Learning to Rank Relevant Files for Bug Reports using Domain Knowledge

Xin Ye, Razvan Bunescu, and Chang Liu
Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, 2014.

Presented by: Juan Florez

Bug reports are not perfect

There is an inherent mismatch between the language in bug reports and source code.

Textual similarity between bug report and related code might even be zero, except maybe if code is comprehensively commented.

Information missing from bug report.

Bug ID: 339286
Summary: Toolbars missing icons and show wrong menus.
Description: The toolbars for my stacked views were: missing icons, showing the wrong drop-down menus (from others in the stack), showing multiple drop-down menus, missing the min/max buttons ...

Figure 1: Eclipse bug report 339286.

```
public class PartRenderingEngine
                    implements IPresentationEngine {
private EventHandler trimHandler = new Even-
tHandler() {
  public void handleEvent(Event event) { ...
    MTrimmedWindow window =
                    (MTrimmedWindow) changedObj;
    ... } ... } ... }
```

Figure 2: Code from PartRenderingEngine.java

What is in bug reports?

Most useful to developers

- Steps to reproduce
- Stack traces
- Test cases
- Observed behavior
- Screenshots

Provided by reporters

- Steps to reproduce
- Observed behavior
- Expected behavior
- Product
- Version

The Idea

Use VSM.

Enrich source code with domain knowledge, in the form of API documentation.

Use other features of source code that have been shown to lead to the discovery of bugs.

Rank of document is weighted sum of features.

Feature: Surface Lexical Similarity

Compute standard cosine similarity between bug report and source file.

However, also compute similarity between bug report and individual methods and choose the highest value of the whole set.

Feature: API-Enriched Lexical Similarity

For each source code file, collect a list of classes and interfaces from all local variable declarations.

Collect textual descriptions of these types using the API specification and also of any direct or indirect supertype.

Same as before, also split by method and choose the maximum value.

Feature: Class Name Similarity

A bug report sometimes mentions a specific class name.

Authors propose to make feature value proportional to the length of the class name.

Other Features

- **Collaborative Filtering Score:** if file was fixed before report was filed, it is likely to need fixing again.
- **Bug-Fixing Recency:** inverse of the difference in months between last fix time and present for file.
- **Bug-Fixing Frequency:** amount of times source file was fixed before the bug report was filed.

The Process

1. Title and description / Source code and comments
2. Remove punctuation and stop words
3. Split identifiers
4. Stem
5. Calculate features
6. Train SVM

SVM Training

For a bug report:

- Choose only top 300 irrelevant documents ranked by cosine similarity
- Split training data chronologically in 10 sets and train in k +1 and test in k
- Determine the parameter that maximizes Mean Average Precision

Discussion

- Lexical gap is a major problem when relating code to the real world
- Features like fix dates and frequency have been shown to simplify localization of bugs. The results shown of this paper back up this assumption.
- Performs better than BugLocator, BugScout, Usual Suspects, and Standard VSM

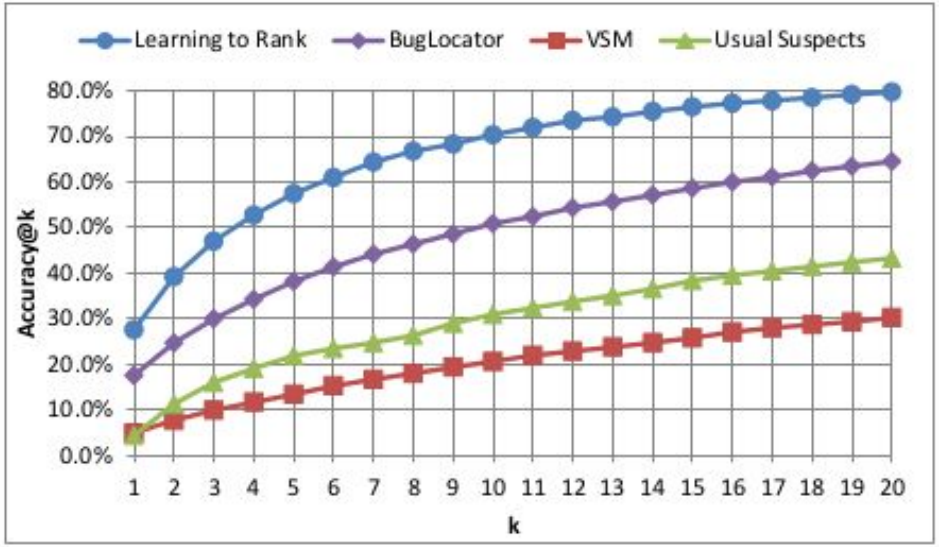

Figure 10: Accuracy graphs on SWT.

