Unsupervised Spam Detection Based on String Alienness Measures

Kazuyuki Narisawa, Hideo Bannai, Kohei Hatano, and Masayuki Takeda

Department of Informatics, Kyushu University, Fukuoka 819-0395, Japan
{k-nari, bannai, hatano, takeda}@i.kyushu-u.ac.jp

Abstract. We propose an unsupervised method for detecting spam documents from a given set of documents, based on *equivalence relations* on strings. We give three measures for quantifying the *alienness* (i.e. how different they are from others) of substrings within the documents. A document is then classified as spam if it contains a substring that is in an equivalence class with a high degree of alienness. The proposed method is unsupervised, language independent, and scalable. Computational experiments conducted on data collected from Japanese web forums show that the method successfully discovers spams.

1 Introduction

Due to its remarkable development, the Web has become a major means of advertisement [1]. Not only normal websites, but CGM (Consumer Generated Media), such as Weblogs, forums and SNS, made and written by the casual user, is also exploited as an advertisement media. *Spam* messages, which are unsolicited, unwanted advertisement messages sent or posted by spammers, is becoming a huge issue on this media, because in general, any user can freely and easily post messages.

There exist various types of spam: webspam (spam in web sites), linkspam (spam used linkfarm), wikispam (spam in Wikis), splog (spam in Weblogs) [2], commentspam (spam in forums), spam mail (spam in email), and more recently, spim (spam over Instant Messaging) [3], and spit (spam over IP Telephony). These spams advertise their goods and websites, mislead users to access other websites, manipulate the PageRank [4] of their sites and so on. Not only do these messages interfere with the user trying to obtain useful information, but they can overload the servers which provide various services to the users. Hence, developing methods to detect such spams automatically is an important problem.

In this paper, we consider an unsupervised and language independent method for the detection of spam in document sets, based on the *alienness* of the substrings contained in each document. In order to effectively transmit their advertisement message to their potential customers (victims), spammers send many identical, or nearly identical spam messages. We assume that such redundancies in spam causes their substring frequencies distribution to deviate from that of other normal messages, and quantify this amount using several measures based

V. Corruble, M. Takeda, and E. Suzuki (Eds.): DS 2007, LNAI 4755, pp. 161–172, 2007.

on the substring equivalence relation defined in [5]. A document is then classified as spam if it contains a substring that is in an equivalence class with a high degree of alienness.

In Section 2, we review related work. In Section 3, we introduce some notations, as well as the substring amplification method [6] which is our previous unsupervised method for detecting spam. In Section 4, we describe our new spam detection method. We show results of computational experiments conducted on Japanese web forum postings in Section 5. In Section 6, we conclude the paper.

2 Related Work

There are roughly three strategies for detecting spam.

Link Analysis: This detects malicious link sets called linkfarms, by analyzing link structures [7,8,9]. It can detect linkspams with high accuracy, and does not depend on languages. However, it suffers from the drawback that it generally has a high computational cost, and that it can only be used for spam messages that contain links.

Machine Learning: There are various machine learning based filters such as Bayesian filters [10], which are fairly effective for spam mails using header information in addition to contents. However, such supervised methods must first be fed with a large amount of training message data marked as spam or nonspam, which may be costly to generate.

Statistical Analysis: This approach detects spams by considering various statistics of words or n-grams in documents [11]. However, word statistics requires word segmentation for languages that do not have word boundaries, such as Japanese or Chinese. Concerning n-gram statistics, a good n must somehow be chosen.

Our proposed method can be classified as a Statistical Analysis strategy, and uses the entire set of substrings instead of words or n-grams. Although the number of substrings in a document is quadratic in its length, our method runs in linear time by grouping the substrings into equivalence classes.

