Some fitting of naive Bayesian spam filtering for Japanese environment

Manabu Iwanaga[1], Toshihiro Tabata[2], and Kouichi Sakurai[2]

[1] Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan
iwanaga@itslab.csce.kyushu-u.ac.jp
[2] Faculty of Information Science and Electrical Engineering, Kyushu University, Japan
{tabata, sakurai}@csce.kyushu-u.ac.jp
http://itslab.csce.kyushu-u.ac.jp/

Abstract. Bayesian filtering is one of the most famous anti-spam measures. However, there is no standard implementation for treatment of Japanese emails by Bayesian filtering. In this paper, we compare several conceivable ways to treat Japanese emails about tokenizing and corpus separation. In addition, we give experimental results and some knowledge obtained by the experiments.

1 Introduction

Recent years, spam is rapidly increasing, because email is much cheaper method to send out some information than other advertising methods such as direct mail or telemarketing. Nowadays spam becomes a terrible obstacle to email communication, hence it is important for users and postmasters to keep spam out of their maildrop.

In order to keep spam out of maildrop, not only users but Internet Service Providers (ISPs) and Mail User Agents (MUAs) have applied anti-spam features into their products and services. For example, Mozilla [1], Eudora [2] and Outlook [3] have applied spam-detection as standard features.

One of the most famous anti-spam measures is Bayesian filtering. It has become famous in the past several years. Many implementations of Bayesian filtering have been developed, motivated by and based on Graham's essay [4]. Since these implementations work alone as a proxy or via other program like procmail[5], they can be applied easily. However, there is no standard implementation to process Japanese emails in Bayesian filtering.

Spam causes a social issue also in Japan (In the case of Japan, spam sent to email address assigned to cell phone is a big problem). Some Japanese email users receive spams written in Japanese, others receive spams written in English, and some unfortunate users receive both of them. For example, authors exchange both Japanese and English emails with an identical email address, and also receive both Japanese and English spams by the address.

In this paper, we consider and make experiments in several conceivable points to treat Japanese emails. First, we focus on how to extract tokens from Japanese sentence. It is important to extract tokens, because the probability that received email is a spam is calculated from probabilities for tokens, that the email with the token is a spam. Second, we consider how to separate corpuses according to language written in email. While most of Japanese implementations of Bayesian filtering separate corpuses between Japanese emails and non-Japanese emails, choosing corpus for each token, not for each email, will bring more accuracy. A part of the result may be useful for users who treat both English emails and non-English emails, not only Japanese.

Corpus is a collection of words appeared in previous spams and nonspams. In detail, it contains two data:

- number of learned spams and nonspams, respectively,
- frequency of each word in spams and nonspams, respectively.

From above two data, each word is given a probability that an email which contains the word is a spam.

2 Related Work

In order to avoid spam, various methods have been proposed and used. Most of proposed methods are classified into following.

- Watching behavior of SMTP connection (e.g. Greylisting [6])
- IP Blacklisting (e.g. ORDB [7])
- Domain authentication (e.g. SPF [8], senderID [9])
- Content filtering: rule-based (e.g. Spamassassin [10]), statistical (e.g. naive Bayes [4, 11, 18–20], Markovian [12]), collaborative (e.g. Razor [13])
- Challenge/response (e.g. [14–17])

Watching behavior of SMTP connection, IP Blacklisting, and domain authentication are mainly used in mail servers. It is pretty hard for users to apply these methods on their PCs, because a mail server can omit some information of sender and a user cannot intervene in transaction when it delivers to users' maildrop (For example, a user cannot apply greylisting without control of his/her mail server). On the other hand, users can easily apply content filtering and challenge/response, without modifying servers and SMTP (needless to say, these methods can work on a server for convenience of users.)

Nowadays, statistical filtering methods have been applied broadly to avoid spam, especially Bayesian filtering (naive Bayes) which has been popular since Graham's essay [4] came out. Bayesian filter calculates the probability for every word that a randomly chosen email containing the word will be a spam, according to past spams and nonspams. Furthermore, Bayesian filter can add tokens appeared in the email to its corpus, according to judgment of the Bayesian filter itself. Therefore, a user only has to train his/her filter in case his/her filter makes a mistake. Even if spammers use obfuscated words, Bayesian filter also learns

these words and obfuscated words are used as obvious evidence. There are many available implementations, for example bsfilter [18], scbayes [19], bogofilter [20] and popfile [21].

