Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 16 (2014) 806 – 812

Procedia Technology

CENTERIS 2014 - Conference on ENTERprise Information Systems / ProjMAN 2014 - International Conference on Project MANagement / HCIST 2014 - International Conference on Health and Social Care Information Systems and Technologies

An automatic generation of textual pattern rules for digital content filters proposal, using grammatical evolution genetic programming

Vitor Basto-Fernandes[a]*, Iryna Yevseyeva[b], Rafael Z. Frantz[c], Carlos Grilo[a], Noemí Pérez Díaz[d], Michael Emmerich[e]

[a] *School of Technology and Management, Computer Science and Communications Research Centre, Polytechnic Institute of Leiria, 2411-901 Leiria, Portugall*

[b] *Centre for Cybercrime and Computer Security, School of Computing Science, Newcastle University, Newcastle-upon-Tyne, NE1 7RU, UK*

[c] *UNIJUÍ University, Department of Exact Sciences and Engineering, Rua Lulu Ilgenfritz, 480 - Bairro São Geraldo, 98700-000, Ijuí, RS, Brazil*

[d] *University of Vigo, Campus As Lagoas S/N, 32004 Ourense, Spain*

[e] *LIACS, Leiden University, 2333CA Leiden The Netherlands*

Abstract

This work presents a conceptual proposal to address the problem of intensive human specialized resources that are nowadays required for the maintenance and optimized operation of digital contents filtering in general and anti-spam filtering in particular. The huge amount of spam, malware, virus, and other illegitimate digital contents distributed through network services, represents a considerable waste of physical and technical resources, experts and end users time, in continuous maintenance of anti-spam filters and deletion of spam messages, respectively. The problem of cumbersome and continuous maintenance required to keep anti-spam filtering systems updated and running in an efficient way, is addressed in this work by the means of genetic programming grammatical evolution techniques, for automatic rules generation, having SpamAssassin anti-spam system and SpamAssassin public corpus as the references for the automatic filtering customization.

Peer-review under responsibility of the Organizing Committee of CENTERIS 2014.

Keywords: Spam filtering, Digital Content Filters, Genetic Programming, Classification, Grammatical Evolution

* Corresponding author. Tel.: +351-244-820-300; fax: +351-244-820-310.
E-mail address: vitor.fernandes@ipleiria.pt

Peer-review under responsibility of the Organizing Committee of CENTERIS 2014.
doi:10.1016/j.protcy.2014.10.030

1. Introduction

The increasing amounts of information exchanges worldwide through Internet requires progressive improvement on resources usage effectiveness and efficiency, and has been particularly felt in the area of anti-spam filtering.

Spammers explore and develop a wide variety of forms to distribute illegal and fraudulent advertisements. Due to continuous advancement of techniques used to distribute spam, anti-spam filters become obsolete in short time periods and need to be updated on a regular basis. Customer satisfaction and emails classification accuracy of anti-spam filtering systems strongly rely on the ability of continuous updating by adding new rules to it, with respect to customer preferences and ability to stay up to date with the latest spam spreading techniques. Behind anti-spam filtering services are groups of experts examining emails and updating anti-spam filters behavior to detect the newest spam contents.

Filtering frameworks such as SpamAssassin [1] or Wirebrush4SPAM [2] assist in filter updates by using a filtering description syntax based on message fields and contents criteria. They provide a great flexibility for easy creation, manipulation and deployment of new customized anti-spam filtering rules.

After performing studies and contributions in previous papers on automatic anti-spam filtering scores optimization [3], the authors are addressing now a more challenging research problem that is the automatic generation of anti-spam filtering rules by the adoption of an innovative approach using genetic programming techniques.

In this paper we propose an evolutionary schema for the creation and deployment of new customized anti-spam filtering rules, by the means of a combination of evolutionary computation techniques targeted for automatic generation of anti-spam filtering rules learned from reference datasets of messages.

In section 2 the SpamAssassin antispam-filtering system is briefly presented, emphasizing its specific features related to the automatic rules generation contribution described in the following sections. Chapter 3 presents SpamAssassin anti-spam filtering automatic tuning/optimization. In chapter 4 genetic programming is generally introduced and grammatical evolution techniques are specifically addressed and presented as most suitable for the SpamAssassin rules automatic generation. ECJ is presented as the reference framework in the area of genetic programming. Chapter 5 introduces our problem formulation and methodology for anti-spam filters rules automatic generation. Finally, conclusions and future work is described in chapter 6.

