Duplicate bug report detection through machine learning techniques

Irving Muller Rodrigues

irving.muller-rodrigues@polymtl.ca

École Polytechnique de Montréal

Laboratoire DORSAL

Duplicate bug reports

- Duplicate bug reports describe the same bug
- Very common in Bug Tracking Systems (BTSs)
- Undetected duplicate bug reports
  - Waste of developer time
- Manually filtered by triage team
  - Beyond team capacity
- Machine learning technique to help triage team
  - Automatic detection of duplicate bug reports
  - **Bug report deduplication**

Our Research

- Address the bug deduplication using three distinct data types:
  1. Textual data
  2. Stack trace
  3. Tracing - user space or kernel space

Our Research

1. Textual data
   - A Soft Alignment Model for Bug Deduplication (MSR 2020)
   - Attention mechanism $\rightarrow$ more powerful model to compare textual data
   - State-of-the-art performance

Our Research

2. Stack trace
   - We are currently working on this problem.

Our Research

Why to use stack traces?

Textual Data Disadvantage

- Heavily dependent on user's expertise
  - Vague and Ambiguous
  - Different technical background $\rightarrow$ different terminologies
- Limited information about the system execution
  - Exterior system behaviors
- Stack Traces
  - More precise and technical information about the bug
  - User independent

Previous Works

- *Dang et. al, 2012*[1]; *Ebrahimi et. al., 2016*[2]; *Koopaei et. al. 2015*[3]; *Kim et. al., 2015*[4]
  - Function calls are the sequence elements
  - Sequence similarity
- *Lerch et. al., 2013*[5]; *Campbell et. al, 2016*[6]
  - Textual similarity
  - TF-IDF
  - Ignore structure information

Previous Works

Bug 15247

Position	Function call
1	localstore.FileSystemResourceManager.read
2	resources.File.getContents
3	resources.File.getContents
4	core.util.SyncFileWriter.readLines
5	core.util.SyncFileWriter.readAllResourceSync
6	EclipseSynchronizer.cacheResourceSyncForChildren
7	EclipseSynchronizer.getResourceSync
8	EclipsePhantomSynchronizer.getResourceSync

Bug 51547

Position	Function call
1	localstore.FileSystemResourceManager.read
2	internal.resources.File.getContents
3	internal.resources.File.getContents
4	core.util.SyncFileWriter.readFirstLine
5	core.util.SyncFileWriter.readFolderSync
6	EclipseSynchronizer.cacheFolderSync
7	EclipseSynchronizer.getFolderSync
8	EclipseFolder.getFolderSyncInfo

