2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering

Improving Email Security with Fuzzy Rules

Sudarshan S. Chawathe
School of Computing and Information Science
University of Maine
Orono, Maine 04469-5711, USA
chaw@eip10.org

***Abstract*—Phishing and other malicious email messages are increasingly serious security threats. An important tool for countering such email threats is the automated or semiautomated detection of malicious email. This paper reports work on using fuzzy rules to classify email for such purposes. The effectiveness of a fuzzy rule-based classifier is studied experimentally on a real dataset and compared with results for other classifiers, including those based on crisp rules and decision trees. The human-readability and editability of the classifiers produced by these methods is also studied.**

***Index Terms*—email, phishing, security, fuzzy logic, machine learning, classification, spam.**

I. Introduction

Email communications are part of the critical infrastructure of most modern organizations. Along with essential and desirable messages, email routinely conveys undesirable messages that range from simply annoying (e.g., promotional messages) to extremely dangerous (spear phishing attacks). The growing volume and malevolence of email seccurity threats is well documented, and has resulted in multiple lines of defense, such as filtering at service providers based on blacklists, signed headers, and spam filtering on server and client machines. It seems clear that effective defense against email threats requires multiple tools and approaches.

This paper describes and evaluates one such tool for improving email security: a semiautomated, rule-based system for detecting malicious email messages. It is important to note at the outset that this tool is not meant to operate in isolation by replacing other tools and approaches such as those noted above. Rather, it is designed to augment those by filling in the gaps left by fully automated systems. In particular, a key design requirement is that it be feasible and practical for a human to quickly understand and modify the criteria used by the system to classify messages. Conventional metrics used to evaluate classifiers, such as accuracy and area under the ROC curve, while important, are secondary considerations.

The main contributions of this paper are (1) design and implementation of a system for improving email security using fuzzy rules and (2) an experimental study of the effectiveness and efficiency of such a system on a well known real dataset.

Section II briefly describes the rule-based system and its prototype implementation, while also highlighting the benefits of a fuzzy rule based system in the context of the above goals. Section III highlights some results of an experimental study of this method. Related work is noted in Section IV and conclusions and ongoing work are addressed in Section V.

II. Rule-based defense

A. Feature extraction

For both training and operational phases of the system, the first step is feature-extraction: computing the values of several attributes from email messages. The choice and computation of attributes are similar to those used by popular email analysis software such as Spamassassin. The attributes are derived, broadly, from three sources: message body, headers, and envelope data. An example of a message body-derived attribute is the average length of maximal substrings composed of all-uppercase letters. An example of a header-derived attribute is the presence of a valid DKIM signature, while an attribute such as the sender's domain comes from the envelope.

Two important considerations for feature-extraction from potential malicious email messages are: safety and efficiency. A malicious adversary must be assumed to be cognizant of the tools used to analyze messages, and to have access the relevant source code. Care must be exercised to ensure that a specially crafted message cannot be used to cause the extraction system to misbehave (crash, execute unexpected code, etc.) Efficiency is important because feature extraction often entails parsing message bodies into words or other fragments, which can be slow but needs to be fast enough to keep up with the offered email load.

B. Training and operation

The first part of the training phase, which is rerun periodically, the rule-based classifier is trained using a combination of curated datasets (known malicious messages) and more recent messages flagged as malicious or benign by users or administrators. This part is almost fully automated and is quite conventional. However, it is followed by a second part in which the trained classifier is examined and potentially modified by a human (with machine assistance).

The primary goal of the second part is to simplify and rationalize the criteria used by the classifier in making its decisions. The key idea here is that a simple, well understood, and easily modifiable method is preferable to an opaque one even if the latter has higher accuracy by some metrics (which certainly need not be the case, since rule-based classifiers are quite competitive).

This design criterion also motivates the use of fuzzy rules over other kinds of classifiers. First, classifiers that are essentially opaque, such as multilayer perceptrons are unsuitable for this purpose. Other classifiers, such as those based on decision trees are more suitable, since one can potentially examine the tree to gain an understanding of the scheme. However, in practice the trees are often too large and complex for such examination to be effective. Rule-based classifiers, whether fuzzy or crisp, tend to provide shorter descriptions of their classification and thus are attractive. The primary advantage of fuzzy rules over crisp rules in this regard is that they better match an intuitive model of the boundary between problematic and benign values of attributes. Other advantages are potentially smaller rule sets due to ideas such as rule-stretching to accommodate instances that would be otherwise uncovered.

