DeepBugs for Python:

Name-Based Bug Detection using Neural Networks

22/08/20

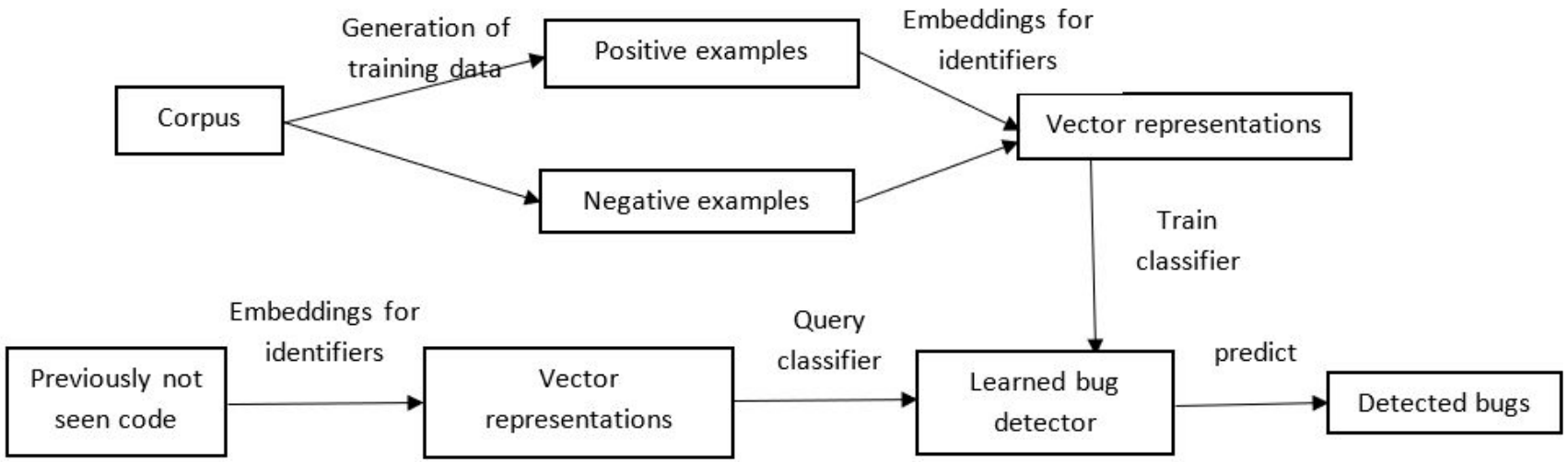
Presented by:
Manasa Sandhya Gunda
Sai Teja Ponugoti

UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

Different Approaches

- Static Analysis Tools
- Name Based Bug Detectors
  - Use Heuristics
  - Hard coded rules

Implementation



BUG DETECTORS

- Swapped Function Arguments

$$x_{pos} = (base, callee, arg1, agr2, argtype1, argtype2, param1, param2)$$

$$x_{neg} = (base, callee, arg2, agr1, argtype2, argtype1, param1, param2)$$

- Incorrect Binary Operator

$$x_{pos} = (left, right, op, lefttype, rightype, parent, grandP)$$

$$x_{neg} = (left, right, op', lefttype, rightype, parent, grandP)$$

- Incorrect Operand in Binary Operation

$$x_{pos} = (left, right, op, lefttype, rightype, parent, grandP)$$

$$x_{neg} = (left', right, op, lefttype', rightype, parent, grandP)$$

or

$$x_{neg} = (left, right', op, lefttype, rightype', parent, grandP)$$

DATA GENERATED

Bug detector	Training examples	Validation examples
Swapped Arguments	196996	91344
Incorrect Binary Operator	231550	98570
Incorrect Binary Operand	133800	52576

