三篇论文

SCIENCE AND TECHNOLOGY

《Deep Code Search》	2018	ICSE
《Dictionary learning based software defect prediction》	2014	ICSE
《It's not a bug, it's a feature: how misclassification impacts bug prediction》	2013	ICSE

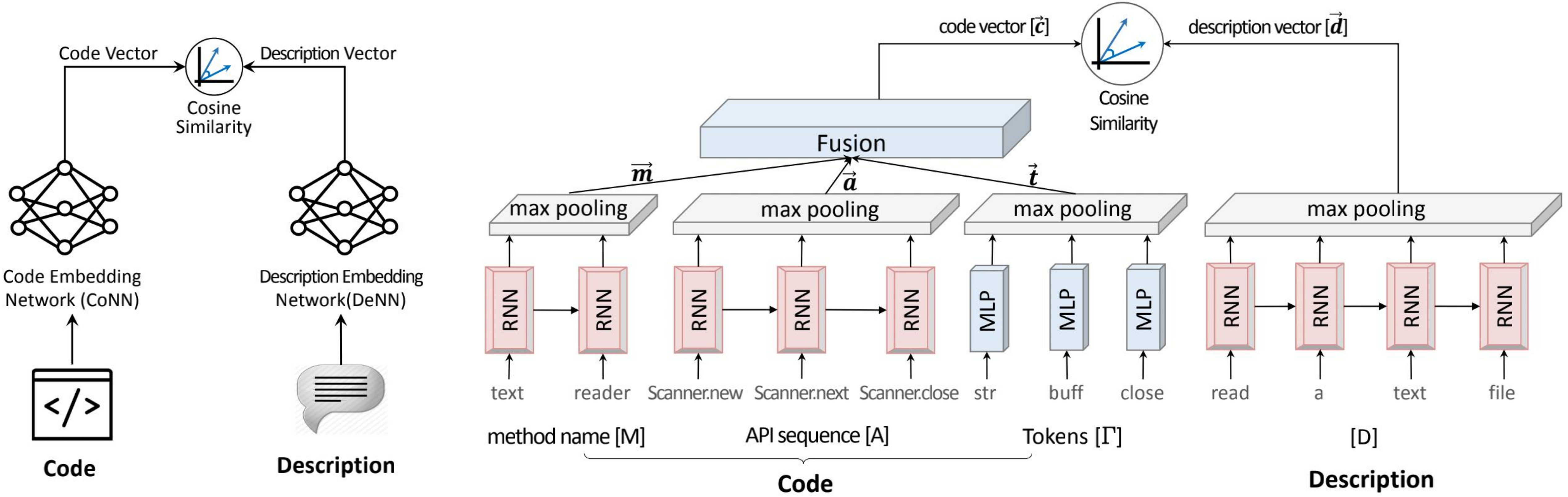
GRADUATION DEFENSE

Deep Code Search

2018 ICSE

CODEnn

<C,D+,D->



(a) Overall Architecture (b) Detailed Structure

Figure 4: The structure of the Code-Description Embedding Neural Network

Deep Code Search

2018 ICSE

Table 1: Benchmark Queries and Evaluation Results (NF: Not Found within the top 10 returned results LC:Lucene CH:CodeHow DCS:DeepCS)

