Monitoring Negative Sentiment-Related Events in Open Source Software Projects

Lingjia Li, Jian Cao* and Qin Qi
Department of Computer Science and Engineering, Shanghai Jiao Tong University, China
Email: {jessie_llj, cao-jian, qi_ng616}@sjtu.edu.cn

***Abstract*—Open source software (OSS) development is a highly collaborative process where individuals, groups and organizations interact to develop, operate and maintain software and related artifacts. The developers' sentiment in this process can have an impact on their working willingness and efficiency. Monitoring sentiment factors can help to improve OSS development and management. However, no method has been proposed to dynamically monitor the sentiment phenomena during the OSS development process. In this paper, an approach to detect Negative Sentiment-related Events (NSE) is proposed. It consists of two steps. The first step is to identify the burst interval of negative comments from open source projects, which corresponds to a NSE. The second step is to annotate this NSE with its event type. To support this approach, the types of NSEs in OSS projects are defined through an empirical study and classifiers are trained to annotate event types automatically. Moreover, conversation disentanglement techniques are employed to make the comments extracted more complete. Finally, the factors that have an influence on NSEs in the OSS project are studied.**

***Index Terms*—Sentiment analysis, Negative Sentiment-related Events, Mining software repositories**

I. Introduction

It has been proved that during the software development process, developers' sentiments or emotions can have an effect on their productivity, creativity and contribution satisfaction [1][2][3][4]. In recent years, open source software (OSS) development, which is a highly collaborative activity carried out by developers in a volunteer-driven and self-organized manner, has become popular. Obviously, developers' sentiments play a very important role during the collaborative development process in OSS projects [5].

Sentiments can be roughly classified into *positive*, *negative* and *neutral*. Studies find that negative sentiments in OSS projects can have a negative impact on developers' working efficiency and collaboration willingness [2]. On the other hand, the events that cause negative sentiments are often associated with some important events. Therefore, detecting a Negative Sentiment-Related Event (NSE) in OSS projects can help project managers take effective measures. At the same time, the statistics of these events can also reveal the overall status of the OSS projects.

In OSS projects, developers are typically geographically distributed and rely on mailing lists or issue tracking systems to coordinate their work. The information exchanged between developers is recorded by the online OSS development platforms and we can exploit it to identify and monitor developers' sentiments using sentiment analysis techniques [6]. In the field of social media, methods have been proposed to detect bursts in sentiment-related topics and to extract the corresponding events. However, in the software engineering domain, there is little research on how to monitor developers' sentiments and detect negative bursts or phenomena during the development process.

In this research, we study the methods to dynamically monitor the NSEs in OSS projects and we also analyze its relationship with various factors during the development process. We intend to answer the following research questions:

RQ1: What are the main negative sentiment-related event types in OSS projects?

This research question seeks to understand what specific events take place in the negative sentiment burst periods that have triggered the burst of sentiment inside the community. We summarize nine main NSE types through empirical studies.

RQ2: How to detect and identify negative sentiment-related events during the OSS development process?

To detect and identify NSEs during the OSS project, we adopt the Kleinberg model and develop multi-label classifiers on a manually labeled dataset. Conversation disentanglement is further employed to improve the completeness and coherence of the detected events.

RQ3: What factors have an influence on negative sentiment-related events during the OSS development process?

We apply association rule mining techniques to investigate the relationship between NSE types and the factors related to projects, issues and developers. This effort aims to help managers adjust collaborators' sentiments strategically.

The rest of this paper is organized as follows. We introduce the related work in Section 2 and the dataset in Section 3. NSE types are presented in Section 4. Section 5 elaborates on the methods we propose to dynamically monitor NSEs. We discuss the factors influencing those events in Section 6. Threats to validity are discussed in Section 7. Finally, we conclude the paper and suggest directions for future work in Section 8.

*Corresponding Author

II. Related Work

A. Sentiment analysis and monitoring for OSS projects

Sentiment analysis [7] uses natural language processing, text analysis and computational techniques to automate the extraction or classification of sentiments from texts. There are a number of existing mature and publicly available tools such as SentiStrength [8], Stanford NLP sentiment analyser [9] and Natural Language Text Processing (NLTK) [10]. Applying sentiment analysis to software engineering (SE) communities is a relative new research field. It has been discovered that sentiment analysis tools trained or evaluated on non-technical datasets can generate unreliable results on SE datasets [11]. Therefore, some tools have been developed specifically for the SE domain, such as SentiStrength-SE [12], Senti4SD [13], Senti4SD [14] and SentiCR [15].

OSS projects are usually built and maintained by teams of developers and testers scattered in different geographic locations, which makes it difficult for project managers to track and regulate developers' sentiments to avoid potential risks during the development process. Fortunately, OSS team members often rely on mailing lists and issue tracking systems for interaction and communication, during which messages are recorded as textual information. Therefore, some researchers have applied sentiment analysis techniques to detect and monitor developers' sentiments from textual information. For example, Jurado et al. [16] introduced sentiment analysis techniques to identify the sentiments of developers in issue reports. It is found that potential sentiments are expressed in issue comments and this information can be monitored like other factors during the development process. Tourani et al.[17] conducted sentiment analysis on the emails of two projects in the Apache software ecosystem and visualized the sentiments of each month. They found that the emails of users and developers carry both positive and negative sentiments and they also summarize the types of sentiments they observed. However, prior studies are mainly descriptive and focus on visualizing sentiment evolution in the project. No method has yet been proposed to dynamically monitor developers' sentiments or to detect the events related to such sentiment phenomena during the OSS development process.

B. Sentiment-based event detection in social media

On social media such as Twitter and Sina Weibo, users usually post a large number of posts containing positive or negative sentiments on various topics. Monitoring topics with bursty sentiments, thus revealing specific sentiment-related events, is of great significance for public opinion monitoring.

