reading report: Analysis and Detection of Information Types of Open Source Software Issue Discussions

Analysis and Detection of Information Types of Open Source Software Issue Discussions

Deeksha Arya*, Wenting Wang*, Jin L.C. Guo*, Jinghui Cheng†
*School of Computer Science, McGill University, Montreal, Canada
†Department of Computer and Software Engineering, Polytechnique Montreal, Montreal, Canada
*{deeksha.arya, wenting.wang}@mail.mcgill.ca, jguo@cs.mcgill.ca †jinghui.cheng@polymtl.ca

1. Citation

```
@inproceedings{DBLP:conf/icse/AryaWGC19,
  author    = {Deeksha Arya and
               Wenting Wang and
               Jin L. C. Guo and
               Jinghui Cheng},
  editor    = {Joanne M. Atlee and
               Tevfik Bultan and
               Jon Whittle},
  title     = {Analysis and detection of information types of open source software
               issue discussions},
  booktitle = {Proceedings of the 41st International Conference on Software Engineering,
               {ICSE} 2019, Montreal, QC, Canada, May 25-31, 2019},
  pages     = {454--464},
  publisher = {{IEEE} / {ACM}},
  year      = {2019},
  url       = {https://doi.org/10.1109/ICSE.2019.00058},
  doi       = {10.1109/ICSE.2019.00058},
  timestamp = {Wed, 16 Oct 2019 14:14:49 +0200},
  biburl    = {https://dblp.org/rec/conf/icse/AryaWGC19.bib},
  bibsource = {dblp computer science bibliography, https://dblp.org}
}
```

2. Abstract

Most modern Issue Tracking Systems (ITSs) for open source software (OSS) projects allow users to add comments to issues. Over time, these comments accumulate into discussion threads embedded with rich information about the software project, which can potentially satisfy the diverse needs of OSS stakeholders. However, discovering and retrieving relevant information from the discussion threads is a challenging task, especially when the discussions are lengthy and the number of issues in ITSs are vast.

In this paper, we address this challenge by identifying the information types presented in OSS issue discussions. Through qualitative content analysis of 15 complex issue threads across three projects hosted on GitHub, we uncovered 16 information types and created a labeled corpus containing 4656 sentences. Our investigation of supervised, automated classification techniques indicated that, when prior knowledge about the issue is available, Random Forest can effectively detect most sentence types using conversational features such as the sentence length and its position. When classifying sentences from new issues, Logistic Regression can yield satisfactory performance using textual features for certain information types, while falling short on others. Our work represents a nontrivial first step towards tools and techniques for identifying and obtaining the rich information recorded in the ITSs to support various software engineering activities and to satisfy the diverse needs of OSS stakeholders.

3. Key Problem

The key problem of extracting relevant information from issue discussions is closely related with identifying and detecting their information types. In order to achieve this, we need to know:

(1) What are the main information types in issue discussions that can satisfy the needs of different stakeholders?

(2) What are the performance and trade-offs of using different automated classification methods to detect information types?

4. Inputs of the approach

The issue discussions of one system.

5. Outputs of the approach

The information type for each sentence in issue comments.

6. Key Contributions

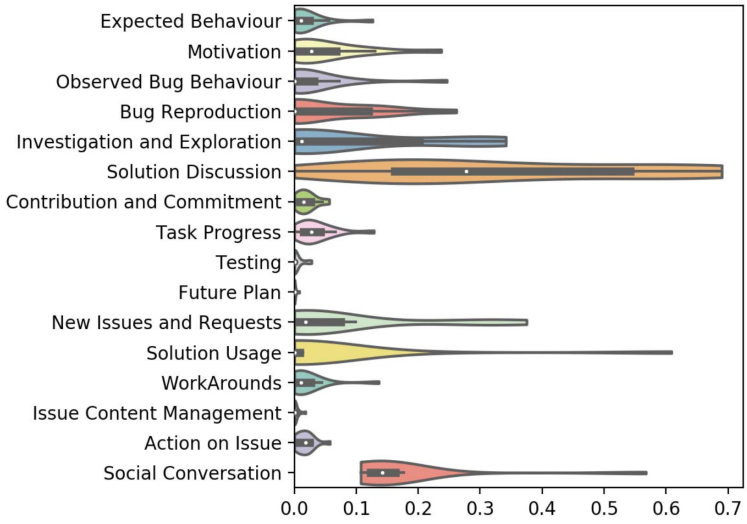
- Usefulness (to current practices):
  The approach outputs could help different participants to retrieve and discover useful information from the discussion threads with the support of the information types. Otherwise, the useful information may bury in the huge amount of discussion threads.
- Novelty/Impact:
  The paper takes the first step toward such goals, providing the suitable techniques to achieve such goals. It also identifies 16 kinds of main information types and provides a labeled corpus containing 4656 sentences, which may benefit other research in future.
- Elegance:
  The well-written paper answers the two target problems clearly with both theorical explanations and sufficient use cases. The whole approach pipeline does not need very complicated methods but utilizes the suitable techniques to obtain interesting research results.

