Automatic Identification of Informative Code in Stack Overflow Posts

Preetha Chatterjee
preetha.chatterjee@drexel.edu
Drexel University
Philadelphia, PA, USA

ABSTRACT

Despite Stack Overflow's popularity as a resource for solving coding problems, identifying relevant information from an individual post remains a challenge. The overload of information in a post can make it difficult for developers to identify specific and targeted code fixes. In this paper, we aim to help users identify informative code segments, once they have narrowed down their search to a post relevant to their task. Specifically, we explore natural language-based approaches to extract problematic and suggested code pairs from a post. The goal of the study is to investigate the potential of designing a browser extension to draw the readers' attention to relevant code segments, and thus improve the experience of software engineers seeking help on Stack Overflow.

KEYWORDS

Stack Overflow, Mining Software Repositories, Natural Language Processing

ACM Reference Format:
Preetha Chatterjee. 2022. Automatic Identification of Informative Code in Stack Overflow Posts. In *Proceedings of the 1st International Workshop on Natural Language-based Software Engineering (NLBSE' 2022).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Throughout all phases of software development and maintenance, developers seek help for avoiding and fixing errors in their applications. A search for a specific error on a Q&A forum or Google search engine would return a list of links related to the query. To gain an insight into the problem, the developer has to click into and read a considerable number of posts to understand the cause of the error and make suitable bug fixes. Additionally, Q&A forums often contain long threads of discussions [18, 21]. Developers include entire classes or methods to provide context for better apprehension of the problems or suggested solutions. The length of posts could vary, but presence of noisy and redundant information is prevalent. In a previous work [5], we observed that a Stack Overflow post (including only the question and two answers) could contain up to 240 lines of code. We also found that software engineers focus on only 27% of code in a post to understand and determine how to apply the relevant information to their context. Identifying targeted code suggestions from such lengthy code segments is not trivial.

Researchers have studied the problem of finding relevant information from Q&A forums [4, 8, 10, 19]. Several techniques and tools have been developed to help information seekers on Stack Overflow, including reformulation of queries [9, 12], high-quality post detection [13, 20], and automatic recommendation of tags [2, 15], all targeted at retrieving better search results. Recently, Nadi and Treude identified essential sentences as cues for navigating Stack Overflow answers in a post [11]. However, to our knowledge, no one has investigated how to help information seekers focus their attention in identifying relevant targeted code changes, once they have narrowed down their search to a post relevant to their task.

In this paper, we aim to automatically identify informative code segments in a Stack Overflow post. Specifically, we focus on identifying the problematic and suggested code pairs from questions and answers in a post. We consider *problematic code* to be a code segment responsible for causing an error or exception, and a *suggested code* to be the targeted relevant code segment which should be used in replacement of the problematic code to solve the error. We explore natural language-based approaches to extract the code pairs. First, we use linguistic patterns to determine the location of suggested code in an answer. Next, we use text similarity heuristics to identify problematic code from the question. The goal of this study is to investigate the feasibility of designing a highlighting interface or a browser extension that could draw the readers' focus to informative code segments of a Q&A forum post, thus leading to a better and quicker understanding of the problem and choosing a solution.

To obtain early feedback on the potential of a highlighting interface to be beneficial for software developers, we conducted a pilot survey with 25 developers. Our survey consists of 2 parts: (1) We asked our participants, which parts of the code in a Stack Overflow post would they recommend highlighting to focus attention quickly? (2) We provided each participant with 4 sample Stack Overflow posts, where we manually highlighted the minimum amount of code that could be helpful for them to understand the problem and choose a potential solution. We asked them to evaluate the usefulness of the highlighted code in the annotated posts. In response to the first question, the participants suggested highlighting the specific lines of code in the question that is causing the error or exception (i.e., problematic code), and the targeted fix from the suggested code snippets in the answers (i.e., suggested code). In response to the second question, all participants agreed that the code highlights could help them understand the problem quickly and choose a potential solution from a Stack Overflow post.

