ErrorDeep: Using an Artificial Neural Network to Detect Syntax Errors and Suggest a Fix

Dhvani Patel, Eddie Santos, Joshua Campbell, Dr. Abram Hindle
University of Alberta, Dept. of Computing Science, Software Engineering Lab

Introduction

- Current tools for interpreting human-written source code (parsers) provide misleading information given an unexpected input.
- Novice programmers thus find it difficult to use the information provided by the parsers to fix their broken code [1], [2], [3].

Purpose

- Provide the location (line number) and fix token given a source file with a single token syntax error
- Use an artificial neural network to automatically find the error

Method

- In order to train the neural network, we needed a large dataset of positive and negative samples.

Collect ~450,000 syntactically-valid JavaScript files

Files were tokenized and standardized to a set vocabulary of 88 tokens.

Chosen sample size for training: 2000
Token Batch Size: 66
Window Size: 10

Method Continued

- Wrote script to mutate the sample size
- 3 types of mutations were created.

Total Training Data: *8000 files*

Input: *Window with 10 Tokens*

if	(	Identifier	==	Identifier	{	Identifier	=	True	;

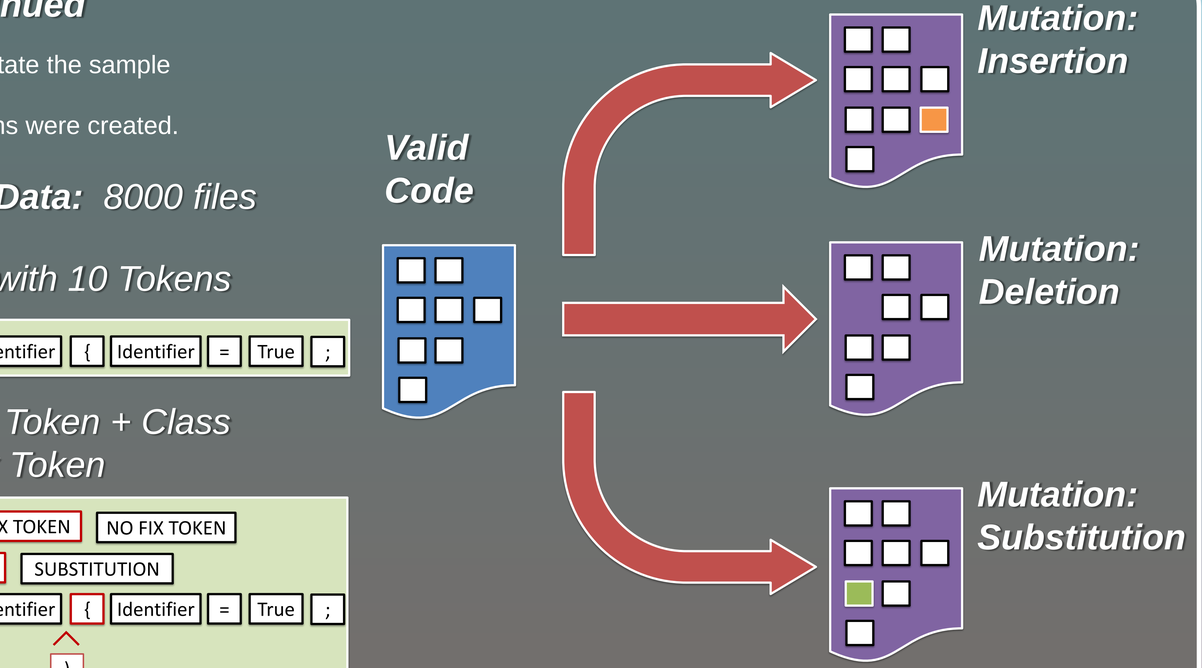
Output: *Error + Token + Class + Location + Fix Token*

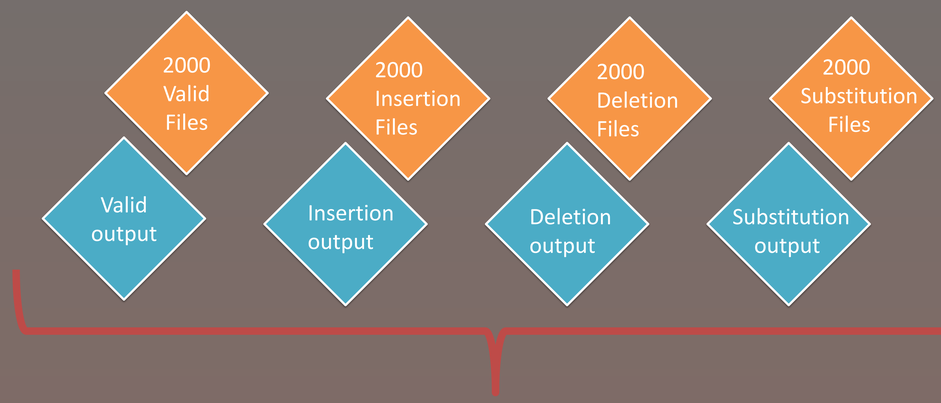
ERROR	NO ERROR	FIX TOKEN	NO FIX TOKEN

INSERTION	DELETION	SUBSTITUTION

if	(	Identifier	==	Identifier	{	Identifier	=	True	;

)





Keras[4] Model Architecture:

Input: One-hot matrix

Layers:

Type	Output dim.	Activation
Dense	4	ReLU
Flatten		
Dense	128	ReLU
Dense	128	ReLU
Dropout		Rate: 0.5
Dense	4	ReLU
Dropout		Rate: 0.5
Activation		softmax

Backend: Theano [5]

Output: Categorical distribution

Loss: Categorical cross-entropy

Optimizer: RMSprop[6], initial learning rate = 0.001, momentum = 0.3

Test Model
Takes future, unknown data and produces the same output format as training.

Results

Conclusions

- Mutations created randomly do not truly represent real errors.
- Machine learning can take a long time to train.
- The model needs more training data and neurons to better learn the language.
- Learning about different types of mistakes will allow for future improvement in teaching programming to others.

References

[1] E. S. Tabanao, M. M. T. Rodrigo, and M. C. Jadud, *"Identifying at*-risk novice *Java programmers through the analysis of online protocols," in* Philippine Computing Science Congress, 2008.
[2] M. C. Jadud, *"A first look at novice compilation behaviour using* bluej," Computer Science Education, vol. 15, no. 1, pp. 25–40, 2005.
[3] E. S. Tabanao, M. M. T. Rodrigo, and M. C. Jadud, *"Predicting at*-risk *novice Java programmers through the analysis of online protocols," in* Proceedings of the seventh international workshop on Computing *education research, ser. ICER '11. New York, NY, USA: ACM, 2011, pp.* 85–92. [Online]. Available: http://doi.acm.org/10.1145/2016911.2016930
[4] *F. Chollet, "Keras," https://github.com/fchollet/keras,* 2015
[5] Theano *Development Team, "*Theano: A Python framework for fast *computation of mathematical expressions," May 2016, preprint. [Online].* Available: http://arxiv.org/abs/1605.02688
[6] T. Tieleman *and G. Hinton, "*RMSprop *gradient optimization," April 2014,* course slides. [Online]. Available: http://www.cs.toronto.edu/ ~tijmen/csc321/slides/lecture slides lec6.pdf

Acknowledgements

- I would like to thank my lab supervisors: Eddie Santos and Joshua Campbell, and my supervisor Dr. Abram Hindle for their support throughout my internship.
- Thank you to the Software Engineering lab and the High School Internship Program (HIP) for presenting this opportunity.