Revista Eletrônica de Sistemas de Informação
ISSN 1677-3071

Revista hospedada em: http://revistas.facecla.com.br/index.php/reinfo
Forma de avaliação: *double blind review*

HISTORICAL ANALYSIS OF MESSAGE CONTENTS TO RECOMMEND ISSUES TO OPEN SOURCE SOFTWARE CONTRIBUTORS

Igor Steinmacher, Igor Wiese, André L. Schwerz, Rafael Liberato
Departamento de Computação – Universidade Tecnológica Federal do Paraná (UTFPR)
{igorfs, igor, andreluis, liberato}@utfpr.edu.br

João Eduardo Ferreira, Marco Aurélio Gerosa
Departamento de Ciência da Computação – Instituto de Matemática e Estatística
Universidade de São Paulo (USP)
{jef, gerosa}@ime.usp.br

ABSTRACT

Developers of distributed open source projects make use of issue tracker tools to coordinate their work. These tools store valuable information, maintaining a log of relevant decisions and bug solutions. Finding the appropriate issues to contribute can be hard, as the high volume of data increases contributors' overhead. This paper shows the importance of the content of issue tracker discussions in an open source project to build a classifier to predict the participation of a contributor in an issue. To design this prediction model, we used two machine learning algorithms called Naive Bayes and J48. We used data from the Apache Hadoop Commons project to evaluate the use of the algorithms. By applying machine learning algorithms to the ten most active contributors of this project, we achieved an average recall of 66.82% for Naïve Bayes and 53.02% using J48. We achieved 64.31% of precision and 90.27% of accuracy using J48. We also conducted an exploratory study with five contributors that took part in fewer issues and achieved 77.41% of precision, 48% of recall, and 98.84% accuracy using J48 algorithm. The results indicate that the content of comments in issues of open source projects is a relevant factor to recommend issues to contributors.

Key-words: Open Source Software; Recommendation System; Issue Tracker; Mining Software Repositories

1 INTRODUCTION

Open Source Software (OSS) projects raised academic and industry interest due to their distributed development method (Raymond, 1999). The voluntary effort of OSS developers spread around the world leveraged this interest. Contributors of OSS projects make use of source code repositories and issue/bug tracking systems to facilitate the planning and discussion among contributors. The use of these tools streamlines bug detection, discussion and fixing processes, improving the software quality (Raymond, 1999; Anvik et al., 2005). These tools hold a lot of data, such as, source code, files' change history, contributors' information, and backlog. The history of messages submitted to the issue tracker maintains a log of important decisions and solutions, such as new features developed, bug solutions and design decisions.

OSS contributors can use the information stored on the issue tracker as a rich project knowledge base to support their decisions. This knowledge base is fed by the projects contributors, who raise issues, proposing new features, and discussing these issues and features. To participate on these discussions, contributors find the issues and tasks that they can contribute to and send their comments or engage on fixing it. In large open source projects, the high volume of issues and comments makes it difficult for the contributor to choose the right ones to collaborate. For example, in March 2012, 1,622 messages were sent to Apache Hadoop issue tracker. Thus, contributors are likely to lose the opportunity to contribute to issues relevant to their profile.

The problem of recommending relevant tasks to contributors is not an exclusive problem of open source software community. Cosley et al. (2007) proposed a system that performs intelligent task recommendation to users on Wikipedia. Abel et al. (2010) recommends relevant information to users in e-learning discussion forums. Our research aims to verify the relevance of comments content in a classifier for recommending issues to contributors. The classifier is based on the vocabulary used by a contributor in the messages submitted to the issue tracker. To analyze their content, each message is converted to a set of words (tokens). This format enables the definition of the vocabulary of a given contributor, issue, or for the entire project.

To conduct our analysis, we collected data from the Apache Hadoop Commons project from the Apache Software Foundation repository. We aimed to answer the following research question:

Is the vocabulary used in issues that a contributor participated effective to predict his/her participation in issues discussions?

For each contributor, we had built a classifier that predicts his/her participation in a particular issue based on the Term Frequency-Inverse Document Frequency (TF-IDF) of the terms used in other issues he/she took part. To address the research question we compared the results achieved when using J48 and the Naïve-Bayes algorithms, detailed in Quinlan (1993) and Mitchell (1997). For each contributor, the issues were split into two classes: "Participate" or "Don't Participate." Then, to analyze the contributor, we defined random training and test data sets. The proportion used to define the training set, and the test was 80:20.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 presents the research method. Results are presented in Section 4. Section 5 presents the discussions of the results. In Section 6, we present the threats to validity and in Section 7, the conclusions.

2 RELATED WORKS

Developers of OSS projects make use of web tools to maintain different types of communication related to the progress of the project. In large communities, the workload assigned to the more experienced developers in screening the information provided by the community is high (Anvik et al., 2005). In this sense, several works of the literature address the workload issue by recommending which tasks a developer should contribute to or to which email a user should answer.