C. Implementation

The prototype implementation runs on the Java Virtual Machine (JVM) platform and is implemented in the Kawa Scheme language, along with the WEKA workbench. Testing and experiments are conducted with OpenJDK 1.8 on a Debian GNU/Linux 9 (Stretch) system, but there are very few system-specific dependencies. Tasks such as feature extraction, communication, and user interaction are implemented in Scheme. The core classification tasks on the processed data are performed using the programmatic interface to the WEKA workbench and its implementations of several popular algorithms. Similarly, evaluation of classifiers using 10-fold cross validation is also performed using WEKA with pre- and post-processing in Scheme. The classifier based on fuzzy rules is built using the FURIA package. (Algorithmic and other details on FURIA appear in the paper introducing it [1].) For comparison, a classifier based on crisp rules is built using the WEKA's JRip implementation of the RIPPER algorithm. Classifiers based on decision tables, C4.5-style trees, and others are also used.

III. Experimental study

A. Datasets

In order to study the performance of the above methods in practice, it is important to use a real dataset. Unfortunately, there is a dearth of datasets that include email messages categorized by phishing or other threats, partly because of the often sensitive nature of such messages. Therefore, we use a proxy dataset that is likely to share key characteristics with the desired ones: viz., the well-documented *spambase* dataset [2]. The spambase dataset consists of attributes derived from email messages received by a small group of people. (The messages themselves are not present.) Each message has been tagged as spam or not-spam (a.k.a. ham) by a human. There are 58 attributes in all. Attributes 1–48 are the relative word frequencies (as percentages) of the following 48 words, in order:

```
make address all 3d our over remove internet order
mail receive will people report addresses free
business email you credit your font 000 money
hp hpl george 650 lab labs telnet 857 data 415
85 technology 1999 parts pm direct cs meeting
original project re edu table conference
```

The percent relative frequency of a word is defined as 100 times the number of occurrences of that word divided by the total number of words in the message. Attributes 49–54 are the relative character frequences (as a percentage) of the six characters: `;`, `(`, `[`, `!`, `$`, and `#`. Similar to word frequencies, the percent relative frequency of a character is defined as 100 times the number of occurrences of that character divided by the total number of characters in the message. Attributes 55–57 relate to the length of runs of uppercase letters, and denote the average, maximum, and total lengths of such runs. They are expressed as absolute values (not as ratios). Finally, attribute 58 is a nominal attribute that is 1 if the message is regarded as spam and 0 otherwise.

B. Initial processing

Approximately 60% of the messages are non-spam, so a *ZeroR* classifier that always classifies a message as non-spam gets 60% accuracy as a baseline. As another baseline, the slightly more discriminating *OneR* classifier, which classifies messages using a single attribute that provides the best results, yields an accuracy of approximately 78% on the spambase data. Examining the trained OneR classifier reveals that the chosen attribute is the one that measures the frequency of the "`!`" character.

In order to evaluate classifiers on inputs of varying levels of difficulty we generate a number of additional datasets from the base spambase dataset. This task is the second purpose of using the OneR classifier (in addition to providing a baseline, with ZeroR), In particular, we generate a sequence of datasets D_i, $i = 0, 1, 2, \ldots, 16$. as follows. The original dataset is used as D_0. For $i > 0$, D_i is obtained by removing from D_{i-1} the attribute chosen by the OneR classifier trained on D_{i-1}.

The following list summarizes the attributes thus removed, in order, using the codes `cf`, `cr`, and `wf` for character frequency, capital-letter run, and word frequency, respectively. For example, `cf_$` is the frequency of the $ character, `wf_money` is the frequency of the word "money," and `cr_average` is the average length of a run of capital (upper-case) letters in the message.

```
cf_! cf_$ cr_average wf_your
wf_free cr_longest wf_remove cr_total
wf_money wf_you wf_our wf_000
wf_all wf_business wf_receive wf_mail
```

Although experiments were conducted on all these datasets, for brevity we report the results on three selected ones: D_0, D_8, and D_{16}. In addition to the FURIA fuzzy rules classifier, we report the results of two other rule-based classifiers: JRip and PART.