3 Preliminaries

Let Σ be a finite alphabet. An element of Σ^* is called a *string*. Strings x, y and z are said to be a *prefix*, *substring*, and *suffix* of the string $u = xyz$, respectively, and the string u is said to be a *superstring* of y. The length of a string u is denoted by $|u|$. The empty string is denoted by ε, that is, $|\varepsilon| = 0$. Let $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. The i-th character of a string u is denoted by $u[i]$ for $1 \leq i \leq |u|$, and the substring of u that begins at position i and ends at position j is denoted by $u[i:j]$ for $1 \leq i \leq j \leq |u|$. For convenience, let $u[i:j] = \varepsilon$ for $j < i$. The set of substrings of a string w is denoted by $Sub(w)$, and let $Sub(S) = \bigcup_{w \in S} Sub(w)$ for a set S of strings. The elements of $Sub(S)$ are called *substrings* of S. Let $Sub_f(S)$ denote the set of substrings appearing f times in S. Let $|S|$ denote the cardinality of S.

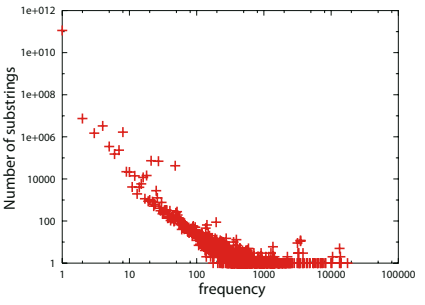

Fig. 1. An f-$|Sub_f(S)|$ plot. Outliers tend to correspond to spams.

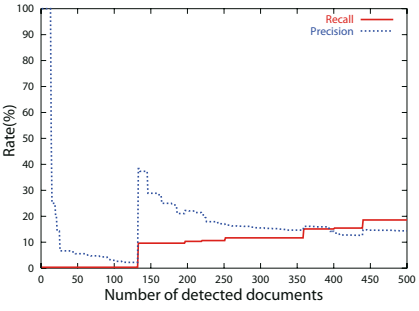

Fig. 2. The performance of the Substring Amplification Method

3.1 Our Previous Method: Substring Amplification

We describe the Substring Amplification Method presented in [6], which is an unsupervised spam detection method. It is conceptually similar to the method in this paper in that it tries to detect spams by finding deviations in occurrence frequencies of substrings in documents. Unlike n-gram analysis and word analysis, it uses the frequency distribution of all substrings of the input documents. We assume that the Zipf's law [12,13] holds between the frequencies f and the number $|Sub_f(S)|$ of distinct substrings with frequency f, and look for outliers. Figure 1 is an example plot for the web data described in Section 4 (forum 4314). Looking more closely at this graph, outliers from the distribution with unexpectedly large $|Sub_f(S)|$ are observed to be due to substrings from spam documents. Figure 2 shows the performance of the Substring Amplification Method run on the same data.

The Substring Amplification Method finds suspicious frequencies f and outputs the set of substrings with frequency f. However, this set is comprised of substrings of essentially identical occurrences, as well as substrings that just happened to have the same frequency. Moreover, it was observed that usually, only a single group of substrings having essentially identical occurrences correspond to spam, and is responsible for large $|Sub_f(S)|$ values. In order to improve the accuracy of the Substring Amplification Method, we formalize this observation and propose a new method using the equivalence relation on substrings defined by [14]. In the next section, we describe our method in detail.

4 New Method

We consider the equivalence relation over substrings, introduced by Blumer et al. [14] based on their occurrences. Intuitively, each equivalence class gathers the substrings whose "occurrences" are the same.

We note that by using the suffix array data structure [15] together with its lcp array, we can enumerate in linear time, the equivalence classes, as well as their values for each of the measures that will be used in this paper as shown

in [16]. The algorithm is a non-trivial extension of the algorithm of [17], but it is beyond the scope of this paper.

4.1 Equivalence Relations on Substrings

In this subsection, we give definitions of the equivalence relations of Blumer et al. [14], and then state some properties.

