三篇论文

SCIENCE AND TECHNOLOGY

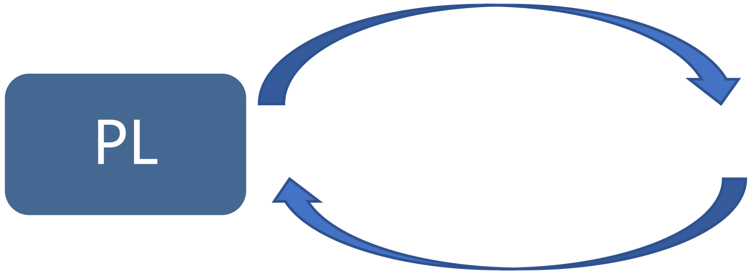
《Unified pre-training for program understanding and generation》	2021	NAACL
《Improving bug localization with word embedding and enhanced convolutional neural networks》	2019	IST
《Personalized Defect Prediction》	2013	ASE

GRADUATION DEFENSE

Unified pre-training for program understanding and generation

2021 NAACL

PLUG
(Program and Language Understanding and Generation)



- Code Summarization
- Code Generation
- Code Translation
- Code Classification

PLBART

PLBART uses the same architecture as BART , it uses the sequence-to-sequence Transformer architecture, with 6 layers of encoder and 6 layers of decoder with model dimension of 768 and 12 heads (~140M parameters).

A bidirectional and autoregressive transformer pre-trained on unlabeled data across PL and NL

Unified pre-training for program understanding and generation

2021 NAACL

- RQ1: *Does PLBART learn strong program and language representations from unlabeled data?*

- RQ2: *Does PLBART learn program characteristics, e.g., syntax, style, and logical data flow?*

- RQ3: *How does PLBART perform in an unseen language with limited annotations?*

Unified pre-training for program understanding and generation

2021 NAACL

Methods	Ruby	Javascript	Go	Python	Java	PHP	Overall
Seq2Seq	9.64	10.21	13.98	15.93	15.09	21.08	14.32
Transformer	11.18	11.59	16.38	15.81	16.26	22.12	15.56
RoBERTa	11.17	11.90	17.72	18.14	16.47	24.02	16.57
CodeBERT	12.16	14.90	18.07	19.06	17.65	**25.16**	17.83
PLBART	**14.11**	**15.56**	**18.91**	**19.30**	**18.45**	23.58	**18.32**

Table 5: Results on source code summarization, evaluated with smoothed BLEU-4 score. The baseline results are reported from Feng et al. (2020).

Improving bug localization with word embedding and enhanced convolutional neural networks

