developer Participation in Issues of Open Source Projects

Predição da Participação de Desenvolvedores em Tarefas em Projetos de Software Livre

André Luis Schwerz, Rafael Liberato,
Igor Scaliante Wiese, Igor Steinmacher
Universidade Tecnológica Federal do Paraná (UTFPR)
Câmpus Campo Mourão
{andreluis,liberato,igor,igorfs}@utfpr.edu.br

Marco Aurélio Gerosa, João Eduardo Ferreira
Universidade de São Paulo
Instituto de Matemática e Estatística (IME)
São Paulo - SP
{gerosa, jef}@ime.usp.br

***Abstract*—Developers of distributed open source projects use management and issues tracking tool to communicate. These tools provide a large volume of unstructured information that makes the triage of issues difficult, increasing developers' overhead. This problem is common to online communities based on volunteer participation. This paper shows the importance of the content of comments in an open source project to build a classifier to predict the participation for a developer in an issue. To design this prediction model, we used two machine learning algorithms called Naive Bayes and J48. We used the data of three Apache Hadoop subprojects to evaluate the use of the algorithms. By applying our approach to the most active developers of these subprojects we have achieved an accuracy ranging from 79% to 96%. The results indicate that the content of comments in issues of open source projects is a relevant factor to build a classifier of issues for developers.**

Content analysis; prediction model; issue tracking classifier; machine learning.

I. INTRODUÇÃO

A natureza descentralizada e colaborativa dos projetos de software livre provocou a necessidade do uso de repositórios e ferramentas de gestão e acompanhamento de tarefas para facilitar o planejamento e a comunicação entre os desenvolvedores.

A disponibilização dessas ferramentas à comunidade de desenvolvedores e usuários agiliza o processo de detecção de falhas e a elaboração de possíveis soluções, melhorando a qualidade do software [5][7]. Dentre as informações mantidas por essas ferramentas, destacamos a lista de tarefas que discute importantes assuntos sobre o projeto, tais como o desenvolvimento de novas funcionalidades, falhas, ajuda aos usuários, decisões de projeto etc.

O conteúdo da lista de tarefa forma uma base de conhecimento do projeto que pode ser utilizada para auxiliar os próprios desenvolvedores. Entretanto, em grandes projetos de software livre há um elevado volume de mensagens, dificultando a escolha das tarefas em que o desenvolvedor deseja participar. Por exemplo, no mês de março de 2012 no projeto Apache Hadoop foram enviadas 1.622 mensagens na lista de tarefas. Assim, os desenvolvedores estão propensos a perderem oportunidades de participar de tarefas relevantes ao seu perfil devido ao grande volume de dados e à dificuldade de seleção.

A dificuldade de sugerir tarefas relevantes aos colaboradores não é um problema exclusivo da comunidade de software livre. Problemas semelhantes são tratados pela comunidade de sistemas colaborativos, como mostram os trabalhos dos autores Cosley *et al*. [3] e Abel *et al*. [6].

Este trabalho mostra a importância do conteúdo dos comentários em um projeto de software livre para construção de um classificador para predizer a participação de um desenvolvedor em uma determinada tarefa. O classificador é baseado na análise de conteúdo do vocabulário utilizado pelo desenvolvedor nas mensagens enviadas à lista de tarefas.

Para realizar a análise foram coletados dados do projeto Apache Hadoop[1] da Apache Software Foundation que está dividido em três subprojetos: Hadoop Commons, Hadoop Distributed File System (HDFS) e Hadoop MapReduce. Essa análise foi realizada com objetivo de responder à seguinte questão de pesquisa:

Q1 - É possível predizer a participação dos desenvolvedores mais ativos na lista de tarefas (*issue tracking*) com base na sua história de participação?

Para cada desenvolvedor foi construído um classificador que prediz sua participação em uma determinada tarefa baseado no vocabulário utilizado em outras participações.

Na composição do classificador proposto comparamos os resultados obtidos pelo algoritmo J48 e pelo classificador bayesiano, detalhados em [9][12]. Para cada desenvolvedor, as tarefas foram divididas em duas classes: "*Participar*" ou "*Não participar*". Em seguida, para a análise do desenvolvedor, definimos aleatoriamente um conjunto de dados para treino e outro para teste. A proporção utilizada para a definição do conjunto de treino e de teste foi 80-20, respectivamente.

Na Seção 2 são apresentados os trabalhos relacionados. Na Seção 3 é apresentado o método de pesquisa utilizado. Os resultados são apresentados na Seção 4. Por fim, na Seção 5 são apresentadas as discussões, conclusões e trabalhos futuros.