Matter *et al*. (2009) present an automatic approach to triage bug reports, assigning the issue to the developer with the best expertise to handle that bug. The authors built an expertise model for each developer, based on the source code produced by him. They used the cosine distance between the expertise model and a vector of words built from the issue report to rank the developers. Authors reported a 33.6% precision and 71.0% recall in the Eclipse project. Similar approaches were presented by Mockus and Herbsleb (2002) and Fritz et al. (2007), building expertise models based on artifacts changes made by software developers.

Siy et al. (2008) and Alonso et al. (2008) propose an expertise model based on the folders in which the source code is stored. In both studies, the authors consider that a developer tends to work on files in folders that are located nearby, and thus can provide a better answer to questions related to that part of the software.

Our research considers the history of messages submitted by the developer to the issue tracker as the only factor to recommend their participation on a specific issue. A similar approach was presented by Cubranic and Murphy (2004). They proposed the use of a Bayesian classifier to support bug triage. During the classification process, the authors conducted a textual characterization considering the description of the bugs and obtained 30% accuracy applying the approach in a large OSS project. Canfora and Cerulo (2005) proposed an approach using an information retrieval technique that achieves recall of 20% on the Mozilla project.

Anvik et al. (2006) present an approach that automates the process of bug reports triage. Based on the set of developers recommended by a SVN machine learning algorithm, the responsible for triaging the bugs can manually select the more skilled developer to fix the bug. The algorithm was trained with textual data extracted from previous bug reports and labeled with the developer that fixed them. The precision achieved with this approach was 57% and 64% for Eclipse and Firefox projects, respectively.

Ibrahim et al. (2010) developed a model that predicts the involvement of contributor in a mailing list thread. This model considers the historical behavior of contributors. The authors showed that the number of messages in a discussion, the contents, the sender, and the time when the message is submitted influence the behavior of a contributor. They used a Bayesian classifier and a decision tree and applied the approach to the mailing lists of Apache, PostgreSQL, and Python projects. As a result, their approach led to accuracy values ranging from 85% to 89.5% for the top 10 contributors. Bird (2012) built a single model for all developers based on a neural network to predict which emails a particular developer would be interested on.

The approaches presented in this section seek to provide automated means to considerably reduce the workload of contributors in OSS projects and present characteristics that are similar to those presented in our research. However, our approach seeks to recommend issues to contributors based only on the vocabulary of the issues that contributor previously contributed to.

3 RESEARCH METHOD

The method used in this research comprises three steps: data collection, data preparation, and data classification. The following subsections details these steps.

3.1 DATA COLLECTION

To analyze and build the classifier, we first collected data from Hadoop projects issue tracker, available at the project's website. To extract the data we had built a tool that accesses the URL of the issues, respecting this pattern: "*https://issues.apache.org/jira/browse/*<PROJECT_NAME>-<ISSUE_NUMBER>". For each issue, we have extracted the following information:

- description;
- issue reporter;
- creation date;
- closing date;
- status;
- comments (containing author, date, and message).

In the first step of this research, we have used a sample composed of the ten contributors that most commented issues. This sample represents the contributors that sent more than one third of the comments in the project. Table 1 summarizes the period and the amount of messages collected and analyzed in this study. It is worth to notice that we count as message all comments submitted to an issue and the summary and description used to open an issue.

Table 1. Summary of Collected Data

	Hadoop Commons
# of issues	8,288
# of messages	70,057
# of contributors	1,181
Start date	01-2006
End date	04-2012
% of messages posted by top 10 contributors	35.7%

In Table 2, we present the participation rate and the number of issues that each of the top 10 contributors took part. It represents the percentage of issues in which the contributor sent at least one comment.

Table 2. Participation rate for the top ten contributors

Contributor	Percentage of issues	Number of issues
Top-1	20.25%	1678
Top-2	18.39%	1524
Top-3	12.90%	1069
Top-4	9.60%	796
Top-5	9.41%	780
Top-6	8.53%	707
Top-7	8,52%	706
Top-8	8,20%	680
Top-9	8.13%	674
Top-10	6.74%	559

To avoid ambiguity we have formally defined the collected data. A list of issues in a given project is defined as the set:

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

in which *n* is the number of issues of project *p*. We have also represented the set of all contributors in a given project *p* as:

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

in which *m* is the number of contributors that submitted a comment to a given issue $i_k \in ISSUE_p$, $1 \leq k \leq n$. We also represented the comments of a project *p* as:

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

in which *q* is the amount of comments submitted to any $i_k \in ISSUE_p$,$1 \leq k \leq n$ of a project *p*. An issue $i_k \in ISSUE_p$, $1 \leq k \leq n$ can be represented by the triplet:

$$i_k = <C_{ik}, d_{first}, d_{last}>,$$

in which $C_{ik} \subseteq C_p$ is a subset of all comments submitted to the issue i_k , and d_{first} and d_{last} are the dates when the first and last comments were sent, respectively.

