**Title:** Towards More Accurate Retrieval of Duplicate Bug Reports
**Authors:** Chengnian Sun* , David Lo† , Siau-Cheng Khoo* , Jing Jiang†
**Published in:** Conference, 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), Lawrence, KS, USA, November 6-10, 2011

**Submitted By**
Md Mynuddin
Id: ASH1825007M

*Question–Answer Form*

***• What is your take-away message from this paper?***

In a bug tracking system, separate testers or users may submit several reports on the same defect, known as duplicates, which may necessitate additional maintenance work in triaging and correcting bugs. So the takeaway message from this paper is to similarity measure the duplicates among the bug reports to reduce the time and cost to fix the bugs in a proper and easy way.

***• What is the motivation for this work (both people problems and technical problems), and its distillation into a research question? Why doesn't the people's problem have a trivial solution? What are the previous solutions and why are they inadequate?***

End users and testers may submit the same defect in the bug reporting system several times. This is problematic since multiple developers should not be assigned the same defect. The main motivation of this work is to detect and prevent it to manage and track the bugs easier to fix them.

I don't know why people don't have a trivial solution, system complexity might be a reason.

There are two primary approaches to detecting duplicate reports. One approach is filtering duplicate reports, preventing them from reaching the triggers. This approach is more complex, as the cutting-edge solution suggested can only remove 8% of the duplicate reports. The remaining 92% of duplicate bug reports would still need to be investigated manually. In another approach, they use SVM where the accuracy is also very low. That's why this approach is inadequate.

***• What is the proposed solution (hypothesis, idea, design)? Why is it believed it will work? How does it represent an improvement? How is the solution achieved?***

In this research, They proposed a retrieval function (REP) to measure the similarity between two bug reports in order to accurately identify such duplicates. It fully utilizes the information provided in a bug report, including not only the similarity of textual material in the summary and description sections, but also the similarity of non-textual fields such as product, component, version, and so on. They extend **BM25F** - an effective similarity formula in the information retrieval field, particularly for duplicate report retrieval - for a more accurate textual similarity assessment.

I think there is no universal formula to solve a problem. It is all about experiments. They tested numerous ways to determine the best one that was internationally similar to other methods. They increase the recall rate @ k (1<= k<= 20) and mean average precision of state-of-the-art automated duplicate bug detection techniques by 10-27 % and 17-23 %, respectively.

**Process:**

Multiple reports from different submitters may correspond to the same bug, resulting in a duplicate bug report problem. They refer to the first report as **master** and the other duplicate ones as **duplicate**. Existing and new bug reports are preprocessed by standard information retrieval techniques, i.e., tokenization, stemming, and stop work removal(NLP). Here is the workflow