Definition 1. *Let S be a non-empty finite subset of Σ^+. For any x in $Sub(S)$, let*

$$BegPos_S(x) = \left\{ \langle w, j\rangle \,\middle|\, w \in S, 0 \le j \le |w|, x = w[j+1 : j+|x|] \right\},$$
$$EndPos_S(x) = \left\{ \langle w, j\rangle \,\middle|\, w \in S, 0 \le j \le |w|, x = w[j-|x|+1 : j] \right\}.$$

For any $x \notin Sub(S)$, let $BegPos_S(x) = EndPos_S(x) = \emptyset$. In this paper, we omit the set S, and write simply *BegPos* and *EndPos*.

For example, if $S = \{\texttt{discover}, \texttt{cover}, \texttt{November}, \texttt{vertical}\}$, then the sets *BegPos* and *EndPos* for their substrings are as follows. $BegPos(\texttt{o}) = BegPos(\texttt{ov}) = BegPos(\texttt{ove}) = \{\langle \texttt{discover}, 4\rangle, \langle \texttt{cover}, 1\rangle, \langle \texttt{November}, 1\rangle\}$, $BegPos(\texttt{c}) = \{\langle \texttt{discover}, 3\rangle, \langle \texttt{cover}, 0\rangle, \langle \texttt{vertical}, 5\rangle\}$, $BegPos(\texttt{co}) = BegPos(\texttt{cov}) = BegPos(\texttt{cove}) = BegPos(\texttt{cover}) = \{\langle \texttt{discover}, 3\rangle, \langle \texttt{cover}, 0\rangle\}$, and $EndPos(\texttt{r}) = EndPos(\texttt{er}) = \{\langle \texttt{discover}, 8\rangle, \langle \texttt{cover}, 5\rangle, \langle \texttt{November}, 8\rangle, \langle \texttt{vertical}, 3\rangle\}$, $EndPos(\texttt{o}) = \{\langle \texttt{discover}, 5\rangle, \langle \texttt{cover}, 2\rangle, \langle \texttt{November}, 2\rangle\}$, $EndPos(\texttt{over}) = EndPos(\texttt{cover}) = \{\langle \texttt{discover}, 5\rangle, \langle \texttt{cover}, 2\rangle\}$.

Definition 2. *Let x and y be arbitrary strings in Σ^*. The equivalence relations $\equiv_L$ and $\equiv_R$ are defined by*

$$x \equiv_L y \Leftrightarrow BegPos(x) = BegPos(y),$$
$$x \equiv_R y \Leftrightarrow EndPos(x) = EndPos(y).$$

The equivalence class of a string x in Σ^* with respect to $\equiv_L$ and $\equiv_R$ is denoted by $[x]_{\equiv_L}$ and $[x]_{\equiv_R}$, respectively.

For example, if $S = \{\texttt{discover}, \texttt{cover}, \texttt{November}, \texttt{vertical}\}$, then $[\varepsilon]_{\equiv_L} = [\varepsilon]_{\equiv_R} = \{\varepsilon\}$, $[\texttt{o}]_{\equiv_L} = [\texttt{ov}]_{\equiv_L} = [\texttt{ove}]_{\equiv_L} = \{\texttt{o}, \texttt{ov}, \texttt{ove}\}$, $[\texttt{c}]_{\equiv_L} = \{\texttt{c}\}$, $[\texttt{co}]_{\equiv_L} = [\texttt{cov}]_{\equiv_L} = [\texttt{cove}]_{\equiv_L} = [\texttt{cover}]_{\equiv_L} = \{\texttt{co}, \texttt{cov}, \texttt{cove}, \texttt{cover}\}$, and $[\texttt{r}]_{\equiv_R} = [\texttt{er}]_{\equiv_R} = \{\texttt{r}, \texttt{er}\}$, $[\texttt{o}]_{\equiv_R} = \{\texttt{o}\}$, $[\texttt{over}]_{\equiv_R} = [\texttt{cover}]_{\equiv_R} = \{\texttt{over}, \texttt{cover}\}$.