DeepLoc

2019 IST

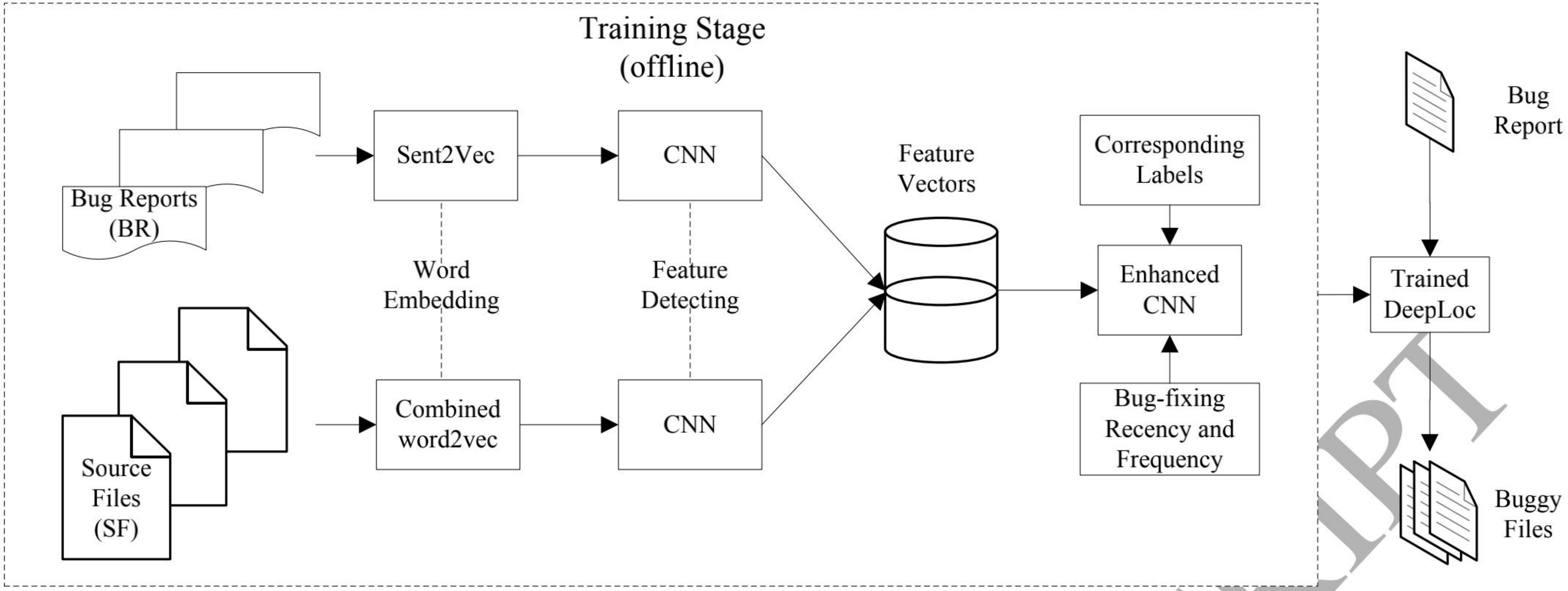


Figure 5: The overall workflow of DeepLoc.

Improving bug localization with word embedding and enhanced convolutional neural networks

- RQ1: *What effect do the model settings have on DeepLoc?*

- RQ2: *What effect does the enhanced CNN have on DeepLoc?*

- RQ3: *How good is DeepLoc's performance compared to state-of-the-art techniques?*

Improving bug localization with word embedding and enhanced convolutional neural networks

2019 IST

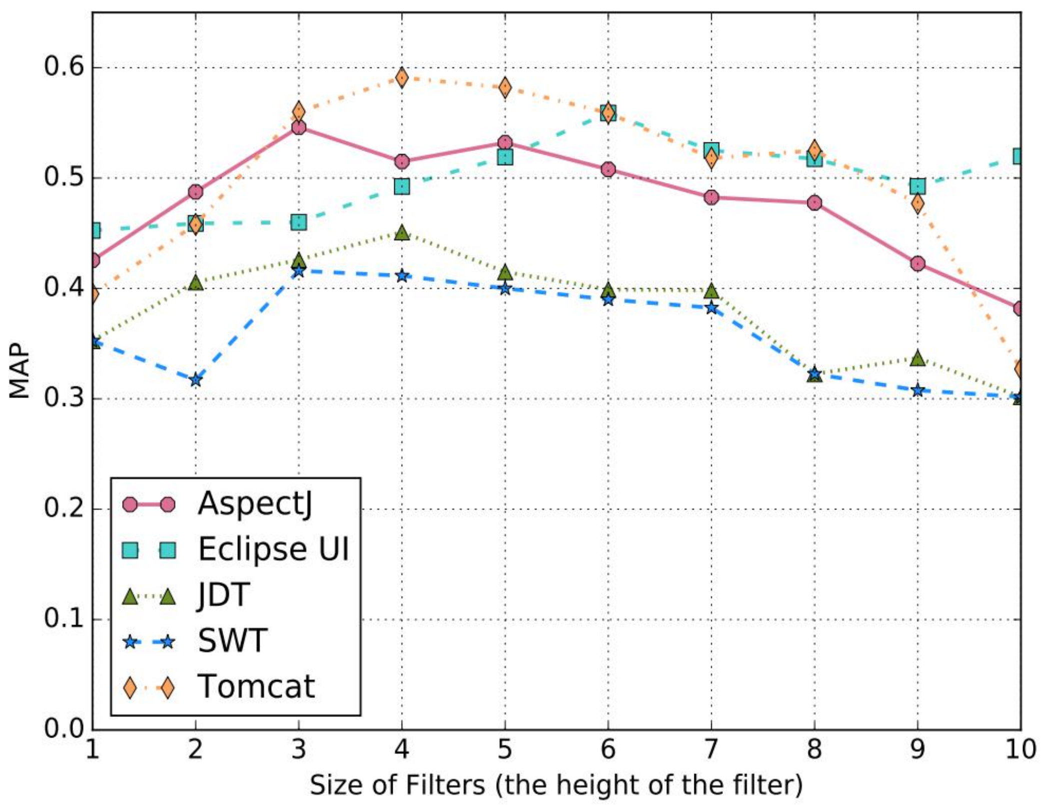


Figure 9: Performance of different filter sizes.