3 Consideration in Bayesian Filtering for Japanese Environment

As Bayesian filtering is used around the world, it is also used in Japanese environment, which treats both Japanese email and English email. In order to apply Bayesian filtering in Japanese environment, some modification on the filtering scheme for improving efficiency of Bayesian filtering should be considered.

There are many Bayesian filtering implementations intended to be used in Japanese environment. These implementations have several features specialized for Japanese environment. We focus on these features, especially about extracting tokens from a sentence and separating corpus according to language written in an email.

3.1 Method for Extracting Tokens

Because Japanese does not have a blank to separate a sentence into tokens, it is not so easy to extract grammatical words from a sentence. In Japanese, the most popular way to extract tokens until now is variants of bigram which utilize the cayegory of Japanese characters (hiragana, katakana and kanji). Basically, bigram is a method that all pairs of consecutive two characters are extracted as tokens. In fact, bigram is usually adapted with some modification, because kanji, hiragana and katakana are used with different way in Japanese. It is also a simple and easy way to implement that regarding each adjacent characters in same category (kanji, hiragana, and katakana) as a token. Bigram seems fairly effective for Bayesian filtering because Japanese has many phrases composed of two, three or four kanjis.

If a user sticks to grammatical word, he/she can also use external tools like KAKASI [22], ChaSen [23] or MeCab [24]. While these tools are originally intended for morphological analysis or kanji-to-hiragana conversion, tokenizing is now one of the most well-known usages of these tools.

Now we give several samples. Bsfilter [18], one of Bayesian filtering implementations written in ruby, has following rules to extract tokens from a Japanese email.

- For a sequence of kanjies, adjoining two kanjies is respectively extracted as a token. If kanji stands alone or continues only two characters, the whole is extracted as one token.
- All contiguous katakanas are extracted as one token as the whole.
- Hiraganas are not extracted as any token.

Scbayes [19], another implementation written in Scheme[3], has following rules.

[3] Scheme is one of varieties of the Lisp programming language.

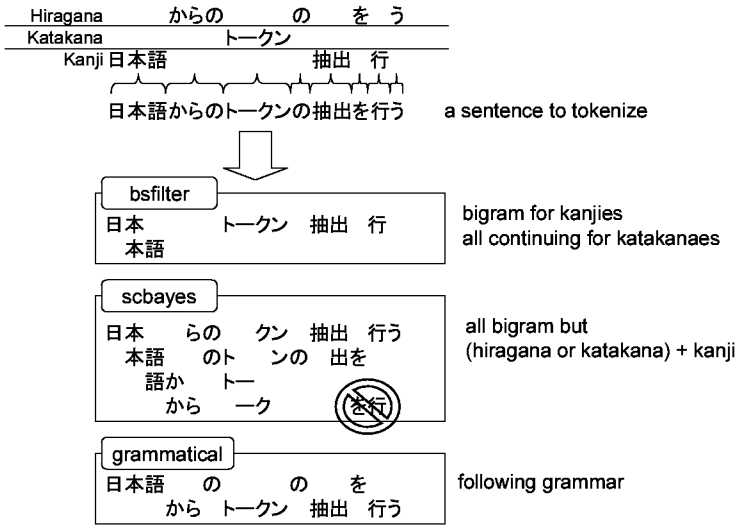

Fig. 1. Three tokenizing methods

– Basically, contiguous two characters are extracted as a token.
– However, combination of hiragana and kanji in that order is ignored.

Some implementations like bogofilter [20] do not support extracting tokens from Japanese sentences, so some tool must convert Japanese email to tokenized text which an implementation can extract tokens from. These tokenizing methods are depicted as Figure 1.