C. Untrimmed data

The first set of results is summarized by Figs. 1 to 4. Figure 1 plots the number of rules generated by the three classifiers as a function of a key parameter: *minimum weight*, which controls the minimum weight of instances in a rule. Note that the horizontal axis is logarithmic and that some unusually high values of the parameter are used to explore the effect more fully. Classifiers with fewer rules are preferred in general because they are easier to comprehend and edit.

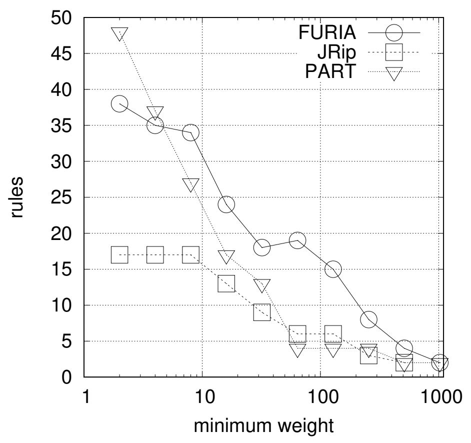

Fig. 1. Effect of minimum-weight parameter on number of rules generated by FURIA, JRip, and PART on the base dataset D_0.

Not surprisingly, the number of rules generated by all three classifiers trends downward with increasing minimum weight per rule. However, this trend is not regular, or even monotonic. Further, in the main portion of the experiments (low to moderate values of minimum-weight), the JRip classifier yields significantly fewer rules than the other two. Part of this parsimony is likely the result of the ordered nature of JRip rules compared with the unordered FURIA rules.

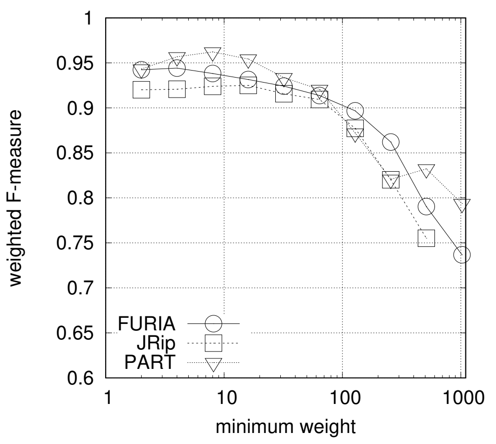

Fig. 2. Effect of minimum-weight parameter on F-measure of the results of FURIA, JRip, and PART on the base dataset D_0.

Figures 2 to 4 summarize effect of the minimum-weight parameter on three key quality metrics for the classifiers: F-measure, area under the ROC (receiver-operator characteristic) curve, and area under the PRC (precision-recall curve). While there are small differences across the metrics and classifiers, overall the curves are very similar: There is an initial small improvement in quality as minimum-weight is increased to around 10 followed by a gradual and then rapid decline. It is useful to compare the points of inflection of these plots with the plot in Fig. 1 to determine suitable parameter settings. For example, for FURIA, a value of approximately 30 is a good choice because it provides high quality metrics as well as a low number of rules.

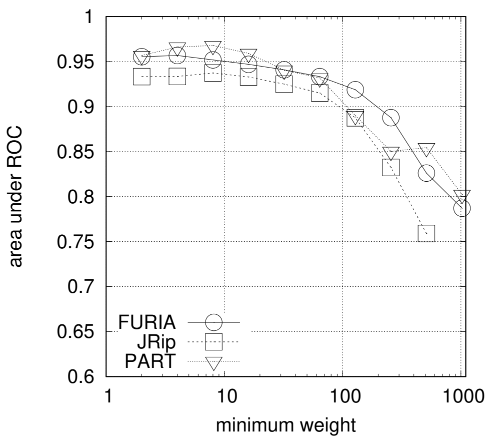

Fig. 3. Effect of minimum-weight parameter on the area under the ROC curve for FURIA, JRip, and PART on the base dataset D_0.

