Personalized Email User Action Prediction Based on SpamAssassin

Ha-Nguyen Thanh[1], Quan-Dang Dinh[2(✉)], and Quang Anh-Tran[3]

[1] Hanoi Department of Information and Communications, Hanoi, Vietnam
nguyenthanhha_sotttt@hanoi.gov.vn
[2] Faculty of Information Technology, Hanoi University, Hanoi, Vietnam
quandd@hanu.edu.vn
[3] Posts and Telecommunications Institute of Technology Hanoi, Hanoi, Vietnam
tqanh@ptit.edu.vn

Abstract. Email overload, even after spam filtering, causes waste of time and reduction of work efficiency to email users. Email prioritization is the general solution for the problem. The idea is to sort incoming emails in a decreasing order of importance so that the most important messages are read and processed first and less significant ones later, if there is enough time. This paper proposed a method to predict the action that a user would take on an email. The method is based on SpamAssassin, a famous spam filter framework. Instead of classifying emails as spam and ham (non-spam message), this method is used to predict amongst the three most common actions: reply, read and delete. Experiments are conducted to measure the effectiveness of the new method on a dataset built by the authors.

Keywords: Personalized email prioritization · Email user action · SpamAssassin

1 Introduction

Communication over the Internet has become crucial in every country and in every aspect of the modern society. Among the many applications of the Internet, email is one of the most used, most important. Email allows people to exchange information in a fast, reliable and cost-effective way. According to statistics reports, the number of emails sent per day in 2015 is approximately 205.6 billion and the figure is expected to be 246.5 billion in 2019 [14]. Along with email's increased usage volume come a great number of unwanted messages called spam (unsolicited bulk email). With a large number of spam, it takes more time for email users to process daily messages.

Spam's bad impacts led to the need for spam filtering. There have been many approaches to spam filtering which can be divided into two main categories: SMTP-based filtering and machine learning [1].

The SMTP-based filtering category addresses the weaknesses of SMTP – the protocol used for sending email. For instance, SMTP does not verify email senders [1], leading to the fact that it is trivial to fake email sender. Attackers can exploit this weakness to send spam or perform online phishing. To tackle that flaw, researchers have

P. Cong Vinh et al. (Eds.): ICCASA 2016, LNICST 193, pp. 161–171, 2017.
DOI: 10.1007/978-3-319-56357-2_17

introduced several methods to verify the sender including Sender Policy Framework (SPF), DomainKeys Identified Mail (DKIM) and SenderID. This category also includes methods such as Blacklisting, Greylisting and so on.

Approaches in the machine learning category focus on analyzing email content using a classifier. Many classifiers and their variations have been applied to detect spam [11, 15], e.g. Naïve Bayes, k-Nearest Neighbor (kNN), Support Vector Machines (SVM), Term Frequency-Inverse Document Frequency (TF-IDF) and so on. Methods in this category have been widely applied and the current state-of-the-art methods are based on Bayesian filters with filtering rates exceeding 99.5% [15].

The popularity of email also leads to the email overload, even after spam filtering. The problem tends to get more serious as Spira and Goldes stated in their study [6] that a typical office worker gets around 200 legitimate emails per day. High-level officers and managers receive even more emails. Emails of different levels of importance are mixed, making important emails easy to be missed.

The above issue can be solved with email prioritization – sorting incoming emails based on their importance. There are two different groups of email prioritization methods. The first group employed regression-based methods [3, 7, 8] because it assumes the linearity of email's importance. The second one considers it a multiclass classification problem. The number of classes is usually three [2, 4] or five [8–10]. User action prediction falls into this category. These studies have yet to achieve a practical performance. This field of study is a new, highly potential one. Therefore, it requires a lot more efforts to be made.

This paper proposes a new method which utilizes SpamAssassin to predict user action on an email. It involves automatic generation and optimization of different SpamAssassin rulesets in order to turn SpamAssassin into a multiclass classifier. Popular multiclass classifier building techniques (OVA, OVO and DAG) are tested against a self-built personal email dataset. Two experiments are conducted to test the prediction rate of the new method as well as its degree of personalization.