NLBSE 2022, May 8, 2022, Pittsburgh, PA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

2 MOTIVATING EXAMPLES

For posts related to errors or exceptions on Stack Overflow, questioners include entire methods or classes because they are not sure where the problem lies, and to provide context for others to help them identify their problem [5]. In this study, we analyzed the answerers' behavior in providing solutions to those questions. We observed that some answerers provide targeted bug fixes, pinpointing where the problem lies. For example, in Figure 1a, both the problematic code and suggested code are provided in the answer. However, in other answers as shown in Figure 1b, one or more code snippets (each often containing multiple lines of code) were suggested, without identifying the specific lines of code that need to be changed. Absence of this information makes it difficult for readers to determine if the suggested code is relevant in their current code focus. Reading and understanding these posts also becomes arduous and time-consuming.

3 METHODOLOGY

We propose a set of natural language-based approaches to automatically extract problematic and suggested code pairs from Stack Overflow posts in two scenarios: (1) presence of problematic and suggested code in the same answer (e.g., Figure 1a), and (2) presence of problematic code in the question and suggested code in the answer (e.g., Figure 1b).

3.1 Data Selection

A set of 2000 Stack Overflow posts related to errors and exceptions were selected using Stack Exchange Data Explorer [6]. Our data set comes from a wide time range (August 2008 - March 2021). We retrieved posts that contain the tag 'Java' and/or 'C++', two commonly used programming languages in Stack Overflow [1]. Since the focus of this study is to automatically find bug fixes, we specifically selected posts where the title contains one or more of the terms "error", "exception", "bug", similar to the procedure in [5].

To ensure selection of good quality posts, we chose posts with a total vote count of 3 or higher. We chose posts in which the question and the answer, each contains at least one code segment. From each post, we selected the question and the accepted answer. If a post does not have an accepted answer, the best voted answer is selected. 1581/2000 (79%) posts in our data set contain an accepted answer.

3.2 Identification of Problematic and Suggested Code Pairs from Answer

When developers suggest short targeted fixes (including both the problematic and suggested code) in answers, they tend to use some recurrent textual patterns. These patterns consist of short phrases or words accompanied with code segments. For instance, in Figure 1a, the code suggestion (highlighted in blue) follows the pattern "change <problematic code> to <suggested code>".

We conducted a manual analysis of 2000 Stack Overflow answers in our data set to identify textual patterns that represent targeted code suggestions. We followed a qualitative content analysis procedure [14] to understand the structure of targeted code suggestions, and identified recurrent keywords towards potential textual patterns. Table 1 lists the 5 most common patterns observed in our data set for identifying the problematic and suggested code pairs in Stack Overflow answers.

Textual Patterns
change/modify <problematic code> to <suggested code>
replace <problematic code> with <suggested code>
instead of <problematic code> add/try/use <suggested code>
switch <problematic code> to <suggested code>
<suggested code> instead of <problematic code>

Table 1: Textual Patterns to Identify Problematic and Suggested Code Pairs from Answer

3.3 Identification of Problematic Code from Question and Suggested Code from Answer

When problematic and suggested code pairs are not located in the same answer, we decompose the problem of code-pair extraction into two sub-problems. First, we identify suggested code from answers using linguistic patterns. Next, we extract the problematic code from the question using textual heuristics.

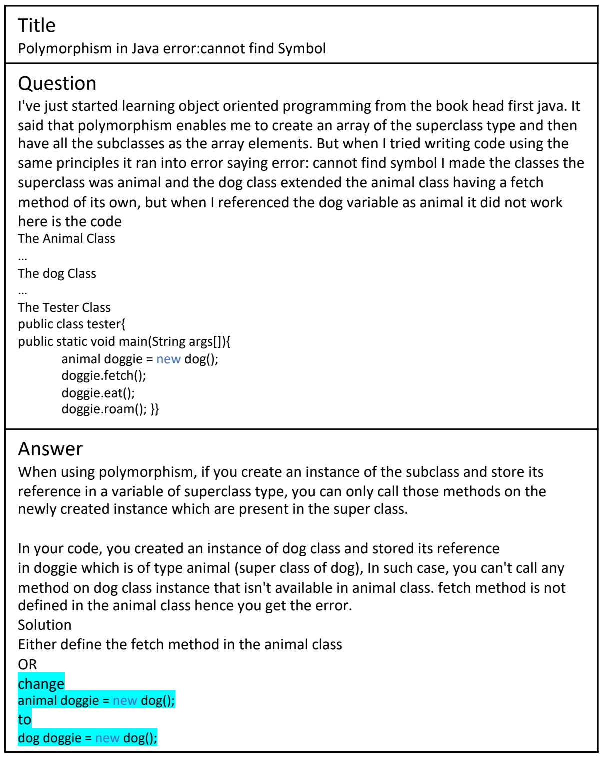
Identification of suggested code from answer: Di Sorbo et al. [16] observed that developers tend to use recurrent linguistic patterns while proposing solutions to known problems. Thus, using natural language parsing, they defined fifty linguistic patterns to identify 'solution proposal' sentences in developer emails. First, to investigate the generalizability of Di Sorbo et al.'s linguistic patterns, we used their replication package of DECA [2] to identify text adjacent to suggested code in Stack Overflow answers. We found that these patterns were able to identify some of the instances in our data set.