To detect bursty patterns from data streams, Kleinberg proposed a 2-state automaton to model the arrival volume of documents in each timestamp of a stream and discover bursty volumes from the underlying state sequence [18]. The Kleinberg model is widely used to detect events in social media.

Zhang Lumin et al. performed sentiment analysis on the information flow in Weibo, used the Kleinberg model and methods based on feature words to detect sentiment burst periods, and extracted the burst events using spectral clustering. The results show that there is a strong correlation between events and public sentiments. Nguyen et al. [19] used sentiment signals and the burst structure in social media to detect sentiment burst events in LiveJournal blogs. Kleinberg and the Latent Dirichlet Allocation (LDA) topic model [20] are applied to further analyze blog posts corresponding to the specific sentiment. The results show that sentiment bursts are consistent with real-world news events. Unfortunately, no study has yet been conducted to analyze or monitor sentiment-related events in OSS projects.

C. Negative events in OSS development

An issue tracking system (ITS) serves as a key knowledge repository, a communication and collaboration hub in OSS development [21]. Issue discussions cover all aspects of software development, and software stakeholders may hold different, even contradictory and conflicting views. Conflict is defined as disagreement, both manifest and latent, among team members and implies incompatible goals or interests [22]. OSS development is a typical virtual work, where conflicts are typically separated into task conflicts, process conflicts, and relationship conflicts [23]. Task conflict involves disagreements about what to add and how to change the contents of the task at hand [24]. It may also be caused by conflicting opinions on choosing project dependencies [25]. Relationship conflict involves interpersonal incompatibilities, and mainly manifest in personal attacks such as online controversy. Developers usually speculate on the process based on indirect hints and the behavior of other developers to promote the development of software projects. However, a lack of direct coordination among team members may lead to process conflict, which mainly evolves out of misunderstandings on how to accomplish the team's goal, such as code specifications and duty delegation [24].

III. Dataset

In this section, we describe our dataset for subsequent analyses.

GitHub is a popular code repository platform for many well-known and active OSS projects. In GitHub, each project has its own repository and the history of the source code, commits, issues, and other related data are publicly accessible. We obtained the dataset through GitHub REST API[1]. TABLE I details the information on the 19 projects on which we focus, which vary in domain, size and programming language. To ensure a high sample coverage [26] of the sampled data, the selection of programming languages is basically in line with the Top Languages[2] on GitHub.

IV. Classification of Negative Sentiment-Related Events in OSS Projects

To answer RQ1, i.e., to understand what kind of NSEs take place, we read a large array of negative sentiment snippets.

[1]https://developer.github.com/v3/
[2]https://octoverse.github.com/

TABLE I
DETAILS OF PROJECTS

Project	Language	#Developers	#Comments	#Issues
Pandas	Python	5712	134014	22854
IPython	Python	3412	55430	10172
gRPC	C++	3141	79484	14828
Scala	Scala	662	45379	6824
Typescript	Typescript	9595	110130	24534
scikit-learn	Python	1912	18633	1978
Vim	C	1442	16490	3705
elasticsearch	Java	1401	10159	2427
Node.js	JavaScript	1168	11152	1136
OpenRA	C#	681	31162	6026
Three.js	JavaScript	2010	25203	5465
Tensorflow	Python	4755	19882	2922
React	JavaScript	1649	5628	522
Go	Go	3938	45508	5254
Moby	Go	7123	28399	3142
Webpack	JavaScript	1763	9988	414
Vue.js	JavaScript	762	1880	372
jQuery	JavaScript	97	538	72
Rails	Ruby	1139	3718	511

The causes of negative sentiments and event types are further summarized through empirical study.

Our first attempt was to extract keywords implying topics of the extracted snippets using LDA dynamic topic models. However, from our observation, the extracted keywords are not understandable enough to characterize the events. The reason for this may be that GitHub comments are usually informal, short, and strongly dependent on the context, thus traditional NLP methods such as LDA cannot achieve good results on this type of text [27]. Therefore, we manually summarized NSE types and then constructed classifiers by extracting features of the negative sentiment snippets to identify specific events and reveal the cause of negative sentiments.

In order to systematically summarize the NSE types, we conducted an empirical case study analysis, referring to theories in the relevant literature on issue resolution and the collaborative problem-solving process, and preliminarily summarized nine NSE types. We present the detailed introduction to the categorization and meaning of each NSE type in this section.

We first divide NSEs into conflict events and non-conflict events. Conflict events are mainly manifested in the incompatibility of opinions during the issue resolution process, which can be divided into task, relationship and process conflict [24]. According to the taxonomy presented in section II-C, task conflict includes disagreements on whether or how to fix, enhance or implement features and third-party problems; process conflict includes working progress and documentation and repository standard problems; relationship conflict involves disruptive comments and personal attacks. Non-conflict events are manifested as NSEs where developers hold similar points of view. According to the types of sentiments in software development, this can be divided into confusion, apology, and frustration. These correspond to confusion and help seeking, apology and bug report and fixing. Detailed explanations and a representative example of each event type are listed in Table II.

V. Detection of Negative Sentiment-related Events

This section answers RQ2. The detection of NSEs can be divided into two steps. The first step is to identify the burst periods of messages that contain negative sentiments. The second step is to annotate these burst periods with NSE types.

A. *Identification of burst periods*

Sentiments are commonly expressed in developers' issue comments [16]. We compare the performance of sentiment analysis over three tools based on supervised learning for SE domain, namely Senti4SD, SentiCR and SentiSW, using a gold standard containing 3,000 manually labeled issue comments of ten OSS projects on GitHub [14]. The evaluation results show that SentiCR has the best performance on negative sentiment and overall classification. Therefore, we select SentiCR as the sentiment analysis tool in subsequent experiments.