This paper's main contribution is as follows: A new method to predict user action on emails based on SpamAssassin is proposed; Experiments to compare different methods using different sets of personalized email data were conducted; Through our experiments, we evaluated the impact of personalized factors on the results of our method.

The remaining of this paper is organized as follows: Section 2 reviews papers related to the user action prediction problem; Section 3 explains the proposed method in details; Section 4 describes two experiments that we conducted and their results; Section 5 concludes our findings and suggests directions for future works.

2 Preliminaries

2.1 Studies on the User Action Prediction Problem

User action prediction aids users in processing daily emails by automatically determine the action that should be done on an email. If a user knows which email should be replied,

read or deleted, he will be able to take suitable actions in order to save time. There have been a few studies to tackle this problem.

The problem was first introduced in 2005 [5]. Authors of [5] evaluated the factors that affect user's responses to incoming emails in order to produce a method to predict user action. Their study was done through an online survey in which questions are divided into three parts. The first part is to collect users' work environment information and the characteristics of their jobs. The second part asks about users' email usage and habits. The last part gathers information on content characteristics, level of importance, characteristics of the sender and associated actions on emails. The study used a self-built dataset with 1100 email messages. 124 persons took part of the survey. 10 features were used to predict the importance of an email. This study found that there is correlation between the importance/probability to get replied of an email and user's relationship with the sender as well as the email's content.

In [2], the authors proposed an email user action recommender system. The system is essentially a Bayesian multiclass classifier where each class represents a user action. There are three user actions: "reply", "read" and "delete".

2.2 SpamAssassin

SpamAssassin is a popular open-source spam filter which operates on multiple platforms. It uses a set of weighted (scored) rules to identify spam. Most of SpamAssassin rules are basically Regular Expressions used to find textual structures which indicate that a message is spam. Figure 1 shows an example of SpamAssassin's rule.

```
body        MONEY_BACK    /money back guarantee/i
describe    MONEY_BACK    Money back guarantee
score       MONEY_BACK    2.910
```

Fig. 1. A typical SpamAssassin body rule.

In Fig. 1, there is a body rule named MONEY_BACK. This rule checks if the body of an email contains a string that matches the RegEx "*/money back guarantee/i*" and adds a score of 2.910 to the total spam score of that email. The higher the total score, the more likely a message is spam. By default, all emails whose total score is equal to or higher than a threshold $T = 5.0$ are considered spam by SpamAssassin. The threshold T can be adjusted by user. Equation (1) shows how SpamAssassin calculates the total score for each message.

$$Score_R(m) = \sum_{i=1}^{k} Match(R_i, m) \times w_i \tag{1}$$

where:

- $Score_R(m)$ returns the total score against ruleset R for the message m.
- R is a set consisting of k rules ($R_1 \ldots R_k$).
- $Match(R_i,m)$ returns 1 if m contains a string that matches rule R_i, 0 otherwise.
- w is a set of k scores ($w_1 \ldots w_k$) corresponding to k rules.

The score return by $Score_R(m)$ is compared to the threshold T to determine if m is spam using Eq. (2).

$$Spam_R(m) = \begin{cases} 1, & Score_R(m) \geq T \\ 0, & Score_R(m) < T \end{cases} \tag{2}$$

2.3 Automatic Generation of SpamAssassin Rules

The study in [12] proposed the method for automatic generation of SpamAssassin rules to detect spam in Vietnamese. That process can be summarized as follows:

Step 1 – Dataset preparation: According to data labels, separate the training set into two parts. The first part, called D_1, contains spam messages and the order part – D_2 – contains ham messages.

Step 2 – Extracting words: The Vietnamese tokenization tool vnTokenizer [13] is used to extract words from the email subjects in D_1 into the set WS_1. Similarly, words from the content of all emails in D_1 are extracted into the set WB_1.

Step 3 – Selecting keywords: The most frequent words from WS_1 and WB_1 are kept and put into two new sets, WS_2 and WB_2.

$$WS_2 = \forall w \in WS_1,\ freq_{WS_1}(w) > \alpha$$
$$WB_2 = \forall w \in WB_1,\ freq_{WB_1}(w) > \beta$$

The function $freq_{WS_1}(w)$ returns the times which the word w appears in WS_1. The two parameters, α and β, should be adjusted according to the size of the dataset. In our experiments (which are described later in Sect. 4), we use the value 2 and 6 for α and β respectively.

