Efficiently Finding Similar Stack Traces

Irving Muller Rodrigues
Polytechnique Montreal
irving.muller-rodrigues@polymtl.ca

Polytechnique de Montréal
Laboratoire DORSAL

Software crash

*https://www.freepik.com/vectors/computer

Software crash

Crash Report

```
1  Date: 2016-01-20T22:11:48.834Z
2  Product: XXXXXXXXXXXX
3  Version: 144.3143
4  Action: null
5  OS: Mac OS X
6  Java: Oracle Corporation 1.8.0_40-release
7  Message: new child is an ancestor
8
9  java.lang.IllegalArgumentException: new child is an ancestor
10   at javax.swing.tree.DefaultMutableTreeNode.insert(DefaultMutableTreeNode.java:179)
11   at javax.swing.tree.DefaultMutableTreeNode.add(DefaultMutableTreeNode.java:411)
12   at com.openapi.application.impl.ApplicationImpl$8.run(ApplicationImpl.java:374)
     .....
41   at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
42   at java.lang.Thread.run(Thread.java:745)
43   at org.ide.PooledThreadExecutor$2$1.run ....
```

Triage Process

- Compare the similarity between stack traces (crash dumps)
- Group stack traces
  - Prioritization bug fixing: focus on the most frequent and critical issues
  - Problem diagnosis: complementary information about an error

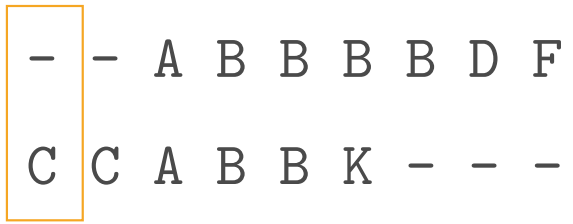
TraceSim

- State-of-the-art method for comparing stack traces
- Based on *optimal global alignment*

Global Alignment

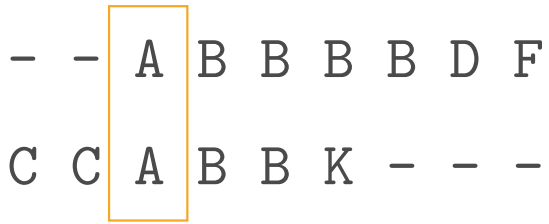
```
- - A B B B B D F
C C A B B K - - -
```

Global Alignment Types



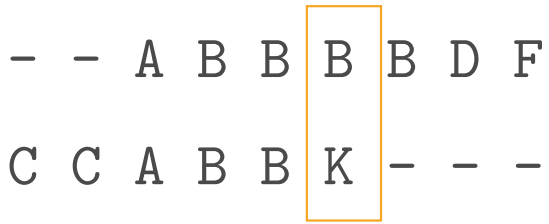
Alignment between an element and a gap

Global alignment



Match

Global Alignment Types



Mismatch

Optimal Global Alignment

- Scoring scheme: define how to compute the value of each match, mismatch and gap alignments.
- Alignment score: the sum of the match values minus the sum of the values of each mismatch and gap alignment.
- Goal: find the highest alignment score

Shortcoming of TraceSim

- Time complexity of $O(n \cdot m)$
  - Expensive to deploy in industrial environments.

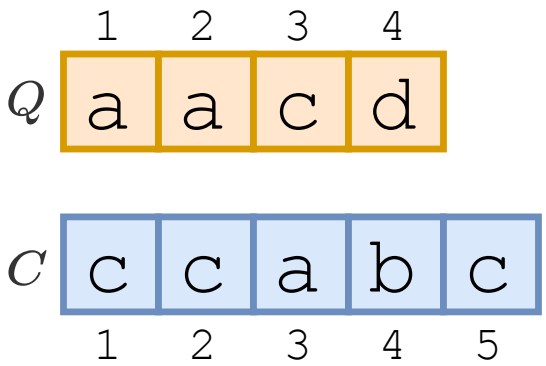
Our solution

- Alignment method for similarity comparison
  - Optimal global alignment is simplified
    1. Optimally requirement removal
    2. Frame dependency is loosed
  - Time complexity: $O(n + m)$