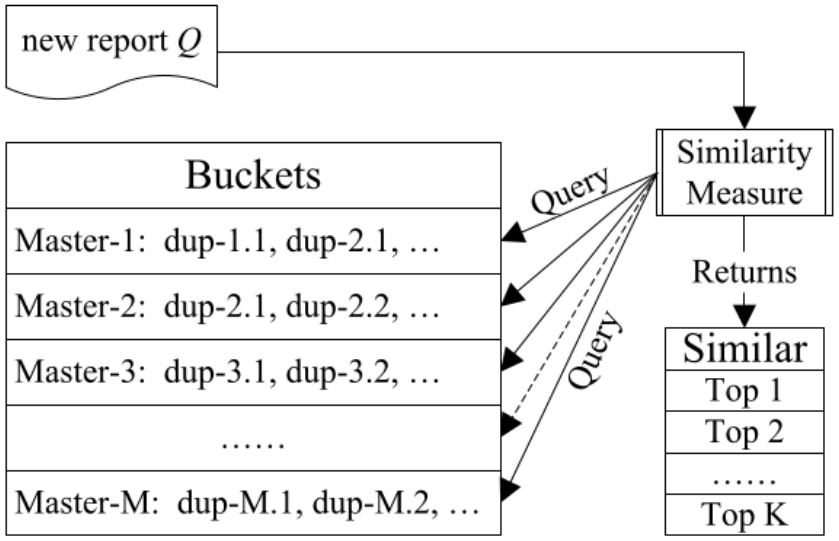


Figure 1: Overall Workflow for Retrieving Duplicates

When a new bug report Q is received, the system computes the similarity between Q and each bucket and returns K master reports to which the top-K buckets have the highest similarity. The similarity between a report and a bucket is the highest similarity estimated by the component Similarity Measure between the report and each report in the bucket.

**BM25F** is an effective textual similarity function for structured document retrieval. And Optimizing the Similarity Functions by using Gradient Descent.

**Step 1:** Extending BM25F for Structured Long Queries

**Step 2:** Retrieval Function

**Step 3:** Optimizing REP with Gradient Descent (Hyperparameter Tuning).

***• What is the author's evaluation of the solution? What logic, argument, evidence, artifacts (e.g., a proof-of-concept system), or experiments are presented in support of the idea?***

They test their method on several large bug report datasets from large open-source projects, including Mozilla, a software platform that hosts several sub-projects such as the Firefox browser and the Thunderbird email client; Eclipse, a popular open-source integrated development environment; and OpenOffice, a well-known open source rich text editor.

| Dataset | Size | Period | | Training Reports | | Testing Reports | |
|---|---|---|---|---|---|---|---|
| | | From | To | #Duplicate | #All | #Duplicate | #All |
| OpenOffice | 31,138 | 2008-01-01 | 2010-12-21 | 200 | 3,696 | 3,171 | 27,442 |
| Mozilla | 75,653 | 2010-01-01 | 2010-12-31 | 200 | 4,529 | 6,725 | 71,124 |
| Eclipse | 45,234 | 2008-01-01 | 2008-12-31 | 200 | 5,863 | 2,880 | 39,371 |
| Large Eclipse | 209,058 | 2001-10-10 | 2007-12-14 | 200 | 3,528 | 27,295 | 205,530 |

Figure 2: Dataset Overview

**Experimental Setup:**

The prototype was developed in C++, and all testing was done on a Linux computer with an Intel Core 2 Quad 3.0GHz processor and 8GB of RAM. We repeated all experiments five times and used the average values for our analysis and conclusion because the training set was created randomly and the gradient descent also involves randomness.

***• What is your analysis of the identified problem, idea, and evaluation? Is this a good idea? What flaws do you perceive in the work? What are the most interesting or controversial ideas? For work that has practical implications, ask whether this will work, who would want it, what it will take to give it to them, and when might it become a reality.***

- ❖ I think if I want to answer the all above questions, I need to read at least 5-6 papers related to the same topics and apply the ways of the solution mentioned in those papers,s and performed an experiment by following their way of solution.

***• What are the paper's contributions (author's and your opinion)? Ideas, methods, software, experimental results, experimental techniques...?***

- ➔ Solution of the problem Ideas
- ➔ Methods
- ➔ Experimental Result
- ➔ Compare their result with their previous work.

***• What are future directions for this research (author's and yours, perhaps driven by shortcomings or other critiques)?***

- ➔ Build an indexing structure of the bug report repository to speed up the retrieval process
- ➔ Integrate the technique into the Bugzilla tracking system.

***• What questions are you left with? What questions would you like to raise in an open discussion of the work (review interesting and controversial points, above)? What do you find difficult to understand? List as many as you can.***

Open discussion of the work: How does BM25F work?

Difficult to understand:

- ➔ Mathematical equation
- ➔ Algorithm 1 Simplified Parameter Tuning Algorithm
- ➔ Algorithm 2 Constructing a Training Set from a Repository
- ➔ Algorithm 3 Tuning Parameters in REP
- ➔ And How BM25F and extended BM25F work.