On the Automatic Construction of Regular Expressions from Examples (GP vs. Humans 1-0)

Alberto Bartoli
DIA - University of Trieste
Italy
bartoli.alberto@units.it

Andrea De Lorenzo
DIA - University of Trieste
Italy
andrea.delorenzo@units.it

Eric Medvet
DIA - University of Trieste
Italy
emedvet@units.it

Fabiano Tarlao
DIA - University of Trieste
Italy
fabiano.tarlao@phd.units.it

ABSTRACT

Regular expressions are systematically used in a number of different application domains. Writing a regular expression for solving a specific task is usually quite difficult, requiring significant technical skills and creativity. We have developed a tool based on Genetic Programming capable of constructing regular expressions for text extraction *automatically*, based on *examples* of the text to be extracted.

We have recently demonstrated that our tool is *human-competitive* in terms of both *accuracy* of the regular expressions and *time* required for their construction. We base this claim on a large-scale experiment involving more than 1700 users on 10 text extraction tasks of realistic complexity. The F-measure of the expressions constructed by our tool was almost always higher than the average F-measure of the expressions constructed by each of the three categories of users involved in our experiment (Novice, Intermediate, Experienced). The time required by our tool was almost always smaller than the average time required by each of the three categories of users. The experiment is described in full detail in "Can a machine replace humans? A case study. IEEE Intelligent Systems, 2016" .

Keywords

Regular Expressions; Entity Extraction; Users Evaluation

1. INTRODUCTION

Regular expressions are systematically used in a number of different application domains. Writing a regular expression is often a complex endeavor requiring significant technical skills and creativity. Along the years, a wealth of research efforts have considered the problem of constructing a regular expression *automatically* based on *examples* of the desired behavior. This problem may be cast in several ways, depending on the intended usage of the regular expression and on the nature of the input data (a systematic literature analysis can be found in [6, 5]). The intended usage may be either binary classification of input items or extraction of chunks from a (possibly very long) input item. In nearly all efforts the constructed expression is expected to generalize a pattern from the available examples, although there have also been proposals aimed at binary classification of input items in two predefined lists, without any generalization requirement [3]. Concerning the nature of input data, there have been proposals focussing on input items expressed in a formal language, on input items consisting of text lines, on input items consisting of an unstructured text stream.

In our multi-year research activity on this topic we have developed several proposals based on Genetic Programming (GP) for automatic construction of regular expressions for text extraction from an unstructured stream. We represent a candidate solution (regular expression) as an abstract syntax tree assembled with the regular expression constructs and we evolve a population of candidate solutions with a multiobjective optimization algorithm in which the fitness of each candidate solution quantifies its accuracy on the available examples (to be maximized) and its length (to be minimized).

Our activity may be summarized in a sort of two epochs: a first tool which improved over the earlier state-of-the-art substantially, demonstrating the ability to address tasks of realistic complexity effectively [1, 2]; a second tool [6], which improved the first tool from a number of points of views, including support for the OR operator based on a form of separate-and-conquer search strategy [4], support for a broader set of regular expression constructs capable of addressing context-dependent extractions, a more sophisticated fitness definition delivering better F-measure and capable of supporting potentially unbounded input items.

We have recently demonstrated that our tool is *human-competitive* in terms of both quality of solutions and time required for their construction. We base this claim on a large-scale experiment involving more than 1700 users on 10 text extraction tasks of realistic complexity. The experiment is described in full detail in [5].

GECCO'16 Companion July 20-24, 2016, Denver, CO, USA

ACM ISBN 978-1-4503-4323-7/16/07.

DOI: http://dx.doi.org/10.1145/2908961.2930946

We developed a web application containing a suite of extraction tasks and challenged users to test their skills with a Reddit post[1]. Each task consisted of a piece of unstructured text annotated with the portions to be extracted (unannotated portions were not to be extracted)[2]. The annotated portions of a task described a certain task-specific pattern: URLs (in two datasets of different nature), phone numbers, HTML href attributes, IP addresses, MAC addresses, HTML headings, HTML heading content (i.e., excluding the delimiting HTML tags), author names in bibtex entries, name of lead author in bibliographic lists.