7. Summary

This paper aims to address the above mentioned two target problems. Therefore we can briefly summarize the overall approach in respects of these two problems:

- To address first target problem

Authors manually conduct a four-step qualitative content analysis process on open-source software issues, and get total 16 information types. Here are the 16 information types and their distribution of the percentage of the identified codes:



[LYU, Siqi: In the figure of 16 information types, I wonder why social conversation starts from 0.1 instead of 0.0? And does the length represent #sentences of each type? What about the width? I notice the shape is a bit strange.]

[Me: Similar to violin plot, the figure shows the distribution of the percentage of the identified information types in the issue threads. In other words, the values in x-axis are percentages and the values in y-axis are information types. The distribution of the percentage is represented in the width. When the width is larger, the number of issue threads with such percentage is bigger.]

In the above information types, some have already been known as important issues in research area in the past, i.e. Observed Bug Behavior and Solution Discussion, for dealing with problems like bug triage and impact analysis. However, many other information types, such as Workarounds, Future Plan, Motivation, and Solution Usage are less studied but can be highly informative for stakeholders in various situations.

[LIU, Zhaosong: the 16 information types are collected manually by the authors. Does the paper mention any kind of metrics to ensure that these 16 types are comprehensive enough to cover all the discussions? How are these 16 types designed and determined?]

[ME: The authors utilize qualitative content analysis process to identify the 16 information types for each sentence in the issue threads. The details of this process are tedious, so I give a general explanation here. If you want to know more, you can reply here and I would give more details and materials for you.

A general explanation is that the authors are co-operating to label the data based on a codebook, which is generated during the cooperation the same time. The codebook is also available in https://github.com/deekshaarya4/Info_Types_in_OSS_Issue_Discussions/tree/master/data (Links to an external site.), which may help future research.

Authors utilize a metric to evaluate the performance of their co-operation, Cohen's kappa coefficient. The overall Kappa for this task is 0.71, representing a substantial agreement between the annotators.]

[LIU, Zhaosong: Thanks for the reply!]

- To address second target problem

Through addressing first problem, authors also label a corpus containing sufficient sentences, which is used as the dataset here.

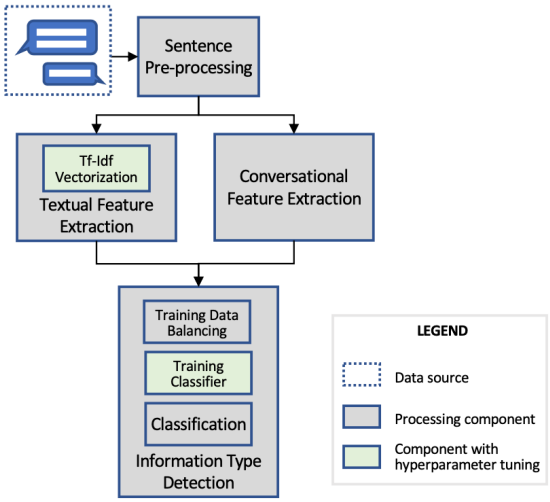
[LIU, Zhaosong: The paper discusses the automated classification method, and based on the Summary section, it seems that this automatic classification requires manually tagging the previous discussion threads, and can only be performed upon new threads on the same application. If that is the case, it seems to be a time-consuming task and not that convenient for actual production. Does the paper mention any possibility that a trained classification model of one application can be transferred to another application to do classification? What is your personal thought?]

[Me: I can't agree with you more. The biggest drawback of this approach is requiring heavy human workload. Since it is the first step to this task, researchers may still need time and effort to alleviate this problem.

The authors of this paper also mentioned that they can't claim general generalizability. In the future, they may further apply the approach to a larger number of OSS projects and issues in order to evaluate the external validity of our results. They also planed to examine the approach on analyzing pull request discussions.

How to obtain high-quality data is actually a general issue in almost AI techniques, especially supervised methods. Once we have high-quality data, like ImageNet, we can indeed achieve a lot. And since OSS issues are still in one certain domain, hopefully some researchers or engineers may come up with a 'ImageNet' in this area, which would solve a lot of problems haha. If you would like to share some of your ideas in this issue, I would be also happy to hear from you.]

