Combining RDR-based Machine Learning Approach and Human Expert Knowledge for Phishing Prediction

Hyunsuk Chung, Renjie Chen, Soyeon Caren Han, and Byeong Ho Kang

School of Engineering and ICT,
Tasmania 7005, Australia
{David.Chung, renjiec, Soyeon.Han, Byeong.Kang}@utas.edu.au

Abstract. Detecting phishing websites has been noted as complex and dynamic problem area because of the subjective considerations and ambiguities of detection mechanism. We propose a novel approach that uses Ripple-down Rule (RDR) to acquire knowledge from human experts with the *modified RDR* model-generating algorithm (Induct RDR), which applies machine-learning approach. The modified algorithm considers two different data types (numeric and nominal) and also applies information theory from decision tree learning algorithms. Our experimental results showed the proposing approach can help to deduct the cost of solving over-generalization and over-fitting problems of machine learning approach. Three models were included in comparison: RDR with machine learning and human knowledge, RDR machine learning only and J48 machine learning only. The result shows the improvements in prediction accuracy of the knowledge acquired by machine learning.

Keywords: phishing prediction, RDR; knowledge-based system; machine learning; decision tree

1 Introduction

An accelerative growth of Internet-based financing increases online fraudulent activity in which malicious people tries to reveal sensitive information of Internet users, also called as phishing. Phishing detection has received great attention but there has been limited research on a way of overall success due to the nature of problems. The problems of detecting phishing websites are very complex and hard to analyze as technical and social problems are joining each other [1]. Either machine learning technique and human expert system has been applied to acquire and maintain the knowledge for phishing website detection and prediction while the results do not show significance. A large number of knowledge-based systems are built for acquiring and maintaining the knowledge for detecting and predicting the phishing website. Phishing website detection knowledge was originally acquired from domain experts. However, acquiring knowledge from an expert in a slow pace cannot meet the demand of the expanding systems since a sophisticated expert system may require an extremely large number of rules. This leads to machine learning based approach as a solution to manage knowledge-based systems. Although machine learning technique can acquire knowledge from data without the help of a domain expert and an abundance of classifier models exist and decision tree based algorithms provide the best performance, over-generalization and over-fitting are still significant problems when sufficient training data are not available so there are not enough patterns which can be found by machine learning. Therefore, large effort usually has to be undertaken to cover those abnormal cases arising from this problem and the cost usually results in repeating reconstruction of the knowledge base [4, 5]. The aim of the research is to find a way to optimize the knowledge acquired by machine learning to deduct the large cost of solving over-generalization and over-fitting problems. Machine learning is used because acquiring human expert knowledge becomes insufficient when a system expands swiftly. In the opposite direction, if machine learning is not perfect, human knowledge might help to improve the knowledge acquired by machine learning. We assume combining two different mechanisms of having machine learning and expert system-style knowledge acquisition will optimize knowledge engineering process. Hence, we focused on developing phishing website detection model by applying Induct RDR approach. The proposed induct RDR (Ripple Down Rules) approach allows to acquire the phishing detection knowledge by machine learning, and maintained by human domain expert

2 Related Work

2.1 Human Knowledge Acquisition

Human knowledge is the knowledge acquired from a domain expert to manage a related knowledge-based system. In typical knowledge-based systems or expert systems, transferring knowledge of the experts is bottleneck of building knowledge base for the systems. This is because the process of transferring expert knowledge into knowledge base of the system requires rich resources and the knowledge engineer requires to fully-understand the domain expert knowledge to construct the knowledge base [6]. These knowledge-based systems are built with large structures of concepts and rules, and the difficulty of interacting new arising circumstances with existing rules. In Ripple-down Rules (RDR), its unique knowledge acquisition process solves the problems lie on knowledge engineering process. RDR is built with rules of hierarchical exceptions [16]. It is a knowledge acquisition and representation technique that allows knowledge of a certain domain to be interpreted as rules. The RDR structure is a finite binary tree where each node can have two distinct branches, which are called *except* and *if-not*. Cases are evaluated from the root node of the RDR tree. Each node in the tree is a rule with the form of if α then β (α is the condition and β is the conclusion). When the system encounters an incorrect classification, a new exception rule is added based on experts' judgment [7] with the given case. Therefore, RDR can incrementally develop a relatively accurate knowledge base, provided the domain is fixed and the experts provides the correct judgments [8]. Since RDR based knowledge base depends on experts' judgment, the correctness of the used language expressed by the expert is the key of developing a good knowledge base [17]. According to Pham and Hoffmann [8], it may cost a long time to classify most of the relevant cases correctly, if the target is linear threshold in the numerical input space then an expert is only allowed to use axis-parallel cuts, since it is unsuitable for him to express accurately.