Step 4 – Building ruleset: A ruleset R_1 is build. Subject rules are generated from the keywords in WS_2 and body rules are generated from keywords in WB_2. Figure 2 shows the structure generated rules. In Fig. 2, <***word***> is replaced by the actual keyword from the two keyword sets.

```
header    ReplySubj_i   Subject ~= /\b<word>\b/i
describe  ReplySubj_i   Subject contains "word"
score     ReplySubj_i   0.1
```

Fig. 2. The structure of auto-generated rules for SpamAssassin.

Step 5 – Rule selection: SpamAssassin's MassCheck tool is executed to see how the rules in R_1 are matched against emails in D_1 and D_2. Bad rules – rules with low hit rate or rules which match both spam and ham – are removed to create a new ruleset called R_2.

Step 6 – Weight (score) optimization: First, the MassCheck tool is executed again for the new ruleset (R_2). In R_2, each rule is initialized with a score value 0.1. A perceptron with a linear transfer function and a logsig activation function is built. Its weights are mapped to rule scores. It is then trained using the Stochastic Gradient Descent method

to achieve highest spam recall and lowest ham error rate [18]. When training completes, the ruleset R_3 is created from R_2's rules and trained scores.

3 Email User Action Prediction Based on SpamAssassin

SpamAssassin, or more specifically, a SpamAssassin ruleset, is able to separate spam from ham. Therefore, it is a binary classifier. In this paper, we make SpamAssassin a multiclass classifier to predict user action (REPLY, READ, DELETE) on an email. Our proposed method can be easily configured to build a system which classifies emails into more than 3 classes. We apply three different approaches of building multiclass classifiers: OVA, OVO and DAG.

The authors of this paper observed that different generated SpamAssassin rulesets have different spam recall and ham error rates at different thresholds. This is not a problem when using a single ruleset. However, to perform multiclass with SpamAssassin, multiple rulesets are required. Therefore, in addition to 6 steps of rule generation described in Sect. 2.3, we add one more step to find the best threshold for each ruleset created. The best threshold for a ruleset is defined as one which achieves the highest *spam recall* while *ham error* remains lower than 1%.

3.1 OVA (One vs. All)

Assume that we have N classes called X_i ($i = 1, 2 \ldots N, N > 2$). N binary classifiers called C_i ($i = 1, 2 \ldots N$) are needed to build a classifier for N classes using OVA method. Each classifier C_i is able to separate data of class X_i (One) from data of the other $(N-1)$ classes (All). As mentioned before, a SpamAssassin ruleset is equivalent to a binary classifier. Therefore, we need to build N rulesets for OVA to work. The algorithm for the prediction process of the OVA method is described in Fig. 3.

Input: An email message m
N rulesets RS_i ($i = 1, 2 \ldots N$)
N thresholds for the rulesets T_i ($i = 1, 2 \ldots N$)
A default class (in case all rulesets return 0)
Output: An integer indicating the class for the message m

1. **Set** $S = new\ Array(),\ max = 0, class = defaultClass$
2. **For** $i = 1 \rightarrow N$
3. **Set** $S_i = Score_{RS_i}(m) \div T_i - 1$
4. **If** $(S_i > max)$ **then** { Set $class = i,\ max = S_i$ }
5. **Return** $(class)$

Fig. 3. OVA prediction algorithm.

When building the ruleset for class X_i using the process described in Sect. 2.3, the D_1 should consist of emails marked as X_i and D_2 should contain the emails from all other classes. After the process, a ruleset RS_i should be created and there should be N rulesets ($RS_1, RS_2 \ldots RS_N$).

3.2 OVO (One vs. One)

In this method, there exists a binary classifier between any pair of different classes. The input data is tested against all classifiers and results them are aggregated to produce the final prediction. However, there are many aggregation models for OVO. In this paper, we adapt three most popular models which are MS (Max Sum, also called "Weighted Voting") [17], MV (Majority Voting) [16] and MC (Most Confident) [8].