II. TRABALHOS RELACIONADOS

Uma desvantagem do uso de ferramentas de gestão está diretamente relacionada à sobrecarga de trabalho atribuída aos desenvolvedores mais experientes na triagem das

[1] http://hadoop.apache.org

978-0-7695-4890-6/12 $26.00 © 2012 IEEE
DOI 10.1109/SBSC.2012.27

informações fornecidas pela comunidade [7]. Neste sentido, o levantamento descrito nesta seção preconiza trabalhos que abordam a redução da carga de trabalho por meio da indicação de quais atividades o desenvolvedor deve participar.

Matter *et al*. [4] apresentam uma abordagem automática para realizar a triagem de relatórios de falhas atribuindo a tarefa ao desenvolvedor com a melhor expertise para lidar com a falha. Os autores construíram um modelo de expertise para cada desenvolvedor baseado no seu código-fonte produzido. Usando essa abordagem, reportaram um *precision* de 33.6% e *recall* 71.0% no projeto Eclipse. Seguindo a mesma linha, os trabalhos [11][1] constroem modelos de expertise baseados nas alterações realizadas no software pelo desenvolvedor.

Uma abordagem semelhante é apresenta por Cubranic e Murphy [2], que propõem a aplicação do classificador bayesiano para auxiliar a triagem de falhas. No processo de classificação, os autores realizaram uma caracterização textual considerando a descrição das falhas e obtiveram 30% de acurácia na aplicação dessa abordagem em um grande projeto de software livre.

Anvik *et al.* [8] apresentam uma abordagem que automatiza parte do processo de triagem dos relatórios de falhas. Baseado no conjunto de desenvolvedores recomendado pelo algoritmo de aprendizagem de máquina SVN, o responsável pela triagem indica um desenvolvedor mais qualificado para resolução de falhas. A *precision* alcançada com essa abordagem foi de 57% e 64% para os projetos Eclipse e Firefox, respectivamente.

Ibrahim *et al.* [13] apresentam um trabalho com uma abordagem semelhante à nossa, no entanto, aplicado para predições em lista de e-mails. Os autores desenvolveram um modelo que prediz a participação do desenvolvedor em uma discussão por e-mail. Esse modelo é desenvolvido baseado no comportamento histórico do desenvolvedor. Esse trabalho também mostrou que a quantidade de mensagens de uma discussão, o conteúdo, o remetente e a época em que a mensagem é submetida influenciam o comportamento do desenvolvedor. O classificador bayesiano e a árvore de decisão foram usadas para aplicar a abordagem em listas de e-mails dos projetos da Apache, PostGreSQL e Python.

As abordagens apresentadas nesta seção buscam fornecer meios automatizados para reduzir a carga de trabalho dos desenvolvedores em projetos de software livre e possuem características similares às apresentadas neste trabalho. No entanto, nossa abordagem procura recomendar tarefas ao desenvolvedor baseadas no seu envolvimento com o projeto em listas anteriores. O desenvolvedor pode colaborar na lista de tarefas indicada pela abordagem de diversas formas, não se limitando a realizar *commits* ou encaminhar novos *patches*.

III. MÉTODO DE PESQUISA

O método é composto de três passos: coleta, preparação e classificação dos dados.

A. Coleta de Dados

Primeiramente realizamos a coleta dos dados da lista de tarefas no website dos projetos mencionados que estão resumidos na Tabela 1. Utilizamos uma amostra dos dez desenvolvedores que mais publicaram comentários em tarefas de cada projeto. Essa amostra representa os desenvolvedores que enviaram mais de um terço do total de comentários de cada projeto.

TABELA 1. RESUMO DOS DADOS COLETADOS.

Subprojetos	***Hadoop Commons***	***HDFS***	***Hadoop MapRaduce***
Tarefas *(issues)*	7.720	2.980	3.591
Comentários	76.065	34.051	36.987
Desenvolvedores	1.035	512	614
Inicio da coleta	01-2006	03-2006	05-2006
Fim da coleta	04-2012	04-2012	04-2012
% de comentários dos desenvolvedores mais ativos (TOP10)	37,3%	57,2%	39%