2.2 Knowledge Acquisition by Machine Learning.

Knowledge is traditionally collected as rules through sustained interaction between domain specialists and knowledge engineers. However, acquiring knowledge from an expert in a slow pace cannot meet the demand of the expanding systems since a sophisticated expert system may require an extremely large number of rules. This leads to machine learning based approach as a solution to developing knowledge.

Decision Tree Learning.

According to Quinlan [2], as one of the technology for building knowledge-based systems by learning from cases, decision tree has been demonstrated successfully. Decision tree based classifiers are used in many areas such as radar signal classification, character recognition, remote sensing, medical diagnosis, expert systems, and speech recognition etc. [3]. ID3 (Iterative Dichotomiser 3) is a typical algorithm to synthesize decision trees in knowledge-based systems. Since there are usually many attributes and the training dataset contains many cases, different decision trees can be created from the same dataset. The fundamental idea is to disperse a complicated decision into a collection of several simpler decision, with the final solution resembling the intended desired solution [3]. A subset of the training dataset is chosen randomly to form a decision tree in which all objects in the subset is correctly classified. Then all other objects are classified using the tree. A subset of those objects is chosen randomly in the same way and the process continues. The random selection is based on two assumptions: 1) The probability of an arbitrary object being determined to belong to class P equals to the proportion of class P in the dataset, and 2) The decision tree is generated by the expected information of the chosen objects. The mentioned expected information is measured by entropy, information gain or gain ratio of the features (attributes) of the chosen objects [9] Compared with categorical attributes, numeric attributes seem to be more difficult to evaluate since they are continuous and the threshold can be arbitrary. In C4.5, the training cases are first sorted by values of the attribute. If the number of the attribute values is m, and the values are sorted as $v_1, v_2, \ldots, v_m$, there are only $m - 1$ ways to split the dataset into two subsets. Each possible split is examined by their information gain or gain ratio to find the best split. The midpoint $(vi + vi+1) / 2$, in which vi is the largest value of the first subset and vi+1 is the smallest value of the second, is not chosen for the threshold in C4.5. It chooses the largest value of the attribute in the entire training dataset that does not exceed the midpoint [10].

Induct RDR.

Induct RDR was introduced by Gaines when illustrating a fundamental relation between techniques that transfer existing knowledge from human experts and those that create new expertise through machine learning [11]. He mentioned a sequence of dispersing knowledge partially from the view of a human expert, which consists of the following seven stages: 1) Minimal Rules, 2) Adequate Rules, 3) Critical Cases, 4) Source of Cases, 5) Irrelevant Attributes, 6) Incorrect Decisions, and 7) Irrelevant Attributes & Incorrect Decisions. The first stage is a complete, minimal set of correct decision rules so no data is required for knowledge acquisition since the correct answer is available from the expert. On the contrary, the last stage is a source of data from which the correct answer might be derived with the greatest probability of correct decisions so the expert has provided little. The stages in the middle from top to bottom show a decrease in existing knowledge though human intervention but an increase in new expertise through machine learning [11]. The main use of existing RDR is close to the top stage. Therefore, Induct RDR which derives rules directly from an extension of Cendrowska's Prism algorithm [12] was made to be close to the bottom. This Induct RDR sums standard binomial distribution as the possibility of selecting correct data at random to measure the correctness of a rule. In supervised learning, there is a risk of over-fitting the noise by memorizing the peculiarities of the training data [4]. Pruning methods are commonly applied to solve the problem. Although Induct RDR recognizes the importance of pruning, it only removes redundant clauses and compresses the structure to some extent. Reducing over-fitting and improving generalization prediction capability has not been considered [13]. Ripple-Down Rules classifier (Ridor) is an implementation of Induct RDR in Weka (Weka is a tool which provides a collection of machine learning algorithms). It first creates the default rule. The exceptions are created for the default rule with the lowest (weighted) error rate [14]. Different from the original Induct RDR, Ridor applies information gain to evaluate each rule and it prunes a rule by reduced error pruning.