For N classes, the number of binary classifiers needed is $N_R = N \times (N-1) \div 2$. For instance, when $N = 2, 3, 4$, $N_R = 1, 3, 6$. This means N_R rulesets $RS_{i,j}$, in which i and j represents two different classes among N classes, should be built. When building $RS_{i,j}$ using the method described in Sect. 2.3, the set D_1 should contain emails from class X_i and D_2 should contain emails from class X_j. The threshold for $R_{i,j}$ is $T_{i,j}$. To predict using the OVO-MS aggregation model, we use the algorithm shown in Fig. 4.

Input: An email message m
N_R rulesets $R_{i,j}$ $(1 \le i \le N,\ 1 \le j \le N,\ i < j)$
N_R corresponding thresholds $T_{i,j}$
A default class (in case all classes get equal weight)
Output: An integer indicating the class for the message m

1. **Set** $S = new\ Array()$
2. **For** $i = 1 \to N$ { **Set** $S_i = 0$ }
3. **For** $i = 1 \to N - 1$
4. **For** $j = i + 1 \to N$
5. **Set** $tmp = Score_{R_{i,j}}(m) \div T_{i,j} - 1$
6. **Set** $S_i = S_i + tmp,\ S_j = S_j - tmp$
7. **Set** $equalCheck = \text{true},\ class = 1,\ max = S_1$
8. **For** $i = 2 \to N$
9. **If** $S_i \neq S_{i-1}$ **then** { Set $equalCheck = \text{false}$ }
10. **If** $S_i > max$ **then** { Set $class = i,\ max = S_i$ }
11. **Return** $(equalCheck\ ?\ defaultClass : class)$

Fig. 4. The algorithm for OVO-MS prediction model.

OVO-MV predicts the class similarly to OVO-MS. The only difference is that OVO-MV counts the votes from classifiers instead of adding up their scores (see Fig. 5).

A binary classifier gives a score to indicate its level of confidence. OVO-MC selects the class that receives the *highest confidence* from any classifier (Fig. 6).

Input: An email message m
N_R rulesets $R_{i,j}$ ($1 \le i \le N$, $1 \le j \le N$, $i < j$)
N_R corresponding thresholds $T_{i,j}$
A default class (in case all classes get equal vote)
Output: An integer indicating the class for the message m

```
1.  Set S = new Array(), max = 1, class = defaultClass
2.  For i = 1 → N { Set S_i = 0 }
3.  For i = 1 → N − 1
4.      For j = i + 1 → N
5.          If Score_{R_{i,j}}(m) ÷ T_{i,j} ≥ 1 then
6.              Set S_i = S_i + 1
7.          Else
8.              Set S_j = S_j + 1
9.  For i = 1 → N
10.     If S_i > max then { Set class = i, max = S_i }
11. Return (class)
```

Fig. 5. The algorithm for OVO-MV prediction model.

Input: An email message m
N_R rulesets $R_{i,j}$ ($1 \le i \le N$, $1 \le j \le N$, $i < j$)
N_R corresponding thresholds $T_{i,j}$
A default class (in case all scores are equal)
Output: An integer indicating the class for the message m

```
1.  Set S = new Array()
2.  For i = 1 → N { Set S_i = 0 }
3   For i = 1 → N − 1
4.      For j = i + 1 → N
5.          Set tmp = Score_{R_{i,j}}(m) ÷ T_{i,j} − 1
6.          If tmp > S_i then
7.              Set S_i = tmp
8.          Else If 1 − tmp > S_j then
9.              Set S_j = 1 − tmp
10. Set equalCheck = true, class = 1, max = S_1
11. For i = 2 → N
12.     If S_i ≠ S_{i−1} then { Set equalCheck = false }
13.     If S_i > max then { Set class = i, max = S_i }
14. Return (equalCheck ? defaultClass : class)
```

Fig. 6. The algorithm for OVO-MC prediction model.

3.3 DAG (Directed Acyclic Graph)

Similar to OVO, DAG requires a binary classifier for each pair of different classes. However, DAG reduces the number of classifiers invoked in the prediction phase to (N – 1) by making use of a binary decision tree. N_R classifiers are arranged in the order given in

Fig. 7. At each level, the prediction process follows either the left or right branch from the current node depending on the outcome of the classifier at that node.