Each user was asked to self-classify his proficiency in regular expressions, either Novice, or Intermediate, or Experienced. We measured the time spent by each user on each task and assessed the F-measure of each constructed expression on a separate testing set. Next, we executed our tool on the very same tasks and the results were as follows.

- For each task (except for one), the *time* spent by our tool for constructing a regular expression was much smaller than the average time required by each category of users.

  The only task in which our tool required more time than human operators was HTML heading content. However, on this task our tool delivered significantly better F-measure than all the three categories of human operators.

- For each task (except for one), the *F-measure* of the regular expression constructed by our tool was higher than the average F-measure obtained by each category of users.

  The only task in which our tool delivered smaller (and unsatisfactory) F-measure was extraction of phone numbers. The reason is because the training data did not describe adequately text that looks like a phone number but is not a phone number. Humans were able to infer the general pattern appropriately from the available examples while our tool was not. With a larger training set, though, our tool was able to obtain F-measure comparable to human operators or better.

Our work is significant, we believe, for at least two reasons. First, there is no other tool for automatic construction of regular expressions capable of delivering human-competitive performance on tasks of realistic complexity. Second, it demonstrates the power of GP on a difficult synthesis problem, requiring technical skills and creativity.

Our recent reference work describing the internals of the tool in full detail includes an experimental comparison to other methods for learning of syntactical patterns that are not specified as a regular expression [6]. Specifically, a method for learning deterministic finite automata [8] and a method included in Windows Powershell for synthesizing programs in a specialized data extraction language [7]. The comparison demonstrates a clear superiority of our approach, on the text extraction tasks considered in our analysis (method [7] can address multifield extraction tasks, i.e., extraction of multiple heterogenous patterns; our tool can only address those tasks as multiple, independent tasks).

The sources of our tool are publicly available on GitHub and a prototype is available online[3].

[1] https://www.reddit.com/r/programming/comments/3eblji/how_good_are_you_in_writing_regex_challange/

[2] A plain and concise description of the web app can be found at http://www.i-programmer.info/news/204-challenges/9586-machine-learning-labs-regular-expression-game.html

[3] https://github.com/MaLeLabTs/RegexGenerator and http://regex.inginf.units.it/

2. REFERENCES

[1] A. Bartoli, G. Davanzo, A. De Lorenzo, M. Mauri, E. Medvet, and E. Sorio. Automatic generation of regular expressions from examples with genetic programming. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '12, pages 1477–1478, New York, NY, USA, 2012. ACM.

[2] A. Bartoli, G. Davanzo, A. De Lorenzo, E. Medvet, and E. Sorio. Automatic synthesis of regular expressions from examples. *Computer*, 47(12):72–80, Dec 2014.

[3] A. Bartoli, A. De Lorenzo, E. Medvet, and F. Tarlao. Playing regex golf with genetic programming. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 1063–1070, New York, NY, USA, 2014. ACM.

[4] A. Bartoli, A. De Lorenzo, E. Medvet, and F. Tarlao. Learning text patterns using separate-and-conquer genetic programming. In *18th European Conference on Genetic Programming*. Springer Verlag, 2015.

[5] A. Bartoli, A. D. Lorenzo, E. Medvet, and F. Tarlao. Can a machine replace humans in building regular expressions? A case study. *IEEE Intelligent Systems*, 2016. To appear.

[6] A. Bartoli, A. D. Lorenzo, E. Medvet, and F. Tarlao. Inference of regular expressions for text extraction from examples. *IEEE Transactions on Knowledge and Data Engineering*, 28(5):1217–1230, May 2016.

[7] V. Le and S. Gulwani. FlashExtract: A framework for data extraction by examples. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, page 55. ACM, 2014.

[8] S. M. Lucas and T. J. Reynolds. Learning deterministic finite automata with a smart state labeling evolutionary algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1063–1074, 2005.