3 Methodology

3.1 Overview

Although machine learning can acquire knowledge from data without the help of a domain expert, over-generalization and over-fitting is still a significant problem when sufficient training data are not available. Because insufficient data may end up with small amount of patterns that cannot be used for rule generation. Therefore, it takes a huge effort to cover abnormal cases arising from the problem [4, 5]. If machine learning is not perfect, human knowledge might help to improve the knowledge base constructed by machine learning approach. In order to see the results following experiments were conducted:

- Although human knowledge acquired can be incorrect or different among different experts, but the knowledge base built can still be further refined. Therefore, for the experiment we allow that the human knowledge we use is not perfect.
- Ideally speaking, when given all the patterns, we can obtain the correct knowledge base by machine learning. However, it is practically not very possible. For the experiment, we consider the dataset we use as whole is sufficient in patterns and the knowledge based on the entire dataset is correct.
- Knowledge based on part of the dataset tends to be incorrect because insufficient data results in difficulties in finding all necessary patterns.

The flow of how to combine human knowledge and machine learning is illustrated in the following way:

- Generate rules by machine learning via training dataset. (Modified Induct RDR)
- Given testing dataset, find out incorrectly classified data.
- Acquire rules from the expert.

- Use those rules (human knowledge) to add exception rules for the machine learning rules where data are incorrectly classified.

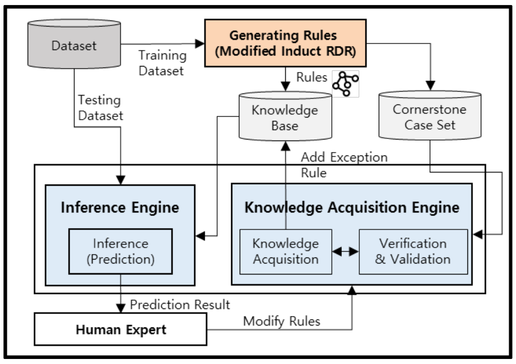


Fig. 1. The flow of combining human knowledge and machine learning

If we need to use human knowledge to improve machine learning, we need to find an intermediate between them. That is the reason that we propose RDR as the method.

3.2 Induct RDR Modification (modified Induct RDR).

The modified Induct RDR is based on Induct RDR algorithm that is the third generation of a family of rule induction algorithms [12]. The big picture of the algorithm is a rule generating function where a RDR structure is constructed through the process. A rule at one single RDR node is called a clause and a clause is a collection of one or more terms with the form of attribute-relation-value. Rule generating function follows the following three steps:

- The class that occurs most frequently in the training data is selected as the default class value for the top-level empty rule.
- It finds the clause at each node by searching any class other than the default class that has the smallest m-value (based on standard binomial distribution) to split the training set into two subsets: all true cases for the rule, and all false cases.
- If either of these subsets contains more than one class, rule-generating function is called recursively on the subset. The selected class is used on the first subset as the default class and the current default class is used on the second subset.

Best Clause Selection

The best clause function is the core of Induct RDR algorithm. Given a specified class, the original function searches all possible combinations of terms to find a particular set of terms which fit the best for covering the class. m function is called to assess the quality of a term. The result of m function is called m-value. The number of cases selected by the combination of terms and z is the number of those examples that are actually needed (true positive). Terms are assessed and qualified terms are added to the clause (combination) until it only selects true positive examples (when z = s). However this procedure might be very computationally intensive because when the number of attributes is large, the number of combinations increases exponentially. In order to solve the problem, terms are first ordered in the modified Induct RDR. Since m-values are the criteria for whether or not adding a term to the clause and the m-value should be minimized, terms can be sorted by m-value in ascending order. Therefore, the possible best terms will be always combined and assessed first, which contributes to finding the best clause within shorter time. This way is much more efficient but it does not prefer combinations such as a good term with a bad term. It might miss the best clause formed by a good term combined with a bad term, but if this combination does not perform better than others, the impact is small.

Best Clause Evaluation

At the end of best clause function, terms are removed from the clause until the m-value is minimized (Induct RDR pruning). Due to the above change, there is no need to search every combination so this part has been moved into the loop to stop the searching at an early age if a new term makes the clause worse than before adding it. The final part of the algorithm is m function, which assesses the correctness of the combination of terms (the clause). m function uses the following formula which sums the standard binomial distribution to calculate m-value.

$$m'(S) = \sum_{i=z}^{\min(s,k)} {}^{s}C_{i}\left(\frac{k}{n}\right)^{i}\left(1-\frac{k}{n}\right)^{s-i} \tag{1}$$

n is the number of the whole training set.

k is the number of data which need to be selected.

s is the number of data which are actually selected.

z is the number of data which are correctly selected.