Para evitar ambiguidade definimos formalmente os dados coletados. Uma lista de tarefas de um determinado projeto p é definida pelo conjunto:

$$ISSUES_p = \{i_1, i_2, i_3, \ldots, i_n\},$$

em que n é a quantidade de tarefas do projeto p. Podemos ainda representar o conjunto de todos os desenvolvedores de um determinado projeto p como:

$$AUTHORS_p = \{a_1, a_2, a_3, \ldots, a_m\},$$

em que m é a quantidade de desenvolvedores que comentaram em alguma tarefa $i_k \in ISSUE_p$ para qualquer $1 \leq k \leq n$. Também podemos representar todos os comentários de um projeto p como:

$$C_p = \{c_1, c_2, c_3, \ldots, c_q\},$$

em que q é a quantidade de comentários publicados no projeto p. Uma tarefa $i_k \in ISSUE_p$ com $1 \leq k \leq n$ é representada pela tripla:

$$i_k = <C_{ik}, d_{in}, d_{out}>,$$

em que $C_{ik} \subseteq C_p$ é o subconjunto de todos os comentários publicados na tarefa i_k e d_{in} e d_{out} são as datas da publicação do primeiro e último comentário, respectivamente.

Um desenvolvedor $a_r \in AUTHORS_p$ sendo $1 \leq r \leq m$ é representado pela tripla:

$$a_r = <C_{ar}, d_{in}, d_{out}>,$$

em que $C_{ar} \subseteq C_p$ é o subconjunto de todos comentários que esse desenvolvedor publicou no projeto e d_{in} e d_{out} são as datas da publicação do primeiro e último comentário desse desenvolvedor. Dessa forma, podemos notar que o conjunto de todos os comentários do projeto p é igual à união de todos os comentários das tarefas, ou ainda, é igual à união de todos os comentários dos desenvolvedores, conforme:

$$C_p = C_{i1} \cup C_{i2} \cup \ldots \cup C_{in} = C_{a1} \cup C_{a2} \cup \ldots \cup C_{am},$$

Um comentário $c \in C_p$ é representado pela tripla:

$$c = < T_c, d, a >,$$

em que T_c é conjunto de *tokens* que compõem o comentário c, d é a data de publicação e $a \in AUTHORS_p$ é o desenvolvedor que publicou esse comentário.

Podemos representar o conjunto de todos os *tokens* do projeto p como:

$$WORLD_p = T_1 \cup T_2 \cup ... \cup T_q \mid \forall c \in C_p ,$$

Analogamente, podemos representar todos os *tokens* de uma tarefa $i_k \in ISSUE_p$ como:

$$TOKENS_k = T_1 \cup T_2 \cup ... \cup T_w \mid \forall c \in C_{ik} ,$$

em que w é a quantidade de elementos do conjunto C_{ik}.

B. Preparação dos Dados

O processo de preparação de dados consistiu em organizar e criar o conjunto $WORLD_p$ dos subprojetos da Apache Hadoop. O primeiro passo foi identificar e remover da nossa amostra os comentários que foram inseridos por sistemas de software de integração contínua, tal como o Hudson[2]. Com esse simples passo foi possível remover uma grande quantidade de comentários irrelevantes a este trabalho como apresentado na Tabela 2.

TABELA 2. RESUMO DO PROCESSO DE PREPARAÇÃO DOS DADOS

Comentário	*Hadoop Commons*	*HDFS*	*Hadoop MapRaduce*
% de comentários extraídos	18,8	24,6	29,9
#comentários restantes	61.769	25.688	25.932
#*tokens*	59.733	27.774	31.294

Em seguida, utilizamos o Apache Lucene™[3] (AL) na versão 3.5 para retirar as marcações HTML dos comentários, realizar o processo de *tokenização,* extração dos *stopwords* e lematização (*stemming*). Consideramos também *urls* e nomes de pacotes de código-fonte como *tokens* únicos. A Tabela 2 mostra o tamanho do conjunto $WORLD_p$ de cada projeto.

Entendemos que o processo de preparação dos dados ainda pode ser mais bem explorado a fim de melhorar os resultados apresentados neste trabalho. Outro aspecto de estudo, é o fato de existir trechos de comentários que são compostos por código-fonte e podem ser tratados de uma melhor maneira.

C. Classificação

Após os dados terem sido preparados, para cada desenvolvedor do projeto, dividimos aleatoriamente o conjunto $ISSUES_p$ em dois subconjuntos, um para treino com 80% das tarefas e outro para teste com os 20% restantes. Os dois subconjuntos foram utilizados para classificar se o desenvolvedor participa ou não em uma determinada tarefa baseado em seu conteúdo. Para a predição utilizamos dois algoritmos: o algoritmo de árvore de decisão J48 e o algoritmo de classificação bayesiana Naïve Bayes, ambos disponibilizados pelo WEKA[4], um ambiente para análise de conhecimento que implementa vários algoritmos de aprendizado de máquina.