Definition 3. *For any string x in $Sub(S)$, let $\overrightarrow{x}$ and $\overleftarrow{x}$ denote the unique longest members of $[x]_{\equiv_L}$ and $[x]_{\equiv_R}$, respectively.*

For example, if $S = \{\texttt{discover}, \texttt{cover}, \texttt{November}, \texttt{vertical}\}$, then $\overrightarrow{\varepsilon} = \overleftarrow{\varepsilon} = \varepsilon$, $\overrightarrow{\texttt{o}} = \overrightarrow{\texttt{ov}} = \overrightarrow{\texttt{ove}} = \texttt{ove}$, $\overrightarrow{\texttt{c}} = \texttt{c}$, $\overrightarrow{\texttt{co}} = \overrightarrow{\texttt{cov}} = \overrightarrow{\texttt{cove}} = \overrightarrow{\texttt{cover}} = \texttt{cover}$, and $\overleftarrow{\texttt{r}} = \overleftarrow{\texttt{er}} = \texttt{er}$, $\overleftarrow{\texttt{o}} = \texttt{o}$, $\overleftarrow{\texttt{over}} = \overleftarrow{\texttt{cover}} = \texttt{cover}$.

Definition 4. *For any string x in $Sub(S)$, let $\overleftrightarrow{x}$ be the string $\alpha x \beta$ such that α and β are the strings satisfying $\overleftarrow{x} = x\beta$ and $\overrightarrow{x} = \alpha x$.*

For example, if $S = \{\texttt{discover}, \texttt{cover}, \texttt{November}, \texttt{vertical}\}$, then $\overleftrightarrow{\varepsilon} = \varepsilon$, $\overleftrightarrow{\texttt{o}} = \overleftrightarrow{\texttt{ov}} = \overleftrightarrow{\texttt{ove}} = \texttt{ove}$, $\overleftrightarrow{\texttt{c}} = \texttt{c}$, $\overleftrightarrow{\texttt{r}} = \overleftrightarrow{\texttt{er}} = \texttt{er}$, and $\overleftrightarrow{\texttt{co}} = \overleftrightarrow{\texttt{cov}} = \overleftrightarrow{\texttt{cove}} = \overleftrightarrow{\texttt{over}} = \overleftrightarrow{\texttt{cover}} = \texttt{cover}$.

Intuitively, $\overleftrightarrow{x} = \alpha x \beta$ means that:

- Every time x occurs in S, it is preceded by α and followed by β.
- Strings α and β are as long as possible.

Definition 5. *Strings x and y are said to be equivalent on S if and only if:*

1. *$x \notin Sub(S)$ and $y \notin Sub(S)$, or*
2. *$x, y \in Sub(S)$ and $\overleftrightarrow{x} = \overleftrightarrow{y}$.*

This equivalence relation is denoted by $\equiv$. The equivalence class of a string x in $Sub(S)$ with respect to $\equiv$ is denoted by $[x]_{\equiv}$.

For example, if $S = \{\texttt{discover}, \texttt{cover}, \texttt{November}, \texttt{vertical}\}$, then the strings in $Sub(S)$ are divided into the equivalence classes: $\{\varepsilon\}$, $\{\texttt{o}, \texttt{ov}, \texttt{ove}\}$, $\{\texttt{c}\}$, $\{\texttt{r}, \texttt{er}\}$, and $\{\texttt{co}, \texttt{cov}, \texttt{cove}, \texttt{over}, \texttt{cover}\}$.

A string x in $Sub(S)$ is said to be *prime* if $\overleftrightarrow{x} = x$. Let $Prime(S)$ denote the set of prime substring of S, that is, $Prime(S) = \{\overleftrightarrow{x} \mid x \in Sub(S)\}$. For example, if $S = \{\texttt{discover}, \texttt{cover}, \texttt{November}, \texttt{vertical}\}$, then $Prime(S) = \{\texttt{c}, \texttt{i}, \texttt{er}, \texttt{ve}, \texttt{ove}, \texttt{ver}, \texttt{cover}, \texttt{discover}, \texttt{November}, \texttt{vertical}\}$.

We regard each prime string x as the *representative* of the equivalence classes $[x]_{\equiv}$.