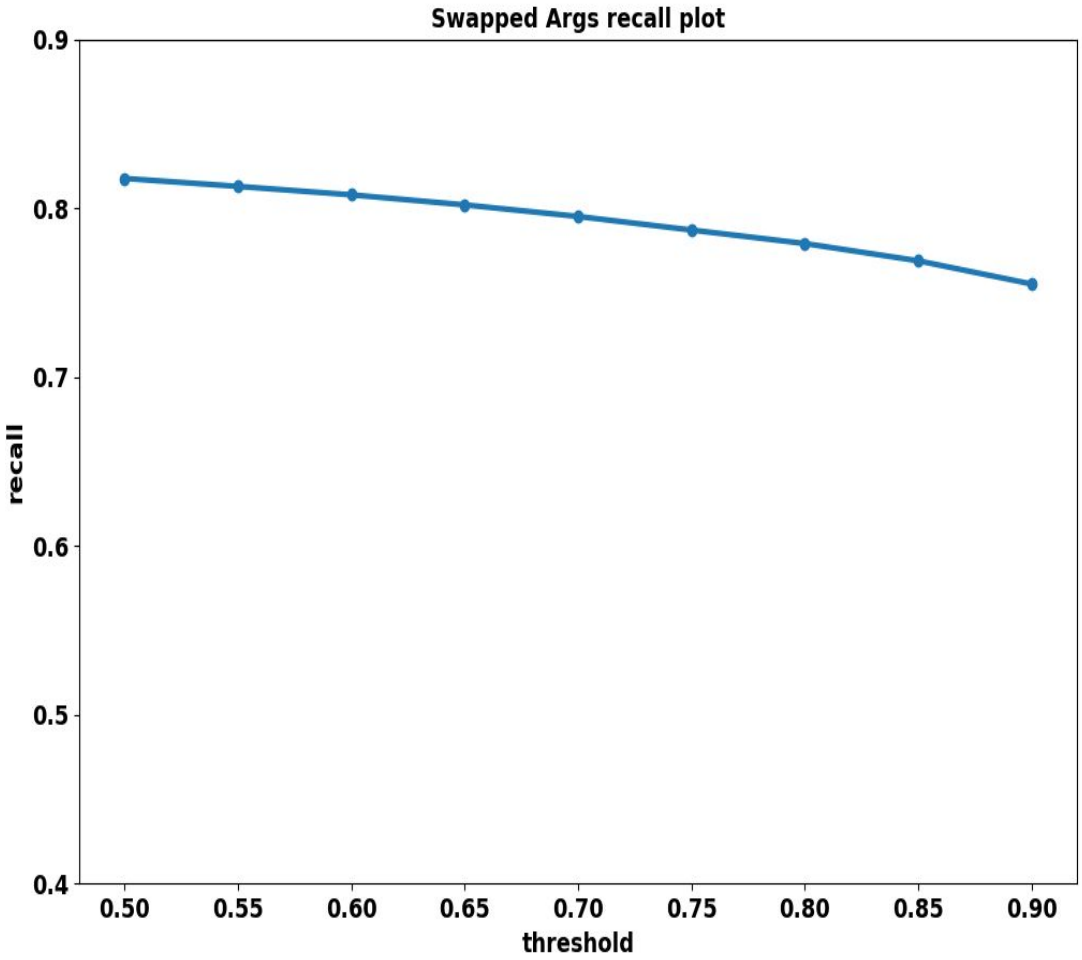
RESULTS

TPR AND ACCURACY

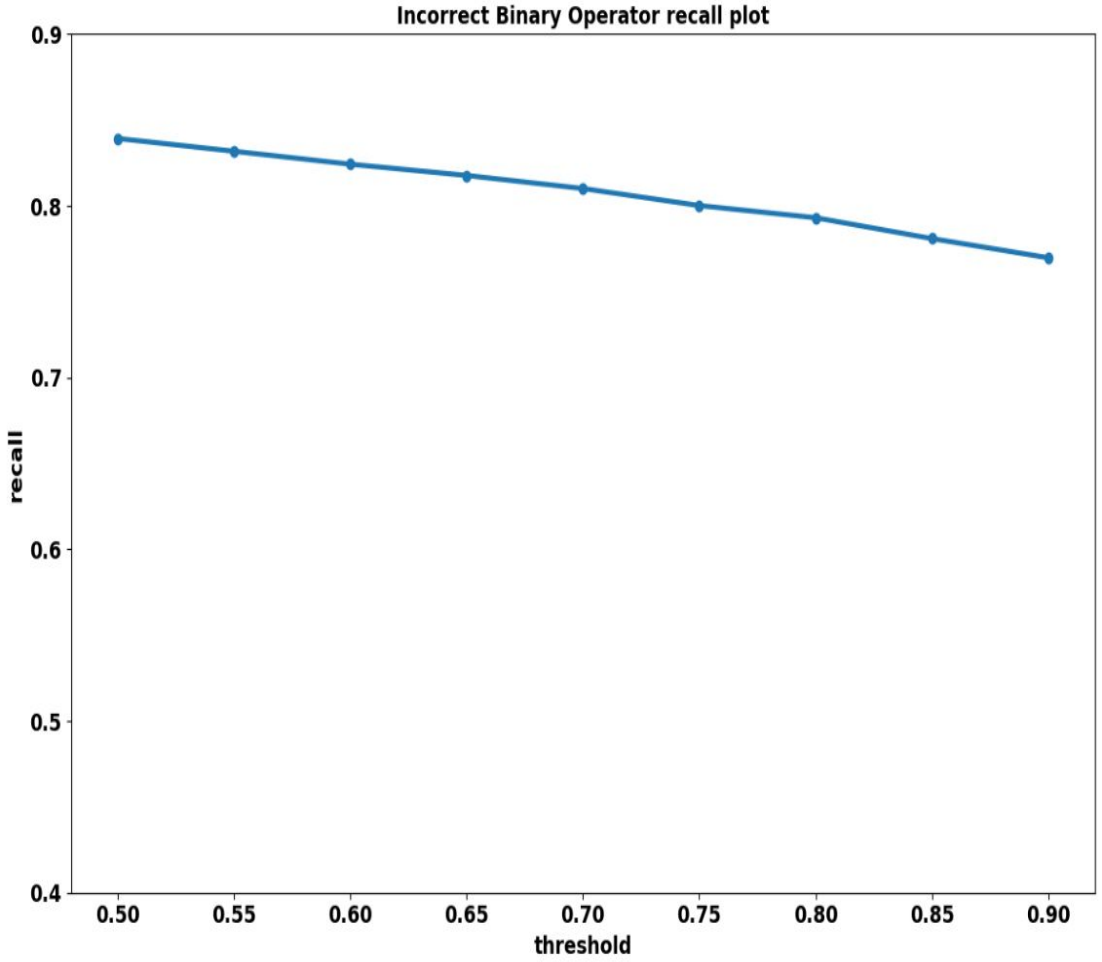
Bug detector	TPR(Recall)	Training accuracy	Validation Accuracy	Validation accuracy for positive samples	Validation accuracy for negative samples
Swapped Arguments	81.76	96.25	87.81	87.78	81.76
Incorrect Binary Operator	88.60	93.89	89.79	90.97	88.60