Para a execução dos algoritmos no ambiente WEKA utilizamos um computador com processador de quatro núcleos com 2.9 GHz, 8GB de memória RAM e um sistema operacional de 64 bits. Devido à grande quantidade de *tokens* do projeto Hadoop Commons e esse limite de memória RAM, não foi possível executar os algoritmos com todos os dados desse projeto. Então, para viabilizar a análise reduzimos o conjunto $WORLD_{common}$ para 70% do original por meio de um algoritmo que calcula a relevância de cada $t \in WORLD_{common}$ de acordo com a equação [10]:

$$score(t) = tf(t,d) * idf(t,D)$$

Vale ressaltar que essa redução foi realizada somente para o projeto Hadoop Common. Para os demais projetos mantivemos o conjunto $WORLD_p$ inalterado. Os resultados são apresentados na Seção 4.

Para execução dos algoritmos os subconjuntos de dados de treino e teste foram formatados em uma matriz de entrada como ilustra a Figura 1.

$ISSUES_p$	t_1	t_2	t_3	⋯	$t_{\lvert WORLD_p \rvert}$	Atributo Alvo (AA)
i_1	0	1	1	⋯	1	1
i_2	1	0	0	⋯	0	0
i_3	0	1	1	⋯	0	0
⋮						
i_n	1	1	1	⋯	1	0

$WORLD_p$

Figura 1. Matriz de entrada dos dados

As linhas da matriz são representadas pelos elementos do conjunto $ISSUES_p$ e as colunas, salvo a última, são representadas pelos elementos do conjunto $WORLD_p$. A matriz foi construída seguindo duas regras. A Regra 1 estabelece o preenchimento das células da matriz, menos as da última coluna.

Regra 1: Seja uma tarefa $i_k \in ISSUES_p$ e um *token* $t_j \in WORLD_p$

$$M[i_k, t_j] = \begin{cases} 1 & se \;\; t_j \in TOKENS_k \\ 0 & se \;\; t_j \notin TOKENS_k \end{cases}$$

Com isso, todos os *tokens* pertencentes à tarefa são assinalados com 1.

As células da coluna *AA*, que representa o atributo alvo a ser predito, são preenchidas respeitando a Regra 2.

Regra 2: Seja uma tarefa $i_k \in ISSUES_p$ e um desenvolvedor $a_r \in AUTHORS_p$

$$M[i_k, AA] \begin{cases} 1 & se \;\; C_{ik} \cap C_{ar} \neq \emptyset \\ 0 & se \;\; C_{ik} \cap C_{ar} = \emptyset \end{cases}$$

[2] http://hudson-ci.org/

[3] http://lucene.apache.org/

[4] http://www.cs.waikato.ac.nz/ml/weka

Aplicamos dois filtros adicionais nos dados de entrada. O primeiro filtro foi aplicado tanto no subconjunto de treino quanto no de teste, descartando as tarefas em que o desenvolvedor não teve a oportunidade de participar. Para isso, definimos o escopo para a tarefa como o intervalo entre a publicação da primeira e da última mensagem publicada. Também definimos o escopo para o desenvolvedor como o intervalo entre a sua primeira e a última mensagem publicada, independente da tarefa. Esse filtro respeita o que foi estabelecido na Regra 3.

Regra 3: Dado uma tarefa $i_k \in ISSUES_p$ e um desenvolvedor $a_r \in AUTHORS_p$

$$\begin{cases} descartar(i_k), & se\ i_k[d_{out}] < a_r[d_{in}]\ OR\ i_k[d_{in}] > a_r[d_{out}] \\ manter(i_k), & caso\ contrário \end{cases}$$

Nessa abordagem identificamos seis tipos de tarefas como ilustra a Figura 2.

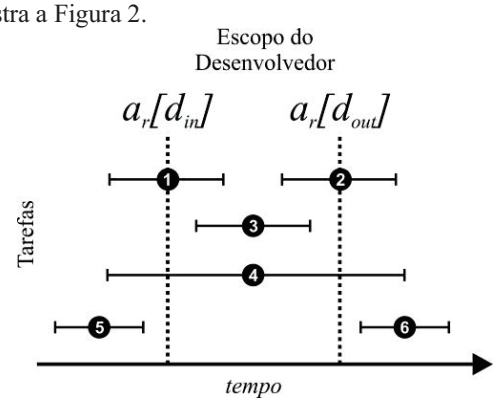

Figura 2. Tipos de Tarefas.

