



An Empirical Study of Bugs in Machine Learning Systems

IEEE 23rd International Symposium on Software Reliability Engineering



Ferdian Thung



Shaowei Wang



David Lo



Lingxiao jiang

紹介者：柏 祐太郎

- 多くのシステムやSE研究で機械学習を利用している
 - SNS, データマイニング, 情報検索
 - bug localization, find bugs, recommend fixes
- 機械学習ライブラリのバグは多くのシステムや研究に影響を与える
 - 出力の誤り, クラッシュ, 遅い応答
- 機械学習のバグについて研究されていない
 - メールクライアント, IDE, ブラウザなどはやってる

Research Questions

RQ1. どの程度の頻度でバグが出現するか？

RQ2. どんな種類のバグが出現するか？

(以下, 種類ごと)

RQ3. どの程度の深刻度か？

RQ4. 修正に要する時間はどの程度か？

RQ5. 修正に要するエフォートはどの程度か？

RQ6. 修正するファイル数はいくつ？ (Impactと定義)

1. データ収集

(1) バグとコミットデータを取得

2. マニュアル分類

(1) 不具合のDescriptionを読んで、
不具合を分類

(2) Descriptionだけでわからない
場合はコミットデータも確認

3. 統計的計算

(1) 深刻度, 修正時間, ファイル数

Table II
BUG CATEGORIES BASED ON CODE DEFECT TYPES IN [34]

Category	Definition
algorithm/method	The implementation of an algorithm/method does not follow the expected behavior.
assignment/initialization	Error in assigning variable values.
checking	Missing necessary checks that lead to an error or a wrong error message.
data	Wrong use of data structure.
external interface	Error in interfacing with other systems or users, such as using deprecated methods from other systems, required updates to own external interfaces for ease of usage, etc.
internal interface	Error in interfacing with another component of the same system, such as violating the contract of inheritance, wrong use of operations from other classes, etc.
logic	Incorrect expressions in conditional statements (e.g., if, while, etc.)
non-functional	Violations in non-functional requirements, such as improper variable or method names, wrong documentation to the implementation of a method, etc.
timing/optimization	Error that causes concurrency or performance issues, such as deadlock, high memory usage, etc.
configuration	Error in non-code (e.g., configuration files) that affects functionality.
others	Other bugs that do not fall into one of the above categories.

分析対象プロジェクト

- Apacheの機械学習に関するプロジェクト



Mahout

- » 機械学習ライブラリ
- » 全不具合：1,000件
- » Closed：300件
- » 抽出：200件



Lucene

- » 検索ライブラリ
- » 全不具合：4,000件
- » Closed：1,500件
- » 抽出：200件



OpenNLP

- » 自然言語ライブラリ
- » 全不具合：500件
- » Closed：110件
- » 抽出：100件

RQ1 : どの程度の頻度でバグが出現するか？ 6

- バグ密度と年間不具合報告数（平均）で評価
 - Luceneが密度および不具合数ともに最も多い
 - MahoutはOpenNLPよりバグ密度が高い
 - OpenNLPはMahoutより年間不具合報告数が高い
 - ただし、**これらが高い＝品質が悪いとは言えない**
 - » ユーザー数なども関係してくる

Table III

BUG DENSITIES IN MACHINE LEARNING SYSTEMS

Application	Bug Count Per kLOC	Bug Count Per Year
Mahout	1.79 bugs/kLOC	73.36 bugs/year
Lucene	2.77 bugs/kLOC	144.76 bugs/year
OpenNLP	1.45 bugs/kLOC	95.68 bugs/year

Table I

NUMBERS OF CLOSED BUGS IN MACHINE LEARNING SYSTEMS

Application	Size – Lines of Code	Bug Count	Duration
Mahout	175,295 (Version 0.6)	314	4.28 years
Lucene	554,036 (Version 3.6)	1,533	10.59 years
OpenNLP	78,224 (Version 1.5.2)	113	1.39 years