We constructed our burst detection model based on the definition of the Kleinberg model. By definition, documents arrive in discrete batches, and a sequence of batched arrivals could be considered bursty if the fraction of relevant documents alternates between reasonably long periods in which the fraction is small and other periods in which it is large. Supposing there are m documents in the batch sequence $B = (B_1, B_2, ...B_m)$, among which the t-th batch of B_t contains d_t texts, and r_t are texts related to the target, the goal is to find the optimal state sequence $q*$ given the text batch sequence B to minimize the cost function:

$$c(q|B) = \sum_{t=0}^{m-1} \tau(i_t, i_{t+1}) + \sum_{t=1}^{m} \sigma(i_t, r_t, d_t) \qquad (1)$$

and eventually the burst intervals are obtained through this optimal state sequence.

In this study, negative and total comments are segmented in units of days separately. Therefore, r_t in the definition represents the number of negative comments on the t-th day, d_t represents the total number of comments on the t-th day, and the expected proportion of negative sentiment comments is $p_0 = \frac{\sum_{t=1}^{m} r_t}{\sum_{t=1}^{m} d_t}$. Then, according to the process, the negative comment generation process is modeled, and the bursty intervals of negative comments are obtained from the optimal state sequence. Figure 1 illustrates the detected burst intervals of negative comments in Pandas from 2011 to 2019. The red line represents the proportion of negative comments (the sequence $p = (p_0, p_1, .., p_m)$); the blue line represents the number of negative comments each day(the sequence $r = (r_0, r_1, ..., r_m)$) and the gray part represents the detected burst interval. Thirty bursty intervals of negative sentiments are detected.

According to the obtained bursty intervals, all the issue discussions of the projects are retrieved during these intervals. A negative sentiment snippet is defined as the snippet composed of all negative sentiment comments of an issue during the interval.

TABLE II
SUMMARY OF NEGATIVE SENTIMENT-RELATED EVENT TYPES

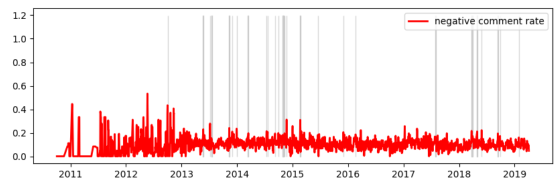
Event Type	**Description**	**Example**
Disagreement on whether to fix, enhance or implement features (DW)	Disagreement on the necessity of a specific task like bug fixing, feature implementation or enhancement	I don't think this is a Tensorflow issue, seems to be caused by your special environment settings. Git inside git is quite a no-no, to be honest, unless you're working with git submodules.
Disagreement on how to fix, enhance or implement features (DH)	Disagreements arising from discussions on the pros and cons of solutions and dissatisfaction with current commits or features	In my opinion raising on integers in not an option. It is a perfectly valid input (as it is also the default for 'Timedelta'), so why would we raise? OK, because we think it will confuse a lot of users. And in that case it is a bit of a balance we should seek here, but I lean to better documenting it (and adding the 'unit' kwarg).
Third-party problems (TP)	Negative sentiments related to third-party dependencies, platforms and browser issues, such as expressing unsatisfied opinions and disagreements on third-party issues	@aardgoose Hmm. Firefox seems to be ok. I'm not super familiar with the Firefox memory profiler, but usage seems relatively consistent between 85dev and 85. Oddly though (maybe a separate issue) randomly (as I can tell) GC goes crazy in both versions in Firefox. Haven't had time to look further into it. Chrome definitely is having issues with this.
Working progress problems (WP)	Dissatisfaction with the workload, task assignments and work schedule and discussions of issues that hinder working progress	Every new proposal has been further away from a solution. Just because I'm not paid doesn't mean my time is worthless. And the disagreement on this whole issue has been a perfect example of a glorious waste of my time. What's the next proposal? Embedding a PHP interpreter into the language and deferring all mathematical operations to it?"
Documentation and repository standards (DR)	Negative sentiments related to technical documentation, naming convention, and version specifications, which frequently co-occur with other event types	I was just trying, but I know too little of nodejsnpm. For instance mocha docs say that tests should be run by:'.node_modulesmochabinmocha',while threejs-mocha docs prefer 'npm test'. The second one doesn't work for me, but I just don't feel competent in mocha or nodejs generally to write docs. Sorry, cannot help.
Confusion and help seeking (CH)	Expressing confusion and seeking help due to ambiguity on requirements and implementation details	How would I do this in TF2? Is there some feature that helps me with this? I need some general guidance (maybe I should've posted on stackoverflow - I wasn't sure).
Apology (AP)	Apologizing or expressing guilt due to delays in work progress or inconvenience inflicted on other developers	I just realized I made a horrible mistake with the issue title. I never meant to say the Demo Truck should be nerfed in general but that's what the discussion coagulates around, and I didn't realize until now. Sorry for wasting your time guys:/ I'll be making a new issue at a later time.
Disruptive comments and personal attacks (DP)	Disruptive comments or personal attacks on other developers	They are irrelevant. I get much better results in 3DS Max and that tells me that the GLTFLoader and/or your PBR model are not properly implemented. Your girlfriend looks bad but you're happy because your boss told you that's normal. OMG ...
Bug report and fixing (BF)	Descriptions of failures and errors in compilation, testing and integration	Yes tried that, but it gets stuck at installing cuda-**** something. Attempted to run this multiple times but for some reason it doesn't find one of the packages. Need to try this again to see which specific package it wouldn't install ...

Issue discussions on GitHub are asynchronous and reply-to relationships between messages are normally not displayed. In order to better understand the logical structure of dialogues, and accurately locate all comments corresponding to the NSE, we use conversation disentanglement techniques to identify the reply-to relationship and dialogue structure.