As tarefas do **tipo 1** são aquelas que começam antes do início do escopo do desenvolvedor, mas terminam dentro do escopo. As tarefas do **tipo 2** são aquelas que iniciam dentro do escopo do desenvolvedor, mas terminam fora. As tarefas do **tipo 3** estão totalmente contidas no escopo do desenvolvedor. As tarefas do **tipo 4** permeiam por todo escopo do desenvolvedor, mas iniciam e terminam fora. As tarefas do **tipo 5** começam e terminam antes do início do escopo do desenvolvedor. E por último, as tarefas do **tipo 6** iniciam e terminam após o término do escopo do desenvolvedor. Com esses seis tipos de tarefas, entendemos que o desenvolvedor teve, em algum momento, a oportunidade de colaborar com as tarefas dos tipos 1, 2, 3 e 4. As tarefas dos tipos 5 e 6 foram descartadas por estarem totalmente fora do escopo do desenvolvedor, não oferecendo a oportunidade de participação. Portanto, para a predição de cada desenvolvedor, o universo de tarefas do projeto foi reduzido de acordo com seu escopo.

O segundo filtro, aplicado somente no subconjunto de teste, descarta todos os *tokens* proferidos pelo desenvolvedor em cada tarefa. Utilizamos essa abordagem baseado na premissa de que os *tokens* proferidos pelo desenvolvedor darão vantagem à tarefa na predição, por fazerem parte de seu vocabulário e, provavelmente, terem sido utilizados no treinamento. Dessa forma, esse filtro tem como objetivo evitar um viés na predição.

IV. RESULTADOS

As Tabelas 3, 4 e 5 mostram as medidas de *recall*, *precision* e acurácia obtidas pelo algoritmo de classificação J48 nos subprojetos Hadoop Common, MapReduce e HDFS, respectivamente. Os resultados alcançados pelo algoritmo J48 foram superiores ao algoritmo Naïve Bayes atingindo uma melhoria da taxa média da acurácia de 11,2%, 9,6% e 13,3% para os subprojetos Hadoop Common, MapReduce e HDFS respectivamente. Por esse motivo, os resultados do algoritmo Naïve Bayes foram omitidos.

Mesmo possuindo as informações sobre o nome e dados de acesso dos desenvolvedores analisados, optamos por preservar suas identidades e referenciá-los apenas por Top-1 a Top-10. Desta forma, o desenvolvedor Top-1 é aquele que mais postou comentários na lista de tarefas do projeto.

A Tabela 3 apresenta o resultado da predição dos desenvolvedores do subprojeto Hadoop Common e mostra que as taxas de *recall* e *precision* para a classe "*Não participar*" são mais satisfatórias que as taxas da classe *"Participar"*. Esse comportamento é similar ao apresentado nas Tabelas 4 e 5. A taxa de *recall* para a classe "*Participar*" varia entre 0,2 e 0,54 com *precision* de 0,53 e 0,91. A classe "*Não participar*" apresenta *recall* entre 0,96 e 1 com taxa de *precision* variando entre 0,83 e 0,95.

TABELA 3. TAXAS DE RECALL, PRECISION E ACURÁCIA – J48 - HADOOP COMMON.

Desenvolvedor	*Participar*		*Não participar*		*Acurácia*
	recall	*precision*	*recall*	*precision*	
Top-1	0.30	0.76	0.97	0.83	0.83
Top-2	0.37	0.73	0.98	0.90	0.88
Top-3	0.30	0.62	0.96	0.86	0.84
Top-4	0.39	0.84	0.99	0.92	0.92
Top-5	0.34	0.75	0.99	0.92	0.91
Top-6	0.30	0.61	0.98	0.92	0.91
Top-7	0.40	0.68	0.98	0.95	0.94
Top-8	0.20	0.53	0.98	0.90	0.89
Top-9	0.54	0.76	0.98	0.95	0.93
Top-10	0.39	0.91	1.00	0.95	0.95

A Tabela 4 apresenta o resultado da predição para os desenvolvedores do subprojeto MapReduce. Destacamos o desenvolvedor Top-9 que apresentou menor taxa de *recall* de toda nossa amostra. Apesar disso, a média da taxa de *recall* e *precision* mantiveram-se semelhantes ao subprojeto Hadoop Commons, como indica a Tabela 6.

TABELA 4. TAXAS DE RECALL, PRECISION E ACURÁCIA – J48 - MAPREDUCE.