Incorrect Binary Operand	84.36	90.57	82.93	81.49	84.36

- **Validation accuracy for positive samples** = correctly predicted positive samples/total positive samples
- **Validation accuracy for negative samples** =correctly predicted negative samples /total negative samples

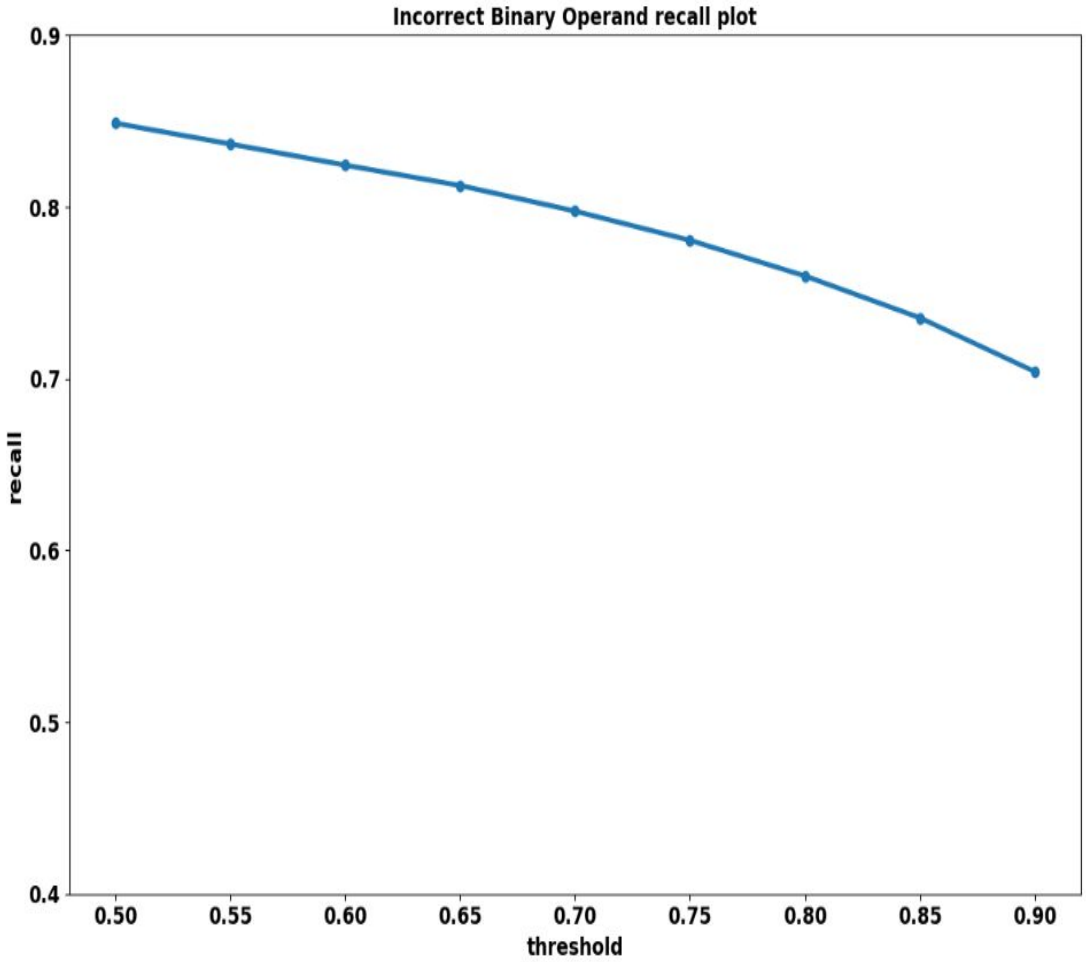
RECALL vs THRESHOLD (Swapped Function Arguments)