2. Anti-spam filtering systems

In this work we assume the open source SpamAssassin filtering system as the reference for the optimization and automatic filtering rules generation research proposal. This choice is due to the public available open source nature of Spam Assassin. It is part of active research and a development community providing contributions in a continuous way. SpamAssassin is the most popular anti-spam filtering system for research purposes and due to its high classification performance it is also popular among the small and medium enterprises [2].

For the sake of simplicity and clearness we present here only the features of SpamAssassin related to the anti-spam content filtering configuration and the corresponding technological and research perspectives. We address in this context one of the most challenging research problems, the continuous update and maintenance required to keep the anti-spam filtering system efficiency and its classification quality at acceptable levels.

As described in [2] and [4], SpamAssassin is a framework for the execution and development of new user defined anti-spam filters and techniques. The techniques are combined in a filter according to user needs, providing great flexibility for the creation and deployment of customized anti-spam filtering solutions. Different types of techniques are used in the SpamAssassin anti-spam filtering, such as intelligent analysis of message contents, collaborative querying and information sharing on spam senders and deliveries, senders legitimacy verification, and regular expressions based messages analysis type.

Each type of technique is not able to provide efficient classification alone, only the combination of techniques of different types can provide acceptable levels of quality for anti-spam filtering classification. Each rule corresponds to a logical test and has a score (weight) associated. The classification process consists of checking all rules against a message, and the verification, if the sum of all the matching rules scores overcomes the required threshold value.

Rules creation and scores setting is mainly performed manually by system administrators based on experience, applying a try-and-error approach.

Continuous creation of new ways to distribute spam (developed by spammers), leads to the need of continuous updating of the anti-spam system by the creation of new rules, corresponding scores (weighting) setting for the new rules and adjustment of the scores (weights) of existing ones.

Neither the generation of rules to detect new spam messages, nor the setting or adjustment/update of rules scores/weights, are straightforward. Generation of new rules has to take into account the existent rules "knowledge base" and the relative importance of rules to assign individual scores (weights). Both of these decision problems are complex, especially because they involve the analysis of thousands of rules.

Additional complexity on rules creation and scores tuning comes from the highly customized nature of anti-spam filtering. Anti-spam filtering is highly dependent on the type of organization, business domain, location, language, culture and other user specific criteria such as if the mail box is used for professional or non professional purposes.

These tasks are performed usually with little or no systematic guidance or support. The need of automation and optimization techniques to assist or substitute system administrators in these tasks has been recently emphasized in the literature. For automatic scores setting (and re-setting) support, advanced optimization techniques are found in the literature, specifically for the SpamAssassin anti-spam filtering. A survey of literature on this subject is found in [5]. Anti-spam filtering scores setting (tuning) problem formulations are studied and presented, together with optimization techniques proposals for the anti-spam filtering optimal rules scores setting.

In section 3 we present briefly the multi-objective formulation of automatic anti-spam filtering tuning problem, and the reasons why that is the problem formulation adopted by us in previous and the current study. In the following sections, we address the problem of automatic rules generation by the means of genetic programming grammatical evolution techniques.

3. Anti-spam filtering optimization (tuning)

In previous [3, 5, 6] and current study we follow a formulation of the scores setting problem as a bi-objective optimization problem. Users wish to minimize both, the number of spam messages not identified by anti-spam filtering techniques, called false negative (FNs), and the number of legitimate messages classified as spam by mistake, called false positives (FPs). Depending on user preferences, the anti-spam systems setup can be tuned to have lowest possible rate of lost legitimate messages (usually at the expenses of higher FN classifications) or to have lowest possible rate of spam messages getting into user mailbox (usually at the expenses of higher FP classifications). In between these two extreme scenarios, there is a wide spectrum of anti-spam filtering configurations trading off these two conflicting objectives (minimizing FP and minimizing FN).

The anti-spam filtering optimization problem could be formulated as a single-objective problem, by weighting objectives according to their importance, and measuring the combined total amount of FP and FN (wrongly classified messages) using TCR (Total Cost Ratio), *precision*, *recall* and other similar metrics.

Modeling multi-objective problems as single-objective problems, either by summing up the objectives in a weighted sum and optimizing (minimizing/maximizing) the weight sum of objectives, or by aggregating the objectives into a single-objective "utility function" (like in the TCR metric), may lead to the loss of flexibility. These techniques may limit the solution space search to convex problems only and prevent the exploration of all regions of optimal solutions throughout the optimization process.