Desenvolvedor	*Participar*		*Não participar*		*Acurácia*
	recall	*precision*	*recall*	*precision*	
Top-1	0.47	0.83	0.97	0.88	0.87
Top-2	0.29	0.66	0.97	0.86	0.84
Top-3	0.38	0.76	0.99	0.93	0.92
Top-4	0.34	0.76	0.98	0.91	0.91
Top-5	0.32	0.93	1.00	0.91	0.91
Top-6	0.38	0.83	0.99	0.94	0.94
Top-7	0.41	0.86	0.99	0.92	0.92
Top-8	0.36	0.71	0.98	0.93	0.92
Top-9	0.07	0.63	1.00	0.90	0.90
Top-10	0.25	0.79	1.00	0.95	0.95

A Tabela 5 apresenta o melhor resultado com as maiores taxas de *recall* e de *precision*. Entre os desenvolvedores do subprojeto HDFS, destacamos o Top-10 que apresentou a maior taxa de *recall* na classe "*Participar*" da amostra.

TABELA 5. TAXAS DE *RECALL*, *PRECISION* E ACURÁCIA – J48 - HDFS.

Desenvolvedor	*Participar*		*Não participar*		*Acurácia*
	recall	*precision*	*recall*	*precision*	
Top-1	0.46	0.78	0.94	0.79	0.79
Top-2	0.33	0.77	0.97	0.81	0.80
Top-3	0.45	0.77	0.97	0.88	0.87
Top-4	0.41	0.73	0.95	0.84	0.83
Top-5	0.31	0.63	0.97	0.90	0.88
Top-6	0.31	0.73	0.99	0.93	0.93
Top-7	0.35	0.87	0.99	0.85	0.85
Top-8	0.43	0.63	0.97	0.93	0.91
Top-9	0.56	0.71	0.99	0.97	0.96
Top-10	0.62	0.97	1.00	0.95	0.95

A Tabela 6 apresenta um resumo com as médias das taxas de *recall*, *precision* e acurácia. As médias das acurácias por projeto foram 0,90, 0,91 e 0,87. No entanto, como visto nas tabelas anteriores, as medidas de acurácia variaram entre 0,79 e 0,96, considerando todos os subprojetos.

TABELA 6. TAXAS MÉDIAS DE *RECALL*, *PRECISION* E ACURÁCIA - J48.

Projetos	*Participar*		*Não participar*		*Acurácia*
	recall	*precision*	*recall*	*precision*	
Commons	0.34	0.71	0.98	0.91	0.90
MapReduce	0.33	0.78	0.99	0.91	0.91
HDFS	0.41	0.76	0.97	0.89	0.87

A Figura 3 apresenta três gráficos que mostram uma comparação entre os resultados das acurácias alcançadas com os subprojetos em estudo para o Naïve Bayes e o J48. Nota-se que o algoritmo J48 é melhor que o algoritmo Naïve Bayes para todos os casos.

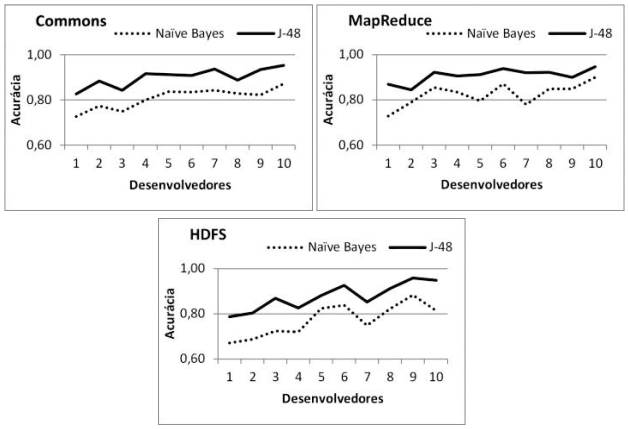

Figura 3. Naïve Bayes versus J48.

A Figura 4 apresenta o gráfico que mostra a relação entre a acurácia obtida pelo algoritmo J48 e os desenvolvedores que estão ordenados de acordo com a taxa de participação.

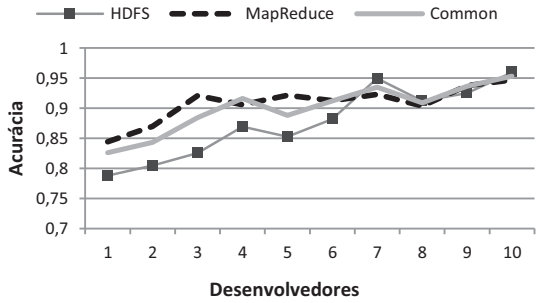

Figura 4. Relação da acurácia dos dez desenvolvedores entre os três projetos.