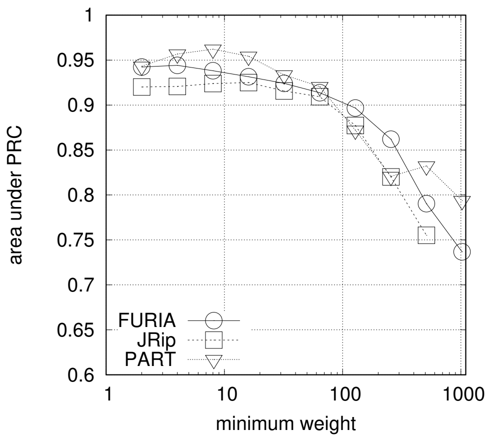

Fig. 4. Effect of minimum-weight parameter on the area under the PRC curve for FURIA, JRip, and PART on the base dataset D_0.

D. Trimmed data

Figures 5 to 8 are analogous to Figs. 1 to 4 and summarize the results on the dataset D_8, i.e, the dataset obtained from D_0 by removing the attribute chosen by the OneR classifier iteratively eight times. The plots of the quality metrics in Figs. 6 to 8 are in general similar to the corresponding plots for the D_0 dataset in Figs. 2 to 4 except that the values are typically lower. However, the effect of the eight missing attributes on the quality metrics is more significant for FURIA and JRip than for PART. The difference between Fig. 5 and Fig. 1 is more significant. On the D_8 dataset, all three classifiers exhibit an initial increase in the number of rules with increasing values of the minimum-weight parameter, although the effect is most pronounced for the FURIA classifier.

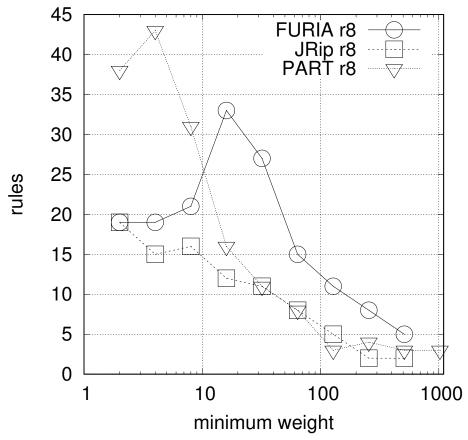

Fig. 5. Effect of minimum-weight parameter on number of rules generated by FURIA, JRip, and PART, with 8 attributes trimmed from the dataset.

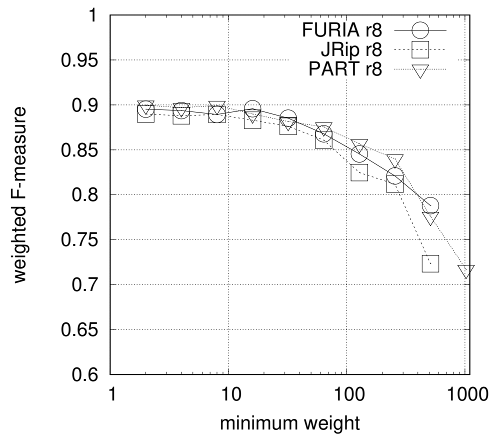

Fig. 6. Effect of minimum-weight parameter on F-measure of the results of FURIA, JRip, and PART, with 8 attributes trimmed from the dataset.

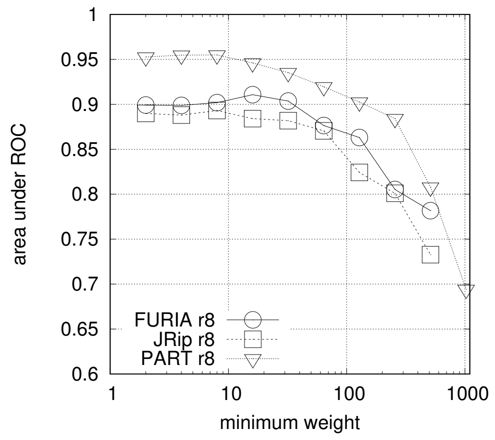

Fig. 7. Effect of minimum-weight parameter on the area under the ROC curve for FURIA, JRip, and PART, with 8 attributes trimmed from the dataset.