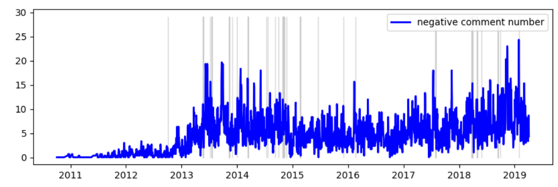
We adopt a feedforward neural network model based on a data set of 77,563 messages of the Ubuntu IRC conversation disentanglement [28]. The model discriminates the graph structure of a nested conversation group, where nodes represent comments, edges represent the reply-to relationships, and each connected component in the graph represents a conversation.

To confirm the reliability of our model, we extracted the conversation disentanglement results of about 50 issue snippets and manually evaluated the accuracy of the results. Of these, 43 are reasonable, that is, the causes and consequences of the events are more completely described and covered, which indicates the model is relatively reliable.

Figure 2 illustrates an example of conversation disentanglement on issue discussion. The content framed by the orange dotted line represents the original negative sentiment snippet; the curves represent the relationships between comments, which discriminate two independent conversations (the blue and green lines). In this case, a new comment is included in the original negative sentiment snippet, which suggests that *optics* should be considered to determine whether to enhance the current performance of *HDBSCAN*. It can be found that this comment serves as the cause of *optics*, which is discussed in the

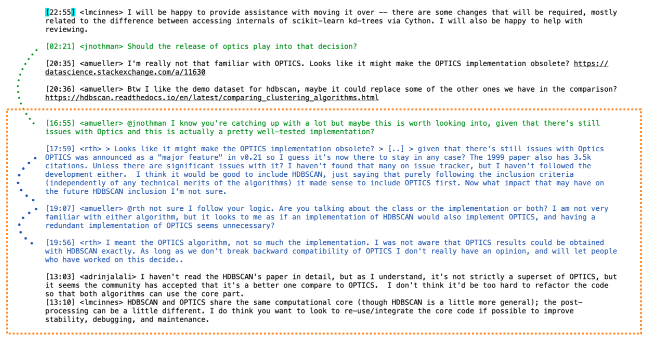


(a) Evolution of the proportion of negative comments in Pandas



(b) Evolution of the number of negative comments in Pandas

Fig. 1. Monitoring results of bursty intervals of negative sentiments in Pandas



```
[22:55] <lmcinnes> I will be happy to provide assistance with moving it over -- there are some changes that will be required, mostly
related to the difference between accessing internals of scikit-learn kd-trees via Cython. I will also be happy to help with
reviewing.

[02:21] <jnothman> Should the release of optics play into that decision?

[20:35] <amueller> I'm really not that familiar with OPTICS. Looks like it might make the OPTICS implementation obsolete? https://
datascience.stackexchange.com/a/11630

[20:36] <amueller> Btw I like the demo dataset for hdbscan, maybe it could replace some of the other ones we have in the comparison?
https://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html

[16:55] <amueller> @jnothman I know you're catching up with a lot but maybe this is worth looking into, given that there's still
issues with Optics and this is actually a pretty well-tested implementation?

[17:59] <rth> > Looks like it might make the OPTICS implementation obsolete? > [..] > given that there's still issues with Optics
OPTICS was announced as a "major feature" in v0.21 so I guess it's now there to stay in any case? The 1999 paper also has 3.5k
citations. Unless there are significant issues with it? I haven't found that many on issue tracker, but I haven't followed the
development either.  I think it would be good to include HDBSCAN, just saying that purely following the inclusion criteria
(independently of any technical merits of the algorithms) it made sense to include OPTICS first. Now what impact that may have on
the future HDBSCAN inclusion I'm not sure.

[19:07] <amueller> @rth not sure I follow your logic. Are you talking about the class or the implementation or both? I am not very
familiar with either algorithm, but it looks to me as if an implementation of HDBSCAN would also implement OPTICS, and having a
redundant implementation of OPTICS seems unnecessary?

[19:56] <rth> I meant the OPTICS algorithm, not so much the implementation. I was not aware that OPTICS results could be obtained
with HDBSCAN exactly. As long as we don't break backward compatibility of OPTICS I don't really have an opinion, and will let people
who have worked on this decide..

[13:03] <adrinjalali> I haven't read the HDBSCAN's paper in detail, but as I understand, it's not strictly a superset of OPTICS, but
it seems the community has accepted that it's a better one compare to OPTICS.  I don't think it'd be too hard to refactor the code
so that both algorithms can use the core part.
[13:10] <lmcinnes> HDBSCAN and OPTICS share the same computational core (though HDBSCAN is a little more general); the post-
processing can be a little different. I do think you want to look to re-use/integrate the core code if possible to improve
stability, debugging, and maintenance.
```

Fig. 2. An example of conversation disentanglement of issue discussion

following event. Including it can improve the logical integrity of the current event snippet.

B. Classification of negative sentiment-related events

Based on the nine NSE types, we performed a data labeling task and constructed a standard dataset. Due to the small number of samples of some event types (only 11 samples are labeled for DP) and high overlapping over some types (DW and DH), we re-organized the nine event types in the dataset. Specifically, DP and DW are merged into one type and DP is removed. Furthermore, an external corpus is imported to expand the size of the training set. For CH, we obtained a standard dataset [29] composed of 219 comments expressing help seeking on Stack Exchange and added them to the existing training set. The distribution of event types in the standard data set is shown in Table III.

Annotating an NSE with an event type is a multi-label classification problem. We trained a binary classifier for each event type. Because different event types have different characteristics, we employed different classification strategies for each type. Of these, the first three types (i.e., disagreements on feature implementation, confusion and help seeking, bug report and fixing) have complex features and relatively fixed linguistic patterns so we adopted the supervised learning model.