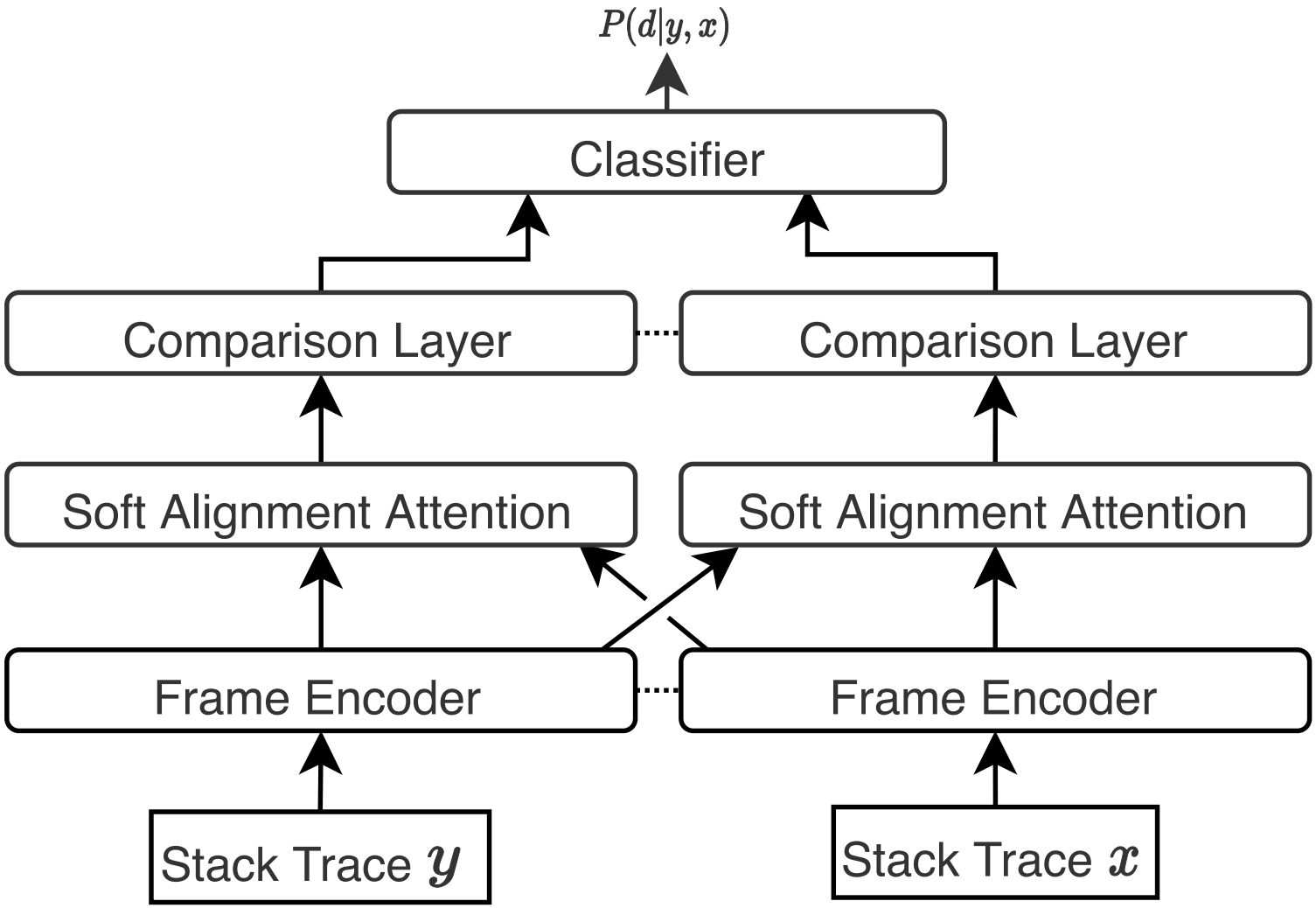
Previous Works

- *Dang et. al, 2012*[1]; *Ebrahimi et. al., 2016*[2]; *Koopaei et. al. 2015*[3]; *Kim et. al., 2015*[4]
  - Function calls are the sequence elements
  - Sequence similarity
- *Lerch et. al., 2013*[5]; *Campbell et. al, 2016*[6]
  - Textual similarity
  - TF-IDF
  - Ignore structure information