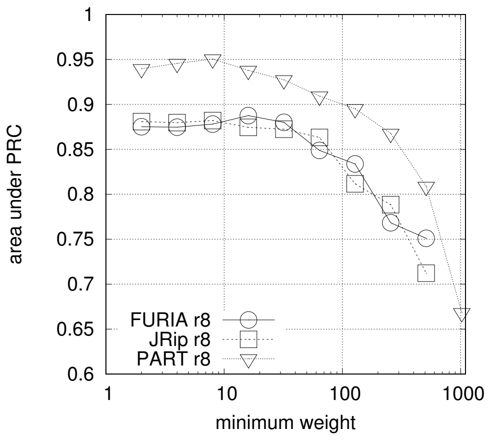

Fig. 8. Effect of minimum-weight parameter on the area under the PRC curve for FURIA, JRip, and PART, with 8 attributes trimmed from the dataset.

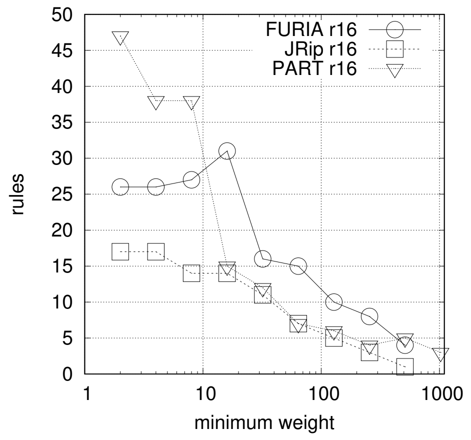

Fig. 9. Effect of minimum-weight parameter on number of rules generated by FURIA, JRip, and PART, with 16 attributes trimmed from the dataset.

Figures 9 to 12 depict results corresponding to Figs. 5 to 8 for the D_{16} dataset. The observations made for the D_8 dataset apply here as well, to a progressively greater extent.

Figure 13 summarizes the results of experiments studying the effect of successive attribute-removal on four classifiers: In addition to FURIA and JRip, the experiments include the simpler decision-table classifier DTab [3] and a standard C4.5-style decision-tree classifer J48 [4]. A general trend in these results is that the performance of FURIA and JRip is similar, higher than DTab but lower than J48. At this juncture, it may seem J48 should be preferred over the other classifiers. However, it is important to note that classifier quality metrics such as these form only one aspect of the overall objective. Another important aspect is the ease with which a human may comprehend and modify a classifier. In this regard, FURIA and JRip have significant advantages over J48 and many other classifiers because the rules they produce are typically both very comprehensible and manageable in number (Fig. 14); in

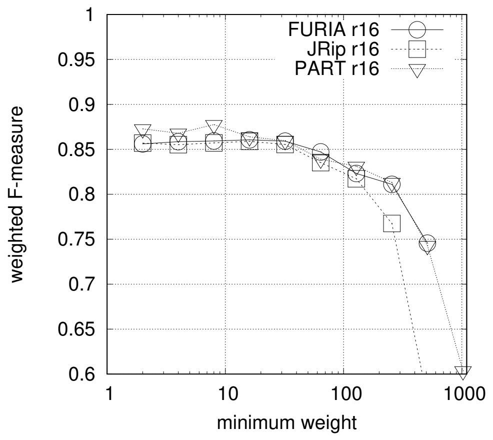

Fig. 10. Effect of minimum-weight parameter on F-measure of the results of FURIA, JRip, and PART, with 16 attributes trimmed from the dataset.

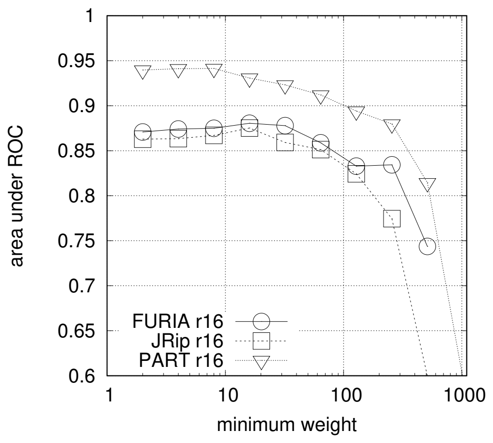

Fig. 11. Effect of minimum-weight parameter on the area under the ROC curve for FURIA, JRip, and PART, with 16 attributes trimmed from the dataset.