For any x, y in Σ^*, we write $x \preceq y$ if x is a substring of y. For any x in $Prime(S)$, let $Minimal(x)$ denote the set of minimal elements of $[x]_{\equiv}$, that is, $Minimal(x) = \{y \in [x]_{\equiv} \mid z \preceq y \text{ and } z \in [x]_{\equiv} \text{ imply } z = y\}$. Let $Maximin(x)$ denote the maximum length of strings in $Minimal(x)$.

For example, if $S = \{\texttt{discover}, \texttt{cover}, \texttt{November}, \texttt{vertical}\}$, then $Minimal(\texttt{cover}) = \{\texttt{co}, \texttt{over}\}$ and $Maximin(x) = |\texttt{over}| = 4$.

The following lemma states that any equivalence class that contains a substring of S is represented by its representative and its minimal elements.

Lemma 1 ([5]). *For any x in $Prime(S)$, let $y_1, \ldots, y_k$ be the elements of $Minimal(x)$. Then, $[x]_{\equiv} = Pincer(y_1, x) \cup \cdots \cup Pincer(y_k, x)$, where $Pincer(y_i, x)$ is the set of strings z with $y \preceq z \preceq x$.*

It can also be shown that such representations of all equivalence classes of substrings in S require only linear space [16].

4.2 Measures with Equivalence Classes

In this subsection, we give three measures for quantifying the alienness of equivalence classes. We use a data set obtained from the Web in order to evaluate these measures. The data set consists of postings from the YahooJapanFinance[1]

[1] `http://quote.yahoo.co.jp/`

forum. This forum is surveyed by the forum administrator, and postings are manually deleted if they are judged to be spam. Therefore, we can obtain spam and non-spam document examples by gathering the postings of a given forum over a certain period of time. We regard the postings which have been deleted as spams, and the writings not deleted as nonspams. This data set contains 1087 postings including 226 spam posts and 861 nonspam posts.

We will not regard strings only occurring once as spam, and only consider the equivalence classes whose elements appear at least twice in the documents.

Length: In general, spams are different from natural sentences in that they tend to be lengthy and appear more frequently. Hence, we first consider the length of the representative of an equivalence class as a measure for spam detection:

$$measure_{Length}(x) = |\overleftrightarrow{x}|$$

There seems to exist a power law between the length of the representative of an equivalence class and the number of equivalence classes with the length (see Figure 3-(1)). In this plot, "Spam" denotes that all equivalence classes with that Length measure are substrings of spam documents only. "NonSpam" denotes that all equivalence classes with that Length measure are substrings of nonspam documents only. "Spam and NonSpam" denotes otherwise, and equivalence classes with that Length measure are included in both spam and nonspam documents. In this plot, we can see that spam equivalence classes are distributed on high length parts. We can say that an equivalence class has a high degree of probability for being spam if the length of the representative of the equivalence class is long.

Figure 3-(4) shows the ROC curve for the Length measure. The x-axis is the negative ratio for each equivalence class, that is

$$negative\ ratio = \frac{\#\text{ of detected nonspam documents}}{\#\text{ of detected documents}},$$

and y-axis is the positive ratio for each equivalence class, that is

$$positive\ ratio = \frac{\#\text{ of detected spam documents}}{\#\text{ of detected documents}}.$$

The graph is drawn by considering all possible Length measure values as a threshold, and plotting the above values by classifying documents which contain the equivalence class whose length is longer than the threshold value as spam, and nonspam otherwise. As can be seen in Figure 3-(4), the length of the representative of an equivalence class seems to be an effective measure for spam detection.