A contributor $a_r \in AUTHORS_p$, $1 \leq r \leq m$, is represented by the triplet:

$$a_r = <C_{ar}, d_{first}, d_{last}>,$$

in which $C_{ar} \subseteq C_p$ is the subset of all comments submitted by a contributor in project *p*, and d_{first} and d_{last} are the dates when the contributor sent his first and last comment, respectively. So, we can say that the set containing all comments of project *p* is equal to the union of all comments sent to all the issues, or, is equal to all comments send by all the contributors:

$$C_p = C_{i1} \cup C_{i2} \cup \ldots \cup C_{in} = C_{a1} \cup C_{a\,2} \cup \ldots \cup C_{am},$$

A comment $c \in C_p$ is represented by the triplet:

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

in which T_c is the set of all tokens that are part of the comment *c*; *d* is the date when the comment was sent; and $a \in AUTHORS_p$ is the contributor who sent the comment. We can represent the set of all project tokens as:

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

Similarly, we can represent all the tokens of the issue $i_k \in ISSUE_p$ as:

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

in which w is the number of elements of C_{ik}.

3.2 DATA PREPARATION

We conducted a data preparation process to organize and create the set WORLD$_p$ of the project. The first step was to identify and remove the comments entered by software systems (bots) from our sample.

After that, we used Apache Lucene™[1] (AL) to remove HTML tags from the comments, tokenize, remove stopwords, and stem. We used AL default tokenizer and stopwords list, and applied the stemmer proposed by Martin Porter (1980). We considered URLs and artifacts full-qualified names as single tokens.

3.3 DATA CLASSIFICATION

After preparing the data, we randomly split the $ISSUES_p$ into two subsets for each contributor: a training set containing 80% of the issues; and a test set containing 20% of the issues. To recommend the participation of a contributor, we used two different algorithms: J48 decision trees and Naïve Bayes. Both algorithms are implemented in WEKA[2], a knowledge analysis environment that implements many machine learning algorithms.

A decision tree is a machine learning model that predicts the value of a target attribute of a new instance based on the values of other available attributes. The target attribute is known as dependent variable as its value depends on (or is defined by) the values of all the other attributes (independent variables). To classify a new instance, it is necessary to create a decision tree based on the values of the attributes of the training set. After generating the tree, the algorithm verifies the values of the attributes of a new instance, and navigates on the tree to predict the target attribute.

The Naive Bayes classification algorithm is based on Bayes rules of conditional probability. The algorithm uses all the attributes presented in the training set and analyses them individually, considering them equally important and independent. The classifier considers each of this attributes separately when it is classifying a new instance.

3.3.1 Input Data Format

Training and test sets were formatted as input matrices, as shown in Figure 1**Error! Reference source not found.**. The lines of the matrices represent the elements of $ISSUES_p$ set. The columns, except the last one, represent the elements of the $WORLD_p$ set. The matrices were built following two rules. Rule 1 establishes cells fulfillment, except for the last column, which represent the target attribute.

Rule 1: For a given issue $i_k \in ISSUES_p$ and a token $t_j \in WORLD_p$

$$M[i_k, t_j] = tf\_idf(i_k, t_j) \;\; if \;\; t_j \in TOKENS_k$$

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

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

WORLD$_p$		t_1	t_2	t_3	. . .	$t_{\lvert WORLDp \rvert}$	Target Attribute (TA)
ISSUES$_p$	i_1	TF-IDF	TF-IDF	TF-IDF		TF-IDF	1
	i_2	TF-IDF	TF-IDF	TF-IDF		TF-IDF	0
	i_3	TF-IDF	TF-IDF	TF-IDF		TF-IDF	0
	⋮						
	i_n	TF-IDF	TF-IDF	TF-IDF		TF-IDF	1

Figure 1. Data Input Matrix

We used two different means to fill the cells. We filled the WORLD with the Text Frequency – Inverse Document Frequency (TF-IDF) value to represent the importance of a given term. Therefore, to follow Rule 1, all tokens that are part of an issue have the value of its cell set to their TF-IDF value, defining issues' weighted vocabulary.

The cells of column Target Attribute (TA), which represent the attribute to be predicted, are filled according to Rule 2.

Rule 2: For a given issue $i_k \in ISSUES_p$ and a given contributor $a_r \in AUTHORS_p$

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

We have applied two additional filters to the input data. The first was applied in both subsets (training and test), discarding the issues in which the contributor had not the chance to participate. We defined the scope of an issue as the time interval between the first and the last message submitted on it. We also defined the scope of a contributor as the time interval between his first and last message submitted to the project. This filter is formalized as the Rule 3.

Rule 3: For a given issue $i_k \in$ ISSUES$_p$ and a contributor $a_r \in$ AUTHORS$_p$

$$\begin{cases}discard(i_k), & if\ i_k[d_{last}] < a_r[d_{first}]\ OR\ i_k[d_{first}] > a_r[d_{last}] \\ keep(i_k), & otherwise\end{cases}$$

By following this approach, we identified six types of issue scopes, as shown in Figure 2.