RQ2 : どんな種類のバグが出現するか？

- 不具合を目視で分類

BUG TYPES

Type	Count	Percentage
algorithm/method	113	22.60%
non-functional	78	15.60%
assignment/initialization	65	13.00%
checking	57	11.40%
external interface	38	7.60%
internal interface	38	7.60%
data	28	5.60%
logic	27	5.40%
configuration	27	5.40%
timing/optimization	24	4.80%
others	5	1%

RQ2 : どんな種類のバグが出現するか？

8

Lucene - Java / LUCENE-124 Fuzzy Searches do not get a boost of 0.2 as stated in "Query Syntax" doc

Description

According to the website's "Query Syntax" page, fuzzy searches are given a boost of 0.2. I've found this not to be the case, and have seen situations where exact matches have lower relevance scores than fuzzy matches.

Rather than getting a boost of 0.2, it appears that all variations on the term are first found in the model, where $\text{dist}^* > 0.5$.

- $\text{dist} = \text{levenshteinDistance} / \text{length of min}(\text{termlength}, \text{variantlength})$

This then leads to a boolean OR search of all the variant terms, each of whose boost is set to $(\text{dist} - 0.5) * 2$ for that variant.

Figure 2. Example of algorithm/method bug

あいまい検索で適切な重み付けができない

Lucene - Java / LUCENE-713 File Formats Documentation is not correct for Term Vectors

Description

From Samir Abdou on the dev mailing list:

Hi,

There is an inconsistency between the files format page (from Lucene website) and the source code. It concerns the positions and offsets of term vectors. It seems that documentation (website) is not up to date. According to the file format page, offsets and positions are not stored! Is that correct?

Figure 3. Example of non-functional bug

ドキュメントが間違っている

Lucene - Java / LUCENE-483 QueryParser.getFieldQuery(String,String) doesn't set default slop on MultiPhraseQuery

Description

there seems to have been an oversight in calling `mph.setSlop(phraseSlop)` in `QueryParser.getFieldQuery()` right (sometimes, ... see below).

when i tried amending `TestMultiAnalyzer` to demonstrate the problem, I discovered that the grammar apar being parsed, in which case it passes the default as if it were specified.

(just to clarify: i haven't comfirmed this from a detailed reading of the grammar/code, it's just what i've ded

The problem isn't entirely obvious unless you have a subclasses of `QueryParser` and try to call `getFieldQ`

Figure 4. Example of assignment/initialization bug

デフォルト値がセットされていない

RQ3 : どの程度の深刻度か？

- 各分類でSeverity（特にBlockerとCritical）を比較
 - デフォルト値のMajorが多く，バイアスがあることも言及

BUG COUNTS FOR VARIOUS BUG SEVERITIES

Type	Severity	Count	Proportion	Type	Severity	Count	Proportion
algorithm/method	Blocker	1	0.88%	internal interface	Blocker	1	2.63%
	Critical	3	2.65%		Critical	0	0.00%
	Major	70	61.95%		Major	22	57.89%
	Minor	33	29.20%		Minor	11	28.95%
	Trivial	6	5.31%		Trivial	4	10.53%
assignment/initialization	Blocker	2	3.08%	logic	Blocker	0	0.00%
	Critical	1	1.54%		Critical	0	0.00%
	Major	34	52.31%		Major	15	55.56%
	Minor	24	36.92%		Minor	8	29.63%
	Trivial	4	6.15%		Trivial	4	14.81%
checking	Blocker	2	3.51%	non-functional	Blocker	0	0.00%
	Critical	1	1.75%		Critical	0	0.00%
	Major	32	56.14%		Major	38	48.72%
	Minor	17	29.82%		Minor	29	37.18%
	Trivial	5	8.77%		Trivial	11	14.10%
configuration	Blocker	0	0.00%	timing/optimization	Blocker	0	0.00%
	Critical	0	0.00%		Critical	0	0.00%
	Major	19	70.37%		Major	19	79.17%
	Minor	6	22.22%		Minor	5	20.83%
	Trivial	2	7.41%		Trivial	0	0.00%
data	Blocker	0	0.00%	others	Blocker	0	0.00%
	Critical	1	3.57%		Critical	0	0.00%
	Major	15	53.57%		Major	3	60.00%
	Minor	12	42.86%		Minor	0	0.00%
	Trivial	0	0.00%		Trivial	2	40.00%
external interface	Blocker	1	2.63%				
	Critical	1	2.63%				
	Major	21	55.26%				
	Minor	12	31.58%				
	Trivial	3	7.89%				