V. DISCUSSÃO E CONCLUSÃO

Em nossa análise utilizamos os comentários publicados nas discussões das listas de tarefas em três subprojetos de um software livre. A amostra foi limitada aos dez desenvolvedores que mais publicaram comentários. Não podemos afirmar que os mesmos resultados serão obtidos para os desenvolvedores com menos participações, pois o vocabulário desses desenvolvedores seria menor. Um possível trabalho futuro é aplicar o algoritmo para uma amostra maior de desenvolvedores para verificar a generalização da abordagem. Os resultados obtidos pelo classificador elaborado usando os métodos descritos na Seção 2 visam responder à seguinte questão de pesquisa:

Q1 - É possível predizer a participação dos desenvolvedores mais ativos na lista de tarefas (issue tracking) com base na sua história de participação?

As Tabelas 3, 4, 5 e 6 mostraram o resultado da predição com uma alta acurácia, variando de 0,79 a 0,96. No entanto, esse classificador apresenta uma baixa taxa de *recall* para a classe "*Participar*". Entendemos que isso é uma deficiência do nosso classificador, visto que ao não atribuir ao desenvolvedor uma tarefa em que poderia participar, deixamos de aproveitar o seu potencial e conhecimento. Entretanto, os resultados mostram que, dentre as tarefas indicadas ao desenvolvedor, a taxa de *precision* apresentou uma média de 0,75 variando entre 0,53 e 0,97. Suspeitamos que melhores resultados para caracterizar um desenvolvedor possam ser obtidos quando outros fatores, tais como os apresentados em [5], são combinados à nossa abordagem.

A alta acurácia do nosso classificador deve-se aos resultados obtidos com a classe "*Não participar*". Essa participação é significativa, visto que nosso classificador foi satisfatório ao identificar tarefas que não são de interesse ou do conhecimento do desenvolvedor. Comportamento semelhante das classes "*Participar*" e "*Não participar*" pode ser visto em Ibrahim *et al.* [13] que apresentaram resultados de um classificador bayesiano para predizer a participação de um desenvolvedor em lista de e-mails.

Ao comparar nossa abordagem com outros trabalhos relacionados, vimos que apresentamos melhores resultados do que aqueles obtidos por Cubranic e Murphy [2]. No entanto, esse trabalho procura identificar qual desenvolvedor é o mais adequado para resolver uma falha específica no projeto. Como nosso objetivo é predizer a participação de

um desenvolvedor em uma determinada tarefa, não é possível comparar os resultados de maneira tão simplificada.

Neste trabalho ainda exploramos o desempenho do classificador construído pelo algoritmo Naïve Bayes. No entanto, como demostrado pela Figura 3, o algoritmo J48 apresentou melhores valores de acurácia sobre o algoritmo Naïve Bayes para os dez desenvolvedores em todos os subprojetos. A maior diferença (14,5%) foi apresentada pelo desenvolvedor Top-3 do projeto HDFS e a menor diferença (4,7%) pelo desenvolvedor Top-10 do subprojeto MapReduce. Ao aplicarmos o algoritmo J48, obtivemos um aumento da acurácia de 11,2%, 9,6% e 13,3% em média para o Hadoop Common, MapReduce e HDFS, respectivamente.

A Figura 4 apresenta um gráfico que faz uma relação entre a participação dos desenvolvedores e a taxa da acurácia do classificador construído pelo algoritmo J48. Podemos notar que o comportamento entre a taxa de acurácia dos dez desenvolvedores mais ativos é semelhante nos três projetos. Um resultado não esperado pode ser visto no crescimento dos valores de acurácia. Esse crescimento indica que entre os desenvolvedores mais ativos da amostra existe uma tendência de quanto menor é a participação melhor é a acurácia na classificação de sua participação ou não em tarefas. Vale ressaltar que mesmo o último entre os dez desenvolvedores, apresenta uma taxa de participação é muito significativa no projeto. Esse comportamento de crescimento foi encontrado nos três subprojetos analisados, no entanto, é necessário mais estudos para investigar se também ocorre para outros conjuntos de desenvolvedores.