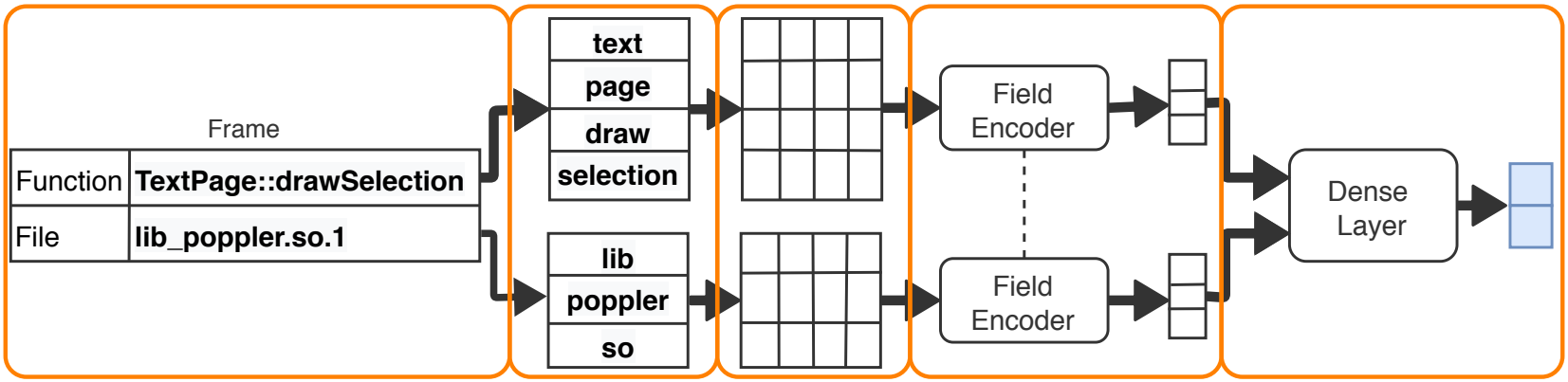
Our Solution

Textual similarity + Sequence Similarity

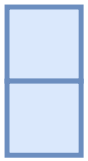
Our Solution



Our Work - Frame Encoder



Our Work - Frame Encoder

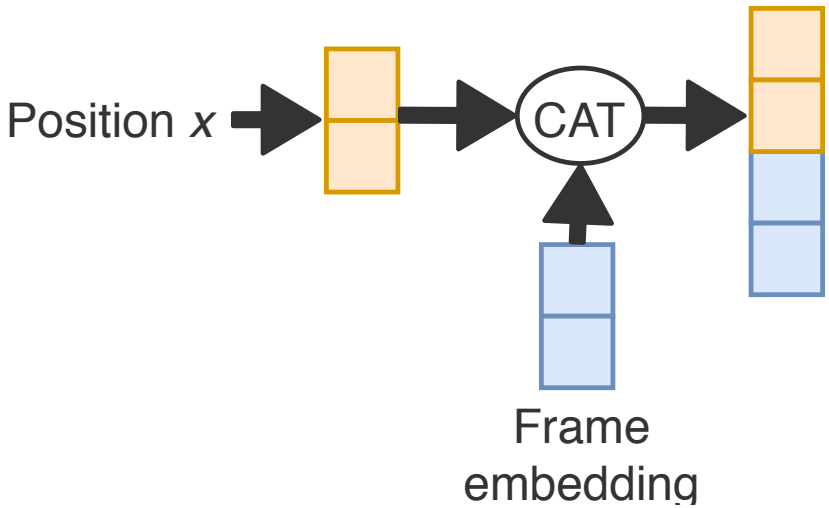


Problem

- Frame embedding only contains textual information
- Lack of the frame position

Our Work - Position embedding

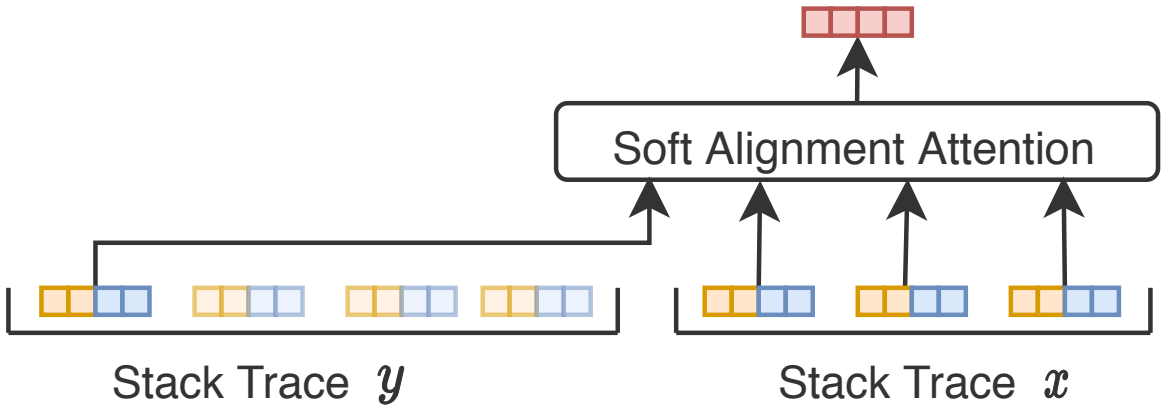
- Solution
  - Position embedding: Positions are converted to vectors
  - Concatenation: Position embedding and Frame embedding