According to Gaines [11], the advantage of using m-value as a measure of the correctness of a rule (terms or clause) is because the probability that the rule could be this good at random, and that it involves no assumptions about the problem such as sampling distributions. He also points out when s = k in the above formula which means all data selected are correct, then log(m) = s log(k/n) which seems to be the basis of 'information-theoretic' measures. However, when the dataset is too large, m-values of all rules become to 0, which means that there is no difference in choosing different rules. In this case, the best clauses are just chosen randomly. Verified by decision tree learning algorithms, in fact choosing attributes having larger information gain can still help improving prediction accuracy. In the modified Induct RDR, when the training dataset is too large so that m-value equals to 0, information gain is used to evaluate the best clause instead.

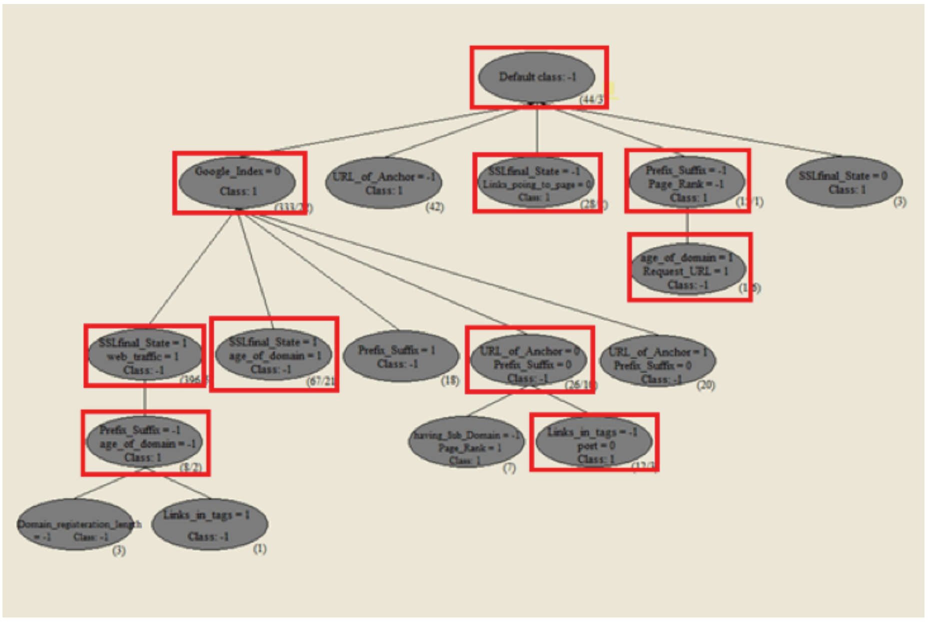
Numeric Number Handling

When the dataset is too large, m-values of all rules become to 0, which means that there is no difference in choosing different rules. In this case, the best clauses are just chosen randomly. Verified by decision tree learning algorithms, in fact choosing attributes having larger information gain can still help improving prediction accuracy. In the modified Induct RDR, when the training dataset is too large so that m-value equals to 0, information gain is used to evaluate the best clause instead. In the original ID3 algorithm, information gain is used not only to divide the numeric data but also to generate classification rules to make the decision tree through both nominal and numeric data. Due to the different nature of Induct RDR algorithm, the method can only be used to handle the numeric data; hence the following two points were proposed.

- Decision tree focuses on each attribute so it only has one rule (term) at each node while Induct RDR focuses on the combination of terms so it has a clause, which contains one or more terms.
- Induct RDR already uses m-value to measure the quality of the clause so it is not suitable to use another method.

3.3 Knowledge Acquisition.

For the dataset used in the research, the actual human experts are less available so we propose to use a simulated expert. Apart from the knowledge acquired by RDR, the knowledge acquired from the simulated expert is treated as human knowledge and the incorrectly or insufficient conclusions will be replaced or supplemented. The whole dataset is divided into a training dataset and a test dataset. The training dataset is used to generate the original knowledge base. The test dataset is used to examine the prediction accuracy of the knowledge. Figure 2 shows the RDR rule tree, where red-circled nodes indicate those nodes with poor prediction of accuracy.



