Locating Relevant Source Files for Bug Reports using Textual Analysis

Reza Gharibi, Amir Hossein Rasekh, Mohammad Hadi Sadreddini

CSSE 2017

October, 2017

Software and Bugs

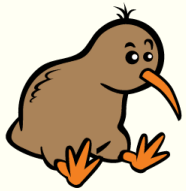
We have things like:

- Software Testing
- Software Inspection
- ...

but... Software still have bugs 🐞

Adding Bug Trackers

Bugzilla



FogBugz

Bug Reports Overflow

- 3389 bug reports for  in 2013[Ye et al., 2014]
- 300 bug reports for mozilla everyday[Shokripour et al., 2013]

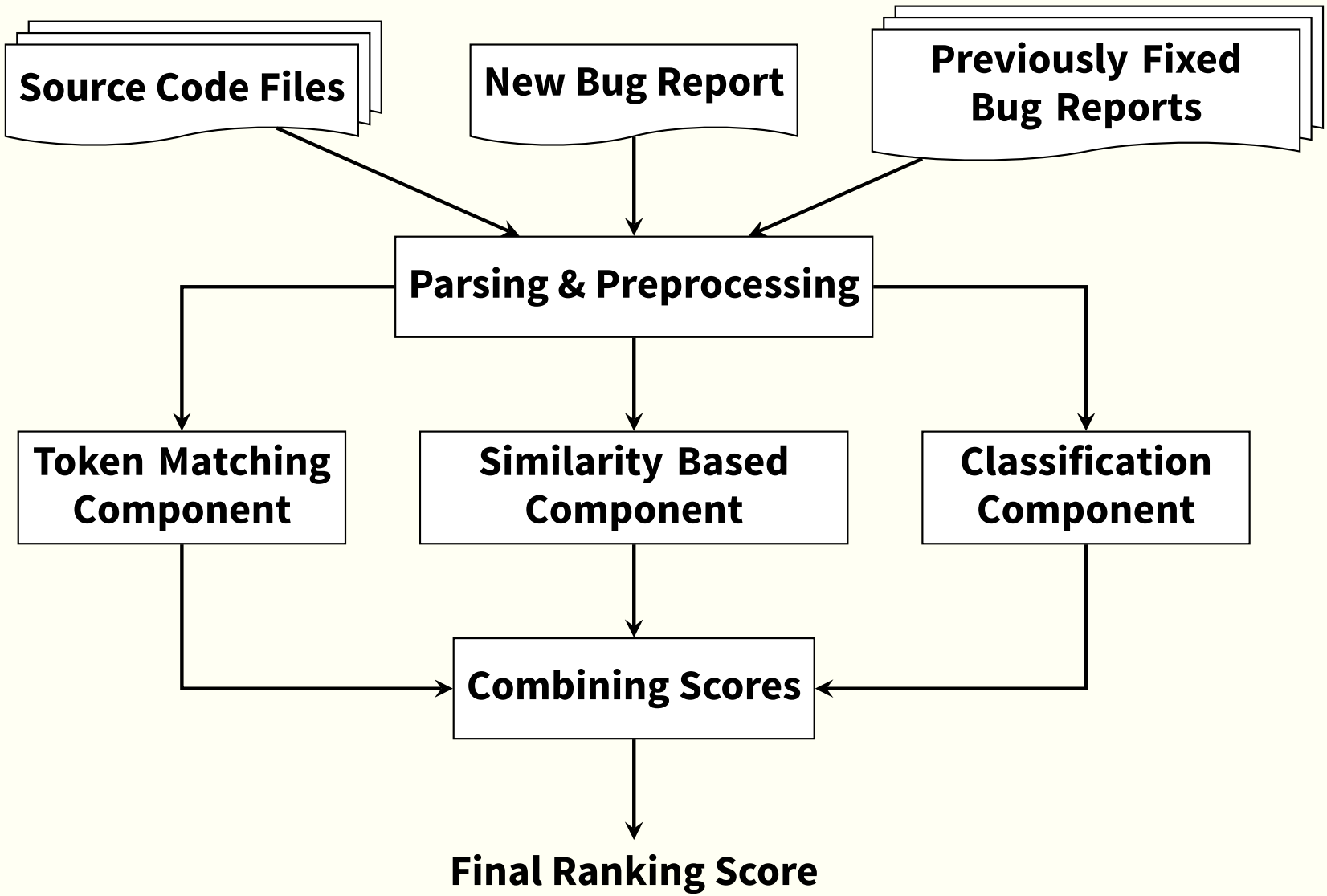
- Text based bug localization to reduce the search space

Motivating Example

Bug ID: 80720
Summary: **Pinned console** does not remain on top
Description:
Open two **console views**, … **Pin** one **console**. Launch another program that produces output. Both consoles **display** the last launch. The **pinned console** should remain **pinned**.

```
public class ConsoleView extends PageBookView
    implements IConsoleView, IConsoleListener {...
        public void display(IConsole console) {
            if (fPinned && fActiveConsole != null) { return;}
        } ...
        public void pin(IConsole console) {
            if (console == null) { setPinned(false);
            } else {
                if (isPinned()) { setPinned(false); }
                    display(console);
                    setPinned(true);
            }
        }
}
```