Our Work - Soft attention alignment

1 Soft attention alignment

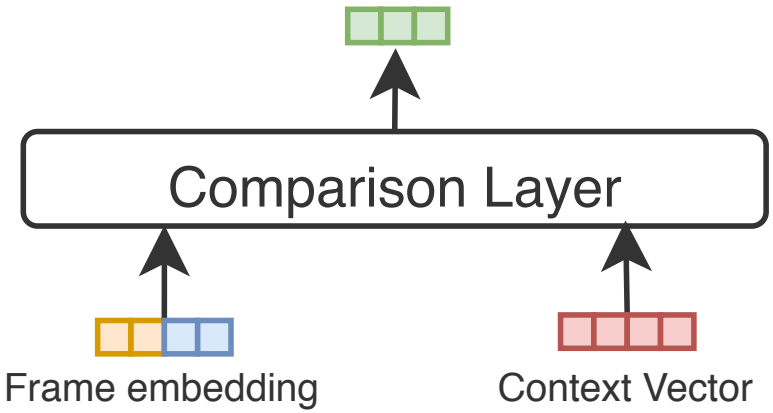
- Attention mechanism
- Summarize frame embedding information in a stack trace x into a fixed-size vector
  - Focus on the information in x that is related to a specific frame in the stack trace y
  - Output: **context vector**



Our Work - Comparison Layer

2 Comparison Layer

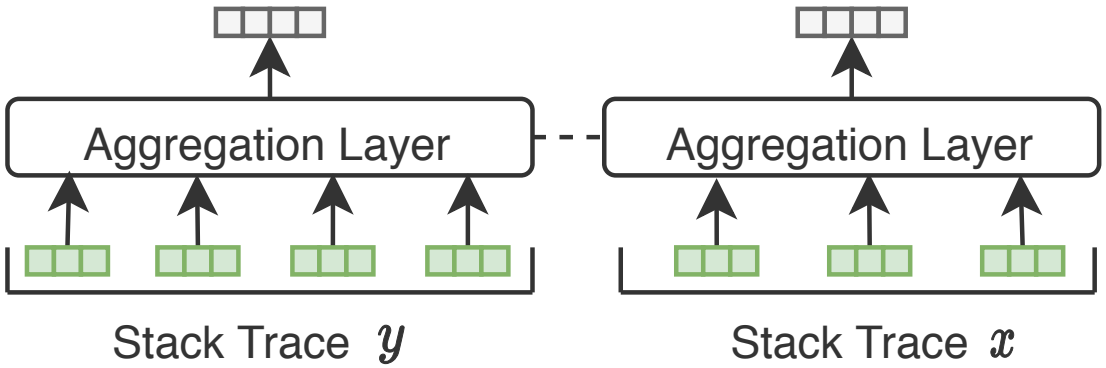
- Compare frame embedding and its context vector
- Dense Layer
- Output: **comparison vector**



