On the Relationship between the Vocabulary of Bug Reports and Source Code

Laura Moreno
Wathsala Bandara
Sonia Haiduc
Andrian Marcus
Bug Location

• Determine where to fix a bug

• Important task during software evolution

• Text Retrieval (TR) based approaches have been proposed to automate this task
It would be nice if...
When the ssh client initializes a connection to the server, it already gets the server's revision. It can throw an exception and tell the user that the connection method can not operate with SSH2.
TR-based Bug Location: Assumption

It would be nice if when the ssh client initializes a connection to the server, it already gets the server's revision. It can then throw an exception and tell the user that the connection method can not operate with SSH2.
Research Questions

RQ1: Is the vocabulary of bug reports reflected in the code?

RQ2: What is the code location of the shared terms between bug reports and patched classes?

RQ3: Is the number of shared terms between bug reports and classes an adequate measure for bug location?
Methodology

- Data previously used in TR-based bug location (corpus)

<table>
<thead>
<tr>
<th>System</th>
<th># of classes</th>
<th># of bug reports</th>
<th># of patched classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADempiere 3.10</td>
<td>1896</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Art of Illusion 2.4.1</td>
<td>570</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>aTunes 1.10</td>
<td>439</td>
<td>17</td>
<td>22</td>
</tr>
<tr>
<td>Eclipse 2.0</td>
<td>7689</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>Eclipse 3.0</td>
<td>22980</td>
<td>40</td>
<td>74</td>
</tr>
<tr>
<td>JEdit</td>
<td>801</td>
<td>18</td>
<td>27</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>34375</strong></td>
<td><strong>114</strong></td>
<td><strong>166</strong></td>
</tr>
</tbody>
</table>
Methodology

• Data previously used in TR-based bug location (corpus)

• Documents as bags of words
  – Bug reports (title + description) and classes (identifiers + comments + literals)
  – Identifier splitting, stop word removal, stemming

• Different measures on pairs of documents
  $<\text{bug report, class}>$
RQ1: Is the vocabulary of bug reports reflected in the code?

- **Measure:** common terms between bug reports and classes
- **Object:** full set, i.e., 1,077,074 pairs <bug report, class>

Bug reports share vocabulary with a large number of classes in a software system.
Is the vocabulary of bug reports reflected in the code?

- **Measure:** common terms between bug reports and classes
- **Object:** patched subset, i.e., 166 pairs <bug report, corresponding patched class>, and non-patched subset

![Box plot showing number of unique shared terms for different subsets](image)

Bug reports have more terms in common with the patched classes than with the non-patched classes.
What is the code location of the shared terms?

- **Measure:** percentage of terms and contribution to shared terms by code location
- **Object:** patched subset, i.e., 166 pairs <bug report, corresponding patched class>

The more verbose a code location is, the more it contributes to the common vocabulary between a patched class and its respective bug report.
RQ2

What is the code location of the shared terms?

- **Measure:** percentage of shared terms by code location
- **Object:** patched subset, i.e., 166 pairs <bug report, corresponding patched class>

The names of the patched classes are more likely to share terms with the respective bug reports than other locations.
RQ3

Is the number of shared terms an adequate measure for bug location?

- **Measure:** effectiveness of LSI and Shared Terms
- **Object:** full set using bug reports as queries

The number of shared terms between bug reports and classes supports bug location better than LSI.
RQ3: Is the number of shared terms an adequate measure for bug location?

- **Measure:** effectiveness of Lucene and Shared Terms
- **Object:** full set using bug reports as queries

The number of shared terms between bug reports and classes supports bug location better than LSI, yet not as well as Lucene.
In Summary

• Is the vocabulary of bug reports reflected in the code?
  – Bug reports share terms with a large number of classes in a software system
  – The number of shared terms is higher for patched classes than for the rest of classes

• What is the code location of the shared terms between bug reports and patched classes?
  – Class names (most likely)

• Is the number of shared terms between bug reports and classes an adequate measure for bug location?
  – Well, it works better than complex TR techniques as LSI but not as well as Lucene
The question...

How can TR-based concept location techniques benefit from these findings?