An Annotated Dataset of Stack Overflow Post Edits

Sebastian Baltes
sebastian.baltes@adelaide.edu.au
@s_baltes

Markus Wagner
markus.wagner@adelaide.edu.au
@MWagnerRedChair
Automated program repair

- Problem-agnostic code mutations: copy, delete, move, ... of lines/statements
- Patches mined from software repositories
Automated program repair

- Problem-agnostic code mutations: copy, delete, move, ... of lines/statements
- Patches mined from software repositories

Genetic Improvement of Software

- Problem-agnostic code mutations: copy, delete, move, ... of lines/statements
- Patched mined from software repositories: no yet?
  
  Justyna Petke (2017) proposed “to mine changes [...] with particular focus on improvement of the software property of interest, such as runtime efficiency. The results can then be sued to devise new mutation operators in the form of templates.”
You have done a lot of redundant work to come here. You can use an `ExecutorService` with a `FixedThreadPool` and submit tasks to the thread pool, instead of hard coding 20 threads.

Also, how was the value of 20 for the number of threads decided? Use

```
Runtime.getRuntime().availableProcessors();
```

You may achieve better results by experimenting with lower number of threads, close to determine the number of cores on your machine/core count in the runtime.

```java
public static void main(String[] args) throws ClassNotFoundException, SQLException, IOException {
    int size = csvData.size();
    int threadCount = 20; //Runtime.getRuntime().availableProcessors();
    ExecutorService executorService = Executors.newFixedThreadPool(threadCount);

    int index = 0;
    int chunkSize = size / threadCount;
    while (index < size) {
        final int start = index;
        executorService.submit(new Runnable() {
            @Override
            public void run() {
                try {
                    ProcessRecords(csvData.subList(start, start + chunkSize));
                } catch (ClassNotFoundException | SQLException | IOException e) {
                    e.printStackTrace();
                }
            }
        });
        index += chunkSize;
    }
    executorService.shutdown();
}
```
Our contribution: a dataset based on Stack Overflow post edits

SO edits are possibly more fine-grained than GitHub commits:
SO post edits are less formal (SO is forum-like), while GH commits are expected to fix a bug or to extend functionality

Research Questions

RQ1: Which aspects do Stack Overflow users mention in their edit comments?

RQ2: Which non-functional properties do users reference in edit comments?
You have done a lot of redundant work to come here. You can use an `ExecutorService` with a `FixedThreadPool` and submit tasks to the thread pool, instead of hard coding 20 threads.

Also, how was the value of 20 for the number of threads decided? Use:

```java
Runtime.getRuntime().availableProcessors();
```

You may achieve better results by experimenting with lower number of threads, close to determine the number of cores on your machine/core count in the runtime.

```java
public static void main(String[] args) throws ClassCastException, SQLException, 
int size = executorService.size();
int threadCount = 20;Runtime.getRuntime().availableProcessors();
ExecutorService executorService = Executors.newFixedThreadPool(threadCount);
int index = 0;
int chunkSize = size / threadCount;
while (index < size) {
        final int start = index;
        executorService.submit(new Runnable() {
            @Override
            public void run() {
                try {
                        ProcessRecords(csvData.subList(start, chunkSize));
                } catch (ClassCastException | SQLException | IOException e) {
                        e.printStackTrace();
                } finally {
                        index += chunkSize;
                }
                executorService.shutdown();
            }
        });
    }
```
Edits on Stack Overflow

- Stack Overflow provides quarterly data dumps, the SOTorrent project extracts information about the edits from those dumps

- SOTorrent version 2020-01-24 contains 7,459,778 post edits where the user provided an (optional) description of the edit:
  - 1,305,323 (17.5%) modified only a code block
  - 4,792,777 (64.2%) only a text block
  - 1,361,678 (18.3%) both text and code blocks
Annotating Edits

- We normalised the edit messages (lower case, normalised whitespace characters)
- Yielding 3,291,268 unique (normalised) edit messages
- Ranked messages according to frequency
- Starting with the most frequent messages, we manually extracted characteristic keywords to build regular expressions matching similar messages
- Stopped the manual analysis as soon as we were able to cluster all messages with at least 1,000 occurrences.
- Example: `Deleting <- grepl(".*\\b((remov|delet|trim)[a-z0-9_-]*).*", edit_comments$Comment, perl=TRUE)`
Annotation Results