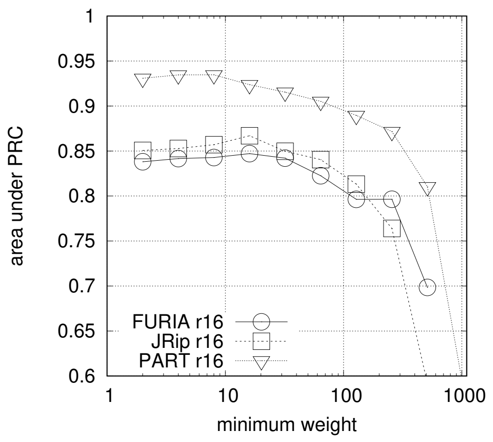

Fig. 12. Effect of minimum-weight parameter on the area under the PRC curve for FURIA, JRip, and PART, with 16 attributes trimmed from the dataset.

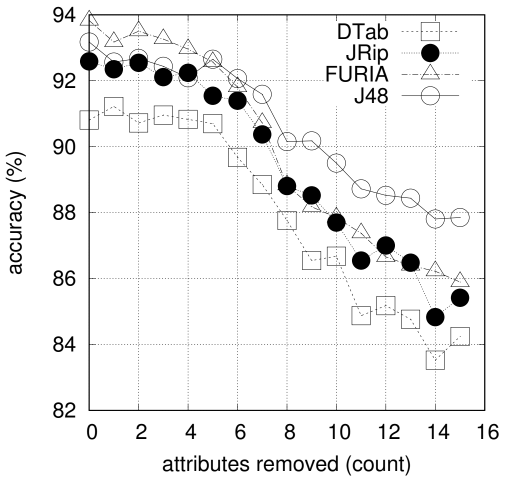

Fig. 13. Change in classification accuracy (weighted mean by class) with removal of OneR-selected attributes from base dataset.

contrast, the trees produced by J48 on similar inputs have over 200 nodes.

IV. Related Work

The need to address phishing and related email threats has been the motivation of much prior work on the topic [5]–[9]. Similarly, email spam has been studied from diverse perspectives, including technical, economic, and social [10]–[15]. Descriptions and analyses of the FURIA algorithm for unordered fuzzy rule induction [1], [16] have indicated performance competitive with other classification algorithms. Our results in Section III are consistent with these observations, although our primary motivations are different. The JRip classifier is based on the RIPPER algorithm [17] and the PART classifier is based on an algorithm that produces compact rules without using global optimization [18]. These and several other classifiers are implemented in the WEKA workbench which was also used in the experiments reported here [19], [20].