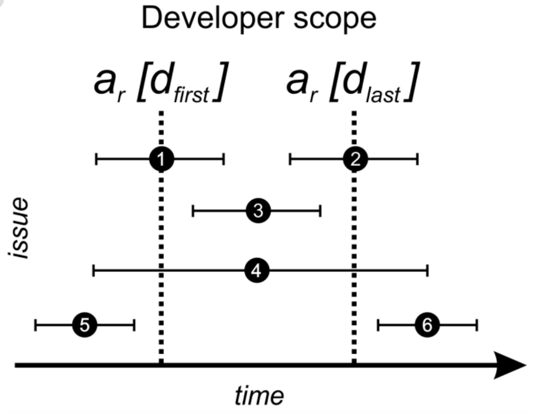

Figure 2. Issue Types

Type 1 issues are those that have their creation date before the beginning of contributor's scope and their last message sent within contributor's scope. **Type 2** issues are those issues that have their first message sent within contributor's scope and the last message sent after the end of contributors' scope. Issues classified as **Type 3** are totally inside the contributor's scope period. **Type 4** issues were created before a contributor's scope and have their last message sent after the end of contributor's scope. Issues classified as **Type 5** are created and have their last message sent before the contributor's scope. Finally, **type 6** issues are created after the contributor's scope period.

We understand that a contributor had the chance to contribute and discuss on issues of types 1, 2, 3 and 4 as the contributor's scope intercepts the issues' scope. The issues classified as types 5 or 6 were not considered as they are totally outside the contributor's scope, and the contributor did not have the chance to join those discussions.

The second filter, applied only to the test set. When predicting the participation of a given contributor we have discarded the messages sent by him/her in each issue. We have applied this approach based on the premise that the tokens used by the contributor himself can introduce some bias to the prediction results as can be part of historical contributor's vocabulary.

3.3.2 Results Evaluation

As we are dealing with a typical usage of machine learning, we use the confusion matrix to evaluate precision, recall, and accuracy of the results. Table 3 depicts a confusion matrix, in which we identify four types of results: true positive (TP), false positive (FP), true positive (TP), and true negative (TN).

Table 3. Confusion Matrix

Classified as	***Known instances***	
	Participate	*Do not participate*
Participate	*TP*	*FP*
Do not participate	*FN*	*TN*

In the context of classification, precision metric is the number of instances correctly classified as a given class, divided by the total number of instances classified as belonging to that class. A precision value of 100% indicates that every instance that was classified as a given class was correctly classified. Recall is the number of instances of a class correctly classified over all of the instances that actually belong to that class. A recall value of 100% would indicate that every instance of a class was classified as belonging to that class.

$$precision_{participate} = \frac{TP}{TP + FP} \qquad precision_{doNotPartipate} = \frac{TN}{FN + TN}$$

$$recall_{participate} = \frac{TP}{TP + FN} \qquad recall_{doNotParticipate} = \frac{TN}{TN + FP}$$

Accuracy is measured by using the true values (true positives and true negatives). Accuracy values close to 100% indicate that the results obtained in the classification are close to those previously known.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Therefore, the metric *precision*$_{participate}$ indicates the probability of an issue classified as an instance of class "Participate" actually being of this class; and the metric *recall*$_{participate}$ indicates the probability of an issue to which a contributor a_k sent a message being classified as an instance of class "Participate". The metrics *precision*$_{donotParticipate}$ and *recall*$_{donotParticipate}$ follow the same understanding. Accuracy can be understood as a quality measure on the true values, independently to which class they belong.

4 RESULTS

In Tables 4 and 5, we present recall and precision metrics of the results obtained when we applied Naïve Bayes and J48 algorithms. We applied the algorithms to the top ten contributors of the project, like Ibrahim et al. (2010). We weighted their vocabularies using Text Frequency – Inverse Document Frequency (TF-IDF), considering each issue a document.

In Table 4, we present the recall and precision obtained when recommending a contributor to "Participate" of an issue. In general, the results presented by the J48 algorithm were better than those presented by Naïve Bayes algorithm. Cells with dark background represent the higher value comparing J48 and Naïve Bayes. For the top ten contributors, J48 presented an average recall of 53.9% and a precision of 64.8%, while for the Naïve Bayes the values are 66.8% and 15.9%. Even presenting higher recall, we can see that the Naïve Bayes performs very badly in terms of precision. It means that Naïve Bayes is recommending a greater number of appropriate issues, however it is also recommending a huge amount of impropriated issues (in average 85%, ranging from 70% for Top-1 to 91% for Top-7).

Table 4. Recall and precision for top ten contributors recommendations on the class Participate

Contributor	***Naive Bayes***		***J48***	
	Precision	*Recall*	*Precision*	*Recall*
Top – 1	0.296	0.715	0.645	0.595
Top – 2	0.204	0.689	0.637	0.525
Top – 3	0.190	0.656	0.680	0.546
Top – 4	0.145	0.745	0.642	0.479
Top – 5	0.148	0.639	0.662	0.632
Top – 6	0.105	0.573	0.529	0.371
Top – 7	0.086	0.615	0.571	0.410
Top – 8	0.102	0.641	0.615	0.523
Top – 9	0.178	0.737	0.688	0.628
Top – 10	0.124	0.545	0.714	0.446
Average	0.159	0.668	0.648	0.539

In Table 5, we present the results of recall and precision for the class “Do Not Participate”. As the amount of issues that a contributor does not take part is much higher than those that he takes part (for the Top-1 the proportion is 80:20, and, for the Top-10, it is around 93:7), the recall and precision for “Do Not Participate” seems to be high. This numbers could be presented as a good result; however our goal is to recommend issues that could be interesting to a contributor. Thus, J48 algorithm presents better results than Naïve Bayes.