The whole process is at the sentence-level. In other words, we train and make classification for each sentence. There are three main steps: sentence pre-processing, feature extraction, and information type detection. It is shown as the blew figure.



(1) Sentence pre-processing:

It is to make the issue comments cleaner by identification and replacement, i.e. Identify reference links to external resources and replace with 'URL' token.

(2) Feature extraction:

Here we have two kinds of features: textual feature and conversational feature. The former one is local feature, at the word-level, using TF-IDF features. The latter one is global feature, using issue features like the participant of this issue from {owner/collaborator/member/other}.

(3) Information Type Detection

Because the dataset is imbalanced, we can either do some sampling with SMOTE method or

adjusting class weight during training process. Then we can build and train the classifier, i.e. Logistic Regression or Random Forest, and make classification with the classifier.

[LIU, Xianghua: Here are 3 steps to classify sentences. In the Feature extraction step, there are two types: textual feature and conversational feature. I wonder what is the difference between local feature and global feature? And what are their main functions in this part?]

[Me: Here we consider the concept of "local" feature and "global" feature:

Textual Features are extracted from the textual content of each individual sentence, "locally". These features are kind of n-grams TF-IDF values, etc.

Conversational Features focus on the characteristics that describe the conversational context where each sentence situates during the issue discussion, "globally". For example, Structural Features describe the location of the sentence in relation to the whole discussion thread, etc.

The reason why we need to extract the features from each comment sentence is that we need the feature vectors and their corresponding labels to train a classifier. Then this classifier can classify the new coming feature vector, so we can automatically get the information types of new comment sentences.]

8. Evaluation

The paper addresses these two target problems so need two evaluations on them respectively:
(1) What are the main information types in issue discussions that can satisfy the needs of different stakeholders?
(2) What are the performance and trade-offs of using different automated classification methods to detect information types?

- The First target problem:

[LIU, Xianghua: In this paper, they only use three open-source software Issues (TensorFlow, scikit-learn, spaCy). What if other software issues? Can this method still perform well in other situation?]

[Me: This is one limitation of this approach. How to generalize and transfer to other domains is a hard and common challenge in AI field.

As discussed with LYU Siqi and LIU Zhaosong earlier, authors can't claim the generalizability. But since these kinds of issues are still in one certain domain, authors say they would try some future work to test the generalizability. They may further apply the approach to a larger number of OSS projects and issues in order to evaluate the external validity of our results.]

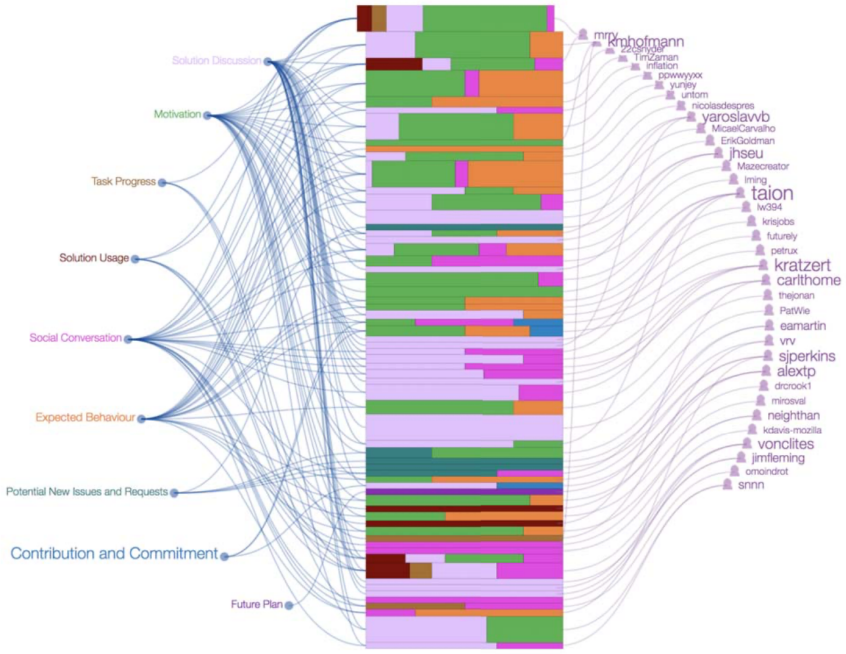
Used Data: Three open-source software Issues (TensorFlow, scikit-learn, spaCy)
Used Tool: ConVis visualization tool