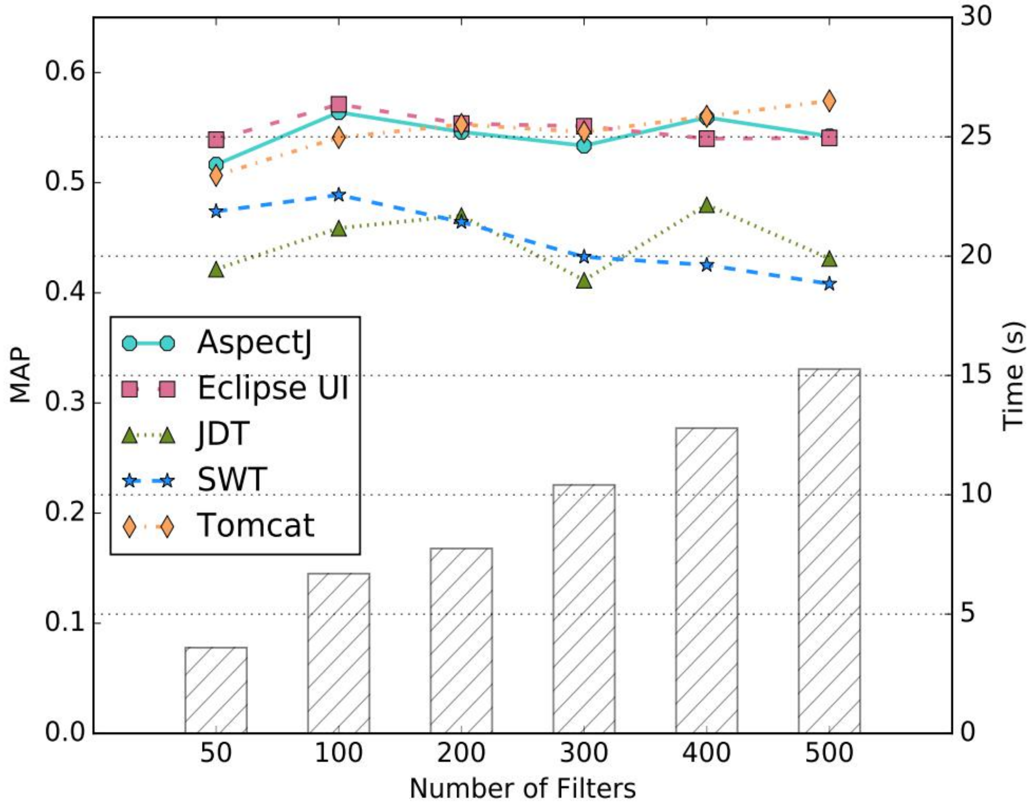


Figure 10: Performance of number of filters.

Improving bug localization with word embedding and enhanced convolutional neural networks