To improve on the existing technique [16], we aim to identify short imperative sentences that suggest a solution, i.e., sentences that start with a verb and followed by its object (e.g., "Perform an interactive rebase"). We perform chunking [1], to identify parts of speech and short phrases present in a given sentence, using NLTK toolkit [3]. We consider a sentence imperative if the root node is a verb (VB) or modal (MD), or if the first chunk is a verb phrase. The code located immediately before or after the 'solution proposal' sentence is identified as the suggested code in the answer.

Identification of problematic code from question: Researchers have compared buggy and fixed code snippets in Q&A forums to generate edit scripts for automated program repair [7, 17]. Based on the intuition that problematic and suggested code pairs in a post are similar, we identify the problematic code using textual similarity.
a) First, we extract a list of code elements from the previously identified suggested code in the answer. Specifically, we isolate each line of code in the suggested code segment on a newline character ("\n"), remove code comments and programming language specific keywords (e.g., 'abstract', 'this'), and then tokenize the code by splitting on white-space, special characters and camel case using regular expressions.

[1]https://insights.stackoverflow.com/survey/2021#technology

[2]https://www.ifi.uzh.ch/en/seal/people/panichella/tools/DECA.html



Title
Polymorphism in Java error:cannot find Symbol

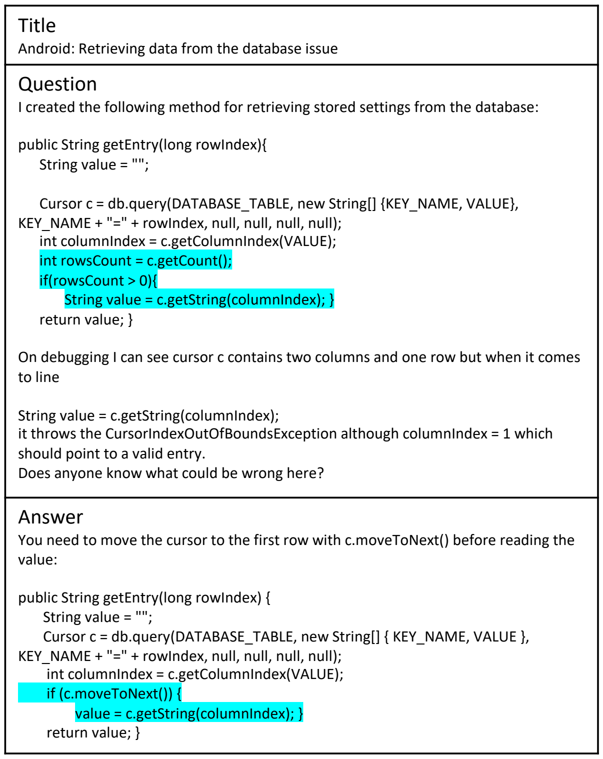
Question
I've just started learning object oriented programming from the book head first java. It said that polymorphism enables me to create an array of the superclass type and then have all the subclasses as the array elements. But when I tried writing code using the same principles it ran into error saying error: cannot find symbol I made the classes the superclass was animal and the dog class extended the animal class having a fetch method of its own, but when I referenced the dog variable as animal it did not work here is the code

```
The Animal Class
...
The dog Class
...
The Tester Class
public class tester{
public static void main(String args[]){
        animal doggie = new dog();
        doggie.fetch();
        doggie.eat();
        doggie.roam(); }}
```

Answer
When using polymorphism, if you create an instance of the subclass and store its reference in a variable of superclass type, you can only call those methods on the newly created instance which are present in the super class.