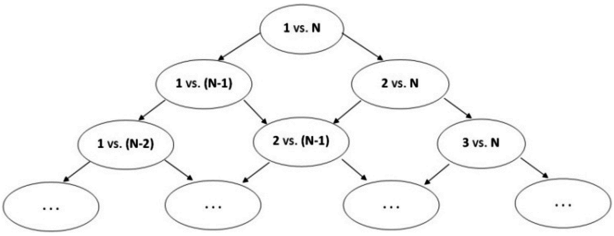

Fig. 7. The binary decision tree for the DAG model.

To apply DAG for the user action prediction problem, we repeat the rule creation part of OVO to build N_R rulesets. After that, we apply the algorithm described in Fig. 8 for prediction.

Input: An email message m
N_R rulesets $R_{i,j}$ $(1 \le i \le N,\ 1 \le j \le N,\ i < j)$
N_R corresponding thresholds $T_{i,j}$
Output: An integer indicating the class for the message m
1. **Set** $i = 1,\ j = N,\ class = 0$
2. **While** $i < j$ **do**
3. **If** $Score_{R_{i,j}}(m) > T_{i,j}$ **then**
4. **Set** $j = j - 1,\ class = i$
5. **Else**
6. **Set** $i = i + 1,\ class = j$
7. **Return** $(class)$

Fig. 8. The prediction algorithm for DAG.

4 Experiments

4.1 Dataset

Our dataset consists of 1408 emails in both English and Vietnamese collected from a personal mailbox. For this same set, 3 different sets of labels are collected. The first label set is directly extracted from the email owner's real data. Two more sets of label are done independently by 2 volunteers. The numbers of emails labeled by 3 users are shown in Table 1.

4.2 Experiment 1

This experiment is conducted to compare 5 multiclass classification models used in our study: OVA, OVO-MS, OVO-MV, OVO-MC and DAG. The authors used 2 measures

Table 1. Number of emails labeled by 3 different users.

User	Reply	Read	Delete	Total
1	183	704	335	1408
2	316	504	402	1408
3	270	402	550	1408

for evaluation: Accuracy and Delete FPR (False Positive Rate, also called "Fall-out"). Accuracy is a common measure used to evaluate multiclass classifiers [2, 15]. In the email user action prediction problem, marking a READ or REPLY email as DELETE is obviously more serious than marking a DELETE email as READ or REPLY. Therefore, the authors decided to include the FPR measure for the DELETE class to more accurately evaluate the methods. Regarding the user action prediction problem, FPR for DELETE can be interpreted as "H*ow many important messages are mistakenly classified as DELETE?*". Equation (3) shows the formula for Accuracy and Eq. (4) illustrates the FPR measure.

$$Accuracy = \frac{\#\ of\ correct\ predictions}{total\ \#\ of\ tests} \tag{3}$$

$$FPR = \frac{false\ positives}{false\ positives + true\ negatives} \tag{4}$$

- *false positives*: the number of REPLY and READ messages marked as DELETE.
- *true negatives*: the remaining number of REPLY and READ messages.

From the test results (see Table 2), OVA has the highest overall Accuracy but also the highest FPR. OVO-MC produces lowest FPR but has poor Accuracy. DAG seems to be the most balanced model with high Accuracy and reasonably low FPR.

Table 2. Accuracy and Delete FPR measured from testing with 5 different methods on 3 users. Shown values are 10-fold cross validation average, represented in percentage (%).

User	OVA		OVO-MS		OVO-MV		OVO-MC		DAG	
	Acc.	FPR	Acc.	FPR	Acc.	FPR	Acc.	FPR	Acc.	FPR
1	85.04	2.36	78.86	2.44	75.14	2.21	70.18	0.51	82.82	0.88
2	69.96	6.40	69.39	4.78	65.25	4.27	58.64	1.47	70.03	2.28
3	72.80	7.09	72.56	4.31	65.11	4.31	54.74	1.48	66.54	2.01

4.3 Experiment 2

This experiment is done to see if a user's rulesets can be effectively applied to other users. We get the rulesets from user 1 for the test because it gives the best results. These rulesets are tested on user 2 and user 3's data. In this experiment, only the Accuracy measure is used (Table 3).