2019 IST

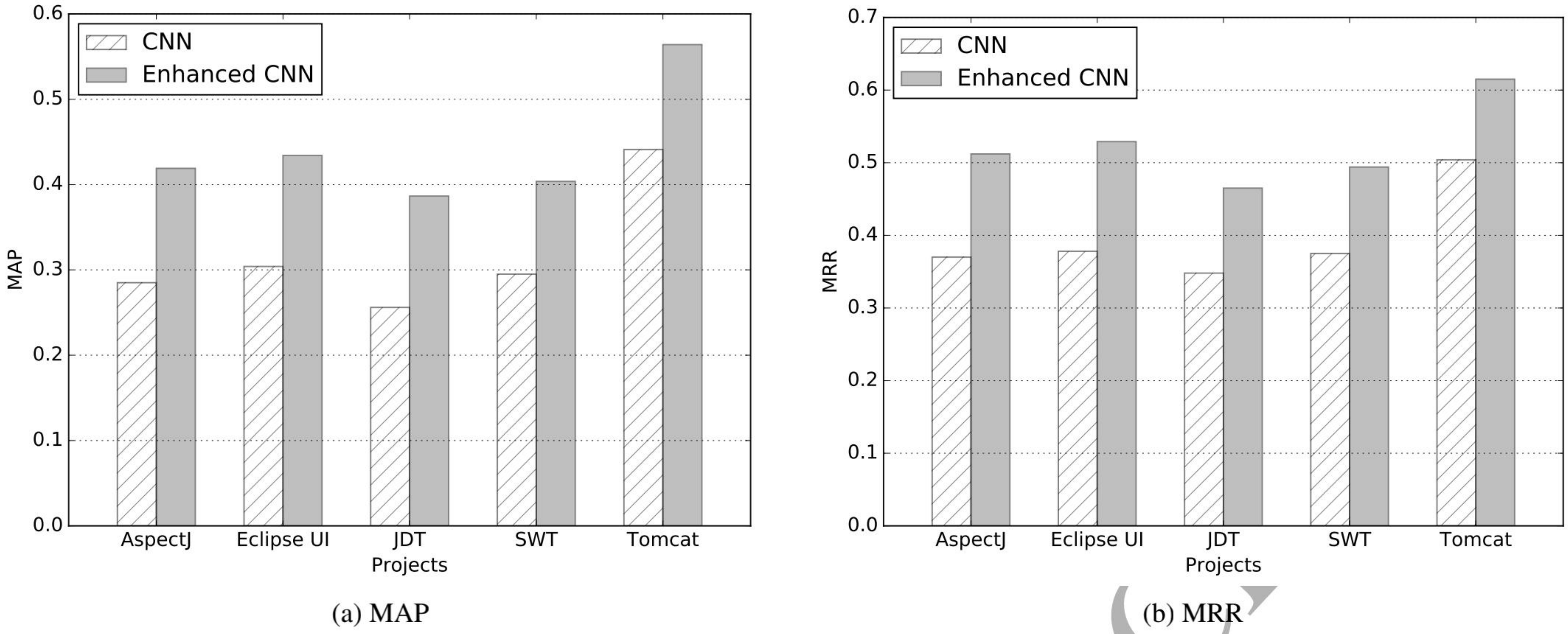


Figure 11: MAP and MRR of conventional CNN and enhanced CNN for five projects.

Improving bug localization with word embedding and enhanced convolutional neural networks

2019 IST

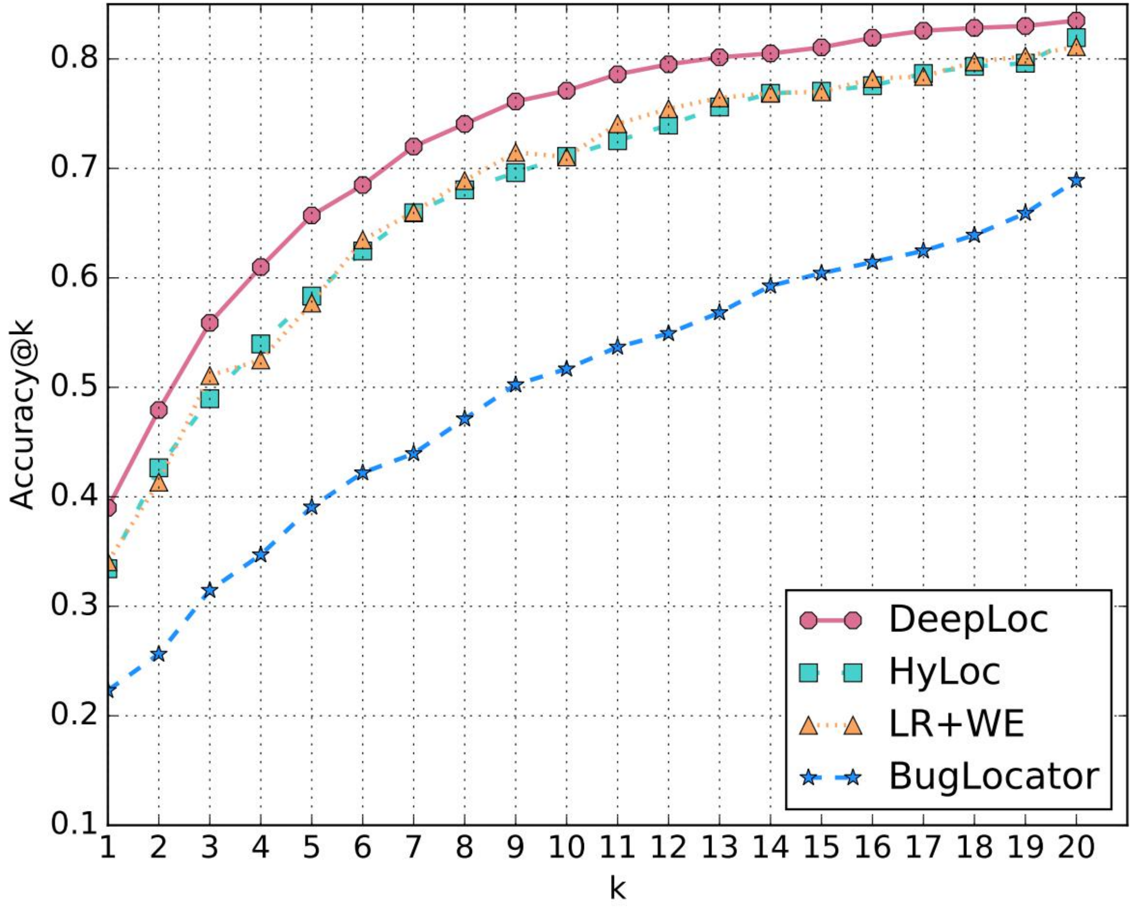


Figure 12: Accuracy@k of the four models on Project SWT.

