Applying Statistical Language Modeling to Genetic Programming

H. Chase Stevens
s1107496

Supervisor: Dr. I. Stark
University of Edinburgh

M.Sc. Project Proposal

April 2016

1 Introduction

2 Background and related work

2.1 Application of language modeling to formal languages

2.2 Genetic programming

3 Methodology

3.1 Compilation of Python corpus

The creation of a language model for Python will require a large and representative corpus of Python source code. While previous work has employed small collections of hand-chosen open-source Python projects for use as a corpus [1] [2], and other work has resulted in the compilation of large corpora for languages such as Java [3], there does not appear to be a suitably large Python source code corpus publicly available for use in this project at present; ergo, the first objective of this project will be to compile one. To do so, source code will be downloaded from open-source projects hosted on the popular code repository host GitHub.

Ideally, the Python corpus compiled for this project will consist of idiomatic, useful, and well-written Python source code; for this project, good-quality code is of special importance, as inexecutable code may still be syntactically valid and parsable. In compiling a similar corpus for the Java programming language, Allamanis and Sutton [3] sought to maintain a minimum level of quality in the GitHub repositories used by filtering out those that had not been forked at least once; in this project, for which I am planning to create a smaller corpus, repositories will instead be used if they exceed some minimum number of stars or forks. To find repositories matching these criteria, the GitHub search API will be used, which allows filtering on not only number of stars and forks but also on project language.

3.2 Creation of language model from Python corpus

3.3 Application of language model to genetic programming system

4 Evaluation

In the literature, many genetic programming approaches are evaluated using grammars tailored to particular domains [4]. However, in this project, by necessity, the genetic programming system to be developed will be operating on the general grammar of the Python programming language. Therefore, a suitable task will

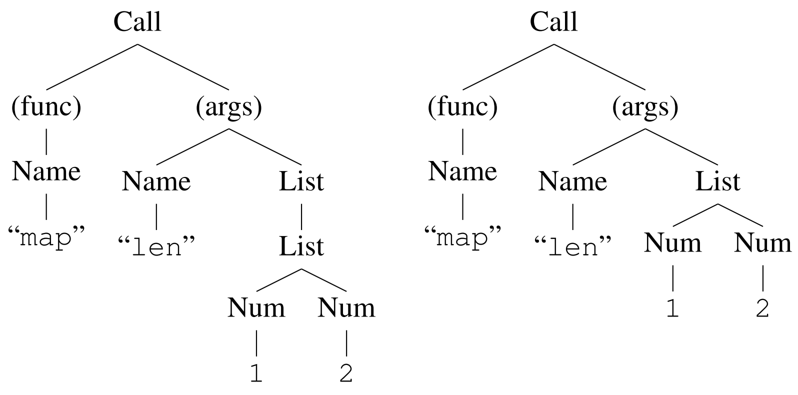

Figure 1: *Comparison of the respective Python AST representations of the semantically valid expression "`map(len, [[1, 2]])`" (left) and the invalid expression "`map(len, [1, 2])`" (right). Note that in both cases the direct descendant node types of "Call" (i.e. Name, Name, and List) are identical.*

be one in which solutions lend themselves to representation and manipulation as Python ASTs.

While, often, genetic programming is used to generate code from scratch, recent work has successfully applied genetic programming to the automated repair of preexisting programs [5]. As in this case manipulation is done on the program's AST, this task would be an ideal evaluative measure for the proposed genetic programming system. Although prior work in this domain has been evaluated on snapshots of open source software during instances in which the software failed to pass a suite of automated tests, given the scope of this project, evaluation will instead either be performed on test-compliant software which has had bugs deliberately introduced through random AST mutation, or on hand-written examples, dependent on the feasibility of the former given the project's time constraints.

The primary means of evaluating the proposed genetic programming system will be demonstrating how many (if any) software repair tasks it is able to solve. Time permitting, the system's performance, both in terms of number of solutions found and speed with which solutions are identified, will be compared against a baseline genetic programming system which does not incorporate an initial language model as induced using the Python corpus.

5 Timeline

6 Conclusions

References

[1] Z. Tu, Z. Su, and P. Devanbu, "On the localness of software," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2014, pp. 269–280.

[2] A. Caliskan-Islam, R. Harang, A. Liu, A. Narayanan, C. Voss, F. Yamaguchi, and R. Greenstadt, "De-anonymizing programmers via code stylometry," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 255–270.

[3] M. Allamanis and C. Sutton, "Mining source code repositories at massive scale using language modeling," in *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 207–216.

[4] J. McDermott, D. R. White, S. Luke, L. Manzoni, M. Castelli, L. Vanneschi, W. Jaskowski, K. Krawiec, R. Harper, K. De Jong *et al.*, "Genetic programming needs better benchmarks," in *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. ACM, 2012, pp. 791–798.

[5] W. Weimer, T. Nguyen, C. Le Goues, and S. Forrest, "Automatically finding patches using genetic programming," in *Proceedings of the 31st International Conference on Software Engineering*. IEEE Computer Society, 2009, pp. 364–374.