Table 5. Recall and precision for top ten contributors recommendations on class Do Not Participate

Contributor	***Naive Bayes***		***J48***	
	Precision	Recall	Precision	Recall
Top – 1	0.870	0.528	0.890	0.909
Top – 2	0.895	0.497	0.914	0.944
Top – 3	0.870	0.451	0.914	0.950
Top – 4	0.937	0.462	0.938	0.967
Top – 5	0.922	0.535	0.954	0.959
Top – 6	0.907	0.460	0.932	0.963
Top – 7	0.932	0.445	0.951	0.974
Top – 8	0.929	0.454	0.955	0.968
Top – 9	0.933	0.520	0.948	0.960
Top – 10	0.902	0.522	0.934	0.978
Average	0.909	0.485	0.931	0.955

Figure 3 and Table 6 present a comparison of the accuracy of Naïve Bayes and J48. As J48 outperformed Naïve Bayes in both classes of prediction, the result for accuracy is straightforward. Figure 3 gives us a graphical demonstration on the difference among the algorithms.

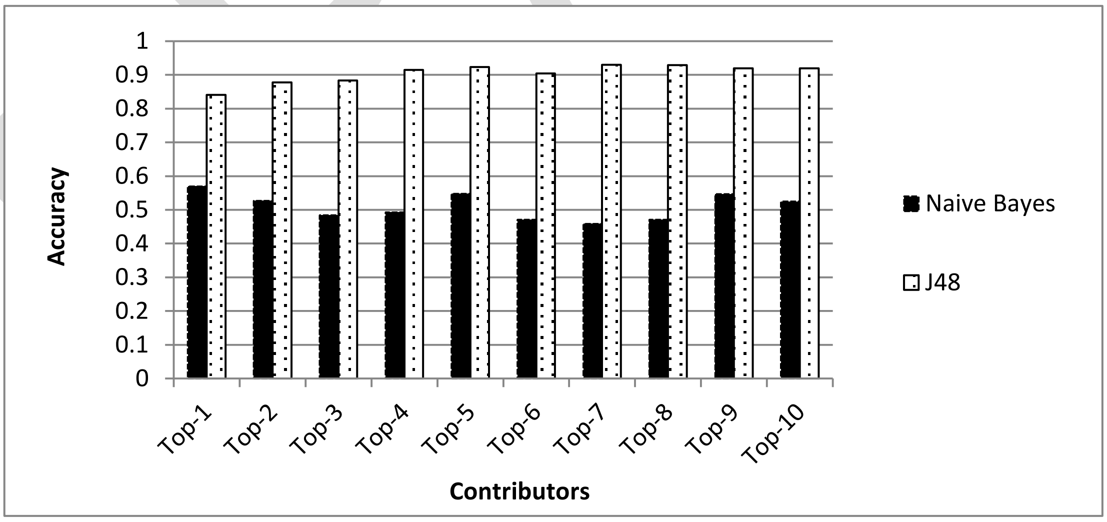

Figure 3. Accuracy: Naïve Bayes versus J48

Table 6. Accuracy: Naïve Bayes versus J48

Contributor	***Naive Bayes***	***J48***
	Accuracy	*Accuracy*
Top – 1	0.569	0.841
Top – 2	0.527	0.878
Top – 3	0.484	0.883
Top – 4	0.493	0.914
Top – 5	0.547	0.923
Top – 6	0.471	0.904
Top – 7	0.458	0.929
Top – 8	0.470	0.929
Top – 9	0.546	0.919
Top – 10	0.525	0.919
Average	0.508	0.902

Our results demonstrate that J48 is a better predictor when dealing with vocabularies of contributors to recommend issues.

5 DISCUSSION

In our analysis, we used the comments submitted to the issue tracker of Hadoop Commons project. We restricted the sample to the ten contributors that commented more issues. The numbers of recall and precision presented to the class "Participate" indicate that J48 can be used satisfactorily as a recommender, with an average accuracy of 90.27%.

One can notice that Naïve Bayes outperforms J48's recall. However, if we check the precision of the contributors, we can see that it is about 10% - 20%. Therefore, Naïve Bayes can recommend more True Positives, but 80% - 90% of the issues recommended are False Positives. In other words, applying Naïve Bayes stills overloading contributors with many issues that do not interest to them. The numbers of correctly recommended and misclassified instances are presented in Table 7. As one of our goals is to reduce overload, we can affirm that J48 does the job better than Naïve Bayes in this case.