Default class: -1 (44/3
Google_Index = 0 Class: 1 (333/2
URL_of_Anchor = -1 Class: 1 (42)
SSLfinal_State = -1 Links_poing_to_page = 0 Class: 1 (28
Prefix_Suffix = -1 Page_Rank = -1 Class: 1 (1 1)
SSLfinal_State = 0 Class: 1 (3)
age_of_domain = 1 Request_URL = 1 Class: -1 (1 6)
SSLfinal_State = 1 web_traffic = 1 Class: -1 (396
SSLfinal_State = 1 age_of_domain = 1 Class: -1 (67/21
Prefix_Suffix = 1 Class: -1 (18)
URL_of_Anchor = 0 Prefix_Suffix = 0 Class: -1 (26/1
URL_of_Anchor = 1 Prefix_Suffix = 0 Class: -1 (20)
Prefix_Suffix = -1 age_of_domain = -1 Class: 1 (8/2)
having_Sub_Domain = -1 Page_Rank = 1 Class: 1 (7)
Links_in_tags = -1 port = 0 Class: 1 (12/
Domain_registeration_length = -1 Class: -1 (3)
Links_in_tags = 1 Class: -1 (1)

Fig. 2. Original RDR rule tree with highlighted nodes to be modified

These nodes should be modified by adding a new branch to the node, deleting the node, or deleting one of the branches of the node. It is supposed that when using the whole dataset, the knowledge acquired is correct so the human knowledge is acquired from the simulated expert based on the whole dataset. In figure 3, the highlighted nodes are the rules, which have same or similar conditions as the ones above.

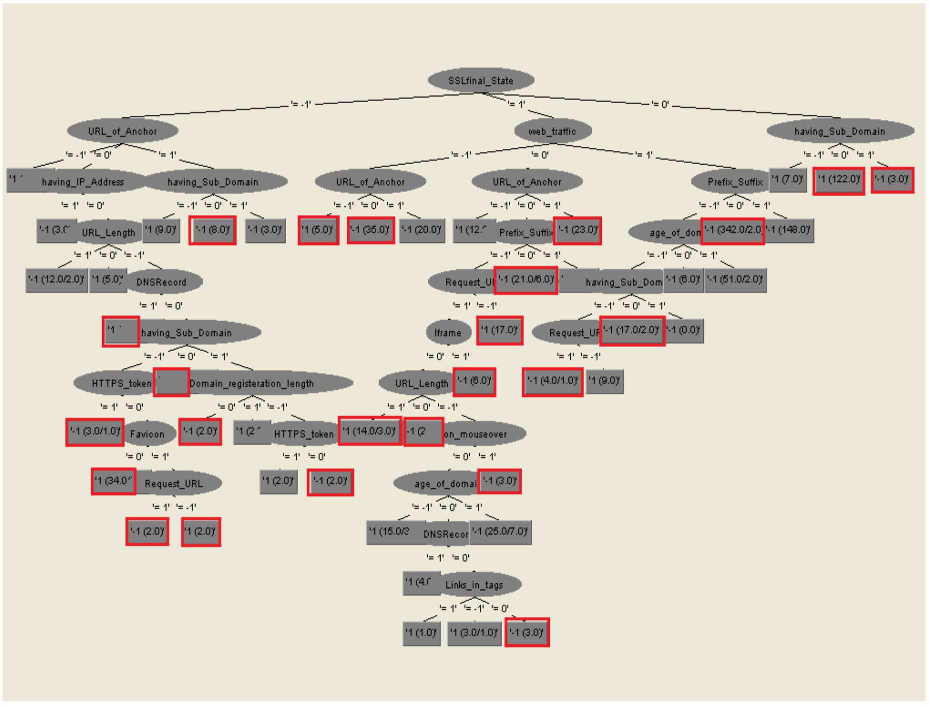


Fig. 3. Knowledge tree built from a simulated expert

Not all of the human knowledge can be applied. There are two reasons summarized in the following list.

- There are data, which have the same vector of attributes but belong to different classes. This is because the existing attributes are not enough to tell the difference. Therefore, the class which the majority belong to will be decided as the conclusion and it is less possible to correct the minority.
- Some rules applied might affect other correctly classified data. The knowledge created by the simulated expert gives a hint about how these rules affect the whole dataset. If a rule has more incorrectly classified data than correctly classified data, it should not be applied.