RECALL vs THRESHOLD [Incorrect Binary Operator]



RECALL vs THRESHOLD [Incorrect Binary Operand]



Bug detector	True Positives (TPR)	False Positives (FPR)	True Negatives (TNR)	False Negatives (FNR)	Total validation examples
Swapped Arguments	40224 (81.76%)	5580 (12.22%)	40092 (87.78%)	5448 (18.24%)	91344
Incorrect Binary Operator	43668 (88.60%)	4448 (9.03%)	44837 (90.97%)	5617 (11.4%)	98570
Incorrect Binary Operand	22177 (84.36%)	4866 (18.51%)	21422 (81.49%)	4111 (15.64%)	52576

- True positive rate (TPR) : TP/(TP+FN)
- False negative rate (FNR) : 1-TPR
- True negative rate (TNR) : TN/(TN+FP)
- False positive rate (FPR) : 1-TNR

EFFICIENCY OF BUG DETECTORS (time in seconds)

Bug detector	Training Data Extract (in sec) for 100k files	Training Model (in seconds) (no.of examples)	Validation Data Extract (in sec) for 50k files	Validation Predict (in sec) (no.of examples)
Swapped Arguments	815	1044 (196996)	428	29 (91344)
Incorrect Binary Operator	316	1195 (231550)	144	20 (98570)
Incorrect Binary Operand	316	705 (133800)	144	6 (52576)

- **Train examples extract Time:** time to tokenize, traverse through AST and extract training examples from each file.
  - Number of files : 100k python code files
  - Average file size : 8.6 KB
- **Training Model time :** time taken to train Neural Network model with layers each of 1500 neurons for 10 epochs on extracted trained examples. (number of examples mentioned in brackets)
- **Validation Data extract :** Similar to train sample extraction procedure.
  - Number of files : 50k python code files
  - Average files size : 8.6 KB
- **Validation prediction time :** Time taken to predict the validation samples(number of examples mentioned in brackets).

➢ Specs of system used : Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz, 2208 Mhz, 6 Core(s), 12 Logical Processor(s), 16GB of memory, an Intel(R) UHD Graphics 630 and a NVIDIA GeForce GTX 1050 Ti.

Conclusion

- Addresses the idea of identifying bugs based on names
- All the three bug detectors have a testing accuracy in between 82% to 90%.
- Using this framework for building bug detectors reduces human intervention for designing heuristics and hard coded rules for developing bug detectors.

Future works

- Comparison with static type tools.
- Further research can be done by using other Deep Learning models like CNN, LSTM, RNN which works very efficiently for Natural Language processing tasks.
- Implementing other types of name-based bugs like incorrect assignment, missing arguments to functions call.

Thank you