As mentioned above, the 16 main information types can help open-source software participants retrieve useful information efficiently, with one visualization tool, ConVis, to provide visual cues that separate parts of each comment into information types. It can be shown as the below figure (a). The paper introduces a use case to illustrate the usefulness:

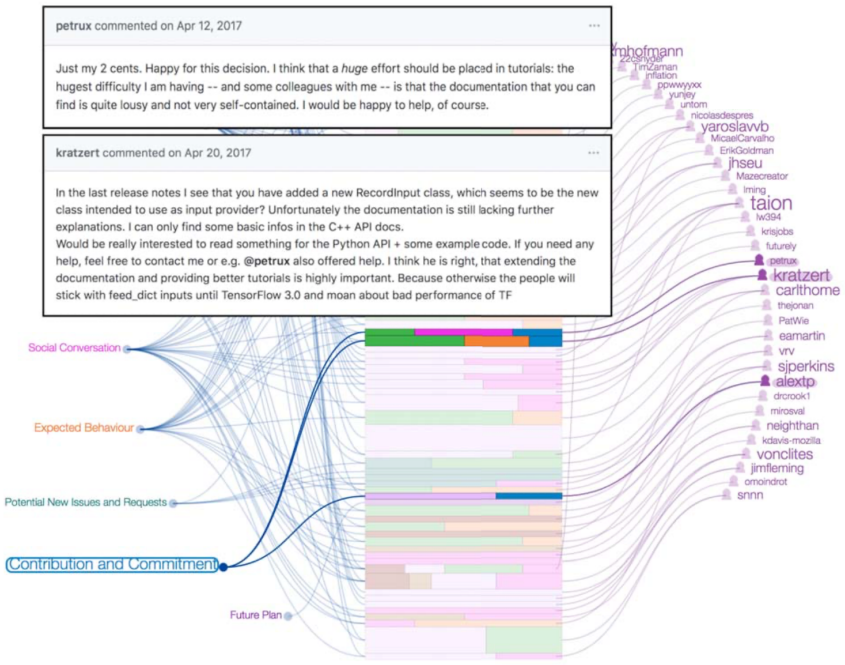
A TensorFlow user wants to resolve some difficulty importing and using data in her TensorFlow program.

With the visualization of issue types, she can quickly discover several comments that provided information on the **Motivation** of redesigning the input pipeline and some **Expected Behaviors**, by core project contributors and other users. She can even find some participants

who were eager to contribute by looking at the **Contribution and Commitment**. It is shown in the below figure (b).



(a) Full visualization



(b) Focused on *Contribution and Commitment*

[LIU, Guanzhou: Because these two diagrams are a little bit unclear, I wonder that whether or not they mean how much an information type contributes to a word root? But what the height of the rectangle means?]

[Me: Sorry that the resolution of these two large pictures is low.

In the center of the visualization, aka the rectangle, each horizontal bar represents a comment. Its height is proportional to the comment length.

The information types are color-coded and appear both on the comment bars (area representing length) and on the left side of the visualization. The right side indicates the participants who made the comments.]

- The Second target problem:

Used Data: The labeled corpus with 4330 annotated sentences.

Two series of experiments: Stratified 5-fold cross validation, and Leave-One-Issue-Out cross validation.

The authors conduct two series of experiments to consider two evaluating scenarios in reality:

• Scenario 1: The sentences in the discussion are partially categorized with information types and the users want to retrieve the missing labels in the same discussion.

• Scenario 2: Comments in a new discussion thread need to be categorized, given knowledge about other threads.

Stratified 5-fold cross validation is designed for Scenario 1, while Leave-One-Issue-Out cross validation is designed for Scenario 2.

Here are the experiment results: (SMOTE and Class Weight Adjustment are two different approaches to handle imbalanced dataset)

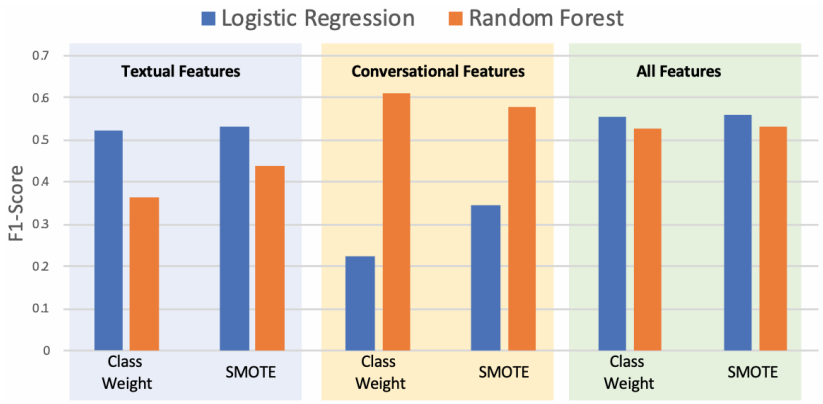


Fig. 5: Comparison of F1-scores in Scenario 1 (Stratified 5-fold cross validation)