Our Work - Aggregation layer

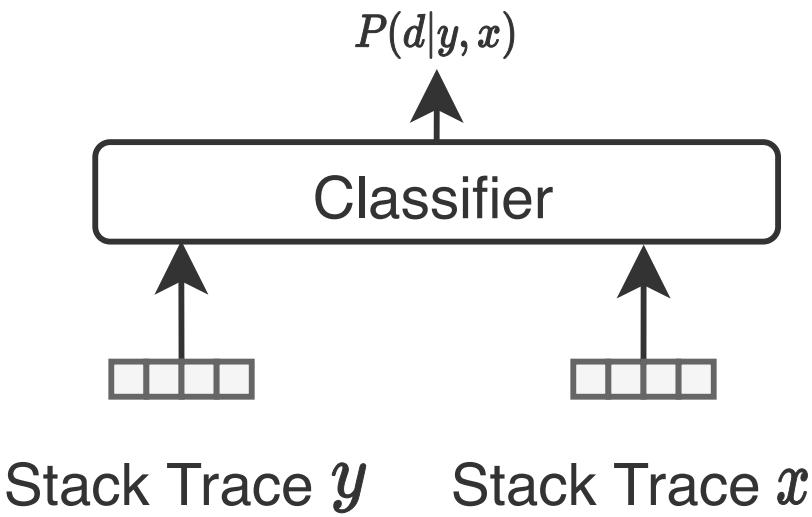
❸ Aggregation layer

- Aggregate all the comparison vector of a stack trace into a vector
- LSTM + mean pooling
- Output: **aggregation vector**



Our Work - Classifier

4. Classifier
   - Input: aggregation vectors of stack traces y and x
   - MLP
   - Output: probability $P(y|q, c)$ of y being a duplicate of x