In your code, you created an instance of dog class and stored its reference in doggie which is of type animal (super class of dog), In such case, you can't call any method on dog class instance that isn't available in animal class. fetch method is not defined in the animal class hence you get the error.
Solution
Either define the fetch method in the animal class
OR
change
```
animal doggie = new dog();
```
to
```
dog doggie = new dog();
```

(a) Problematic and Suggested Code in Answer



Title
Android: Retrieving data from the database issue

Question
I created the following method for retrieving stored settings from the database:

```
public String getEntry(long rowIndex){
    String value = "";

    Cursor c = db.query(DATABASE_TABLE, new String[] {KEY_NAME, VALUE},
KEY_NAME + "=" + rowIndex, null, null, null, null);
    int columnIndex = c.getColumnIndex(VALUE);
    int rowsCount = c.getCount();
    if(rowsCount > 0){
        String value = c.getString(columnIndex); }
    return value; }
```

On debugging I can see cursor c contains two columns and one row but when it comes to line

```
String value = c.getString(columnIndex);
```
it throws the CursorIndexOutOfBoundsException although columnIndex = 1 which should point to a valid entry.
Does anyone know what could be wrong here?

Answer
You need to move the cursor to the first row with c.moveToNext() before reading the value:

```
public String getEntry(long rowIndex) {
    String value = "";
    Cursor c = db.query(DATABASE_TABLE, new String[] { KEY_NAME, VALUE },
KEY_NAME + "=" + rowIndex, null, null, null, null);
     int columnIndex = c.getColumnIndex(VALUE);
     if (c.moveToNext()) {
         value = c.getString(columnIndex); }
     return value; }
```

(b) Problematic Code in Question and Suggested Code in Answer

Figure 1: Examples of Error/Exception-related Posts on Stack Overflow (problematic and suggested code are highlighted)

b) Next, we determine the approximate location of problematic code in question. As before, we isolate each line of code in the question on a newline character, remove code comments and programming language specific keywords. If one or more code elements found in the suggested code co-occur in the question code, we extract that line of code in question as a potential problematic code. Since, the size of code in a question is often higher than the code in the answer, this step helps us in reducing the search space and getting an approximate location of the problematic code in the question.
c) Finally, we measure the textual similarity between each line of approximate problematic code in the question with each line of suggested code in answer. Towards generating edit scripts for bug fixes, Gao et al. [7] proposed a formula to measure code similarity. They calculate the edit distance between each line, and denote the length of problematic code as len_problematic, and the length of suggested code as len_fixed as follows:

$$Sim\,(Text) = 1 - \frac{edit\ distance}{\sqrt{len\_problematic \times len\_suggested}}$$

We found this similarity measure to be insufficient to identify problematic code in our data set. Hence, we are exploring a separate similarity measure. We count the occurrence of common words in each pair of line of code in question and answer, and normalize the measure using the following formula:

$$Sim\,(Text) = \frac{Number\ of\ common\ words}{\sqrt{len\_problematic \times len\_suggested}}$$

For each line in suggested code, we select the code in question with highest textual similarity as the problematic code.

4 EVALUATION: PRELIMINARY SURVEY

To obtain early feedback on the potential of our idea to be beneficial for software developers, we conducted a pilot survey to understand *"How effective is our approach in helping developers find a relevant solution to a bug-related issue from a Stack Overflow post?"*. We reached out to 25 developers from the software industry by emails. All 25 survey participants had prior programming experience of at least 4 years in Java and/or C++. Each participant was asked to indicate their Stack Overflow usage frequency as: (1) Never (never heard of it), (2) Seldom (at least once in six months), (3) Periodically (at least once in a month), or (4) Frequently (at least once a week). The responses indicate that 18 out of 25 (72%) use Stack Overflow frequently, 5 out of 25 (20%) use it periodically, and 2 out of 25 (8%) use it seldom.