In previous and current study we follow a multi-objective optimization problem formulation, assuming that minimization of FP and FN are equally important in the optimization process. Instead of adopting so called *a priori* methods, we follow *a posteriori* methods, meaning that decision maker preferences must be considered on the output of the optimization technique and not by the means of input variables simplification/combination.

Due to the combinatorial nature of the automatic anti-spam filtering tuning problem, it cannot be solved via traditional exact optimization methods. However, heuristic methods can be used to find approximations to solutions, e.g. evolutionary algorithms that mimic the improvement process driven by selection and variation (mutation, crossover) as found in the biological evolution. Several multi-objective evolutionary algorithms (EMOA) were tested in previous studies [3, 5], revealing promising results for the anti-spam filtering rules scores setting problem.

These studies provided recommendations on EMOA to be followed and applied in the context of the methodology and framework proposed.

4. Genetic Programming

4.1. Introduction

Genetic programming (GP) is a systematic, domain-independent, evolutionary computation technique for computational problem solving [7]. In genetic programming a population of computer programs is evolved from generation to generation, stochastically trying to transform programs into new, better, populations of programs.

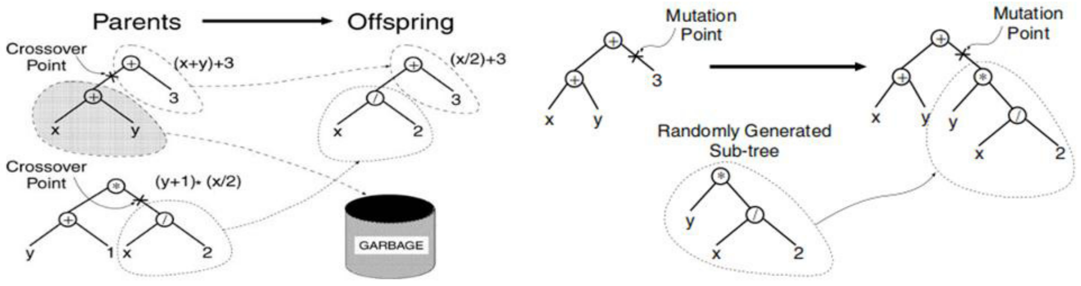

Figure 1 - GP Arithmetic expressions example of crossover and mutation operations [7].

GP simple programs are usually represented by syntax trees composed by terminal (leaves or end nodes) and internal nodes called functions, as shown in Figure 1 for arithmetic expressions example. The sets of functions and terminals constitute the primitive set of the GP system. The primitive set and the programs quality measurement function are problem dependent and must be specified by the GP system designer.

GP execution follows a pretty general evolutionary algorithm schema: An initial population of programs is randomly generated from available initialization primitives (e.g. full, grow, Ramped half-and-half initialization models). Quality assessment of programs belonging to the population is performed by running them and comparing its quality (fitness) to some ideal reference. The best quality performing programs are chosen deterministically or in a probabilistically based fitness (e.g. tournament selection, fitness-proportionate selection, selection based selection) to breed and produce new programs for the next generation. Applying crossover and mutation to the selected programs results in creation of offspring programs, which are subject to mutation operations (Figure 1). Quality assessment of offspring is done in order to repeat the cycle until some stopping condition (maximum number of generations) is met. Finally, GP returns the best individual resulting from the evolutionary process.

4.2. Grammatical evolution GP

In some GP systems the evolutionary process must be constrained, based on some problem domain knowledge. If the domain knowledge implies a particular syntactic or grammar-based constraint on the programs, ignoring this constraint causes a wrong evaluation function or bad performance of the algorithm to find promising solutions. In these cases, both programs initialization, crossover and mutation operations must not lead to incompatible connections between nodes, preventing the generation of illegal programs, that would make harder to approach or find optimal solutions.

A simple and formal way of expressing constraints is via grammars (e.g. Extended Backus-Naur Form [8]). The grammar is typically used to ensure that the initial population is made up of legal programs and also to guide the operations of the genetic operators to generate new programs. Although constraint GP systems reduce the search space for valid solutions, it comes at the cost of higher algorithm complexity and introduces biases in the search space exploration that should be carefully analyzed. Grammatical evolution GP is one type of GP systems corresponding to the class of constraint based GP, being the most suitable to support the methodology proposed in our study for the anti-spam filtering rules generation.