Table 7. Comparison of Correctly and Incorrectly Recommendations for Class "Participate"

Contributor	***Naive Bayes***		***J48***	
	Misclassified	Correct	Misclassified	Correct
Top – 1	565	238	109	198
Top – 2	654	168	73	128
Top – 3	611	143	56	119
Top – 4	725	123	44	79
Top – 5	571	99	50	98
Top – 6	604	71	41	46
Top – 7	765	72	36	48
Top – 8	722	82	42	67
Top – 9	467	101	39	86
Top – 10	430	61	20	50

Table 6 presented in Section 4 shows the result of the approach using J48 with a high accuracy, varying from 84.1% to 92.9%. However, the classifiers present lower recall values to class “Participate” than to class “Do Not Participate.” We understand that this is a deficiency of the classifiers, since when we do not recommend an issue to which a contributor should participate we may be losing his skills and knowledge to discuss or help fix the issue.

We have applied the same approach as other recommendation studies that use OSS projects data and compared the recommendations to the actual participation on the issues. This should not represent the most accurate test data. In OSS projects, a volunteer collaborator is not obligated to keep contributing and searching for issues to collaborate. For example, there should be issues that were predicted by the algorithms and the contributor should have participated, but he/she was not able to at that time. A collaborator that became away from the project for a period could have missed some interesting issues.

The results obtained are valid just for the top ten contributors, which took part in a high number of issues. The Top-1 contributor submitted comments to more than 1600 issues, and the Top-10 to 534 issues. To verify if the pattern obtained maintains for contributors that took part in fewer issues, we conducted the same steps of the method for other five contributors. For this exploratory study, we chose the subjects following these steps: (i) retrieve the number of issues that each contributor of the project took part; (ii) discard all contributors that participated in just one issue; (iii) performed a quartile analysis; and (iv) select the five top contributors of the third quartile. They have contributed to 52 or few issues.

The results for this exploratory study are presented in Table 8. The table shows the recall and precision of Naïve Bayes and J48 algorithms for the class “Participate” and also the overall accuracy of both algorithms. Comparing to the Top-10 contributors we can see that the precision is presenting better results, in average 12% better. It is also possible to notice that recall is around 50%. In two cases, J48 achieved 100% precision with high recall values. For ThirdQuartile-3, the recall was 75%, recommending 6 out of 8 possible instances. For ThirdQuartile-5, the recall was 54.5%, with 6 issues correctly predicted out of 11 possible. The accuracy present outstanding results for J48 (98.8%), however this value is influenced due to the class imbalance. The number of issues on class “Do not Participate” is much larger those one on class “Participate” (the proportion is around 1:160), what considerably influences the accuracy measure.

Table 8. Recall and precision for recommendations made to the third quartile contributors

Contributor	***Naive Bayes***			***J48***		
	Precision “Participate”	*Recall “Participate”*	*Accuracy*	*Precision “Participate”*	*Recall “Participate”*	*Accuracy*
ThirdQuartile-1	0.034	0.538	0.653	0.800	0.308	0.983
ThirdQuartile-2	0.027	0.500	0.579	0.800	0.400	0.984
ThirdQuartile-3	0.003	0.125	0.560	1.000	0.750	0.998
ThirdQuartile-4	0.000	0.000	0.609	0.444	0.500	0.987
ThirdQuartile-5	0.018	0.182	0.634	1.000	0.545	0.984
Average	0.013	0.300	0.602	0.774	0.480	0.988

In general, J48 stills performing satisfactorily, with a very high precision rate and a fair recall. As a preliminary result, we can say that the classifier maintained its fair results even for contributors with fewer contributions. To find a pattern for a project or an ecosystem it would be necessary to conduct a case study covering all the contributors of a project and more projects.

On previous works (Schwerz et al., 2012; Schwerz et al. 2013), we conducted the analysis considering the vocabulary of the issues without assigning weight to the terms. We created the WORLD using a binary scale to determine if a term appeared or not during a discussion of an issue. Table 9 presents the comparison among the best results obtained on the previous study and the results obtained with J48 in the current study (using TF-IDF). We can see that recall values for the current study are much better, doubling the value in some cases. Even with lower results for precision, we can say that the current approach can lead to better results. Checking the raw number of issues classified correctly for top ten in total, we have 949 for the current approach, versus 585 using previous approach. For example, for Top-1 using the current approach, we retrieved 198 correct instances, while in the previous approach we retrieved just 100.

Table 9. Comparing results with our previous work (SCHWERZ et al. 2013)

Contributor	***Current Approach***			***Previous approach (SCHWERZ et al. 2013)***		
	J48			***J48***		
	Precision "Participate"	*Recall "Participate"*	*Accuracy*	*Precision "Participate"*	*Recall "Participate"*	*Accuracy*
Top – 1	0.645	0.595	0.841	0.763	0.303	0.826
Top – 2	0.637	0.525	0.878	0.619	0.303	0.843
Top – 3	0.680	0.546	0.883	0.733	0.368	0.884
Top – 4	0.642	0.479	0.914	0.610	0.303	0.909
Top – 5	0.662	0.632	0.923	0.750	0.342	0.912
Top – 6	0.529	0.371	0.904	0.527	0.204	0.888
Top – 7	0.571	0.410	0.930	0.681	0.405	0.936
Top – 8	0.615	0.523	0.929	0.907	0.390	0.953
Top – 9	0.688	0.628	0.919	0.841	0.390	0.916
Top – 10	0.714	0.446	0.919	0.760	0.543	0.935
Average	0.648	0.539	0.899	0.711	0.340	0.898