For the survey, we manually annotated a set of 100 posts selected randomly from our data set described in Section 3.1. In each post, we highlighted specific lines of code that is causing an error or exception (problematic code), and the lines of code proposed to fix the issue (suggested code). Figure 1b shows an example of an annotated post. We provided 4 annotated posts to each of our participants, which took them approximately 20 minutes to evaluate. To gather the participant responses, we designed an online survey form which consisted of the following questions:

(1) Once you have identified a Stack Overflow post relevant to your question related to an error or exception, which part of the code in the question and/or answer would you recommend highlighting to focus attention quickly?
(2) For each of the following annotated posts (provided as pdf), we have highlighted the minimum amount of code that could be helpful for you to understand the problem and choose a potential solution. Do you find the highlighted code useful? Please select a response from the following options: (1) Not useful, (2) Somewhat useful, (3) Useful, (4) Very useful.

Observations: Responses to question (1) in the survey suggest highlighting the specific lines of code in the question that is causing the error or exception, and the targeted fix from the suggested code snippets in the answers. Below are few excerpts from the responses:

- *"In the question, it can help if the section of the* ***code that is causing the problem*** *is highlighted. In the answer, highlighting the exact lines of* ***code that should solve the problem****, can help focus attention. Highlighting specific error messages could also be useful."*
- *"If a huge code snippet is given, it is helpful to highlight the* ***section of the code that throws the exception****... Sometimes the answer attached to the post rewrite the whole code snippet with only one or two lines changing, highlighting only the* ***code snippets that has changed*** *in the answer is very helpful."*
- *"It is sometimes hard to identify the exact line that is incorrect. If there is no accepted answer, it is far more challenging to identify an appropriate solution. I recommend highlighting anything in the question that explains:* ***this is what is going wrong****. Then, in the answer, this is the* ***change you need to make*** *and why."*

In response to question (2), all participants agreed that the code highlights could help them understand the problem and choose a potential solution. Specifically, 14 out of 25 (56%) participants indicated 'Very useful', 8 out of 25 (32%) indicated 'Useful', and the remaining 3 out of 25 (12%) indicated 'Somewhat useful'. We plan to conduct a short interview with each participant to gain more insights and incorporate their feedback in designing our approach.

5 CONCLUSION AND FUTURE WORK

This paper presents a natural language-based approach to extract informative code segments, i.e., problematic and suggested code pairs from an individual Stack Overflow post. First, we use linguistic patterns to determine the location of suggested code in an answer. Next, we use text similarity heuristics to identify problematic code from the question. The goal of this research is to investigate the feasibility of designing a highlighting tool or a browser extension that can draw the reader's attention to the relevant parts of a post, and pinpoint targeted code fixes. Our preliminary pilot survey suggests that designing such a highlighting interface would be useful for developers in understanding the problem and choosing a solution that is relevant to their tasks.

REFERENCES

[1] Steven P. Abney. 1992. *Parsing By Chunks.* Springer Netherlands, Dordrecht, 257–278. https://doi.org/10.1007/978-94-011-3474-3_10

[2] S. Beyer and M. Pinzger. 2015. Synonym Suggestion for Tags on Stack Overflow. In *2015 IEEE 23rd International Conference on Program Comprehension.* 94–103. https://doi.org/10.1109/ICPC.2015.18

[3] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

[4] P. Chatterjee, K. Damevski, L. Pollock, V. Augustine, and N.A. Kraft. 2019. Exploratory Study of Slack Q&A Chats as a Mining Source for Software Engineering Tools. In *Proceedings of the 16th International Conference on Mining Software Repositories (MSR'19)* (Montreal, Canada). https://doi.org/10.1109/MSR.2019.00075

[5] Preetha Chatterjee, Minji Kong, and Lori Pollock. 2020. Finding Help with Programming Errors: An Exploratory Study of Novice Software Engineers' Focus in Stack Overflow Posts. *Journal of Systems and Software* 159 (2020), 110454. https://doi.org/10.1016/j.jss.2019.110454

[6] Stack Exchange Data Explorer. 2021. https://data.stackexchange.com//.