The Overall Structure



Parsing

Parsing and extracting specific parts.

```
<bug id="80720"
  opendate="2005-03-10 12:28:00"
  fixdate="2005-03-10 14:19:00">

  <buginformation>
  <summary>Pinned console does not
  remain on top</summary>

  <description>Open two console
  views, Pin one console. Launch
  another program that produces
  output. Both consoles display
  the last launch.</description>
  </buginformation>
</bug>
```

```
public class ConsoleView {
  public void display(console) {
      if (fPinned &&
        fActiveConsole != null)
          { return; }
  }
  public void pin(console) {
      if (console == null)
      { setPinned(false);
      } else {
          if (isPinned())
          { setPinned(false); }
          display(console);
          setPinned(true);
      }
  }
}
```

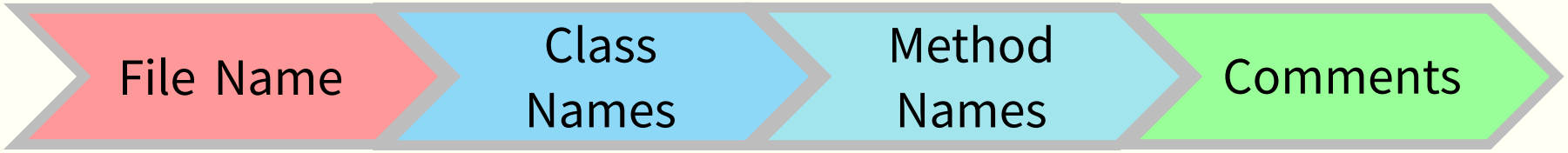
Preprocessing

- Extracting *NOUNS* from bug reports and source code comments.

  Open two console $\overbrace{\text{views}}^{\text{NNS}}$, pin one $\overbrace{\text{console}}^{\text{NN}}$. The $\overbrace{\text{pinned console}}^{\text{NN}}$ should remain pinned.

- $\text{PinnedConsole} \xrightarrow[\text{CamelCase}]{\text{Split}} \text{Pinned Console} \xrightarrow[\text{Numbers}]{\text{Punctutation}} \ldots \xrightarrow[\text{Programming Lang. keywords}]{\text{Stop words}} \ldots \xrightarrow{\text{Stem}} \text{pin consol}$

Token Matching Component



- Boosts the bug localization by finding the exact matching tokens.

```
<summary>Pinned console does not remain on top</summary>
```

Source code file: ConsoleView.java

- More score for specific matches.

Similarity Based Component

- Constructing tf-idf vectors:

$\vec{V}_{SourceFiles}$:
`[Class Names + Method Names + Comments Nouns]`
$\vec{V}_{BugReports}$: `[Summary Nouns + Description Nouns]`

- Calculating similarity for each report-src pair:

$$cos(s, b) = \frac{\vec{V_s} \cdot \vec{V_b}}{|\vec{V_s}| \times |\vec{V_b}|}$$

Classification Component

- Using previously fixed bug reports.
- A multi-label classification algorithm:

  Features: tf-idf weights of summary & nouns set
  Labels: each bug report's fixed source files

- Probabilities from an OvR with a LinearSVM base classifier.

Combining Scores

Final ranking of source files:

$$\begin{aligned} \mathit{FinalScore} = \alpha \times \mathit{TokenMatchingScore} \\ + \beta \times \mathit{SimilarityBasedComponent} \\ + \gamma \times \mathit{ClassificationComponent} \end{aligned}$$

$$\alpha, \beta, \gamma \in [0, 1]$$

Experimental Evaluation

Benchmark Dataset:

Project	BR Period	#Fixed Bugs	#Source Files
SWT	Oct 2004 - Apr 2010	98	484
ZXing	Mar 2010 - Sep 2010	20	391

Previous Works:

BugLocator: Vector Space Model + previous bug reports.

BLUiR: Structured BM25 + previous bug reports.

Results

Project	Approach	Top@1	Top@5	Top@10	MRR	MAP
SWT	BugLocator	39	66	80	0.53	0.45
	BLUiR	55	75	**86**	0.66	0.58
	Our Approach	**62**	**78**	84	**0.71**	**0.61**
ZXing	BugLocator	8	12	14	0.50	0.44
	BLUiR	8	13	14	0.49	0.39
	Our Approach	**10**	**14**	**16**	**0.59**	**0.50**

Conclusion

- Searching through a huge amount of source files is time consuming and inefficient.
- A bug localization technique can automate this process.
- Our approach presented an automatic bug localization technique.
- The approach improved the ranking of faulty source files.

Thanks for your attention

Any Questions?

References

- X. Ye, R. Bunescu, and C. Liu, “Learning to rank relevant files for bug reports using domain knowledge,” in Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2014, pp. 689–699.
- R. Shokripour, J. Anvik, Z. M. Kasirun, and S. Zamani, “Why so complicated? simple term filtering and weighting for location-based bug report assignment recommendation,” in Proceedings of the 10th Working Conference on Mining Software Repositories, 2013, pp. 2–11.
- J. Zhou, H. Zhang, and D. Lo, “Where should the bugs be fixed? more accurate information retrieval-based bug localization based on bug reports,” in 2012 34th International Conference on Software Engineering (ICSE), 2012, pp. 14–24.
- R. K. Saha, M. Lease, S. Khurshid, and D. E. Perry, “Improving bug localization using structured information retrieval,” in Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on, 2013, pp. 345–355.