Information Processing and Management 54 (2018) 303–317

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Using evolutionary computation for discovering spam patterns from e-mail samples

David Ruano-Ordás[a,b,*], Florentino Fdez-Riverola[a,b], José R. Méndez[a,b]

[a] *Department Computer Science, University of Vigo, ESEI, Campus As Lagoas, 32004, Ourense, Spain*
[b] *Centro de Investigaciones Biomédicas (Centro Singular de Investigación de Galicia), Campus Universitario Lagoas-Marcosende, 36310, Vigo, Spain*

ARTICLE INFO

Keywords:
Genetic programing
Regular expressions
Automatic generation
E-mail
Spam filtering

ABSTRACT

One of the most relevant problems affecting the efficient use of e-mail to communicate worldwide is the spam phenomenon. Spamming involves flooding Internet with undesired messages aimed to promote illegal or low value products and services. Beyond the existence of different well-known machine learning techniques, collaborative schemes and other complementary approaches, some popular anti-spam frameworks such as SpamAssassin or Wirebrush4SPAM enabled the possibility of using regular expressions to effectively improve filter performance. In this work, we provide a review of existing proposals to automatically generate fully functional regular expressions from any input dataset combining spam and ham messages. Due to configuration difficulties and the low performance achieved by analysed schemes, in this work we introduce DiscoverRegex, a novel automatic spam pattern-finding tool. Patterns generated DiscoverRegex outperform those created by existing approaches (able to avoid FP errors) whilst minimising the computational resources required for its proper operation. DiscoverRegex source code is publicly available at https://github.com/sing-group/DiscoverRegex.

1. Introduction and motivation

Although Internet provided a wide variety of forms and services to communicate people worldwide, the introduction, evolution and popularization of smartphones and their operating systems (e.g. Android OS, Blackberry OS, iOS, Cyanogen OS) have really improved the experience of users. However, different spamming forms have drastically affected the achieved sophistication of Internet communications during last years. In this context, hoax, fraud, phishing, or pharming are just a few samples of spam that coexists in different services such as instant messaging, e-mail, social networking, and/or weblogs. Whilst hoaxing is mainly focused in the divulgation of false information, the remaining tricks mainly seek stealing money.[1]

With the goal of alleviating this situation, some progress in content-based filtering methods took place over the last years. In this context, machine learning (ML) became very popular as a set of reliable techniques able to efficiently fight against spam (Guzella & Caminhas, 2009). Earlier ML approaches were mainly based on the use of Naïve Bayes classifiers (Borodin, Polishchuk, Mahmud, Ramakrishnan, & Stent, 2013; Graham, 2009; Metsis, Androutsopoulos, & Paliouras, 2006; Rennie, 2000) being successfully introduced in some popular filtering frameworks (Apache Software Foundation, 2007; Pérez-Díaz, Ruano-Ordás, Fdez-Riverola, & Méndez, 2013). Additionally, the scientific community also evaluated other ML algorithms with the potential to filter spam including

* Corresponding author at: Centro de Investigaciones Biomédicas (Centro Singular de Investigación de Galicia), Campus Universitario Lagoas-Marcosende, 36310, Vigo, Spain
E-mail address: drordas@uvigo.es (D. Ruano-Ordás).
URL: http://www.drordas.info/ (D. Ruano-Ordás).
[1] Some popular scams are studied in http://www.hotscams.com/ or in https://www.scamwatch.gov.au/.

https://doi.org/10.1016/j.ipm.2017.12.001
Received 27 April 2017; Received in revised form 2 November 2017; Accepted 1 December 2017

Support Vector Machines (SVM) (Amayri & Bouguila, 2010; Bouguila & Amayri, 2009; Li, Li, & Zhou, 2009; Pérez-Díaz, Ruano-Ordás, Fdez-Riverola, & Méndez, 2016), Artificial Immune Systems (AIS) (Borodin et al., 2013; Guzella, Mota-Santos, Uchõa, & Caminhas, 2008), Memory-Based Systems (Delany, Cunningham, & Tsymbal, 2006; Delany, Cunningham, Tsymbal, & Coyle, 2005; Fdez-Riverola, Iglesias, Díaz, Méndez, & Corchado, 2007; Pang & Jiang, 2013; Sakkis et al., 2003) or Rough Sets (RS) (Pérez-Díaz, Ruano-Ordás, Méndez, Gálvez, & Fdez-Riverola, 2012; Zhao & Zhang, 2005). Alongside, some other collaborative filtering schemes also emerged from particular companies or Internet communities to fight against spam (Damiani, di Vimercati, Paraboschi, & Samarati, 2004; Del Castillo & Serrano, 2005; Prakash, 2007).