Personalized Defect Prediction

Author: Tian Jiang; Lin Tan; Sunghun Kim

汇报人：陈冰婷

导师：邹卫琴

GRADUATION DEFENSE

Content

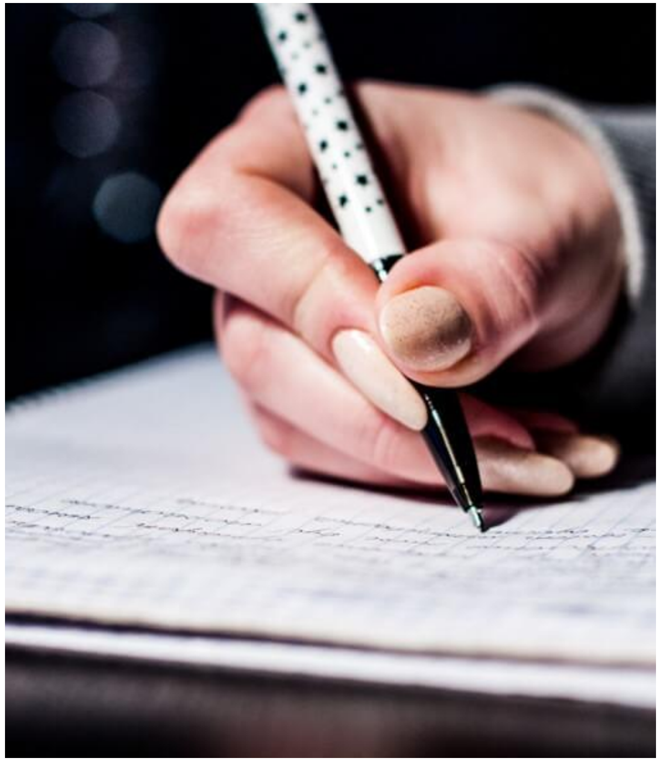
SCIENCE AND TECHNOLOGY

GRADUATION DEFENSE

Introduction

SCIENCE AND TECHNOLOGY

*While many of the previous defect prediction studies take the author of the code into consideration, none of these studies **build separate prediction models for individual developers.** They combine all developers' changes to build a single prediction model.*