3.2 Method of Separating Corpus

Relative proportions of spams and nonspams are not same between in one's Japanese emails and in one's English emails. For example, one of the authors' email addresses receives many Japanese spams but only a few English spams, while another receives many English spams but a few Japanese spams. It is thought that the former address was gathered by and exchanged among Japanese spammers, and the latter one by/among spammers in English-speaking market. On the other hand, we can receive important English emails by the latter email address, for example, acceptance letter from the international conference and confirmation of hotel reservation. This situation causes high spam probabilities for whole words of particular language. It is undesirable that this bias causes more false-positives (FP) in emails of particular language, because Bayesian filtering is based on the assumption that we do not want a classification to be affected by the relative proportion of spams and nonspams.

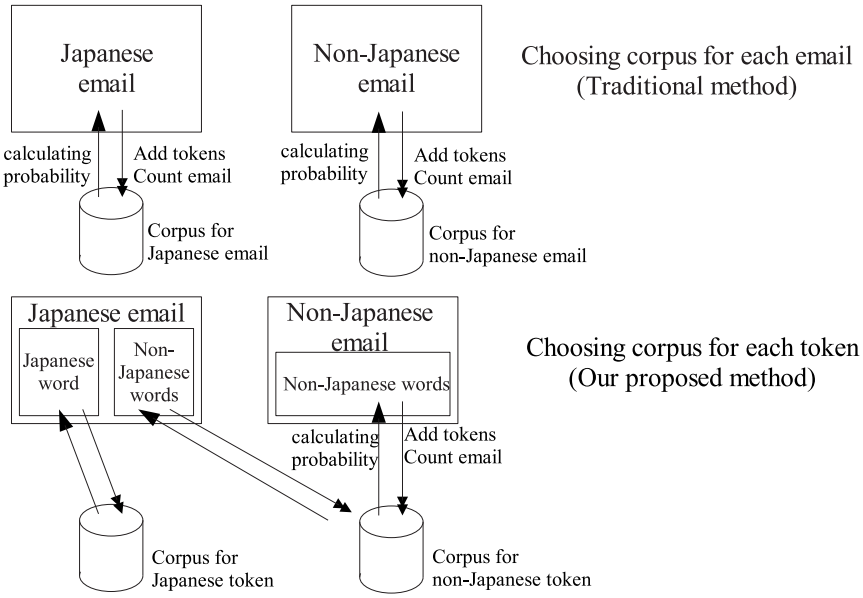

Fig. 2. Our Proposal for separating corpus

To cope with this situation, most of Japanese implementations separate corpuses for Japanese emails and non-Japanese emails. In these implementations, filters first make a decision whether an email is written in Japanese or not. That is to say, the implementation distinguishes language written in an email, and then a spam probability is calculated with one corpus corresponding to the language. It is reasonable to reduce false-positives arising from language in which email is written. It is easy to distinguish Japanese emails from non-Japanese emails, reading "charset" header field [25] or checking character code.

However, we wanted to detect language-independent evidence, which is contained in header, etc. To realize this, we tried to distinguish language of each token, not each email. This method is expected to reduce both false-positives and false-negatives (FN). This method can also deal with an email which contains two or more languages as a message body.

Our method is represented as Figure 2. Upper one shows traditional separation for each email and lower one shows our proposed separation for each token. In our method, non-Japanese tokens in Japanese emails are learned by a corpus for non-Japanese tokens. On the other hand, corpus should also count number of learned spams and nonspams. Because tokens in Japanese emails are learned by either Japanese and non-Japanese corpuses, Japanese email should be counted by both of the corpuses. In this experiment, we leave this matter open and use simple way. We count an email as both Japanese and English email, in proportion to ratio between Japanese tokens and English tokens. For example, an email

Table 1. Number of emails which is used to experiment about extracting tokens

initial training	nonspam Japanese	spam Japanese
Condition 1	293	293
Condition 2	1659	293

Table 2. Comparison between token-extracting methods

initial training	method	Condition 1		Condition 2	
		FP	FN	FP	FN
3/4	**bsfilter**	0.41%	1.42%	0.00%	6.69%
	ChaSen	0.27%	1.94%	0.00%	8.72%
	scbayes	0.27%	1.42%	0.00%	15.88%
1/2	**bsfilter**	0.27%	1.39%	0.02%	11.22%
	ChaSen	0.27%	1.94%	0.02%	14.18%
	scbayes	0.34%	2.11%	0.00%	23.64%

which contains 300 Japanese tokens and 100 English tokens is counted as 0.75 Japanese emails and 0.25 English emails.