Despite those efforts, authorities, bank entities and other institutions have simultaneously designed their own awareness campaigns to encourage users to thoroughly examine messages before clicking their links. Nevertheless, a study of FBI recently announced that 8179 victims of scam e-mail loosed 788,897,959.28$ between October 2013 and august 2015 (Federal Bureau of Investigation, 2015). This is possible because spammers simultaneously take advantage of different strategies to ensure their success: (*i*) use of diverse tricks to avoid the rapid identification of some key words (‘V¡agra’, ‘Viaggra’, ‘V i a g r a’ or ‘v < bre/ > iagra’), (*ii*) add common ham (legitimate) terms using the same colour for text and background, (*iii*) URL hiding by URL encoding, site redirectors or URL shorteners and (*iv*) use of images to represent text, among others. Moreover, spammers periodically change the combination of the above-mentioned trickery together with the message format, making the identification of spam even more difficult.

All the stratagems commented above obviously limit the effectiveness of most content-based techniques to properly fight against spam but, in such a situation, the use of manually generated regular expressions (regex) becomes an appropriate method to identify those messages obfuscated by spammers. In this context, a regular expression is a compact way to describe sets of sentences that conforms to a pattern and, therefore, they can be locally used as a content-based filtering scheme (Apache Software Foundation, 2007; Jones, 1997) or shared in a peer-to-peer (P2P) network to build up a collaborative filtering strategy (Segal, Crawford, Kephart, & Leiba, 2004). Although historically, regular expressions have been used to reduce the misclassification of spam messages (FN errors). Due to their intrinsic capability to easily capture and reveal those tricks used by spammers to hide inappropriate terms, they can be also used as a pre-filter strategy (i.e. if a given e-mail match any pattern, no further analysis is required and the message is classified as spam) reducing the computational cost derived from the use of more sophisticated techniques. In particular, regular expressions are useful to reduce misclassification errors, especially false positive ones, which entails the elimination of valuable messages without the recipient's knowledge.

However, the manual identification of fully featured regular expressions matching spam messages is extremely complex, so there is an acute need for automatic and/or support tools to accomplish this task. Additionally, the changing nature of spam (concept drift) forces regular expression to be periodically updated in order to ensure their adaptation to the new spam trends.

In response to this need, and with the aim of automating the entire process, this work studies the use of evolutionary computation to automatically generate regular expressions from a set of e-mails that can be further used for spam filtering. In detail, it provides a deep analysis about the drawbacks and robustness of actual regular expression generation approaches. As long as results achieved using the current approaches were not accurate enough (great amount FP errors), we designed a new regular expression finding tool (DiscoverRegex) which is also a contribution of this work.

While this section has motivated the work, the rest of the paper is structured as follows: Section 2 introduces the state of the art in evolutionary computation and regular expression extraction from texts; Sections 3 and 4 describe our approach to discover knowledge in the form of regular expressions and validate its utility as pre-filter alternative. Finally, Section 5 outlines the main conclusions of our work and defines future research lines.

2. State of the art: Generation of regular expressions through genetic programing

The first method found in the literature able to discover patterns from spam e-mails was the Chung-Kwey algorithm (Rigoutsos & Huynh, 2004). Given the ease of sharing patterns between computers, it was soon adapted by SpamGuru (Segal et al., 2004), a collaborative filtering platform developed in the mid-2000 s. Despite their unquestionable initial value, patterns discovered by applying Chung-Kwey and other approaches are less flexible than regular expressions, because only the wildcard symbol “.” is allowed. In contrast, standard regular expressions provide a more rich and flexible syntax to support the representation of multiple pattern types. Furthermore, the main limitation of this approach lies in the need to configure a great amount of parameters (e.g. length of the pattern, minimum number of matches, maximum amount of wildcards, etc.) that are difficult to guess in advance. Both flaws (specially the latest one) suggested the use of evolutionary computational methods to specifically address the challenge of regular expression generation.

In this way, previous works proved the utility of Genetic Programming (GP) as a reliable way to generate regular expressions (Basto-Fernandes et al., 2014). GP is an evolutionary computational method particularly suitable for solving complex problems (Poli, Langdon, & McPhee, 2008) based on the use of and initial population of individuals (i.e. regular expressions) that is stochastically evolved into new, hopefully better populations by using *crossover* and *mutation* operators (see Fig. 1). To evaluate the suitability of a given individual, a *fitness* function needs to be defined. Crossover and mutation operators allow the combination and random modification of individuals in a population, guaranteeing the generation of a wide variety of alternative regular expressions. Complementarily, the fitness function assess the quality of an individual as the intended problem solution, and allows the selection of the best individuals for mating or mutation purposes.

As seen in Fig. 1, during the first and second step a population (collection of individuals) is initially generated and further evaluated from those samples included in the original corpus. During the third phase, individuals are combined and randomly altered using both crossover and mutation operators. The selection of which individuals are going to be combined can be carried out keeping