TABLE III
DISTRIBUTION OF NSE TYPES IN THE STANDARD DATASET

Event Type	Event Number
Confusion and help seeking	387
Bug report and fixing	214
Disagreements on feature implementation	206
Apology	121
Third-party problems	65
Documentation and repository standards	60
Working progress problems	37
Total	754

The latter four types (third-party issues, work schedule issues, apologies, and documentation and code base standardization issues) have fewer samples and the keyword features are more explicit, which could be reflected by a single sentence. The supervised learning method has a poor performance in these contexts as noise will be introduced when the entire comment snippet is inputted to the classifier. Therefore, a rule-based classification method is adopted in which we manually selected features corresponding to each event type. The specific keywords and features of each type are listed as follows.

- Disagreements on feature implementation: The extracted features include the number of keywords expressing strong opposition or proposing alternative solutions and the number of typical linguistic patterns (such as ”SUPPORT + BUT” , ”BUT + NEGATION”).
- Confusion and help seeking: The extracted features include the number of question words, question marks, and keywords in the title and comments related to the expression of confusion.
- Bug report and fixing: The extracted features include the number of code lines, code blocks, @-mention behaviors and keywords related to bug resolution in issue title and comments.
- Apology: Extract related keywords including ”apologize”, ”sorry”, ”apology”, ”pardon”, ”culprit”, etc.
- Third-party problems: Extract dependency names in issue title, label and comments. These names are obtained from the dependency information listed in the project.
- Documentation and repository standards: Extract related keywords including ”doc”, ”document”, ”documentation”, ” rename”, ”abstract”, ”rationale”, etc.
- Working progress problems: Extract related keywords including ”cannot do”, ”not have time”, ”progress”, ”assign”, ”tomorrow ”, ”soon”, ”delay”, ”busy”, ”rework”, etc.

We performed hierarchical sampling for each type and formed a training set (70%), a test set (20%) and a validation set (10%). The text preprocessing of the data set includes:

(1) Remove punctuation marks and stop words;
(2) Convert each word to lowercase, and perform stemming;
(3) Extract code blocks, code lines, quotes, links, pictures, and @-mention behaviors.

For the preprocessed text, TF-IDF is used for encoding, and different features are extracted and added to the feature vector. We used the linear support vector machine model for training and 10-fold cross-validation on the validation set. Finally, the results were evaluated on the test set. The rule-based method is implemented by extracting corresponding keywords in issue comments, titles and labels. A comment is classified as apology and work schedule issues if the comment contains keywords. For third-party and documentation and repository standards problems, we additionally considered keywords in the issue title and issue label because they are routine maintenance activities in software engineering. The event is classified if there are keywords in the issue comment or issue title and label.

The accuracy and AUC scores of each classifier are listed in Table IV. It shows that the model has achieved relatively reliable classification performance for each event type.

TABLE IV
NSE CLASSIFICATION RESULT

Event type	ACC	AUC	method
Disagreements on feature implementation	0.83	0.801	SVM
Confusion and help seeking	0.786	0.787	SVM
Bug report and fixing	0.726	0.686	SVM
Apology	0.906	0.928	Rule
Third-party problems	0.69	0.734	Rule
Documentation and repository standards	0.708	0.704	Rule
Working progress problems	0.778	0.655	Rule

Based on this classifier, we used the dialogue disentanglement technique introduced in Section V-A to study whether the completeness of snippets can improve the classification performance. Specifically, we extracted the reply-to relationship from all the comments in the issue where the negative sentiment segment was detected. Then, all the related comments are added to the original segment. Finally, the model is trained and tested on the new data set. The results are shown in Table V. It can be seen that the completeness of snippets does not improve the performance. For confusion and help seeking and disagreements on feature implementation, the performance is better when the reply-to relationship is not added. After observation, it is found that the newly included comments for confusion and help-seeking events mainly describe the cause and answer of the problem; the newly included comments for disagreements on feature implementation mainly propose the corresponding use cases or to state that this issue can be closed after discussion. The inclusion of these comments is helpful to understand the causes and consequences of negative sentiment events, but they do not contain event features and do not contribute to the performance of classification. For bug report and fixing, the classifier's performance is slightly improved after applying dialogue disentanglement. The newly added comments generally propose the scenarios of the bug or explain the fix process and provide the final solution. The meaning of comments is consistent, which helps to classify the event. For rule-based classifiers, the inclusion of new comments increases the hit rate of keywords, but the number of false positives increases, so the overall performance is not improved. **Therefore, the performance of NSE classification mainly depends on the contents and features of core comments**.

TABLE V
NSE CLASSIFICATION RESULT WITH CONVERSATION DISENTANGLEMENT

Event type	ACC	AUC
Disagreements on feature implementation	0.761	0.724
Confusion and help seeking	0.746	0.748
Bug report and fixing	0.747	0.704
Apology	0.861	0.899
Third-party problems	0.634	0.703
Documentation and repository standards	0.801	0.677
Working progress problems	0.717	0.61

C. A Tool to monitor negative sentiment-related events

We also implemented a tool for NSE monitoring, whose interfaces are shown in Figure 3.

Firstly, the evolution of the sentiments of each polarity is displayed for each week. Secondly, the number of NSEs are summarized according to each type for each week. Finally, the comments corresponding to each NSE are displayed.

VI. FACTORS INFLUENCING NEGATIVE SENTIMENT-RELATED EVENTS

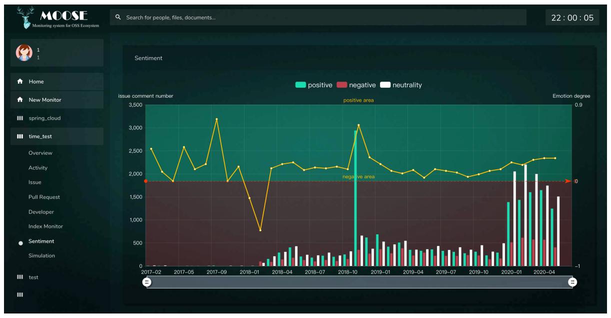
In this section, we answer RQ3, i.e., what factors have an influence on NSEs? We use association rule mining [30], which is a data mining technique to reveal the implicit information of different attributes in data, as a tool.