Algumas limitações e ameaças à validade dos resultados são: (i) Os resultados da análise são específicos para os projetos analisados, não podendo ser generalizados. Os projetos podem possuir diferentes características decorrentes da estrutura organizacional ou do domínio da aplicação. A análise com uma maior quantidade de projetos, ou com o ecossistema do Apache Software Foundation tornaria os resultados mais genéricos. (ii) A análise foi realizada baseada em dados coletados desde a criação do projeto e no início não havia uma separação formal dos subprojetos. Assim, suspeitamos que algumas das tarefas que eram destinadas ao HDFS ou ao MapReduce possam ter sido publicadas na lista do Hadoop Common. (iii) Devido a uma restrição imposta pela limitação física do equipamento utilizado para executar os algoritmos, tivemos que reduzir em 30% o conjunto $WORLD_{common}$. (iv) O classificador tem como base a análise do vocabulário do desenvolvedor utilizado em comentários anteriores, e pode não proporcionar os mesmos resultados para novos desenvolvedores. (v) A abordagem para a extração dos *tokens* que compõem o conjunto $WORLD_p$ pode ser melhor explorada, principalmente com relação aos códigos-fonte publicados nas tarefas. (vi) Não é possível também afirmar que a abordagem tem os mesmos resultados para os desenvolvedores com menor quantidade de comentários enviados. Como dito anteriormente, uma análise com uma amostra maior e com desenvolvedores escolhidos aleatoriamente deve ser conduzida futuramente para verificar a efetividade da abordagem em outros casos.

Em resumo apresentamos resultados que indicam que o conteúdo dos comentários publicados em tarefas (*issues*) de projetos de software livre é um importante fator para construir um classificador de predição de tarefas aos desenvolvedores. Baseado neste fator, mostramos que é possível construir um classificador com alta acurácia que pode diminuir a carga de trabalho do desenvolvedor indicando a sua participação em tarefas de seu interesse. Entretanto, reconhecemos que a taxa de *recall* obtida para a classe "*Participar*" não foi satisfatória. Por último, mostramos que nossa abordagem é mais efetiva se utilizarmos o algoritmo de classificação J48 em comparação ao Naïve Bayes.

A respeito dos trabalhos futuros, além daqueles já citados na análise das ameaças à validade, entendemos que outros fatores que auxiliem a caracterização dos desenvolvedores devem ser explorados e combinados para melhorar o desempenho da predição. Pretende-se ainda comparar a nossa abordagem com outras apresentadas na literatura.

REFERÊNCIAS

[1] A. Mockus and J. D. Herbsleb, "Expertise browser: a quantitative approach to identifying expertise," Proc. of the 24th International Conference on Software Engineering, pp.503-512, May 2002.

[2] D. Cubranic and G. C. Murphy, "Automatic bug triage using text classification," Proc. of Software Engineering and Knowledge Engineering, pp.92-97, 2004

[3] D. Cosley, D. Frankowski, L. Terveen, and J. Riedl, "SuggestBot: using intelligent task routing to help people find work in wikipedia," Proc. of the 12th international conference on intelligent user interfaces. New York, NY, USA, pp.32-41, 2007.

[4] D. Matter, A. Kuhn, and O. Nierstrasz, "Assigning bug reports using a vocabulary-based expertise model of developers," Proc. of 6th IEEE International Working Conference on Mining Software Repositories. Washington, DC, USA, pp.131-140, May 2009.

[5] E. S. Raymond, "The cathedral and the bazaar," Tim O'Reilly (Ed.). O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1999.

[6] F. Abel, I. I. Bittencourt, E. Costa, N. Henze, D. Krause, and J. Vassileva, "Recommendations in online discussion forums for e-learning systems," IEEE Transactions on Learning Technologies, vol.3, no.2, pp.165-176, April-June 2010.

[7] J. Anvik, L. Hiew, and G. C. Murphy, "Coping with an open bug repository," Proc. of the OOPSLA workshop on Eclipse technology eXchange. ACM, New York, NY, USA, 35-39. 2005.

[8] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?," Proc. of the 28th international conference on software engineering, New York, NY, USA, pp.361-370, 2006.

[9] R. Quinlan, "C4.5: Programs for Machine Learning," Morgan Kaufmann Publishers, San Mateo, CA, USA, 1993.

[10] Spärck Jones, Karen. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation* **28** (1): 11–21. doi:10.1108/eb026526.

[11] T. Fritz, G. C. Murphy, and E. Hill, "Does a programmer's activity indicate knowledge of code?," Proc. of the 6th european software engineering conference, pp.341-350, New York, NY, USA, 2007.

[12] T. M. Mitchell, "Machine Learning," McGraw-Hill, New York, NY, USA, 1997.

[13] W. M. Ibrahim, N. Bettenburg, E. Shihab, B. Adams, and A. E. Hassan, "Should I contribute to this discussion?," Proc of 7th IEEE working conference mining software repositories, pp.181-190, May 2010.