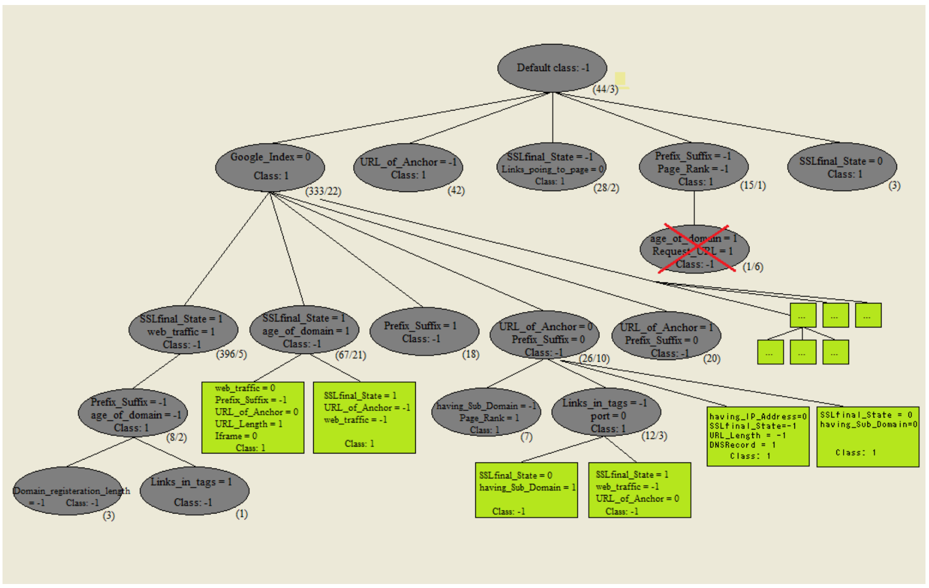

Default class: -1
(44/3)
Google_Index = 0 Class: 1
(333/22)
URL_of_Anchor = -1 Class: 1
(42)
SSLfinal_State = -1 Links_poing_to_page = 0 Class: 1
(28/2)
Prefix_Suffix = -1 Page_Rank = -1 Class: 1
(15/1)
SSLfinal_State = 0 Class: 1
(3)
~~age_of_domain = 1 Request_URL = 1 Class: -1~~
(1/6)
SSLfinal_State = 1 web_traffic = 1 Class: -1
(396/5)
SSLfinal_State = 1 age_of_domain = 1 Class: -1
(67/21)
Prefix_Suffix = 1 Class: -1
(18)
URL_of_Anchor = 0 Prefix_Suffix = 0 Class: -1
(26/10)
URL_of_Anchor = 1 Prefix_Suffix = 0 Class: -1
(20)
...
Prefix_Suffix = -1 age_of_domain = -1 Class: 1
(8/2)
web_traffic = 0 Prefix_Suffix = -1 URL_of_Anchor = 0 URL_Length = 1 Iframe = 0 Class: 1
SSLfinal_State = 1 URL_of_Anchor = -1 web_traffic = -1 Class: 1
having_Sub_Domain = -1 Page_Rank = 1 Class: 1
(7)
Links_in_tags = -1 port = 0 Class: 1
(12/3)
having_IP_Address=0 SSLfinal_State=-1 URL_Length = -1 DNSRecord = 1 Class: 1
SSLfinal_State = 0 having_Sub_Domain=0 Class: 1
Domain_registeration_length = -1 Class: -1
(3)
Links_in_tags = 1 Class: -1
(1)
SSLfinal_State = 0 having_Sub_Domain = 1 Class: -1
SSLfinal_State = 1 web_traffic = -1 URL_of_Anchor = 0 Class: -1

Fig. 4. RDR rule tree with modified nodes

4 Evaluation

4.1 Dataset Selection

We chose 75 datasets from UCI machine learning repository for evaluating the modified Induct RDR with other machine learning models, including Ridor, C4.5 Decision Tree, and NB Tree. The performance of the RDR algorithm combined with human knowledge was verified for the purpose of our experiment. UCI has been published the training dataset that includes important 31 features in detecting and predicting phishing websites. The training dataset contains 11063 websites [15].

- Features/Attributes: having_IP_Address, URL_Length, Shortining_Service, having_At_Symbol, double_slash_redirecting, Prefix_Suffix, having_Sub_Domain, SSLfinal_State, Domain_registeration_length, Favicon, port, HTTPS_token, Request_URL, URL_of_Anchor, Links_in_tags, SFH, Submitting_to_email, Abnormal_URL, Redirect,on_mouseover, RightClick,popUpWidnow, Iframe, age_of_domain, DNSRecord, web_traffic, Page_Rank, Google_Index, Links_pointing_to_page, and Statistical_report.
- Class: Phishing/Non-Phishing

4.2 The Modified RDR

For the evaluation of the *modified RDR*, we chose the following machine learning models to be included in the comparison. For the *modified RDR*, the minimum number of a subset was set to 2 and the result was collected from its output.