V. Conclusion

We have motivated an email analysis system based on fuzzy rules for detecting phishing, spam, and other malicious messages. An important benefit of using classifiers based on fuzzy rules is that they are amenable to human understanding and modification. We studied the performance of the classifier on a real dataset and its trimmed variants and compared it to other similar classifiers. The performance is competitive with the best classifiers in the experiments, although higher accuracy may be achieved using alternatives in some cases. However, raw accuracy of an unattended classifier is not the primary metric of interest. The focus is on the overall performance with a human in the loop, for which ease of understanding is the key factor, albeit harder to quantify. Examination of the rules generated by the fuzzy-rules classifier on actual data suggest that rules may be understood and modified with only modest

```
(wf_hp in [0.09, 0.1, inf, inf]) => spam=0 (CF = 0.96)
(wf_our in [-inf, -inf, 0, 0.43]) and (wf_george in [0.2, 0.23, inf, inf]) => spam=0 (CF = 1.0)
(wf_our in [-inf, -inf, 0.04, 0.12]) and (wf_edu in [0.04, 0.09, inf, inf]) => spam=0 (CF = 0.97)
(wf_you in [-inf, -inf, 0.11, 0.15]) and (cf_( in [-inf, -inf, 0, 0.004]) => spam=0 (CF = 0.9)
(wf_money in [-inf, -inf, 0, 0.18]) and (wf_george in [0, 0.01, inf, inf]) => spam=0 (CF = 1.0)
(wf_money in [-inf, -inf, 0, 0.04]) and (wf_our in [-inf, -inf, 0.2,
0.21]) and (wf_000 in [-inf, -inf, 0.11, 0.13]) and
    (wf_business in [-inf, -inf, 0.08, 0.1]) => spam=0 (CF = 0.85)
(wf_money in [0, 0.02, inf, inf]) and (wf_receive in [0.05, 0.06, inf, inf]) => spam=1 (CF = 0.98)
(wf_our in [0.19, 0.21, inf, inf]) and (wf_hp in [-inf, -inf, 0, 0.21]) and
    (wf_business in [0, 0.02, inf, inf]) => spam=1 (CF = 0.96)
(wf_hp in [-inf, -inf, 0.11, 0.13]) and (wf_our in [0.08, 0.09, inf, inf]) and (wf_mail in [0, 0.05, inf, inf]) and
    (wf_edu in [-inf, -inf, 0, 0.2]) and (wf_george in [-inf, -inf, 0, 0.01]) => spam=1 (CF = 0.97)
(wf_hp in [-inf, -inf, 0.09, 0.2]) and (wf_000 in [0.04, 0.06, inf, inf]) => spam=1 (CF = 0.96)
(wf_hp in [-inf, -inf, 0.05, 1.27]) and (wf_you in [0.74, 0.76, inf, inf]) and (wf_money in [0, 0.09, inf, inf]) and
    (wf_re in [-inf, -inf, 0.2, 0.23]) => spam=1 (CF = 0.97)
(wf_hp in [-inf, -inf, 0.09, 0.11]) and (wf_george in [-inf, -inf, 0, 0.16]) and (wf_edu in [-inf, -inf, 0, 0.09]) and
    (wf_you in [0, 0.07, inf, inf]) and (wf_our in [0.1, 0.14, inf, inf]) => spam=1 (CF = 0.92)
(wf_hp in [-inf, -inf, 0.09, 0.19]) and (wf_george in [-inf, -inf, 0, 0.01]) and (wf_edu in [-inf, -inf, 0, 0.19]) and
    (wf_you in [0, 0.05, inf, inf]) and (wf_email in [0, 0.09, inf, inf]) and (wf_will in [0.28, 0.3, inf, inf]) =>
    spam=1 (CF = 0.94)
(wf_hp in [-inf, -inf, 0.09, 0.13]) and (wf_george in [-inf, -inf, 0, 0.08]) and (wf_edu in [-inf, -inf, 0, 0.09]) and
    (wf_business in [0.05, 0.1, inf, inf]) => spam=1 (CF = 0.96)
(wf_hp in [-inf, -inf, 0.38, 0.39]) and (wf_george in [-inf, -inf, 0, 0.07]) and (wf_edu in [-inf, -inf, 0, 0.12]) and
    (wf_will in [-inf, -inf, 0, 2.4]) and (cf_( in [0, 0.082, inf, inf]) and (cf_( in [-inf, -inf, 0.215, 0.216]) =>
    spam=1 (CF = 0.81)
```

Fig. 14. The verbatim set of unordered fuzzy rules produced by FURIA on the D_8 dataset. The attribute-name prefixes `cf_`, `cr_`, and `wf_` refer to character frequency, capital-letter run, and word frequency, respectively, of the corresponding suffixes. In the notation $[a, b, c, d]$, the four values anchor the vertices of a trapezoidal fuzzy interval (lower left, upper left, upper right, lower right).

human effort. The experiments suggest that performance is close enough to the best-performing classifier in all cases so that there is no significant penalty incurred in preferring a fuzzy rule-based system. In ongoing work, we are expanding the experimental study to larger and more diverse datasets and quantifying the load-handling capacity of the system.