[7] Q. Gao, H. Zhang, J. Wang, Y. Xiong, L. Zhang, and H. Mei. 2015. Fixing Recurring Crash Bugs via Analyzing Q&A Sites (T). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE).* 307–318. https://doi.org/10.1109/ASE.2015.81

[8] Swapna Gottipati, David Lo, and Jing Jiang. 2011. Finding Relevant Answers in Software Forums. In *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE '11).* IEEE Computer Society, Washington, DC, USA, 323–332. https://doi.org/10.1109/ASE.2011.6100069

[9] Zhixing Li, Tao Wang, Yang Zhang, Yun Zhan, and Gang Yin. 2016. Query Reformulation by Leveraging Crowd Wisdom for Scenario-based Software Search. In *Proceedings of the 8th Asia-Pacific Symposium on Internetware* (Beijing, China) *(Internetware '16).* ACM, New York, NY, USA, 36–44. https://doi.org/10.1145/2993717.2993723

[10] Alessandro Murgia, Daan Janssens, Serge Demeyer, and Bogdan Vasilescu. 2016. Among the Machines: Human-Bot Interaction on Social Q&#38;A Websites. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI EA '16).* ACM, New York, NY, USA, 1272–1279. https://doi.org/10.1145/2851581.2892311

[11] Sarah Nadi and Christoph Treude. 2020. Essential Sentences for Navigating Stack Overflow Answers. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER).* 229–239. https://doi.org/10.1109/SANER48275.2020.9054828

[12] L. Nie, H. Jiang, Z. Ren, Z. Sun, and X. Li. 2016. Query Expansion Based on Crowd Knowledge for Code Search. *IEEE Transactions on Services Computing* 9, 5 (Sept 2016), 771–783. https://doi.org/10.1109/TSC.2016.2560165

[13] L. Ponzanelli, A. Mocci, A. Bacchelli, M. Lanza, and D. Fullerton. 2014. Improving Low Quality Stack Overflow Post Detection. In *2014 IEEE International Conference on Software Maintenance and Evolution.* 541–544.

[14] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. 2012. *Case Study Research in Software Engineering: Guidelines and Examples* (1st ed.). Wiley Publishing.

[15] A. K. Saha, R. K. Saha, and K. A. Schneider. 2013. A Discriminative Model Approach for Suggesting Tags Automatically for Stack Overflow Questions. In *2013 10th Working Conference on Mining Software Repositories (MSR).* 73–76. https://doi.org/10.1109/MSR.2013.6624009

[16] Andrea Di Sorbo, Sebastiano Panichella, Corrado A. Visaggio, Massimiliano Di Penta, Gerardo Canfora, and Harald C. Gall. 2015. Development Emails Content Analyzer: Intention Mining in Developer Discussions (T). In *Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE) (ASE '15).* IEEE Computer Society, Washington, DC, USA, 12–23. https://doi.org/10.1109/ASE.2015.12

[17] Jia Tong, Li Ying, Tang Hongyan, and Wu Zhonghai. 2016. Can We Use Programmer's Knowledge? Fixing Parameter Configuration Errors in Hadoop through Analyzing Q amp;A Sites. In *2016 IEEE International Congress on Big Data (BigData Congress).* 478–484. https://doi.org/10.1109/BigDataCongress.2016.73

[18] B. Xu, Z. Xing, X. Xia, and D. Lo. 2017. AnswerBot: Automated generation of answer summary to developers' technical questions. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE).* 706–716.

[19] Bowen Xu, Zhenchang Xing, Xin Xia, and David Lo. 2017. AnswerBot: Automated Generation of Answer Summary to Developersź Technical Questions. In *Proceedings of the 32Nd IEEE/ACM International Conference on Automated Software Engineering* (Urbana-Champaign, IL, USA) *(ASE 2017).* IEEE Press, Piscataway, NJ, USA, 706–716. http://dl.acm.org/citation.cfm?id=3155562.3155650

[20] Yuan Yao, Hanghang Tong, Tao Xie, Leman Akoglu, Feng Xu, and Jian Lu. 2015. Detecting High-quality Posts in Community Question Answering Sites. *Inf. Sci.* 302, C (May 2015), 70–82. https://doi.org/10.1016/j.ins.2014.12.038

[21] Y. Zhang and D. Hou. 2013. Extracting problematic API features from forum discussions. In *2013 21st International Conference on Program Comprehension (ICPC).* 142–151. https://doi.org/10.1109/ICPC.2013.6613842