Table 1. Models for Comparison

Model name	Based algorithm
Modified RDR	Induct RDR
Ridor	Induct RDR
J48	C4.5 (decision tree)
NBTree	Naive Bayes & decision tree

The result of comparing the *modified RDR*, Ridor, J48 and NBTree is listed in the following table. It can be concluded that the *modified RDR* works better on 66% of the datasets than Ridor, an existing algorithm based on Induct RDR. Although it is 39% for J48 and 43% for NBTree, some datasets with the same accuracy are not counted. Therefore, at least the *modified RDR* has a comparable performance to J48 and NBTree. Besides, it is an inspiring result that the *modified RDR* performs best on 30% (more than one out of four) of the entire datasets.

Table 2. Comparison Result

When	Number of datasets	Proportion
Modified RDR is the best	22	30% (out of 75 datasets)
Modified RDR is better than Ridor	45	66% (out of 68 datasets)
Modified RDR is better than J48	29	39% (out of 75 datasets)
Modified RDR is better than NBTree	27	43% (out of 63 datasets)

4.3 Combining Human Knowledge with Machine Learning

In order to solve over-generalization and over-fitting problems, which usually affect the prediction accuracy of the knowledge base when unrecognized patterns occur, new knowledge needs to be added to the knowledge base.

The first task of this evaluation was to compare the prediction accuracy between the hybrid way (combing human knowledge and machine learning) and the pure machine learning way. It was because it would become meaningless if adding human knowledge did not help improving prediction. Prediction accuracy of three different models (algorithms) were compared each other: RDR machine learning modified by human knowledge, RDR machine learning only, and J48.

The second task was adding new knowledge to the existing knowledge base. In a knowledge-based system, knowledge is stored in a tree-like structure which consists of nodes, conditions, conclusions and branches. The amount of knowledge can be quantified as the numbers of nodes and conditions which are the main components of a knowledge base, so how much knowledge are reconstructed or added can be quantified as how many nodes and conditions are reconstructed or added. Therefore, the numbers of nodes and conditions can be the objects for the comparison purpose since the larger the numbers are, the more cost has to be spent on constructing knowledge bases. For RDR (machine learning and human rules), the comparison object was the increased number of nodes and conditions by adding new human rules. Human rules were acquired from a simulated expert. For RDR (machine learning only) and J48, the comparison object was the reconstructed number of nodes and conditions by adding new data. Practically speaking, new data are usually found gradually, so we added data cases to the knowledge base one by one and it was reconstructed several times. The total reconstructed number of nodes and conditions was our comparison object. We found that RDR with machine learning only achieved 93.18% of prediction accuracy, while after adding human rules, the result was improved up to 95.09%. Although J48 had the best prediction accuracy (94.45%), RDR with machine learning and human rules outperformed it eventually. Therefore, it can be concluded that adding human knowledge to the knowledge base created by machine learning does improve the quality of the knowledge base.

Table 3. result of prediction accuracy

Models	Prediction accuracy
RDR (ML and human rules)	95.09%
RDR (ML only)	93.18%
J48	94.45%

The following table summarizes the result of reconstructed or increased nodes and conditions by comparing the above mentioned three models. By applying human knowledge, the increased ratio of nodes for improving 1% of accuracy was 33.54%, much smaller than those of RDR (machine learning only) and J48, 111.80% and 99.92% respectively. Similarly the increased ratio of conditions for improving 1% of accuracy was 69.41%, much smaller than those of RDR (machine learning only) and J48, 193.45% and 195.49% respectively. As mentioned above, the reason that pure machine learning models cost much, is because they abandon the existing knowledge base and create a new one every single time that it encounters a new data case which cannot be explained by the existing knowledge base.

Table 4. Evaluation result of knowledge increased

Models	RDR (ML and human rules)	RDR (ML only)	J48
Number of nodes original	16	16	28
Number of conditions original	26	26	73
Number of nodes after solving the stated problems	27	77	80
Number of conditions after solving the stated problems	63	119	210
Improved ratio of predication accuracy	2.05%	3.41%	0.96%
Increased ratio of nodes	68.75%	381.25%	185.71%
Increased ratio of conditions	142.30%	340.74%	187.67%
Increased ratio of nodes per 1% of accuracy improvement	33.54%	111.80%	193.45%
Increased ratio of conditions per 1% of accuracy improvement	69.41%	99.92%	195.49%

5 Conclusion

We aimed at finding how to optimize the knowledge acquired by machine learning to deduct the large cost of solving over-generalization and over-fitting problems for having the better knowledge base of phishing prediction. The experiment investigated how an approach based on RDR can be the intermediate between human knowledge and machine learning. First comparing with existing machine learning models, our experiments show some interesting facts. These are:

- The *modified RDR* performs better than Ridor which is also based on Induct RDR, especially when a dataset has both numeric and nominal attributes.
- For some datasets, the *modified RDR* performs much better than decision tree learning algorithms.
- The *modified RDR* tends to perform slightly better when a dataset has only numeric or only nominal attributes.
- The *modified RDR* tends to perform better when the ratio of training data to test data is small.