To study the factors that affect NSEs in OSS communities, we perform association rule mining on event types, project factors and developer factors during development with the Apriori algorithm [31] implemented by the Weka tool [32]. For numerical variables, we adopted the unsupervised instance filter in Weka and discretized continuous values into three types according to the frequency distribution to ensure that all variables are nominal variables.

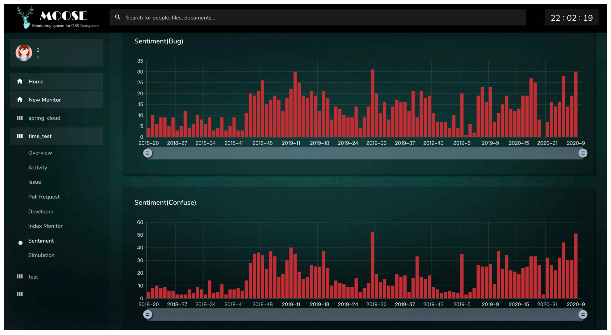
The relevance of association rules is mainly measured by three metrics: Support, Confidence and Lift.

- Support of rule X→Y is defined as the percentage of instances that satisfy both the conditions of the antecedent and consequent, i.e., Sup(X→Y)=P(X,Y).
- Confidence of rule X→Y represents the probability of occurrence of the consequent, given the occurrence of the antecedent, i.e., Conf(X→Y)=P(Y|X)=P(X,Y)/P(X).
- Lift of rule X→Y indicates how more frequently the conditions in Y occur given that the conditions in X occur. Higher Lift indicates a stronger correlation between X and Y, i.e., Lift(X→Y)=P(X,Y)/P(X)P(Y).

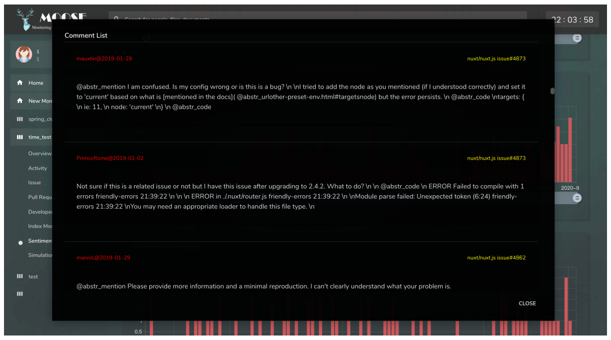
The minimum Support in the experiment is set as 0.1% and the extracted association rules are sorted by Lift in descending order.



(a) Sentiment evolution trends of each polarity



(b) The statistics of NSEs of an event type


Comment List

mauxtin@2019-01-29 nuxt/nuxt.js issue#4873

@abstr_mention I am confused. Is my config wrong or is this is a bug? \n \nI tried to add the node as you mentioned (if I understood correctly) and set it to 'current' based on what is [mentioned in the docs](@abstr_urlother-preset-env.html#targetsnode) but the error persists. \n @abstr_code \ntargets: { \n ie: 11, \n node: 'current' \n} \n @abstr_code

PrimozRome@2019-02-02 nuxt/nuxt.js issue#4873

Not sure if this is a related issue or not but I have this issue after upgrading to 2.4.2. What to do? \n \n @abstr_code \n ERROR Failed to compile with 1 errors friendly-errors 21:39:22 \n \n \n ERROR in ./.nuxt/router.js friendly-errors 21:39:22 \n \nModule parse failed: Unexpected token (6:24) friendly-errors 21:39:22 \nYou may need an appropriate loader to handle this file type. \n

manniL@2019-01-29 nuxt/nuxt.js issue#4862

@abstr_mention Please provide more information and a minimal reproduction. I can't clearly understand what your problem is.

CLOSE

(c) The comments corresponding to an NSE

Fig. 3. A tool for monitoring NSEs in OSS projects

A. The impact of project factors on NSEs

This section introduces the extracted rules related to the project factors, which include the number of Forks, Stars and developers. TABLE VI shows the extracted strong association rules.

- For projects with a large number of Forks and Stars, there are more NSEs for third-party problems, documentation and repository standard problems.
- For projects with a small number of developers, Forks and Stars, there are more NSEs involving working progress problems and disagreements on whether or how to implement corresponding features.
- Projects with a medium number of Forks and a small number of Stars have more NSEs regarding confusion and help seeking.

It can be concluded that problems about third-party, documentation and repository standards are more likely to occur in large-scale projects; problems such as confusion and help seeking, working progress problems, and debating whether or how to implement features are more likely to occur in small-to-medium projects. These conclusions suggest that project managers can monitor and prevent NSEs in a more targeted manner based on the project factors.

B. The impact of developer factors on negative sentiment-related events

Considering the variety of projects, we extract and analyze association rules related to developers in each single project. We study the impact of a specific developer, the collaboration between two developers and the developer's role in different event types.

1) The impact of a specific developer on negative sentiment-related events: It can be seen in TABLE VII that the participation of specific developers increases the probability of certain types of NSEs, and the degree of influence varies in each example. For example, rule one indicates that in the issue discussion of Typescript, with the participation of *rotemdan*, the possibility of disagreement events on feature implementation is increased by 133%. In addition, the confidence of each rule (X→Y) is significantly greater than the confidence of its reverse rule (Y→X). Therefore, it can be concluded that the participation of specific developers has an impact on the occurrence of NSEs in issue discussions.

2) The impact of different collaborators on negative sentiment-related events: We also investigated whether collaboration between developers affects negative sentiment events. Table VIII shows the extracted association rules. It can be found that the participation of specific collaborators in different projects can increase the probability of certain types of NSEs, indicating that collaboration between developers has an impact on the occurrence of different NSEs in issue discussions.