Acknowledgments

The paper benefited from the referees' careful comments. The work was supported in part by the US National Science Foundation grants EAR-1027960 and PLR-1142007

References

[1] J. Hühn and E. Hüllermeier, "FURIA: an algorithm for unordered fuzzy rule induction," *Data Mining and Knowledge Discovery*, vol. 19, no. 3, pp. 293–319, Dec 2009. [Online]. Available: https://doi.org/10.1007/s10618-009-0131-8

[2] M. Hopkins, E. Reeber, G. Forman, and J. Suermondt, "SPAM e-mail database," Jun. 1999, hewlett-Packard Labs, Palo Alto, CA.

[3] R. Kohavi, "The power of decision tablesa," in *Proceedings of the 8th European Conference on Machine Learning (ECML)*, Apr. 1995, pp. 174–189.

[4] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[5] A. Almomani, B. B. Gupta, S. Atawneh, A. Meulenberg, and E. Almomani, "A survey of phishing email filtering techniques," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 2070–2090, 2013, fourth quarter.

[6] M. Volkamer, K. Renaud, B. Reinheimer, and A. Kunz, "User experiences of torpedo: Tooltip-powered phishing email detection," *Computers & Security*, vol. 71, pp. 100–113, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404817300275

[7] B. Harrison, E. Svetieva, and A. Vishwanath, "Individual processing of phishing emails: How attention and elaboration protect against phishing," *Online Information Review*, vol. 40, no. 2, pp. 265–281, 2016. [Online]. Available: https://doi.org/10.1108/OIR-04-2015-0106

[8] M. U. Chowdhury, J. H. Abawajy, A. V. Kelarev, and T. Hochin, "Multilayer hybrid strategy for phishing email zeroday filtering," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 23, p. e3929, 2017, e3929 cpe.3929. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.3929

[9] J. Wang, T. Herath, R. Chen, A. Vishwanath, and H. R. Rao, "Research article phishing susceptibility: An investigation into the processing of a targeted spear phishing email," *IEEE Transactions on Professional Communication*, vol. 55, no. 4, pp. 345–362, Dec. 2012.

[10] W. Z. Khan, M. K. Khan, F. T. B. Muhaya, M. Y. Aalsalem, and H. C. Chao, "A comprehensive study of email spam botnet detection," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2271–2295, Fourth quarter 2015.

[11] D. Wang, D. Irani, and C. Pu, "A study on evolution of email spam over fifteen years," in *9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, Oct 2013, pp. 1–10.

[12] T. C. Loder, M. W. V. Alstyne, and R. Wash, "Information asymmetry and thwarting spam," Social Science Research Network Library http://ssrn.com/., Jan. 2004.

[13] J. M. Rao and D. H. Reiley, "The economics of spam," *The Journal of Economic Perspectives*, vol. 26, no. 3, pp. 87–110, 2012. [Online]. Available: http://www.jstor.org/stable/41581133

[14] W. Liu and T. Wang, "Utilizing multi-field text features for efficient email spam filtering," *International Journal of Computational Intelligence Systems*, vol. 5, no. 3, pp. 505–518, 2012.

[15] Y. Shao, M. Trovati, Q. Shi, O. Angelopoulou, E. Asimakopoulou, and N. Bessis, "A hybrid spam detection method based on unstructured datasets," *Soft Computing*, vol. 21, no. 1, pp. 233–243, Jan 2017. [Online]. Available: https://doi.org/10.1007/s00500-015-1959-z

[16] J. C. Hühn and E. Hüllermeier, *An Analysis of the FURIA Algorithm for Fuzzy Rule Induction*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 321–344. [Online]. Available: https://doi.org/10.1007/978-3-642-05177-7_16

[17] W. W. Cohen, "Fast effective rule induction," in *Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann, 1995, pp. 115–123.

[18] E. Frank and I. H. Witten, "Generating accurate rule sets without global optimization," in *Proceedings of the Fifteenth International Conference on Machine Learning*, J. Shavlik, Ed. Morgan Kaufmann, 1998, pp. 144–151.

[19] E. Frank, M. A. Hall, and I. H. Witten, "The WEKA workbench." 2016, online appendix for Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, Fourth Edition, 2016. http://www.cs.waikato.ac.nz/ml/weka/Witten_et_al_2016_appendix.pdf.

[20] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The WEKA data mining software: An update," vol. 11, pp. 10–18, 11 2008.