Introduction

SCIENCE AND TECHNOLOGY

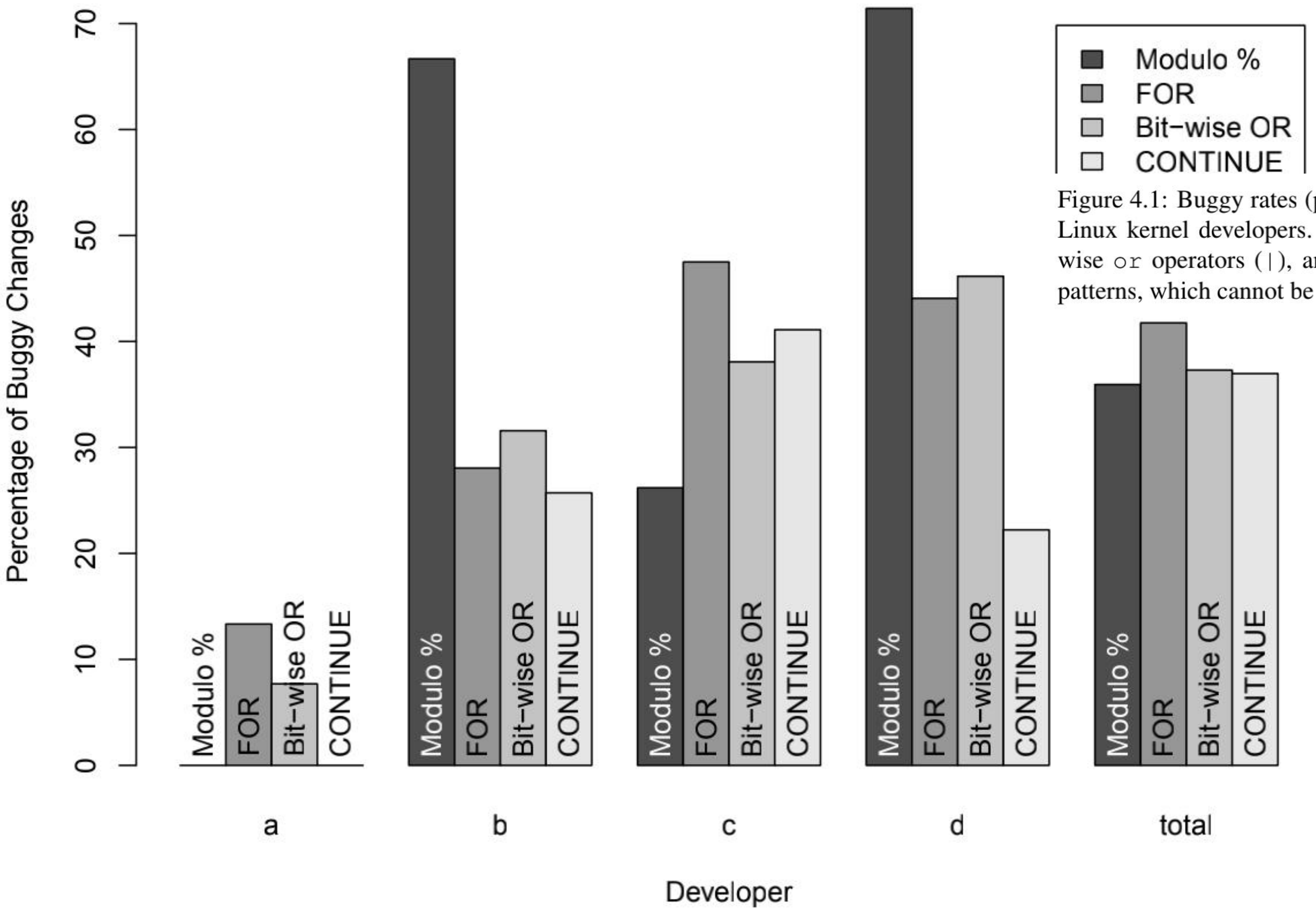


Figure 4.1: Buggy rates (percentages of buggy changes) of different syntactic structures of four Linux kernel developers. The syntactic structures are modulo operators (`%`), `for` loops, bit-wise `or` operators (`|`), and `continue` statements. Developers have different buggy change patterns, which cannot be observed if we combine different developers' changes ("total").

CC (Change Classification)

1. Label each change clean or buggy by mining the project's revision history.
2. Extract features.
3. Use a classification algorithm to build a model from the labelled changes based on the extracted features.
4. Predict new changes as buggy or clean using the model.

Implement

SCIENCE AND TECHNOLOGY

Feature Exaction

- ***Characteristic Vector***
- Bag-of-Words
- Meta-Data

developer, commit hour (0, 1, 2, ..., 23), commit day (Sunday, Monday, ..., Saturday), cumulative change count, cumulative buggy change count, source code file/path names, and file age in days in a way similar to that of Kim et al.

```
// Sum up positive entries
for (int i = 0; i < array.length; ++i) {
  for (int j = 0; j < array[i].length; ++j) {
    if (array[i][j] > 0) sum += array[i][j];
  }
}
```

Characteristic Vector: (1, 2, 0)

Figure 3.1: The characteristic vector for the above code segment contains one `if` statement, two `for` loops and zero `while` loops.

Implement

SCIENCE AND TECHNOLOGY

PCC

"

*Note that **the idea behind PCC is not adding the feature—developer—as an advantage over CC, but instead building separate models for different developers.***

"

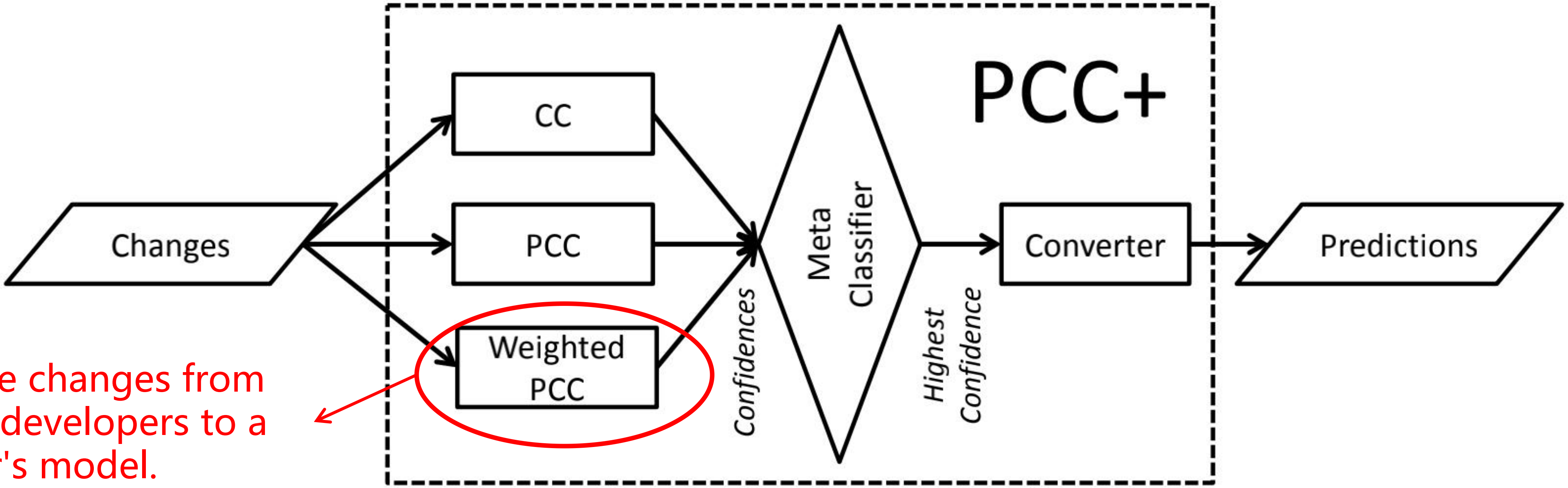
PCC follows exactly the same steps as CC except that PCC groups changes by developers and builds a separate model for each developer.