Size: Next, we consider the size of an equivalence class as a measure for spam detection:

$$measure_{Size}(x) = |[x]_{\equiv}|$$

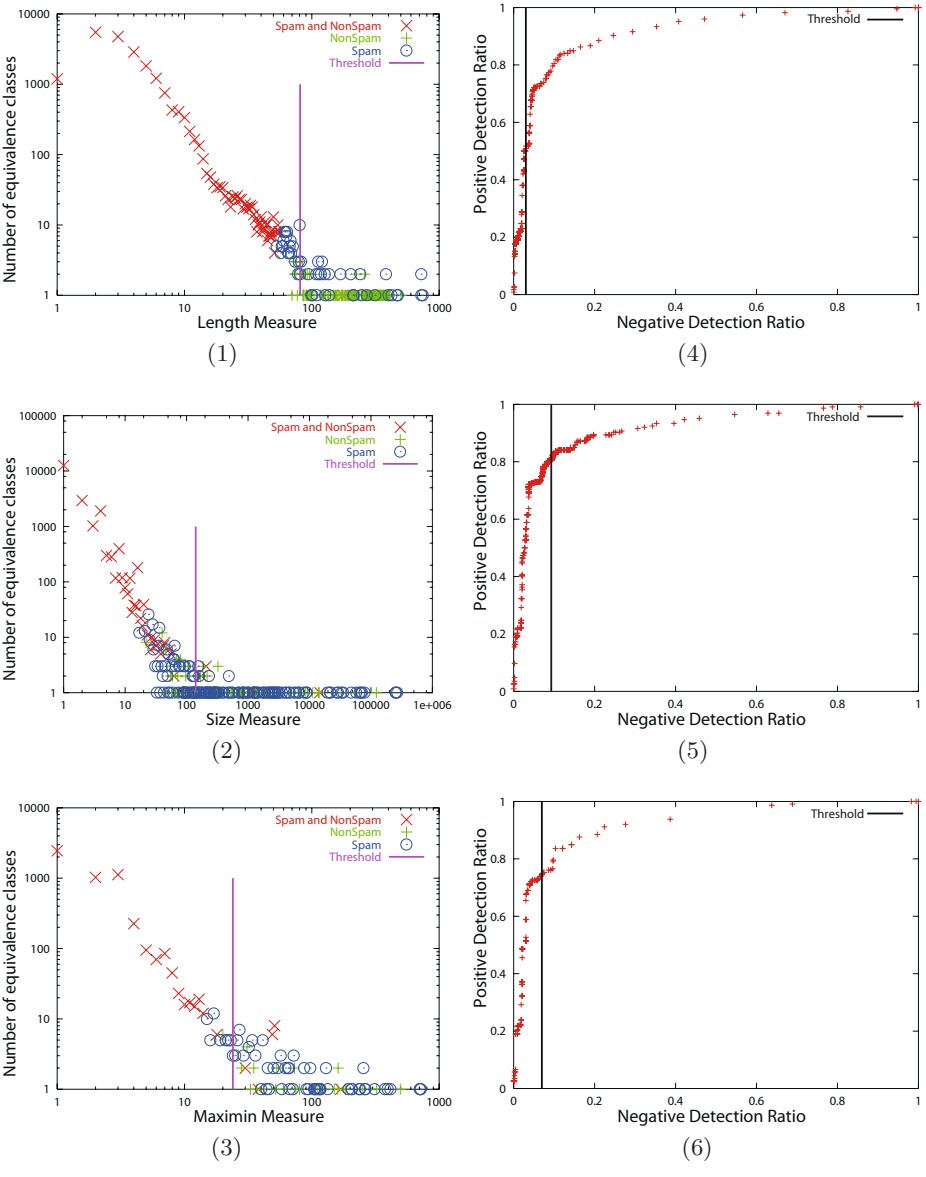

Fig. 3. Measure value distribution of equivalence classes for (1)Length, (2)Size and (3)Maximin. The threshold line is obtained by our method described in Section 4.3. (4)Length, (5)Size and (6)Maximin measure ROC curve for discriminating between spam and nonspam documents.

Although there is a strong relationship between the length of the representative of equivalence classes and the size of equivalence classes (see Figure 4), there are equivalence classes whose representative is long, but whose size is not large.