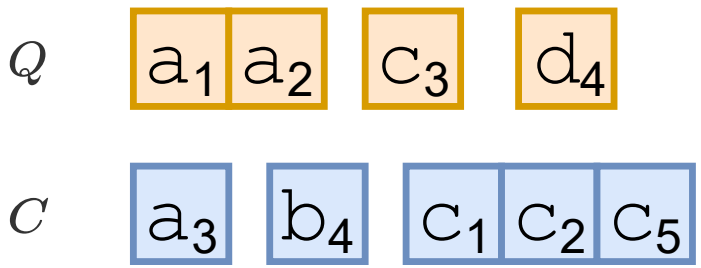
Our Algorithm Steps

1. Group the frame with same subroutine names (sorted by frame position)
2. Individually align each group of frames with same subroutine names
    1. Match frames from top positions to the bottom
    2. Unmatched frames are aligned to gaps

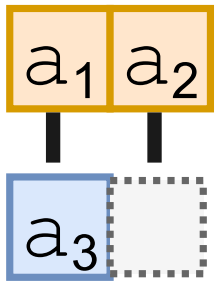
Our Algorithm Example



Our Algorithm Example - Step 1

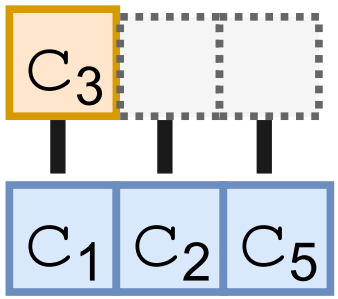


Our Algorithm Example - Step 2



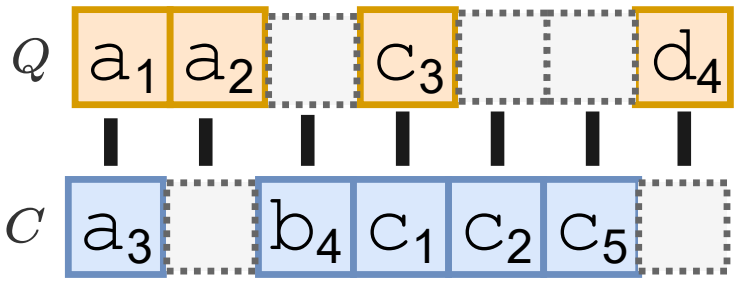
Our Algorithm Example - Step 2

Our Algorithm Example - Step 2



Our Algorithm Example - Step 2

Our Algorithm Example - Final



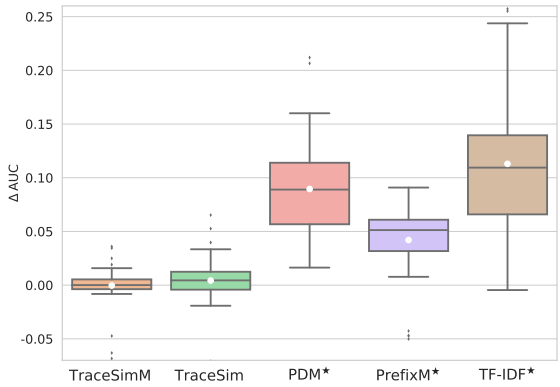
Experiments

- Task: Duplicate crash reports detection
- Datasets: Netbeans, Eclipse, OpenOffice, and Gnome
- 50 different runs
- Metrics
  1. Area Under the ROC Curve (AUC)
  2. MAP: measure how good is our method on ranking.
  3. Recall Rate@1 (RR@1): percentage of queries whose the first report in the recommend list is the correct duplicate.

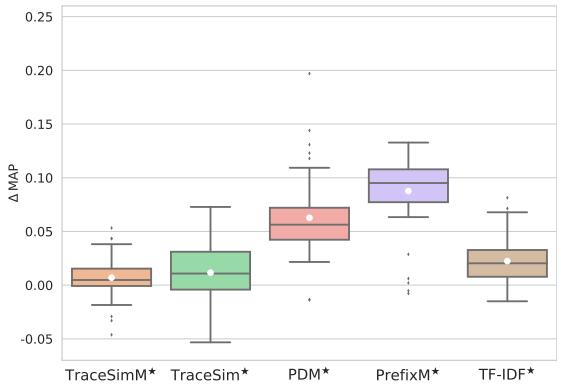
Experiments

- Our algorithm is compared to five competing methods
  - Difference: ΔAUC, ΔMAP ΔRR@1
  - Wilcoxon signed-rank test: statistical significance is indicated by $\bigstar$