5. Anti-spam rules generation based on GP

As described in section 1 and 2, the continuous creation of new ways to distribute spam (developed by spammers), leads to the need of continuous updating of the anti-spam system by the creation of new anti-spam filtering rules. Although there are several different types of techniques being used in the SpamAssassin anti-spam filtering, we concentrate our research in the regular expressions based messages analysis type of rules. The reason for us to concentrate on this type of rules is grounded in the fact that highly customization of anti-spam filtering is mostly achieved by rules that belong to this technique type.

A regular expression is represented by a sequence of characters to express a search text pattern. Each character in a regular expression can be a metacharacter with special meaning or a regular character with literal meaning [9]. Some research has been done in the area of automatic generation of regular expressions from examples [10], including for spam detection purposes [11]. However, the studies found in the area, addressed generation of regular expression based on simple short textual examples (not reference datasets) using GP, or when targeting spam detection they used (numeric) genetic algorithms (GA) approaches (not GP or grammatical evolution GP). Those studies are excessively simplified for the purposes of anti-spam filtering classification, when targeting regular expression generation to find patterns in simple short sentences (such as in [10]), or their GA style problem representation raises serious difficulties in capturing the grammar based constraints of regular expressions, leading to poorer search space exploration [12]. Additionally, these studies assume a single-objective problem formulation, as the weighted sum minimization of FN and FP, contrasting to the multi-objective optimizer that we aim for.

To address the problem of anti-spam filtering rules generation with the grammatical evolution GP system approach, we propose the following problem formulation and methodology:

1) Proceed with data preparation (e.g. remove email html annotations and binary attachments, isolate human-readable text, etc). Spam messages of public SpamAssassin corpus must be subject to some data preparation and cleansing for the GP execution with ECJ framework executions;

2) Tokenize the spam messages from the prepared corpus in step 1. Assume the spam messages tokens as potentially representative components (terminals) of spam messages text patterns in the grammatical evolution of the GP system;

3) Tokenize the legitimate messages from the prepared corpus in step 1. Assume the spam messages tokens as potentially representative components (terminals) of legitimate (ham) messages text patterns, in the grammatical evolution GP system;

4) Discard tokens that are present at high balanced rates both in spam and ham messages. Discarded tokens are considered not contributing in a relevant way for spam vs ham differentiation, therefore they are not used as inputs to the (terminal set) proposed grammatical evolution GP system;

5) Set the relevant tokens for spam vs ham differentiation and regular expressions terminal tokens as terminals for the grammatical evolution GP system representation;

6) Define the regular expressions metacharacters (with special meaning) and basic boolean operators sets, as functions for the grammatical evolution GP system representation;

7) Define the quality of a program (fitness of a generated regular expression) according to a multi-objective optimization problem formulation, minimizing both the FP and FN messages classification.

8) Test and select most suitable bloat (trend of GP techniques to generate increasingly complex solutions without corresponding quality improvement [13]) prevention mechanisms for the current problem formulation;

9) Remove messages from the dataset and programs from the GP system responsible for successful classifications, and restart the evolutionary GP system process for the remaining messages;

10) Define a stopping criterion for the GP systems specialization training process;

11) Apply the multi-objective optimization techniques studied and developed tools published in previous studies [3,5], for the anti-spam filtering automatic tuning (classifier optimal rules scores setting).

The above presented methodology serves as the basis for the experimental stage being performed currently with the ECJ framework [14]. Among the evolutionary computation frameworks available for the scientific community, ECJ revealed to be the one most suited for genetic programming support. It is a mature, stable, free open-source evolutionary computation framework written in Java, designed for large, heavy-weight experimental needs, providing many popular evolutionary computation algorithms and conventions, with particular emphasis towards genetic programming. ECJ is design to easily accommodate additional algorithms and extensions. It is widely used by the genetic programming community, and also includes implementations of the most relevant multi-objective optimization algorithms. Due to the previously mentioned reasons, ECJ was selected to perform the experimental stage of our ongoing study.

6. Conclusions and future work

Continuous improvements of spammers techniques to distribute spam and anti-spam filters' highly customized nature with respect to several criteria (such as institutional, location, language and user specific criteria), requires permanent efforts of system administrators creating anti-spam filter rules to keep acceptable levels of spam classification performance and efficiency. Among the type of filtering rules present in the SpamAssassin anti-spam system adopted for our study, those based on regular expressions textual pattern detection have a major role in filter updating and customization.