Given a new unlabelled change, the final model first checks the author of the change, and then uses this developer's model to predict this change.

Implement

SCIENCE AND TECHNOLOGY

PCC+



adding the changes from the other developers to a developer's model.

Figure 5.1: The flow diagram of PCC+. First, CC, PCC and weighted PCC predict the changes and pass the confidences to the meta classifier. Then, the meta-classifier picks the highest confidence. Last, the converter converts the highest confidence to a prediction (buggy or clean).

Implement

SCIENCE AND TECHNOLOGY

Dataset

Project	Language	LOC	First Commit Date	Last Commit Date	# of Changes	% of Buggy Changes
Linux	C	7.3M	2005-04-16	2010-11-21	429K	14.0%
PostgreSQL	C	289K	1996-07-09	2011-01-25	89K	23.6%
Xorg[a]	C	1.1M	1999-11-19	2012-06-28	46K	12.9%
Eclipse[b]	Java	1.5M	2001-06-05	2012-07-24	73K	16.9%
Lucene	Java	828K	2010-03-17	2013-01-16	76K	9.4%
Jackrabbit	Java	589K	2004-09-13	2013-01-14	61K	23.6%

Baseline

- CC
- Mars

Measure

- Cost Effectiveness：NofB20, PofB20
- F1

Evaluation

SCIENCE AND TECHNOLOGY

- RQ1: *How much do PCC and PCC+ improve the classification performance over CC and MARS?*

Project	Method	P	R	F1	NofB20	PofB20
Linux	CC	0.59	0.49	0.54	160	0.51
	MARS	0.46	0.39	0.42	121	0.39
	PCC	0.61	0.50	0.55(+0.01)	179(+**19**)	0.57(+**0.06**)
	PCC+	0.62	0.49	0.55(+0.01)	172(+**12**)	0.55(+**0.04**)
PostgreSQL	CC	0.65	0.58	0.61	55	0.08
	MARS	0.60	0.55	0.57	76	0.11
	PCC	0.63	0.58	0.60(−0.01)	210(+**155**)	0.29(+**0.21**)
	PCC+	0.66	0.59	0.63(+0.02)	175(+**120**)	0.24(+**0.16**)
Xorg	CC	0.69	0.62	0.65	96	0.23
	MARS	0.65	0.52	0.57	152	0.37
	PCC	0.69	0.66	0.67(+**0.02**)	159(+**63**)	0.39(+**0.16**)
	PCC+	0.73	0.66	0.69(+**0.04**)	161(+**65**)	0.39(+**0.16**)
Eclipse	CC	0.59	0.48	0.53	116	0.20
	MARS	0.55	0.43	0.48	20	0.03
	PCC	0.63	0.55	0.59(+**0.06**)	207(+**91**)	0.36(+**0.16**)
	PCC+	0.68	0.56	0.61(+**0.08**)	200(+**84**)	0.35(+**0.15**)
Lucene	CC	0.58	0.46	0.51	176	0.28
	MARS	0.51	0.41	0.45	131	0.21
	PCC	0.60	0.53	0.56(+**0.05**)	254(+**78**)	0.40(+**0.12**)
	PCC+	0.64	0.54	0.59(+**0.08**)	258(+**82**)	0.41(+**0.13**)
Jackrabbit	CC	0.72	0.72	0.72	411	0.37
	MARS	0.72	0.70	0.71	411	0.37
	PCC	0.72	0.72	0.72(+0.00)	449(+**38**)	0.40(+**0.03**)
	PCC+	0.74	0.74	0.74(+**0.02**)	459(+**48**)	0.41(+**0.04**)
Average				(+**0.03**)	(+**74**)	(+**0.12**)

Table 7.1: Results of evaluated projects. The values in parentheses show the F1, NofB20 and PofB20 differences against CC. The "Average" row contains the average improvement of PCC over CC across all projects. Statistically significant improvements are bolded.