RQ4： 修正に要する時間はどの程度か？

- 修正時間で比較
 - algorithmとtimingとconfigurationが最も時間がかかった
 - algorithmとinternal interfaceでは高い深刻度より低い深刻度の不具合の方が早く修正された

BUG-FIXING DURATIONS IN TERMS OF DAYS

Type	Min	Max	Mean	<u>Median</u>
algorithm/method	0.0022	2433.7033	91.7238	3.8740
assignment/initialization	0.0003	160.9271	9.9160	0.5000
checking	0.0017	195.7766	17.1335	<u>1.1175</u>
configuration	0.0016	195.9011	22.3583	2.8032
data	0.0014	676.3825	40.8279	2.2666
external interface	0.0006	1700.5871	69.2463	0.4275
internal interface	0.0029	1688.5275	93.3543	2.4852
logic	0.0016	59.8305	6.8892	1.2537
non-functional	0.0006	1330.4142	47.8057	0.6949
timing/optimization	0.0017	569.7309	71.6649	<u>3.3596</u>
others	0.0005	1.3128	0.3344	0.0594

RQ5 : 修正に要するエフオートはどの程度か？11

- 何回のコミットで修正を完了しているか

Table IX
NUMBERS OF REVISIONS NEEDED TO FIX A BUG

Type	Min	Max	Mean	Median
algorithm/method	1	9	<u>1.9646</u>	1
assignment/initialization	1	4	1.2923	1
checking	1	11	1.7544	1
configuration	1	5	1.7037	1
data	1	5	1.6429	1
external interface	1	6	1.7368	1
internal interface	1	8	1.8947	1
logic	1	4	1.2593	1
non-functional	1	6	1.5641	1
timing/optimization	1	7	<u>2.4167</u>	<u>2</u>
others	1	1	1.0000	1

RQ6 : 修正するファイル数はいくつか？

12

- 修正したファイル数を比較
 - non-functionalでMaxが大きい理由はコーディングスタイルの変更
 - dataでMaxが大きくなった理由はデータ構造の変更
 - timingは70%が2ファイル以上の変更で、最もImpactが大きい

NUMBERS OF IMPACTED FILES PER BUG

Type	Min	Max	Mean	Median
algorithm/method	1	117	11.7434	3
assignment/initialization	1	74	5.2459	2
checking	1	115	7.2982	2
configuration	1	40	5.2692	1
data	1	124	12.3571	3
external interface	1	92	13.3514	3
internal interface	1	52	7.6842	4.5
logic	1	15	2.7778	2
non-functional	1	676	16.7143	2
others	1	47	10.2000	1
timing/optimization	1	69	11.5833	5

- Algorithm/methodに関する不具合
 - 出現頻度が多い
 - 深刻度が高い
 - 修正に時間がかかる
 - 複数回のコミットで修正されやすい
- Timing/optimizationに関する不具合
 - 出現頻度は低い
 - 複数回のコミットで修正されやすい
 - 最も修正時間が長い
 - 修正するファイル数も多い

⚙️ 論文の骨組みは， 参考になる

⚙️ 分類の数が多すぎず， 結果が理解しやすい.