Creating anti-spam filtering rules in general, and textual pattern based ones in particular, is a cumbersome, complex error-prone task requiring specific expertise and experience. In this paper we propose a grammatical evolution approach for the automatic generation of anti-spam filtering regular expressions rules. SpamAssassin public corpus is adopted as the reference dataset used for training and performance assessment of the proposed approach.

Although the problem of generating regular expressions from examples has been studied from several different points of view, the GP approach was not found in the literature for the anti-spam filtering problem, and was therefore addressed and studied in the current work.

A variety of other published proposals were analyzed, but they present simplified models of the anti-spam filtering problem, based on genetic algorithms (not genetic programming). These approaches suffer from search space exploration disadvantages, due to difficulties on problem representation and operators design. Other studies following GP approaches were found in the literature, but they use oversimplified single sentences textual examples for training/testing and require text labeling, which makes them of limited interest for the anti-spam filtering problem solving.

The grammatical evolution GP anti-spam filtering problem formulation and methodology presented here aims to be the reference for the experimental stage of the study to follow. Results comparison are to be done against other approaches proposed in the literature, and also having as a reference the SpamAssassin default rules configuration released in public, free SpamAssassin software distributions.

Although we concentrated our study specifically on the problem of digital content filtering problem for email messages spam classification, we see our approach and methodology generalizable for other digital content filtering application domains. Messaging services filtration in the content management systems domain and data leakage prevention in the security domain, are two examples of potential applications of our digital content filtering proposal, based on automatic generation of textual patterns rules generation, using grammatical evolution GP.

Acknowledgements

References

[1] The Apache SpamAssassin Project. The Powerful #1 Open-Source Spam Filter. SpamAssassin <http://spamassassin.apache.org> 2011.

[2] Pérez-Díaz N, Ruano-Ordas D, Fdez-Riverola F, Méndez JR. Wirebrush4SPAM: a novel framework for improving efficiency on spam filtering services. Software: Practice and Experience 2012; 43(11): 1299-1318.

[3] Yevseyeva I, Basto-Fernandes V, Ruano-Ordás D, Méndez JR. Optimising anti-spam filters with evolutionary algorithms. Expert Syst. Appl 2013; 40(10): 4010-4021.

[4] Pérez-Díaz N, Ruano-Ordas D, Méndez JR, Gálvez JF, Fdez-Riverola F. Rough sets for spam filtering: Selecting appropriate decision rules for boundary e-mail classification. Appl. Soft Comput. 2012; 12(11): 3671-3682.

[5] Basto-Fernandes V, Yevseyeva I, Méndez JR. Anti-spam multiobjective genetic algorithms optimization analysis. International Resource Management Journal 2012; 26(1): 54-67.

[6] Yevseyeva I, Basto-Fernandes V, Ruano-Ordás D, Méndez JR. Survey on Anti-spam Single and Multi-objective Optimization. In: Cruz-Cunha MM, Varajão J, Powell P, Martinho R editors. ENTERprise Information Systems International Conference Proceedings, Part II, 2011, Springer-Verlag GmbH Berlin Heidelberg 2011. p 120-129.

[7] Poli R, Langdon WB, McPhee NF, Koza JR, A Field Guide to Genetic Programming, March 2008 - http://www.gp-field-guide.org.uk

[8] EBNF: A notation to descibe a sintax Pattis RE http://www.ics.uci.edu/~pattis/misc/ebnf.pdf

[9] The Single UNIX Specification, Regular Expressions,Version 2. The Open Group. 1997.

[10] Bartoli A, Davanzo G, De Lorenzo A, Mauri M, Medvet E, Sorio E. Automatic generation of regular expressions from examples with genetic programming. In: Soule T, editor, Proceedings of the 14th annual conference companion on Genetic and evolutionary computation(GECCO '12). ACM, New York, NY, USA, 2012. p. 1477-1478.

[11] Conrad E, Detecting Spam with Genetic Regular Expressions SANS Institute InfoSec Reading Room, Copyright SANS Institute, 2007.

[12] Greenstadt R, Kaminsky, M Evolving Spam Filters Using Genetic Algorithms. Final Project. 2002, web.mit.edu/greenie/Public/spamfilter-GA-paper.ps

[13] S. Silva and E. Costa. Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. Genetic Programming and Evolvable Machines, 10(2):141-179, 2009.

[14] The ECJ Owner’s Manual. A User Manual for the ECJ Evolutionary Computation Library, Luke S, Dept of Computer Science, George Mason University, Manual Version 21, May, 2013, http://cs.gmu.edu/_eclab/projects/ecj/