Evaluation

SCIENCE AND TECHNOLOGY

- RQ2: *Is PCC's performance gain over CC generalizable to other experimental setups?*

Project	Approach	F1			NofB20		
		ADTree	N. B.	L. R.	ADTree	N. B.	L. R.
Linux	CC	0.54	0.39	0.39	160	138	102
	PCC	0.55	0.40	0.49	179	147	137
	Delta	+0.01	**+0.01**	**+0.10**	**+19**	**+9**	**+35**
PostgreSQL	CC	0.61	0.51	0.56	55	89	46
	PCC	0.60	0.52	0.56	210	113	56
	Delta	−0.01	+0.01	+0.00	**+155**	**+24**	+10
Xorg	CC	0.65	0.55	0.63	96	84	52
	PCC	0.67	0.60	0.65	159	101	29
	Delta	**+0.02**	**+0.05**	**+0.02**	**+63**	**+17**	**−23**
Eclipse	CC	0.53	0.43	0.53	116	65	54
	PCC	0.59	0.47	0.51	207	108	55
	Delta	**+0.06**	**+0.04**	**−0.02**	**+91**	**+43**	+1
Lucene	CC	0.51	0.42	0.44	176	152	30
	PCC	0.56	0.45	0.50	254	139	200
	Delta	**+0.05**	**+0.03**	**+0.06**	**+78**	−13	**+170**
Jackrabbit	CC	0.72	0.56	0.72	411	420	261
	PCC	0.72	0.66	0.68	449	414	370
	Delta	+0.00	**+0.10**	**−0.04**	**+38**	−6	**+109**
Average	Delta	**+0.03**	**+0.04**	**+0.02**	**+74**	**+12**	**+50**

Table 7.2: F1 and NofB20 for different classifiers. The delta between CC and PCC are shown in the "Delta" row. The "Average" row contains the average delta between PCC and CC across all projects for each classification algorithm. For simplicity, the p-values are not shown. Instead, statistically significant deltas are bolded.

Evaluation

SCIENCE AND TECHNOLOGY

- RQ3: *What is the difference between using different combinations of feature classes?*

Project	Approach	F1 M	F1 MB	F1 MC	F1 MBC
Linux	CC	0.56	0.52(−**0.04**)	0.55(−0.01)	0.54(−0.02)
	PCC	0.58	0.56(−**0.02**)	0.58(+0.00)	0.55(−**0.03**)
PostgreSQL	CC	0.62	0.61(−0.01)	0.61(−**0.01**)	0.61(−0.01)
	PCC	0.61	0.60(−**0.01**)	0.62(+0.01)	0.60(−0.01)
Xorg	CC	0.65	0.65(+0.00)	0.63(−**0.03**)	0.65(+0.00)
	PCC	0.66	0.67(+0.01)	0.66(+0.00)	0.67(+0.01)
Eclipse	CC	0.54	0.55(+0.01)	0.54(+0.00)	0.53(−0.01)
	PCC	0.57	0.59(+**0.02**)	0.58(+0.01)	0.59(+**0.02**)
Lucene	CC	0.53	0.51(−**0.02**)	0.51(−**0.02**)	0.51(−**0.02**)
	PCC	0.60	0.57(−**0.03**)	0.59(−0.01)	0.56(−**0.04**)
Jackrabbit	CC	0.71	0.72(+0.01)	0.70(−0.01)	0.72(+0.01)
	PCC	0.74	0.72(−**0.02**)	0.71(−**0.03**)	0.72(−**0.02**)
Average	CC	-	(−**0.01**)	(−0.01)	(−**0.01**)
	PCC	-	(−0.01)	(−**0.01**)	(−**0.01**)

Table 7.3: The effect of different classes of features on F1. M represents meta-data features. B represents bag-of-words features. C represents characteristic vector features. The values in parentheses show the deltas against the predictions using only meta-data features. For simplicity, the p-values are not shown. Instead, statistically significant deltas are bolded.

南京航空航天大学
Nanjing University of Aeronautics and Astronautics

2022

THANK YOU

THNAK YOU

GRADUATION DEFENSE