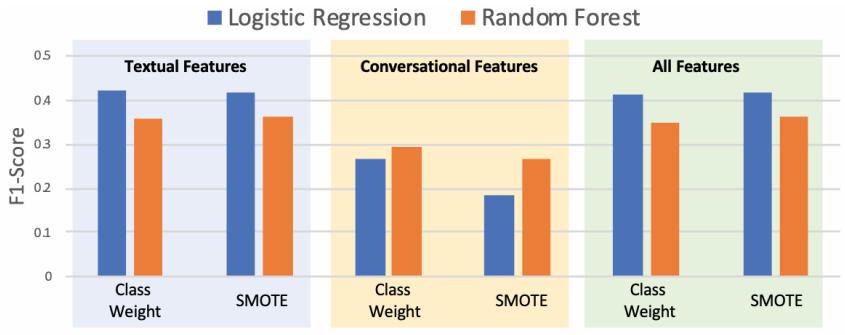


Fig. 6: Comparison of F1-scores in Scenario 2 (Leave-One-Issue-Out cross validation)

From the results, we can see that Random Forest using only conversational features with class weight adjustment performs best in Scenario 1. The reason may be Random Forest cannot effectively handle the sparsity of the textual features in our dataset.

However, in Scenario 2, both Logistic Regression and Random Forest models perform worst using conversational features. This finding illustrates that textual features are more reliable at detecting information types for sentences from new issues.

[LIU, Guanzhou: In your opinion, why the textual feature performs better than the conversational feature? Because you know, the global feature indeed contains more information such as the connection between different issues.]

[Me: Yes, I agree with you. That's why Conversational features outperform Textual features in scenario 1. But in scenario 2, what we are considering is:

Comments in a new discussion thread need to be categorized, given knowledge about other threads.

We use issue thread to indicate the complete discussion on the posted issue; issue comment to indicate a single post written by one participant of the issue, and sentence to indicate one sentence within a comment.

Therefore, the conversational information in other discussion threads may not be always useful to determine the information types of the new discussion thread, which is what we care about in scenario 2. That's why we think, from the evaluation in scenario 2, textual features are more reliable than conversational features in this case.]

9. Reflection

There are some similar works conducting in this area:

[1] A. J. Ko and P. K. Chilana. Design, discussion, and dissent in open bug reports. In Proceedings of the 2011 iConference on - iConference '11, pages 106–113, New York, New York, USA, 2011. ACM Press.

[2] I. Morales-Ramirez, F. M. Kifetew, and A. Perini. Speech-acts Based Analysis for Requirements Discovery from Online Discussions. Information Systems, Aug 2018.

Ko and Chilana [1] have conducted a qualitative analysis of bug reports to identify the topics involved in the discussion threads. But what they want is to understand the focus and the dynamics of those discussions. The target problems are quite different.

Itzel Morales-Ramirez [2] utilizes NLP and linguistic parsing techniques to identify requirements-relevant information in open-source software issue discussions. This target problem is quite similar to this paper we talked about, and it also achieves good performance with 0.81 F-measure. But what this paper proposes is more general than theirs, we not only focus the requirements-relevant info, but try to identify all interesting info to meet different needs.

[LYU, Siqi: The approach seems effective. Are there some limitations? Can this approach work for all kinds of projects?]

[Me: Though the authors didn't mention too many limitations, I think we have a great number of them since this work is the first step to such problem or task. Let's just consider the following:

Firstly, the training data, the labeled corpus, is manually annotated by authors. Due to the headvy workload of manual annotaion, the process is not efficient enough and we can't guarantee to generate a big corpus. The quality of the labeled data is also quite significant. So how can we get it in a more efficient way, how can we claim the good quality of the data and how to transfer the data into other domains? They're all limitations.

Secondly, since authors have done the first step to this kind of task, the performance is actually not good enough with best 0.6 F1-score or so. There is still some space to improve.
Thirdly, the detection of information types is to help diverse OSS participants more effectively utilize those tools. But do they actually improve their effectiveness? How to measure the help quantitatively? Therefore, the authors' future work is to further design such tools based on this, to understand the "help" to OSS participants from information types.]