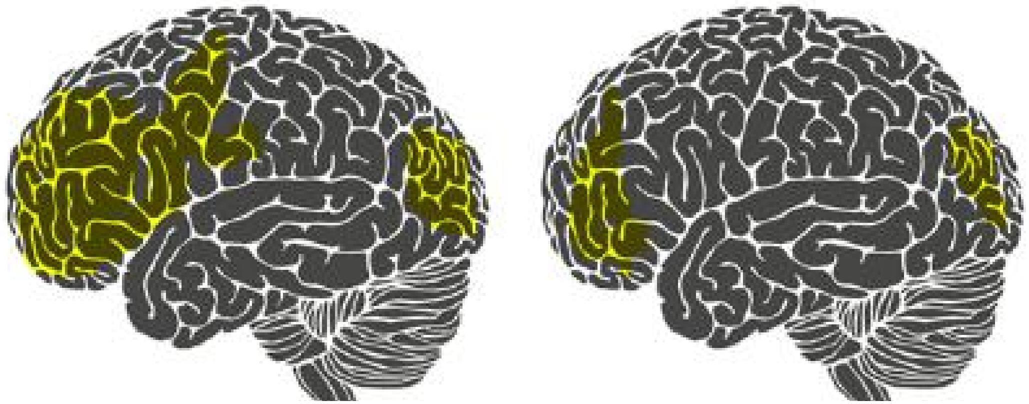
No.	Question ID	Query	FRank LC	FRank CH	FRank DCS
1	309424	convert an inputstream to a string	2	1	1
2	157944	create arraylist from array	NF	NF	2
3	1066589	iterate through a hashmap	NF	4	1
4	363681	generating random integers in a specific range	NF	6	2
5	5585779	converting string to int in java	NF	10	1
6	1005073	initialization of an array in one line	NF	4	1
7	1128723	how can I test if an array contains a certain value	6	6	1
8	604424	lookup enum by string value	1	NF	10
9	886955	breaking out of nested loops in java	NF	NF	NF
10	1200621	how to declare an array	NF	NF	4
11	41107	how to generate a random alpha-numeric string	NF	1	1
12	409784	what is the simplest way to print a java array	6	NF	1
13	109383	sort a map by values	NF	1	3
14	295579	fastest way to determine if an integer's square root is an integer	NF	NF	NF
15	80476	how can I concatenate two arrays in java	NF	1	1
16	326369	how do I create a java string from the contents of a file	8	NF	5
17	1149703	how can I convert a stack trace to a string	3	1	2
18	513832	how do I compare strings in java	1	3	1
19	3481828	how to split a string in java	1	1	1
20	2885173	how to create a file and write to a file in java	2	1	NF
21	507602	how can I initialise a static map	7	1	2
	223918	iterating through a collection, avoiding concurrentmodificationexception when removing in loop	3	3	2
	415953	how can I generate an md5 hash	1	3	6
	1069066	get current stack trace in java	3	1	1
	2784514	sort arraylist of custom objects by property	1	1	1
	153724	how to round a number to n decimal places in java	1	1	4
	473282	how can I pad an integers with zeros on the left	NF	3	1
	529085	how to create a generic array in java	NF	NF	3
	4716503	reading a plain text file in java	4	NF	7
	1104975	a for loop to iterate over enum in java	NF	NF	NF
	3076078	check if at least two out of three booleans are true	NF	NF	NF
	4105331	how do I convert from int to string	2	1	NF
	8172420	how to convert a char to a string in java	5	10	3

Dictionary learning based software defect prediction

2014 ICSE

Dictionary Learning

Dictionary learning (DL) aims to learn from the training samples'space where the given signal could be well represented or coded for processing.



- “以前的知识”，更专业一点，我们称之为**原始样本**，用矩阵$\mathbf{Y}$表示；
- “字典”，我们称之为**字典矩阵**，用$\mathbf{D}$表示，“字典”中的词条，我们称之为**原子（atom）**
- “查字典的方法”，我们称为**稀疏矩阵**，用$\mathbf{X}$；
- “查字典的过程”，我们可以用矩阵的乘法来表示，即$\mathbf{DX}$。

Y=DX

class-imbalance problem

misclassification cost issue

$$\min_{\mathbf{D},\mathbf{X}} \|\mathbf{Y}-\mathbf{DX}\|_F^2, \quad \text{s.t. } \forall i,\ \|\mathbf{x}_i\|_0 \le T_0$$

Dictionary learning based software defect prediction

2014 ICSE

Sparse Representation based Classification (SRC)

1. Sparsely code y over A via l_1-norm minimization

$$\hat{\alpha} = \arg\min_{\alpha} \left\{ \|y - A\alpha\|_2^2 + \gamma \|\alpha\|_1 \right\}. \quad (1)$$

2. Do classification by using

$$\text{identity}(y) = \arg\min_{i} \{e_i\}, \quad (2)$$

$A = [A_1, A_2, \ldots, A_c] \in R^{m \times n}$ training samples (labeled software modules),

$A_i = [s_{i,1}, s_{i,2}, \ldots, s_{i,n_i}] \in R^{m \times n_i}$ the subset of training samples from class i

y testing samples

$e_i = \|y - A_i\alpha_i\|_2, \ \alpha_i = [\alpha_1, \alpha_2, \ldots, \alpha_c]^T$ αi is the coefficient vector associated with class i

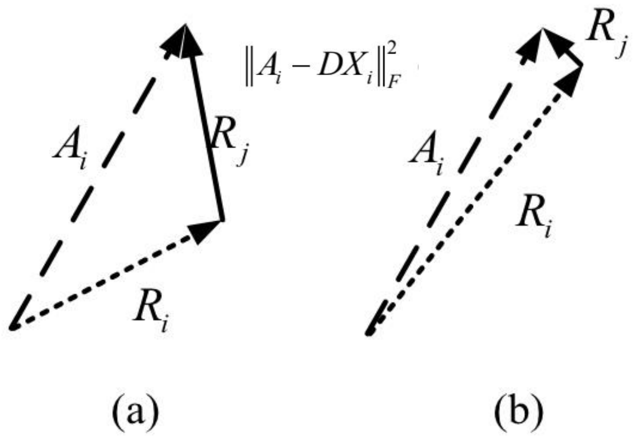


Fig. 2: Illustration of the discriminative fidelity term.