Discussion (cont.)

- **Class Name Similarity:** weight is class name length. Is this the right approach? There is no test of hypothesis in the paper or references. Ranks as least useful feature in the results.
- **Using only top 300 irrelevant documents:** risks biasing the SVM towards higher textual similarity. Random approach?
- **No generic model:** has to be tuned to the individual project, which requires a comprehensive history of reports and fixes.

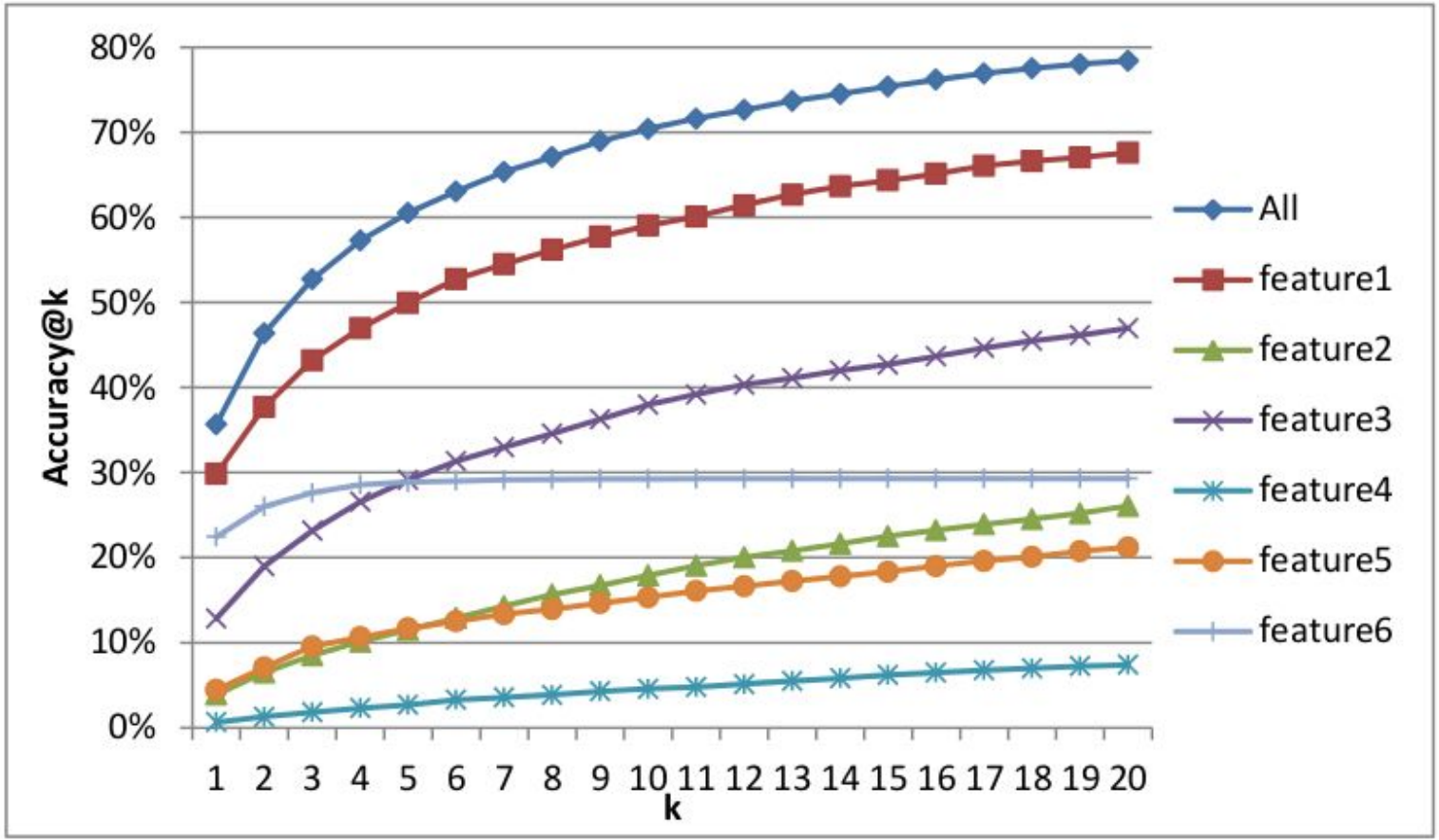

Figure 14: Single feature performance on Eclipse.

References

- Ye, Xin, Razvan Bunescu, and Chang Liu. "Learning to rank relevant files for bug reports using domain knowledge." *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2014.
- Bettenburg, Nicolas, et al. "What makes a good bug report?." *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*. ACM, 2008.