• We were able to assign edit messages to 25 categories using customised regular expressions
• One edit can have multiple categories
• We were able at assign 6,704,541 of the 7,459,778 edits (89.9%) to at least one category

• **User actions:** adding, updating, deleting, fixing, improving, clarifying, simplifying, explaining, editing, copy-editing, active reading, refactoring
• **Targets of the edit:** formatting, typo, grammar, spelling, code, bug, link, image, example, syntax, solution, tag
• **Meta:** sarcasm
RQ1: Aspects mentioned in edit messages

- Formatting
- Adding
- Fixing
- Code
- Improving
- Grammar
- Editing
- Deletion
- Updating
- Spelling
- Copy Editing
- Link
- Typo
- Clarifying
- Example
- Explaining
- Image
- Solution
- Syntax
- Bug
- Simplifying
- Active Reading
- Tag
- Refactoring
- Sarcasm

n=6,704,541
RQ1: Aspects mentioned in code edit messages

n=933,340
RQ1: Co-occurrence of categories for code edits

<table>
<thead>
<tr>
<th>pair</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>formatting, code</td>
<td>152,721</td>
</tr>
<tr>
<td>improving, formatting</td>
<td>98,339</td>
</tr>
<tr>
<td>fixing, code</td>
<td>76,026</td>
</tr>
<tr>
<td>fixing, formatting</td>
<td>65,544</td>
</tr>
<tr>
<td>adding, code</td>
<td>50,795</td>
</tr>
<tr>
<td>fixing, typo</td>
<td>32,711</td>
</tr>
<tr>
<td>improving, code</td>
<td>31,463</td>
</tr>
<tr>
<td>editing, code</td>
<td>28,844</td>
</tr>
<tr>
<td>updating, code</td>
<td>24,910</td>
</tr>
<tr>
<td>deleting, code</td>
<td>20,106</td>
</tr>
</tbody>
</table>

Table 1: Top 10 pairs of tags (pairs ordered only for presentation purposes).
RQ2: Non-functional properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>2,658</td>
</tr>
<tr>
<td>Size</td>
<td>2,284</td>
</tr>
<tr>
<td>Memory</td>
<td>1,084</td>
</tr>
<tr>
<td>Energy</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 2: Number of code edits where the user mentioned one of the four non-functional properties we have considered (n=7,024).
Examples


   ```java
   private string getctl(Control master)
   {
       StringBuilder sb = new StringBuilder();
       foreach (Control child in master.Controls)
       {
           sb.AppendFormat("[ {0} - {1} ]", child.ClientID.ToString(), child.GetType().ToString());
           if (child.HasControls())
           {
               sb.AppendFormat(getctl(child));
           }
       }
       return sb.ToString();
   }
   
   string controls = getctl(this.Page);
   ```
Examples

(1) “using john saunders tip for more performance” (https://stackoverflow.com/a/23481309): the edit replaced a String with a StringBuilder
Examples found within 15 minutes (1/2)

(1) “using john saunders tip for more performance” (https://stackoverflow.com/a/23481309):
the edit replaced a String with a StringBuilder.

(2) “added debounce to improve performance when app scales” (https://stackoverflow.com/a/44000037):
the edit added a JavaScript debounce function.

(3) “evaluating x 0 first solves for type errors and gives better performance than if” (https://stackoverflow.com/a/19400435):
the edit updated an if-statement – interestingly, there is a brief discussion on the performance attached to this post.
Examples found within 15 minutes (2/2)

(4) “some small **performance** improvements always a good idea to have a fast primality test” (https://stackoverflow.com/a/8539774): the edit added a few **hard-coded** scenarios for a particular problem.

(5) “Improved **performance**, by getting [...] outside the loop” (https://stackoverflow.com/a/11535593): the edit **lifted code** outside of a loop, which is an approach that is commonly taught in undergraduate courses.
Summary / Outlook

Our Stack Overflow post edits vs. GitHub commits: our edits are likely to be more fine-grained → potential to reveal insights on SE in practice at a higher resolution

Millions of SO edits might be a treasure trove for fine-grained code patches

Move from code edits to text edits: suggest typical grammar fixes or frequent formatting improvements

Call for participation:

- How can we improve the dataset?
- What support can we provide?
Our dataset

Available online:

- Zenodo:
  https://doi.org/10.5281/zenodo.3754159

- Google BigQuery:

- Live Demo:
  https://www.youtube.com/watch?v=2GqMONIAX2U