3) The impact of developers' roles on negative sentiment-related events: We extracted the developer role field (CommentAuthorAssociation) provided by GitHub API, which contains three types, i.e., *Contributor*, *Member* and *None*. *Contributor* represents the direct contributor of the project, that is, the developer who has commit history to the code. *Member* represents the team member of the organization to which the project belongs. *None* represents the external developer not associated with the project. As can be concluded from TABLE IX, the developer's role has an impact on NSEs in issue discussions. In **Typescript**, bug report and fixing, documentation and repository standards issues often occur with team members, while apology issues often occur with external developers. In **Pandas**, working progress issues occur with the contributors of the project, while disagreements on feature implementation occur with external developers. In **Go**, confusion and help seeking usually occur with project contributors and team members.

Based on these results, project managers can better manage and allocate developers, by the targeted monitoring of developers, collaborators and roles that are more likely to

TABLE VI
ASSOCIATION RULES RELATED TO PROJECTS AND NSES

X	Y	Conf(X→Y)	Conf(Y→X)	Lift(X→Y)
Forks>1914	Documentation and repository standards	0.08	0.38	1.13
386<Forks<1914	Confusion and help seeking	0.07	0.37	1.11
Stars<1391	Confusion and help seeking	0.07	0.37	1.11
Forks<386	Work progress	0.17	0.36	1.07
Stars<1391	Work progress	0.18	0.36	1.09
nDevelopers<39	Work progress	0.18	0.36	1.07
Forks>1914	Third-party	0.24	0.35	1.05
Stars>6758	Third-party	0.25	0.36	1.09
nDevelopers<39	Disagreements on feature implementation	0.2	0.36	1.07

TABLE VII
ASSOCIATION RULES RELATED TO DEVELOPERS AND NSES IN TYPESCRIPT, PANDAS AND GO

Project	X	Y	Conf(X→Y)	Conf(Y→X)	Lift(X→Y)
Typescript	rotemdan	Disagreement on feature implementation	0.36	0.08	2.33
Typescript	mhegazy	Confusion and help seeking	0.21	0.13	1.91
Pandas	jnmclarty	Bug report and fixing	1	0.02	12.91
Pandas	mdmueller	Work progress	0.5	0.02	7.73
Pandas	kornilova	Apologize	1	0.01	8.24
Go	hirchachacha	Bug report and fixing	1	0.02	13.16
Go	hyangah	Disagreement on feature implementation	1	0.01	4.96

TABLE VIII
ASSOCIATION RULES RELATED TO COLLABORATORS AND NSES IN PANDAS AND GO

Project	X	Y	Conf(X→Y)	Conf(Y→X)	Lift(X→Y)
Go	aclements & bradfitz	Confusion and help seeking	1	0.08	54.5
Pandas	hadim & jreback	Bug report and fixing	1	0.06	35.13
Pandas	gfyoung & jowens	Confusion and help seeking	1	0.17	29.58
Go	bcmills & worldiety	Bug report and fixing	1	0.03	1.24

induce NSEs to promote the harmony and efficiency of the OSS development.

VII. THREATS TO VALIDITY

In this preliminary study, we only studied a sample of projects. We selected some representative projects and took into account different programming languages to ensure the diversity of samples and the generalizability of our results.

We used SentiCR, a customized sentiment analysis tool for the SE domain for sentiment analysis and retrained it with a gold standard on GitHub issues. However, as discussed in prior work, we cannot expect 100% accuracy from sentiment analysis tools. Misclassifications may still exist and bring noise to our dataset.

Limited by time and effort, the dataset we labeled and constructed for NSEs is relatively limited in the number of samples. We integrated and re-organized some minor event types and imported external corpus to reduce this threat. However, the generalizability and performance of the developed classifiers may still be affected. In the future, we plan to expand our dataset to further improve our classifiers.

VIII. CONCLUSIONS

Monitoring sentiment factors is of great significance for project management during the OSS development process. We proposed a method to identify and monitor NSEs in OSS communities, and analyzed its relationship with the development process. First, we employed a Kleinberg model to extract burst periods of negative sentiments, summarized nine NSE types through case studies, and manually constructed the standard dataset. Second, we developed and evaluated our classifiers based on the coded dataset. We also applied conversation disentanglement techniques to improve the completeness of the snippets we extracted. Finally, we performed statistical analysis and association rule mining on a wider range of dataset. It is found that the type of NSEs is associated with the attributes of projects, developers and issues during the development process.

In the future, we plan to expand our dataset through semi-supervised learning and apply deep learning models to improve the performance of classifiers. In addition, we can further study methods to directly identify events that have a negative effect on the software development process to improve monitoring efficiency.

TABLE IX
ASSOCIATION RULES RELATED TO DEVELOPERS' ROLE AND NSES IN TYPESCRIPT,PANDAS AND GO

Project	X	Y	Conf(X→Y)	Conf(Y→X)	Lift(X→Y)
Typescript	Team member	Documentation and repository standards	0.1	0.25	1.25
Typescript	External developer	Apology	0.13	0.64	1.17
Typescript	Team member	Bug report and fixing	0.18	0.23	1.17
Pandas	Contributor	Working progress	0.61	0.1	1.18
Pandas	External developer	Disagreement on feature implementation	0.26	0.17	1.05
Go	Contributor	Confusion and help seeking	0.11	0.48	1.33
Go	Team member	Confusion and help seeking	0.1	0.28	1.24

ACKNOWLEDGEMENT

This work is partially supported by National Key Research and Development Plan(No.2018YFB1003800).