When we compare our results with Ibrahim et al. (2010), we found that using simple textual analysis bring similar results as they obtained. Their accuracy for the top ten contributors, when recommending to which mailing list threads they should contribute was 85% - 89%. Our results in terms of accuracy are very similar, as we achieved accuracy ranging from 84.1% to 92.9%. Differently from their work, we are not using any contextual or profiling factors to recommend the participation. We suspect that better results can be obtained if we use other contextual factors to profile a contributor – like those presented in (Matter et al., 2009) and (Ibrahim et al., 2010).

6 THREATS TO VALIDITY AND FUTURE WORK

The results of our analysis are specific for the analyzed project and cannot be directly generalized. Projects have different characteristics due to the organizational structure or application domain, but the same method can be applied to identify what algorithm and configuration best suit for each case. As future work, an analysis with more projects or using the whole Apache Foundation ecosystem may look for patterns among different projects and find results that are more generalizable.

We did not investigate the real reasons of false positives (FP) and false negatives (FN). As we are dealing with a large OSS project, FPs can be caused by many reasons, including the unavailability of the volunteer contributors at the time the issue was under discussion, the nature of the issue (some issues just report a problem already fixed), or the overload of information that make it difficult for contributors to find issues of interest. We tried to reduce this threat by reducing the scope of issues to the period between the first and last messages sent by the contributors. A manual analysis on false positives and false negatives can explain some of the recommender errors. It is also possible to look for the long periods when the volunteers did not contribute to the project and verify if the FPs occurred in these periods.

We just analyzed two machine algorithms with no tuning to conduct our study. We can check the efficiency of other algorithms and different settings of these algorithms. There is also the possibility of using raw Term Frequency (TF) instead of using TF-IDF to verify if it brings better results, once, according to Canfora et al. (2012), TF-IDF is useful to discriminate different corpus rather than match similar corpus.

The method used to extract the tokens that compound WORLD_p set can be better explored, mainly regarding the source code sent in the comments. We also can combine the vocabulary with other factors, as described at the end of Section 5.

Even conducting the exploratory study presented on Section 6, we cannot say that our approach will present similar results to all the contributors of this project. We can also say that the classifiers applied need to receive historical data as input, so they cannot produce the same results for newcomer contributors (cold start problem).

Regarding future works, in addition to what we presented before, we understand that other factors that support contributors profiling – such as social, work rhythms and temporal analysis – can be combined to the vocabulary to bring better results. We also aim to conduct studies that apply Social Network Analysis to the issues in order to recommend based on previous social interactions.

7 CONCLUSIONS

Although source code repositories and issue tracking systems facilitate the management of activities in OSS projects, the large amount of information can overload more experienced contributors. In particular, choosing appropriate issues to contribute can be a complex and time-consuming activity. In this paper, we present an alternative for minimizing the negative effects of this activity by exploring the history of messages sent to issues to recommend appropriate issues to contributors.

The results obtained in this paper indicate that the content of comments posted in issues of an OSS project is an important factor to build an issue recommender for contributors. Based on this factor, we showed that it is possible to use well known classification algorithms to reduce the overload of a contributor, indicating to which issues he should be interested. By using J48 algorithm to classify the issues, we achieved average precision of 64.31%, an average recall of 53.02% and accuracy of 90.27%.

We recognize that the recall values for J48 still need to be improved to recommend better results for contributors. However, we improved the recall if we compare to the results obtained in our previous study (Schwerz et al., 2013) that used the same algorithms, but using different inputs. We could see that applying the same approach to contributors that are not the top ten we keep obtaining satisfactory values of recall and precision using J48. For two contributors we have 100% recall and precision. Therefore, we have evidence that J48 classifier, using information about the vocabulary, can also bring satisfactory results to contributors that took part in fewer issues.

ACKNOWLEDGEMENTS

The authors thank Fundação Araucária for the financial support (DINTER IME/USP and grant 344/12). Marco Aurélio Gerosa and João Eduardo Ferreira receive individual grant from Brazilian National Research Council (CNPq).

REFERENCES

ABEL, Fabia; BITTENCOURT, Ig Ibert; COSTA, Evandro; HENZE, Nicola; KRAUSE, Daniel; VASSILEVA, Julita. Recommendations in Online Discussion Forums for E-Learning Systems. IEEE Trans. Learn. Technol. v. 3, n. 2, pp. 165-176. April 2010

ALONSO, Omar; DEVANBU, Premkumar; GERTZ, Michael. Expertise identification and visualization from CVS. In Mining Software Repositories, 08. Proceedings of the 2008 Intl. working conference on Mining Software Repositories, IEEE Computer Society, Washington, DC, USA, 2008.