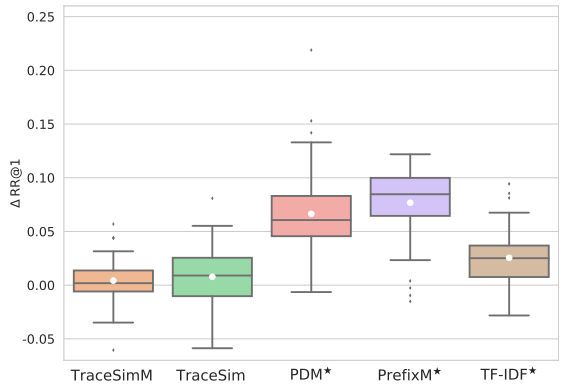
Ubuntu



(a) ΔAUC

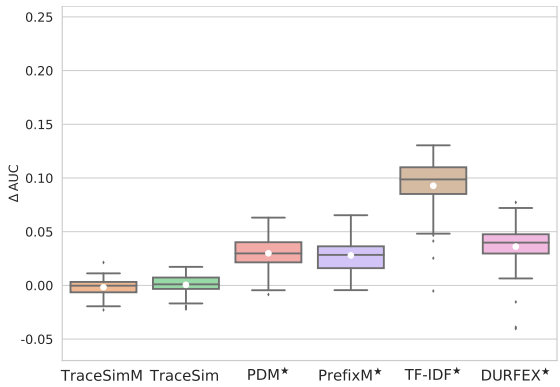


(b) ΔMAP

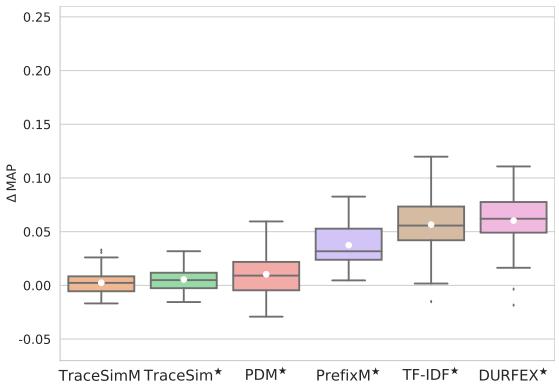


(c) ΔRR@1

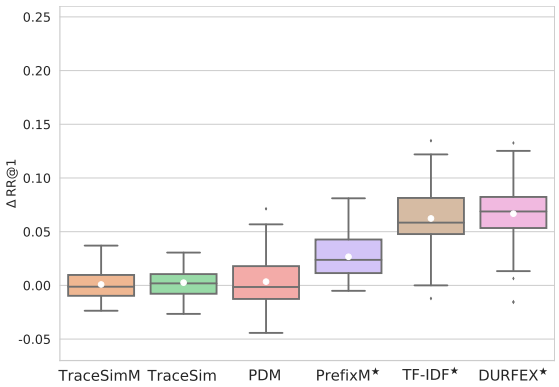
Eclipse



(a) ΔAUC

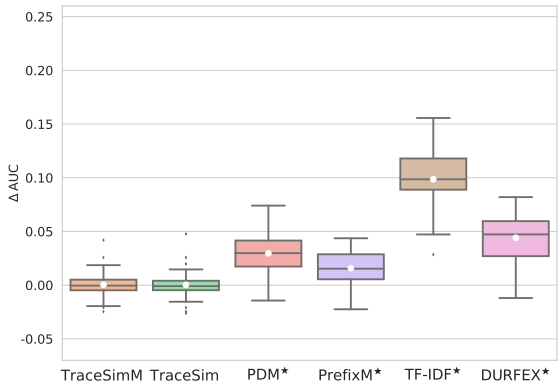


(b) ΔMAP

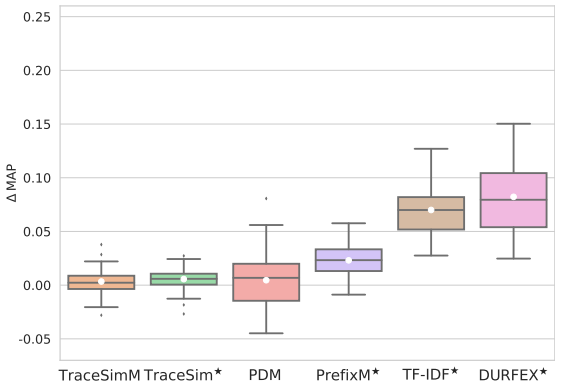


(c) ΔRR@1