4 Experiment

We performed experiments on techniques concerning about treating Japanese emails mentioned above. In every experiment, emails were divided randomly into ones for training and ones for testing. We call the proportion of emails for training on all emails in the experiment initial training ratio. For example, we have 1000 nonspams and 600 spams, and initial training ratio is 1/5, we train 200 nonspams and 120 spams. We changed initial training ratio to several values to know characteristic of our method associated with amount of training. At first, emails for training are learned as spams and nonspams by a Bayesian filter, respectively. Then emails for testing are classified to spam or nonspam by the filter, and we count numbers of false-negatives and false-positives. Because corpuses are cleared at the end of each testing, each experiment is independent.

Performance is evaluated by rate of false-positives and it of false-negatives. The former is an error that a nonspam is wrongly classified as a spam, and the latter is an error that a spam is wrongly classified as a nonspam.

In the following experiments, we used bsfilter (Revision 1.35.4.3, [18]) as an implementation of Bayesian filtering, with Graham's method of calculation and 0.9 as threshold, and then simulated the other methods when needed. It is because we want to look at core differences between methods and omit the other differences.

Table 3. Number of emails for comparison about corpus separation

Init. training	nonspam			spam		
	JA	non-JA	Total	JA	non-JA	Total
Condition 3	1167	55	1222	293	929	1222
Condition 4	1680	257	1937	75	164	239

JA: Japanese emails, non-JA: non-Japanese emails.

4.1 Method for Extracting Tokens

We performed experiment with three tokenizing methods, bsfilter-styled bigram, scbayes-styled bigram (simulated on bsfilter) and grammatical tokenizing. We used ChaSen [23] as a grammatical tokenizer.

The number of emails used for this experiment is shown in Table 1. Condition 1 assumes that there are spams as many as nonspams, and Condition 2 assumes that there are more nonspams than spams. The emails are collected from spams / nonspams we received and spams we caught by a honeypot email account. Note that emails used in this experiment are all Japanese emails, because tokenizing methods are intended only for Japanese.

Table 2 shows the result of experiment between tokenizing methods. In case there are spams as many as nonspams (Condition 1), there were about 0.34% of false-positives and 1.70% of false-negatives on an average, and difference between tokenizing methods was little (Roughly speaking, difference between tokenizing methods is only $\pm 0.07\%$ of false-positives and $\pm 0.41\%$ of false-negatives from average). In this case, difference between tokenizing methods was little.

In case there are more nonspams than spams, false-positive decreased and false-negative increased (Condition 2) from above case, and difference of false-negatives between tokenizing methods was not negligible (difference of false-negative reached about at $\pm 5\%$ from average.) Bsfilter-styled bigram had the best performance with 6.69% and 11.22% of false-negatives; ChaSen had second-best with 8.71% and 14.18%. However, scbayes-styled bigram made the most errors, about twice as many as bsfilter. From this result, we can say that grammatical tokenizing does not always yield good performance in Japanese.

4.2 Method of Separating Corpus

Next, we performed experiments to compare between two methods of separating corpus. One is a way that selects a corpus for each email, and the other is a way that selects corpus for each token. The number of emails used for this experiment is shown in Table 3. Condition 3 represents a case that there are spams as many as nonspams, and Condition 4 represents a case that there are more nonspams than spams. Similar to the experiment in Section 4.1, the emails are collected from spams / nonspams we received and spams we caught by a honeypot email account. However, this time we included non-Japanese spams / nonspams.

Table 4 shows the result of comparison between corpus separation styles. Through Condition 3 and 4, the result shows that our method, the way that

Table 4. Experiment about separating corpus

Number of emails	Initial training	**for each email**		**for each token**	
		FP	FN	FP	FN
Condition 3	4/5	0.99%	3.84%	0.48%	3.86%
	3/5	0.97%	4.35%	0.55%	4.28%
	2/5	1.45%	5.26%	0.81%	5.49%
	1/5	2.29%	7.10%	1.46%	6.84%
Condition 4	4/5	0.32%	4.59%	0.24%	4.75%
	3/5	0.37%	6.09%	0.23%	5.42%
	2/5	0.47%	7.39%	0.25%	7.87%
	1/5	1.03%	11.68%	0.24%	12.63%

selects a corpus for each token, got less false-positives with almost same amount of false-negatives. We think it is because spam has more characteristics which are language-independent than nonspam.