Therefore, as whole, the *modified RDR* performs better than the existing RDR model Ridor. It is used to improve prediction accuracy which might be worsened by over-generalization and over-fitting problems.

Three models were compared: RDR with ML and human knowledge, RDR ML only and J48 ML only. The result shows that applying human knowledge do improve prediction accuracy of the knowledge acquired by machine learning. Our example shows the increased ratio of nodes for improving 1% of accuracy is 33.54%, much smaller than using RDR alone and J48 (111.80% and 99.92% respectively).

6 Acknowledgement

This paper was supported by the grant FA2386-15-1-6061, funded by Asian Office of Aerospace Research and Development (AOARD), Japan. This work was supported by the Industrial Strategic Technology Development Program, 10052955, Experiential Knowledge Platform Development Research for the Acquisition and Utilization of Field Expert Knowledge, funded by the Ministry of Trade, Industry & Energy (MI, Korea).

References

1. Aburrous, M. and A. Khelifi. *Phishing detection plug-in toolbar using intelligent Fuzzy-classification mining techniques*. in *The international conference on soft computing and software engineering [SCSE'13], San Francisco State University, San Francisco, California, USA*. 2013.
2. Quinlan, J.R., *Induction of decision trees.* Machine learning, 1986. **1**(1): p. 81-106.

3. Safavian, S.R. and D. Landgrebe, *A survey of decision tree classifier methodology.* 1990.
4. Dietterich, T., *Overfitting and undercomputing in machine learning.* ACM computing surveys (CSUR), 1995. **27**(3): p. 326-327.
5. Pham, H.N.A. and E. Triantaphyllou, *The impact of overfitting and overgeneralization on the classification accuracy in data mining*, in *Soft Computing for Knowledge Discovery and Data Mining*. 2008, Springer. p. 391-431.
6. Compton, P. and R. Jansen, *Knowledge in context: A strategy for expert system maintenance*. 1988: Springer.
7. Nguyen, D.Q., et al., *Ripple down rules for part-of-speech tagging*, in *Computational Linguistics and Intelligent Text Processing*. 2011, Springer. p. 190-201.
8. Pham, S.B. and A. Hoffmann, *A new approach for scientific citation classification using cue phrases*, in *AI 2003: Advances in Artificial Intelligence*. 2003, Springer. p. 759-771.
9. Mazid, M.M., S. Ali, and K.S. Tickle. *Improved C4. 5 algorithm for rule based classification*. in *Proceedings of the 9th WSEAS international conference on Artificial intelligence, knowledge engineering and data bases*. 2010. World Scientific and Engineering Academy and Society (WSEAS).
10. Ruggieri, S., *Efficient C4. 5 [classification algorithm].* Knowledge and Data Engineering, IEEE Transactions on, 2002. **14**(2): p. 438-444.
11. Gaines, B.R. *An Ounce of Knowledge is Worth a Ton of Data: Quantitative studies of the Trade-Off between Expertise and Data Based On Statistically Well-Founded Empirical Induction*. in *ML*. 1989.
12. Cendrowska, J., *PRISM: An algorithm for inducing modular rules.* International Journal of Man-Machine Studies, 1987. **27**(4): p. 349-370.
13. Joshi, M.V. and V. Kumar. *CREDOS: Classification Using Ripple Down Structure (A Case for Rare Classes)*. in *SDM*. 2004. SIAM.
14. Devasena, C.L., et al., *Effectiveness evaluation of rule based classifiers for the classification of iris data set.* Bonfring International Journal of Man Machine Interface, 2011. **1**: p. 5.
15. Mohammad RM, Thabtah F, McCluskey L. Predicting phishing websites based on self-structuring neural network. Neural Computing and Applications. 2014 Aug 1;25(2):443-58.
16. Han SC, Yoon HG, Kang BH, Park SB. Using MCRDR based Agile approach for expert system development. Computing. 2014 Sep 1;96(9):897-908.
17. Han SC, Mirowski L, Kang BH. Exploring a role for MCRDR in enhancing telehealth diagnostics. Multimedia Tools and Applications. 2015 Oct 1;74(19):8467-81.