ANVIK, John; HIEW, Lyndon; MURPHY, Gail C. Coping with an open bug repository. In: Eclipse'05. Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange, ACM, New York, NY, USA, pp. 35 – 39, 2005.

ANVIK, John; HIEW, Lyndon; MURPHY, Gail C. Who should fix this bug? In: International Conference on Software Engineering 2006. Proceedings of the 28th international conference on Software engineering (ICSE '06). ACM, New York, NY, USA, pp. 361-370, 2006.

BIRD, Cristian. Predicting email response using mined data, Disponível em: http://www.cabird.com/papers/mlpaper.pdf, last accessed, August, 2013.

CANFORA, Gerardo; CERULO, Luigi. How software repositories can help in resolving a new change request. In: Workshop on Empirical Studies in Reverse Engineering. Proceedings of the Workshop on Empirical Studies in Reverse Engineering (STEP 2005), IEEE Computer Society, Washington, DC, USA, 2005.

CANFORA, Gerardo; DI PENTA, Massimiliano; OLIVETO, Rocco; PANICHELLA, Sebastiano. Who is going to mentor newcomers in open source projects? In: 20th International Symposium on the Foundations of Software Engineering (FSE '12). Proceedings of the ACM SIGSOFT 20th International Symposium on the

Foundations of Software Engineering (FSE '12). ACM, New York, NY, USA, Article 44 , 11 pages, 2012

COSLEY, Dan; FRANKOWSKI, Dan; TERVEEN, Loren; RIEDL, John. SuggestBot: using intelligent task routing to help people find work in wikipedia. International Conference on Intelligent User Interfaces, 12th. Proceedings of the 12th International Conference on Intelligent User Interfaces (IUI '07). ACM, New York, NY, USA, pp. 32-41, 2007.

CUBRANIC, Davor; MURPHY, Gail C. Automatic bug triage using text classification. In: International Conference on Software Engineering (SEKE04). Proceedings of International Conference on Software Engineering and Knowledge Engineering SEKE’04), pages 92-97, KSI, Illinois, USA, 2004.

FRITZ, Thomas; MURPHY, Gail C.; HILL, Emily. Does a programmer’s activity indicate knowledge of code? In: ESEC-FSE ’07. Proceedings of the 6th European software engineering conference, ACM, New York, NY, USA, pp. 341–350, 2007.

IBRAHIAM, Walid M.; BETTENBURG, Nicolas; SHIHAB, Emad, ADAMS, B.; HASSAN, Ahmed E. Should I contribute to this discussion? In: Mining Software Repositories, 2010. Proceedings of the Working Conference on Mining Software Repositories, IEEE Computer Society, Washington, DC, USA, pp181–190, 2010.

JONE, Karen Spärck. A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation v. 28 n. 1, pp. 11–21. 1972.

MATTER, Dominique; Adrian Kuhn; Oscar Nierstrasz. 2009. Assigning bug reports using a vocabulary-based expertise model of developers. In: Working Conference on Mining Software Repositories Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR '09). IEEE Computer Society, Washington, DC, USA, pp. 131-140, 2009.

MITCHELL, Tom M. Machine Learning. McGraw-Hill, New York, NY-USA, 1997.

MOCKUS, Audris; HERBSLEB, James D. Expertise browser: a quantitative approach to identifying expertise. In International Conference on Software Engineering 2002 (ICSE’02). Proceedings of the 24th Intl. Conference on Software Engineering, ACM, New York, NY, USA, pp. 503–512, 2002.

PORTER, Martin F. An algorithm for suffix stripping. Program, v.14, n.3, pp. 130–137, 1982.

QUINLAN, Ross. Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

RAYMOND, Eric S. The Cathedral and the Bazaar (1st ed.). Tim O'Reilly (Ed.). O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1999.

SCHWERZ, André L.; LIBERATO, Rafael; WIESE, Igor S.; STEINMACHER, Igor; GEROSA, Marco A.; FERREIRA, João E., Prediction of Developer Participation in Issues of Open Source Projects. In: Simpósio Brasileiro de Sistemas Colaborativos 2012. In: Proceedings of Brazilian Symposium on Collaborative Systems (SBSC 2012), IEEE, Washington, DC, USA, pp.109-114, 2012.

SCHWERZ, André L.; LIBERATO, Rafael; WIESE, Igor S.; STEINMACHER, Igor; GEROSA, Marco A.; FERREIRA, João E., Predizendo a Participação de

Desenvolvedores em Discussões em Projetos de Software Livre. In: Workshop Internacional de Software Livre 2013, 2013.

SHIHAB, Emad. An Exploration of Challenges Limiting Pragmatic Software Defect Prediction. 2012. PhD Thesis. School of Computing, Queen's University, Kingston, Ontario, Canada, 2012

SIY, Harvey; CHUNDI, Parvathi; SUBRAMANIAM, Mahadevan. Summarizing developer work history using time series segmentation: challenge report. In Mining Software Repositories MSR’08. Proceedings of the 2008 Intl. working conference on Mining software repositories (MSR’08), ACM, New York, NY, USA, pp. 137–140, 2008.