Table 3. Accuracy measured from using user #1's rulesets to test against the other users' test data. Shown values are 10-fold cross validation average, represented in percentage (%).

User	OVA	OVO-MS	OVO-MV	OVO-MC	DAG
1	85.04	78.86	75.14	70.18	82.82
2	63.60	60.49	56.98	55.03	61.08
3	49.72	47.98	47.21	44.67	47.40

Accuracy is decreased dramatically in all 5 methods. This means the rulesets built for a user are personalized and it is not feasible for other users to use those rulesets.

5 Conclusion

In this paper, we proposed a method to predict user action on email using SpamAssassin. From our experiments, a few conclusions can be made. Firstly, among five prediction models that we studied, DAG has the highest overall performance. Second, the rate of false positives for the Delete action is still not practical. We should attempt to repeat the experiments using different parameters to reduce Delete FPR. Finally, our new method is intended for personal email data and experiment result has proven so. For future studies, we would like to consider experimenting on a large dataset and apply more features besides email content.

References

1. Caruana, G., Li, M.: A survey of emerging approaches to spam filtering. ACM Comput. Surv. (CSUR) **44**(2), 9 (2012)
2. Ha, Q.M., Tran, Q.A., Luyen, T.T.: Personalized email recommender system based on user actions. In: Asia-Pacific Conference on Simulated Evolution and Learning, pp. 280–289 (2012)
3. Hasegawa, T., Ohara, H.: Automatic priority assignment to E-mail messages based on information extraction and user's action history. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pp. 573–582 (2000)
4. Dabbish, L.A., Kraut, R.E., Fussell, S., Kiesler, S.: Understanding email use: predicting action on a message. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 691–700 (2005)
5. Bennett, P.N., Carbonell, J.: Detecting action-items in e-mail. In: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 585–586 (2005)
6. Spira, J.B., Goldes, D.M.: Information overload: We have met the enemy and he is us. Basex Inc. (2007)
7. Neustaedter, C., Brush, A.B., Smith, M.A., Fisher, D.: The social network and relationship finder: social sorting for email triage. In Proceedings of 7th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS) (2005)
8. Yoo, S., Yang, Y., Carbonell, J.: Modeling personalized email prioritization: classification-based and regression-based approaches. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pp. 729–738 (2011)

9. Yang, Y., Yoo, S., Lin, F., Moon, I.C.: Personalized email prioritization based on content and social network analysis. IEEE Intell. Syst. **25**(4), 12–18 (2010)
10. Yoo, S., Yang, Y., Lin, F., Moon, I.C.: Mining social networks for personalized email prioritization. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 967–976 (2009)
11. Blanzieri, E., Bryl, A.: A survey of learning-based techniques of email spam filtering. Artif. Intell. Rev. **29**(1), 63–92 (2008)
12. Dinh, Q.D., Tran, Q.A., Jiang, F.: Automated generation of ham rules for Vietnamese spam filtering. In: the 2014 Seventh IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), pp. 1–5 (2014)
13. Huyen, N.T.M., Roussanaly, A., Vinh, H.T.: A hybrid approach to word segmentation of Vietnamese texts. In: International Conference on Language and Automata Theory and Applications, pp. 240–249 (2008)
14. The Radicati Group. Email Statistics Report, 2015–2019. Palo Alto, CA, USA (2015)
15. Carpinter, J., Hunt, R.: Tightening the net: A review of current and next generation spam filtering tools. Comput. Secur. **25**(8), 566–578 (2006)
16. Friedman, J.: Another approach to polychotomous classification, vol. 56. Technical report, Department of Statistics, Stanford University (1996). http://statweb.stanford.edu/~jhf/ftp/poly.pdf
17. Hastie, T., Tibshirani, R.: Classification by pairwise coupling. The annals of statistics **26**(2), 451–471 (1998)
18. Stern, H.: Fast SpamAssassin Score Learning Tool (2004). https://svn.apache.org/repos/asf/spamassassin/trunk/masses/README.perceptron