cost-sensitive discriminative

Dictionary learning based software defect prediction

2014 ICSE

Pf: false positive
Pd: recall

Table 6. Experimental results: Pd and Pf comparisons on NASA's ten datasets

Data set	M	SVM	CC4.5	NB	CEL	CBNN	CDDL
CM1	Pd	0.15	0.26	0.44	0.43	0.59	**0.74**
	Pf	0.04	0.11	0.18	0.15	0.29	0.37
JM1	Pd	0.53	0.37	0.14	0.32	0.54	**0.68**
	Pf	0.45	0.17	0.32	0.14	0.29	0.35
KC1	Pd	0.19	0.40	0.31	0.37	0.69	**0.81**
	Pf	0.02	0.12	0.06	0.13	0.30	0.37
KC3	Pd	0.33	0.41	0.46	0.29	0.51	**0.71**
	Pf	0.08	0.16	0.21	0.12	0.25	0.34
MC2	Pd	0.51	0.64	0.35	0.56	0.79	**0.83**
	Pf	0.24	0.49	0.09	0.38	0.54	0.29
MW1	Pd	0.21	0.29	0.49	0.25	0.61	**0.79**
	Pf	0.04	0.09	0.19	0.11	0.25	0.25
PC1	Pd	0.66	0.38	0.36	0.46	0.54	**0.86**
	Pf	0.19	0.09	0.11	0.13	0.17	0.29
PC3	Pd	0.64	0.34	0.28	0.41	0.65	**0.77**
	Pf	0.41	0.08	0.09	0.13	0.25	0.28
PC4	Pd	0.72	0.49	0.39	0.48	0.66	**0.89**
	Pf	0.16	0.07	0.13	0.06	0.18	0.28
PC5	Pd	0.71	0.50	0.32	0.37	0.79	**0.84**
	Pf	0.22	0.02	0.14	0.13	0.08	0.06

It's not a bug, it's a feature: how misclassification impacts bug prediction

2013 ICSE

Issue Report

bug

non-bug

"perfective and adaptive maintenance, refactoring, discussions, requests for help, and so on"

TABLE I
PROJECT DETAILS.

	Maintainer	Tracker type	# reports
HTTPClient	APACHE	Jira	746
Jackrabbit	APACHE	Jira	2,402
Lucene-Java	APACHE	Jira	2,443
Rhino	MOZILLA	Bugzilla	1,226
Tomcat5	APACHE	Bugzilla	584