Because false-positive is serious problem on using Bayesian filtering, it is significant that our method can decrease false-positives without increase of false-negatives.

5 Conclusion

In this paper, the authors have explained issues on Bayesian filtering for Japanese. To adapt Bayesian filtering into Japanese email environment, some modification is usually made to Bayesian filtering. The authors have looked at two factors and experimented on several methods which have applied by existing implementation or which the authors have proposed.

First, we have focused on methods of extracting tokens from sentence. Grammatical tokenizer does not always yield the best performance for Japanese emails.

Second, we have focused on methods of separating language-specified corpuses. False-positives can be decreased by choosing language-specified corpus for each token, not for each email, without increase of false-negatives. Because false-positive is serious problem on using Bayesian filtering, it is significant that choosing corpus for each token can decrease false-positives without increasing false-negatives.

While the authors have obtained good result by choosing corpus for each token, modification for counting email learned by Bayesian filter is very intuitive. More sophisticated way for counting emails may yield better performance than our way.

Acknowledgement

This research was partly supported from the grant of Secom Science and Technology Fundation.

References

1. Mozilla 1.3 Release Notes, modified February 2004, http://www.mozilla.org/releases/mozilla1.3/.
2. QUALCOMM Releases Eudora(R) 6.0 - Significant Version Upgrade with New Advanced Time-Saving Tools, September 2003, http://www.eudora.com/press/2003/09_04_03.html.
3. Help Prevent Junk E-Mail Messages with Outlook 2003, April 2003, http://www.microsoft.com/office/editions/prodinfo/junkmail.mspx.
4. P. Graham, "A Plan for Spam," http://paulgraham.com/spam.html.
5. Procmail, http://www.procmail.org/.
6. E. Harris, The Next Step in the Spam Control War: Greylisting, 2003, http://projects.puremagic.com/greylisting/.
7. Open relay database, http://www.ordb.org/.
8. Sender Policy Framework, http://spf.pobox.com/.
9. MTA Authentication Records in DNS, *Internet-Draft*, May 2004, http://xml.coverpages.org/draft-ietf-marid-core-01.txt.
10. Spamassassin, http://spamassassin.org/.
11. P. Graham, Better Bayesian Filtering, *Spam conference*, Boston, USA, January 2003, http://paulgraham.com/better.html.
12. W. Yerazunis, The Spam-Filtering Accuracy Plateau at 99.9% Accuracy and How to Get Past It., *2004 Spam Conference*, Boston, USA, January 2004, http://crm114.sourceforge.net/Plateau_Paper.pdf
13. Vipul Razor, http://razor.sourceforge.net/.
14. E. Gabber, M. Jakobsson, Y. Matias, A. Mayer, Curbing Junk Email via secure Classification, *Financial Cryptography '98*, Anguilla, British West Indies, 1998, 198–213.
15. R. J. Hall, Channels: Avoiding unwanted electronic mail, *the 1996 DIMACS Symposium on Network Threats*, Piscataway, USA, 1996, 85–103.
16. Mailblocks, http://about.mailblocks.com/.
17. M. Jakobsson, J. Linn, J. Algesheimer, "How to Protect Against a Militant Spammer," *Cryptology ePrint archive*, report 2003/071, 2003.
18. bsfilter, http://www.h2.dion.ne.jp/~nabeken/bsfilter/.
19. scbayes, http://www.shiro.dreamhost.com/scheme/wiliki/wiliki.cgi?Gauche%3ASpamFilter&l=jp.
20. bogofilter, http://bogofilter.sourceforge.net/.
21. POPFile, http://popfile.sourceforge.net/.
22. KAKASI, http://kakasi.namazu.org/.
23. ChaSen, http://chasen.aist-nara.ac.jp/.
24. MeCab, http://cl.aist-nara.ac.jp/~taku-ku/software/mecab/.
25. Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples, *RFC2049*, Nov. 1996, http://www.ietf.org/rfc/rfc2049.txt