REFERENCES

[1] Emitza Guzman, David Azócar, and Yang Li. "Sentiment analysis of commit comments in GitHub: an empirical study". In: *Proceedings of the 11th Working Conference on Mining Software Repositories*. 2014, pp. 352–355.

[2] Michal R Wrobel. "Emotions in the software development process". In: *2013 6th International Conference on Human System Interactions (HSI)*. IEEE. 2013, pp. 518–523.

[3] David Garcia, Marcelo Serrano Zanetti, and Frank Schweitzer. "The role of emotions in contributors activity: A case study on the Gentoo community". In: *2013 International Conference on Cloud and Green Computing*. IEEE. 2013, pp. 410–417.

[4] Marco Ortu et al. "Are bullies more productive?: empirical study of affectiveness vs. issue fixing time". In: *Proceedings of the 12th Working Conference on Mining Software Repositories*. IEEE Press. 2015, pp. 303–313.

[5] Karim R Lakhani and Eric Von Hippel. "How open source software works:"free" user-to-user assistance". In: *Produktentwicklung mit virtuellen Communities*. Springer, 2004, pp. 303–339.

[6] Alessandro Murgia et al. "An exploratory qualitative and quantitative analysis of emotions in issue report comments of open source systems". In: *Empirical Software Engineering* 23.1 (2018), pp. 521–564.

[7] Bing Liu et al. "Sentiment analysis and subjectivity." In: *Handbook of natural language processing* 2.2010 (2010), pp. 627–666.

[8] Mike Thelwall et al. "Sentiment strength detection in short informal text". In: *Journal of the American Society for Information Science and Technology* 61.12 (2010), pp. 2544–2558.

[9] Richard Socher et al. "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of the 2013 Conference on Empirical mMthods in Natural Language Processing*. 2013, pp. 1631–1642.

[10] Edward Loper and Steven Bird. "NLTK: the natural language toolkit". In: *arXiv preprint cs/0205028* (2002).

[11] Robbert Jongeling et al. "On negative results when using sentiment analysis tools for software engineering research". In: *Empirical Software Engineering* 22.5 (2017), pp. 2543–2584.

[12] Md Rakibul Islam and Minhaz F Zibran. "Leveraging automated sentiment analysis in software engineering". In: *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE. 2017, pp. 203–214.

[13] Fabio Calefato et al. "Sentiment polarity detection for software development". In: *Empirical Software Engineering* 23.3 (2018), pp. 1352–1382.

[14] Jin Ding et al. "Entity-level sentiment analysis of issue comments". In: *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering*. ACM. 2018, pp. 7–13.

[15] Toufique Ahmed et al. "SentiCR: a customized sentiment analysis tool for code review interactions". In: *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press. 2017, pp. 106–111.

[16] Francisco Jurado and Pilar Rodriguez. "Sentiment Analysis in monitoring software development processes: An exploratory case study on GitHub's project issues". In: *Journal of Systems and Software* 104 (2015), pp. 82–89.

[17] Parastou Tourani, Yujuan Jiang, and Bram Adams. "Monitoring sentiment in open source mailing lists: exploratory study on the Apache ecosystem". In: *CASCON '14 Proceedings of 24th Annual International Conference on Computer Science and Software Engineering* (2014).

[18] Jon Kleinberg. "Bursty and hierarchical structure in streams". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2002. DOI: 10.1145/775060.775061.

[19] Thin Nguyen et al. "Event extraction using behaviors of sentiment signals and burst structure in social media". In: *Knowledge and information systems* 37.2 (2013), pp. 279–304.

[20] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation". In: *the Journal of Machine Learning Research* 3 (2003), pp. 993–1022.

[21] Dane Bertram et al. "Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams". In: *Proceedings of the 2010 ACM conference on Computer supported cooperative work*. 2010, pp. 291–300.

[22] Stephen P Robbins. *Managing organizational conflict: A nontraditional approach*. Englewood Cliffs: NJ, Prentice-Hall, 1974.

[23] Pamela J Hinds and Diane E Bailey. "Out of sight, out of sync: Understanding conflict in distributed teams". In: *Organization Science* 14.6 (2003), pp. 615–632.

[24] Karen A Jehn and Elizabeth A Mannix. "The dynamic nature of conflict: A longitudinal study of intragroup conflict and group performance". In: *Academy of Management Journal* 44.2 (2001), pp. 238–251.

[25] Anna Filippova and Hichang Cho. "Mudslinging and manners: Unpacking conflict in free and open source software". In: *CSCW 2015 - Proceedings of the 2015 ACM International Conference on Computer-Supported Cooperative Work and Social Computing*. 2015. ISBN: 9781450329224. DOI: 10.1145/2675133.2675254.

[26] Meiyappan Nagappan, Thomas Zimmermann, and Christian Bird. "Diversity in software engineering research". In: *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*. ACM. 2013, pp. 466–476.

[27] Marco Ortu et al. "Would you mind fixing this issue?" In: *International Conference on Agile Software Development*. Springer. 2015, pp. 129–140.

[28] Jonathan K. Kummerfeld et al. "A Large-Scale Corpus for Conversation Disentanglement". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy, July 2019, pp. 3846–3856. DOI: 10.18653/v1/P19-1374. URL: https://aclweb.org/anthology/papers/P/P19/P19-1374/.

[29] Sebastian Cross et al. "Classifying help seeking behaviour in online communities". In: *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*. 2017, pp. 419–423.

[30] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. "Mining association rules between sets of items in large databases". In: *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*. 1993, pp. 207–216.

[31] Rakesh Agrawal, Ramakrishnan Srikant, et al. "Fast algorithms for mining association rules". In: *Proc. 20th int. conf. very large data bases, VLDB*. Vol. 1215. Citeseer. 1994, pp. 487–499.

[32] Mark Hall et al. "The WEKA data mining software: an update". In: *ACM SIGKDD explorations newsletter* 11.1 (2009), pp. 10–18.