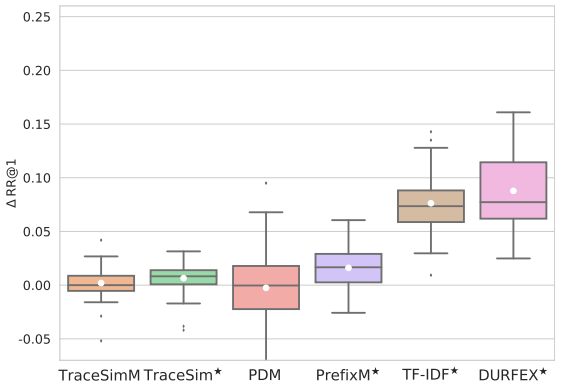
Netbeans



(a) ΔAUC

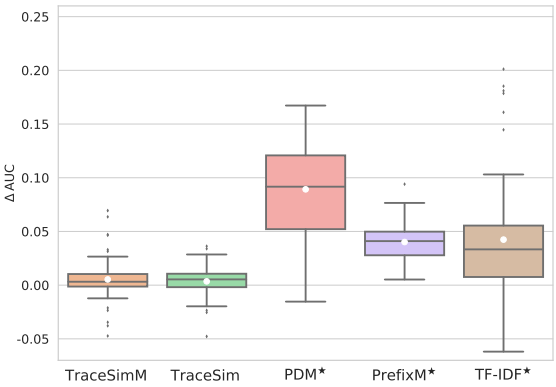


(b) ΔMAP

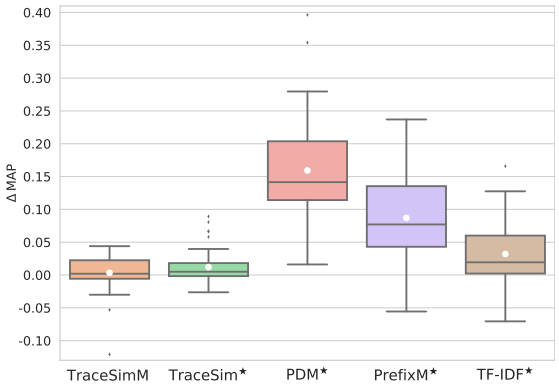


(c) ΔRR@1

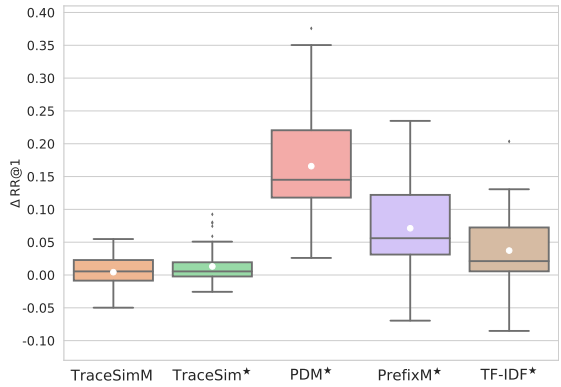
Gnome



(a) ΔAUC

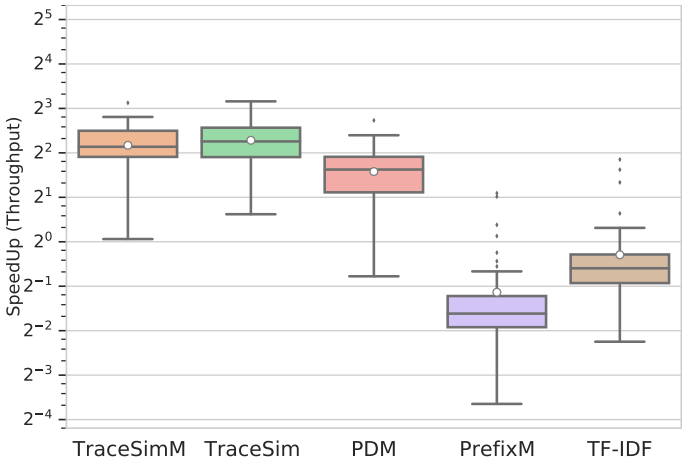


(b) ΔMAP

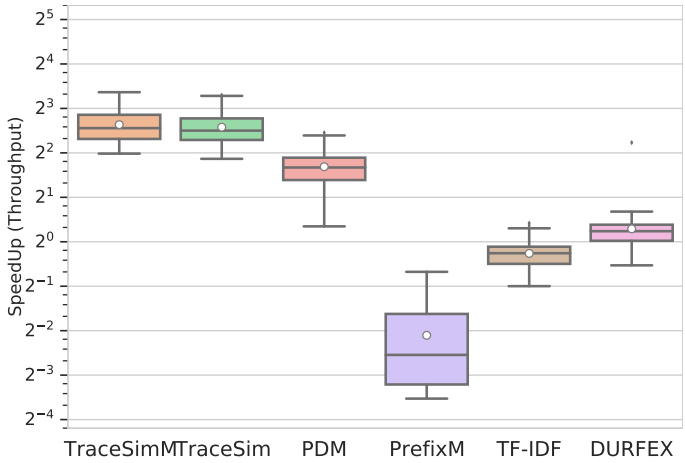


(c) ΔRR@1