It's not a bug, it's a feature: how misclassification impacts bug prediction

2013 ICSE

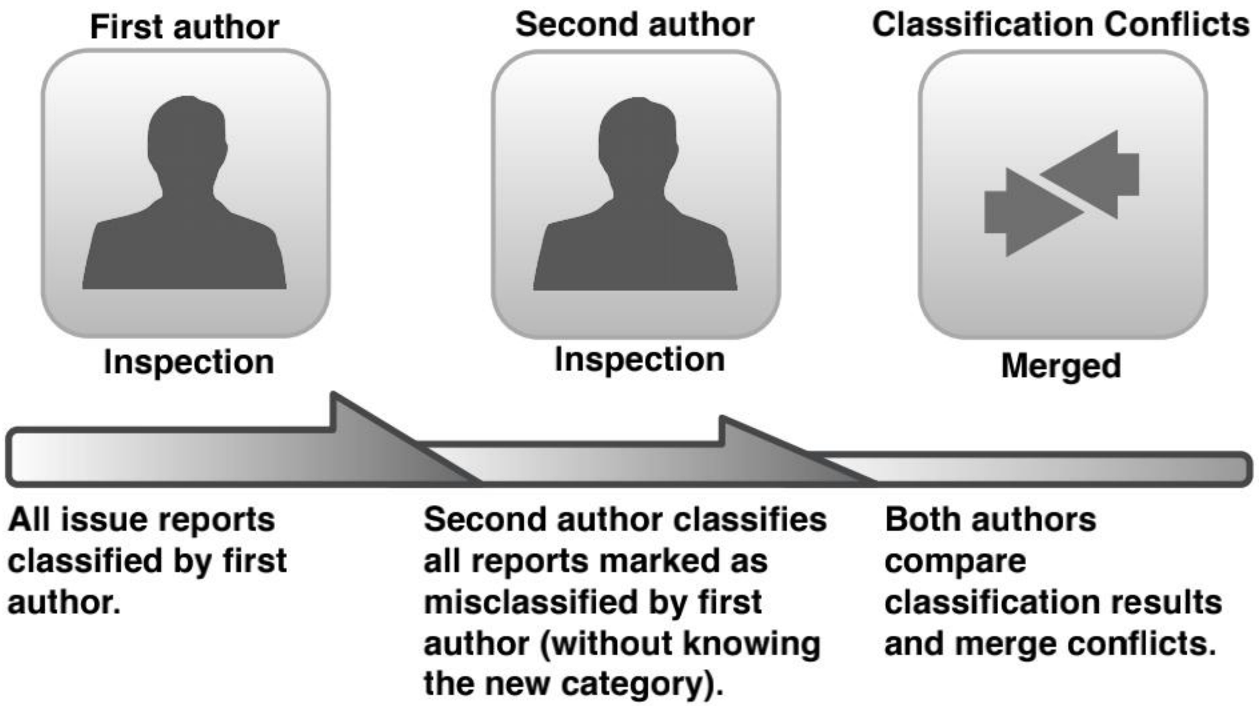


Fig. 1. The manual report inspection process.

TABLE II
THE ISSUE REPORT CATEGORIES USED FOR MANUAL CLASSIFICATION.

Category	Description
BUG	Issue reports documenting corrective maintenance tasks that require semantic changes to source code.
RFE	Issue reports documenting an adaptive maintenance task whose resolving patch(es) implemented new functionality (request for enhancement; feature request).
IMPR	Issue reports documenting a perfective maintenance task whose resolution improved the overall handling or performance of existing functionality.
DOC	Issue reports solved by updating external (e.g. website) or code documentation (e.g. JavaDoc).
REFAC	Issues reports resolved by refactoring source code. Typically, these reports were filed by developers.
OTHER	Any issue report that did not fit into any of the other categories. This includes: reports requesting a backport (**BACKPORT**), code cleanups (**CLEANUP**), changes to specification (rather than documentation or code; **SPEC**), general development tasks (**TASK**), and issues regarding test cases (**TEST**). These subcategories are found in the public dataset accompanying this paper.

It's not a bug, it's a feature: how misclassification impacts bug prediction

2013 ICSE

- RQ1: *Do bug databases contain data noise due to issue report misclassification, and how much?*

Over all five projects researched, we found 42.6% of all issue reports to be wrongly typed.

- RQ2: *Which percentage of issue reports associated with a category was marked as misclassified? Which category do these misclassified reports actually belong to?*

Every third bug report is no bug report.

It's not a bug, it's a feature: how misclassification impacts bug prediction

2013 ICSE

" "

Our motivation for this work was to have a well-classified set of bug reports and features, which we now can leverage (and share) for future research.

TABLE V
RECLASSIFICATION OF REPORTS

(a) Reports originally filed as **BUG**

Classified category	HTTPClient	Jackrabbit	Lucene-Java	Rhino	Tomcat5	combined
BUG	63.5%	75.1%	65.4%	59.2%	61.3%	66.2%
RFE	6.6%	1.9%	4.8%	6.0%	3.1%	3.9%
DOC	8.7%	1.5%	4.8%	0.0%	10.3%	5.1%
IMPR	13.0%	5.9%	7.9%	8.8%	12.0%	9.0%
REFAC	1.7%	0.9%	4.3%	10.2%	0.5%	2.8%
OTHER	6.4%	14.7%	12.7%	15.8%	12.9%	13.0%
Misclassifications	36.5%	24.9%	34.6%	40.8%	38.7%	33.8%

(b) Reports originally filed as **RFE**

Classified category	HTTPClient	Jackrabbit	Lucene-Java	Rhino	Tomcat5	combined
BUG	0.0%	0.7%	0.0%	3.6%	8.1%	2.8%
RFE	100.0%	91.3%	97.0%	42.9%	39.6%	72.6%
DOC	0.0%	2.0%	0.0%	0.0%	18.1%	5.3%
IMPR	0.0%	0.7%	0.6%	19.0%	20.8%	8.6%
REFAC	0.0%	0.0%	0.0%	15.5%	3.4%	3.2%
OTHER	0.0%	5.3%	2.4%	19.0%	10.1%	7.5%
Misclassifications	0.0%	8.6%	3.0%	57.1%	60.4%	24.7%

(c) Reports originally filed as **IMPR**.

Classified category	HTTPClient	Jackrabbit	Lucene-Java	Rhino	Tomcat5	combined
BUG	2.6%	2.8%	1.8%	0.0%	0.0%	2.3%
RFE	45.3%	18.8%	28.6%	0.0%	0.0%	26.1%
DOC	11.6%	3.7%	7.2%	0.0%	0.0%	6.2%
IMPR	26.7%	45.6%	35.2%	0.0%	0.0%	38.8%
REFAC	4.3%	9.2%	14.2%	0.0%	0.0%	10.9%
OTHER	9.5%	19.8%	13.0%	0.0%	0.0%	29.4%
Misclassifications	73.3%	54.4%	64.8%	0.0%	0.0%	61.2%