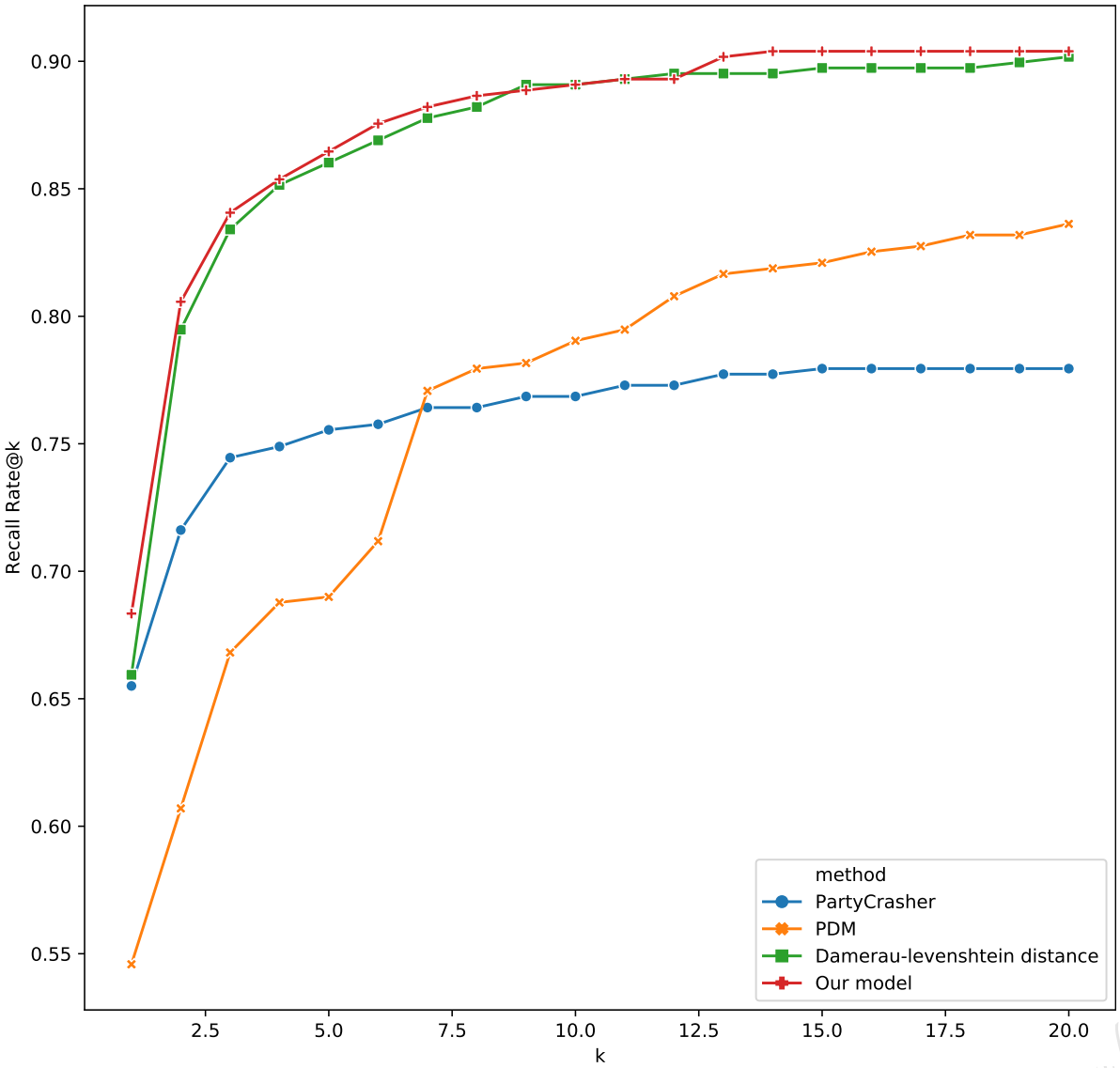
Experiments

- Ubuntu launchpad
  - 15,293 bug reports in 3,824 buckets
  - 70% training and 30% test
  - Validation: 5% training data set
- Our method is compared to:
  - Damerau-levenshtein distance
  - PartyCrasher [6] (TF-IDF)
  - Position Dependent Model (PDM) [1]

Experiments

- Metrics
  1. Ranking metrics
     - Recommendation list for each duplicate report
     - Mean average precision: inversely proportional to the correct report positions in the recommendation lists
     - Recall Rate @k: portion of duplicate report whose the correct reports are in top$-k$ positions of the recommendation lists.
  2. Multi-class classification: accuracy (threshold of top-1 report)
  3. Clustering metrics: ARI and AMI (threshold of top-1 report)

Preliminar Results - Recall rate @k



Preliminar Results - MAP

Method	MAP
PartyCrasher	0.7006
PDM	0.621522
Damerau-levenshtein distance	0.752491
Our model	0.767613

Preliminar Results - Accuracy

Method	Accuracy
PartyCrasher	0.4421
PDM	0.294118
Damerau-levenshtein distance	0.419355
Our model	0.493359

Preliminar Results - ARI and AMI

Method	AMI	ARI
PartyCrasher	0.689413	0.70271
Damerau-levenshtein distance	0.711171	0.729981
PDM	0.563246	0.422052
Our model	0.753107	0.745103

Future Work

- Overfitting
  - Dropout, Layer norm and Early stopping
- Test different frame encoder architectures
  - Current: Mean pooling operator
- Categorical data
- Ablation Study

Thank you for your attention!

Y. Dang, R. Wu, H. Zhang, D. Zhang, and P. Nobel, "Rebucket: A method for clustering duplicate crash reports based on call stack similarity," in *2012 34th International Conference on Software Engineering (ICSE)*, June 2012, pp. 1084–1093.

N. E. Koopaei, M. S. Islam, A. Hamou-Lhadj, and M. Hamdaqa, "An effective method for detecting duplicate crash reports using crash traces and hidden markov models," in *Proceedings of the 26th Annual International Conference on Computer Science and Software Engineering*, ser. CASCON '16. Riverton, NJ, USA: IBM Corp., 2016, pp. 75–84. [Online]. Available: http://dl.acm.org/citation.cfm?id=3049877.3049885

N. E. Koopaei and A. Hamou-Lhadj, "Crashautomata: An approach for the detection of duplicate crash reports based on generalizable automata," in *Proceedings of the 25th Annual*

International Conference on Computer Science and Software Engineering, ser. CASCON '15. Riverton, NJ, USA: IBM Corp., 2015, pp. 201–210. [Online]. Available: http://dl.acm.org/citation.cfm?id=2886444.2886474

Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014, pp. 1746–1751. [Online]. Available: http://aclweb.org/anthology/D14-1181

J. Lerch and M. Mezini, "Finding duplicates of your yet unwritten bug report," in *Proceedings of the 2013 17th European Conference on Software Maintenance and Reengineering*, ser. CSMR '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 69–78. [Online]. Available: http://dx.doi.org/10.1109/CSMR.2013.17


J. C. Campbell, E. A. Santos, and A. Hindle, "The unreasonable effectiveness of traditional information retrieval in crash report deduplication," in *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, May 2016, pp. 269–280.