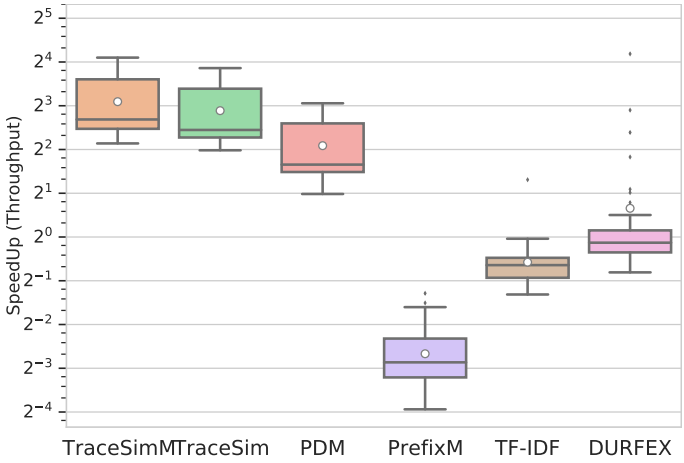
Speedup



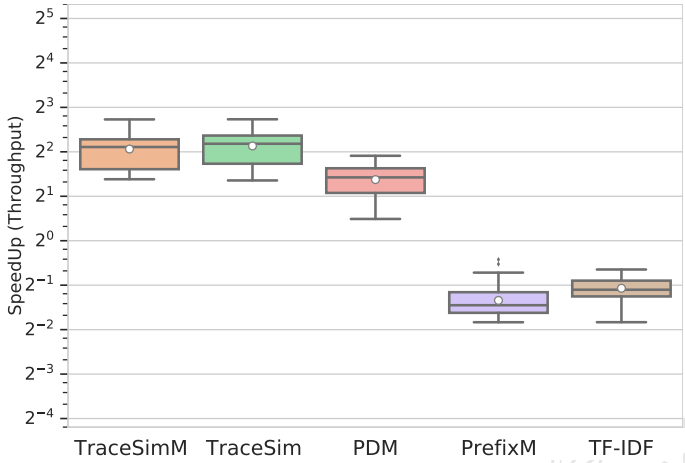
(a) Ubuntu



(b) Eclipse



(c) Netbeans



(d) Gnome

Concluding Remarks

- We propose a simple alignment method
- Our algorithm achieves state-of-art performance in all datasets
- Our algorithm is much faster than TraceSim

Thank you for your attention!

Questions?