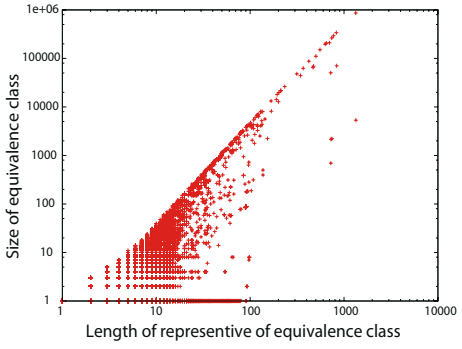

Fig. 4. Relation between the length of the representative of the equivalence class and the size of the equivalence class

There also seems to exist a power law between the size of an equivalence class and the number of equivalence classes with the size (see Figure 3-(2)). In this plot, there are more spam specific points than that of the length measure when the measurement is high.

Figure 3-(5) is the ROC curve for the Size measure. This curve shows that the Size measure is also effective.

Maximin: The Maximin measure is defined as the difference between the length of the representative and the length of the longest minimal element of the equivalence class:

$$measure_{Maximin}(x) = |\overleftrightarrow{x}| - Maximin(x)$$

This measure represents a lower bound on how much a representative string can be shortened (by removing its prefix and/or suffix) and still be in the same equivalence class.

Figure 3-(3) shows the relation between the $measure_{Maximin}(x)$ and the number of the equivalence classes with the measure.

Figure 3-(6) shows the ROC curve for this measure, showing that this measure is also effective for spam detection. In addition, the area under the ROC curve given the negative ratio is less than 0.3 is larger than those of other measures (see Table 1). Hence, we expect that this measure has lower false-positive error.

4.3 Determining the Threshold Value

Our method takes an unlabeled document set as input. We simply use a threshold to determine whether an equivalence class with some measure value is "alien" or not. Documents are then judged as spam if it includes an "alien" substring equivalence class. Below, we describe an unsupervised method for determining the threshold value.

Looking at each of the measure plots, Figure 3-(1), (2), (3), we can see that there are roughly two parts. One is the spam specific part on right area, and

Table 1. The area under the ROC curve for each measure

negative ratio	≤ 0.3	≤ 0.5	≤ 1.0 (all)
Length	0.282	0.485	0.913
Size	0.243	0.469	0.915
Maximin	0.318	0.559	0.919

the rest is on the left area. We propose heuristics to find a point that separates the spam part from the nonspam part, and regard the point as the threshold. More precisely, we model each of the two parts using a linear model, and we look for the point of separation where the two linear models best explains the data points.

Let $S = ((x_1, y_1), \ldots, (x_n, y_n))$ be a sequence of n points, where $x_1 \leq x_2 \leq \cdots \leq x_n$. For $1 \leq k < n$, let S_1^k be the sequence of the first k points in S, and let S_2^k be the remaining sequence in S. We choose the k^*-th point that minimizes the sum of the least square errors in the left and the right sides of points. That is, we choose

$$k^* = \operatorname*{argmin}_{1 \leq k < n} (LSE(S_1^k) + LSE(S_2^k)),$$

where

$$LSE(S') = \min_{a,b} \sum_{i=1,\ldots,n'} (y'_i - ax'_i - b)^2$$

for $S' = ((x'_1, y'_1), \ldots, (x'_{n'}, y'_{n'}))$. It is well known that

$$LSE(S') = \sum_{i=1,\ldots,n'} (y'_i - \hat{a}x'_i - \hat{b})^2,$$

where

$$\hat{a} = \frac{n' \sum_{i=1}^{n'} x'_i y'_i - \sum_{i=1}^{n'} x_i \sum_{i=1}^{n'} y'_i}{n' \sum_{i=1}^{n'} {x'_i}^2 - (\sum_{i=1}^{n'} x'_i)^2},$$

and

$$\hat{b} = \frac{n \sum_{i=1}^{n'} {x'_i}^2 \sum_{i=1}^{n'} y'_i - \sum_{i=1}^{n'} x'_i y'_i \sum_{i=1}^{n'} x'_i}{n' \sum_{i=1}^{n'} {x'_i}^2 - (\sum_{i=1}^{n'} x'_i)^2}.$$

5 Computational Experiments

We detect spams in four forums of Yahoo Japan Finance [2], which are collected in the same way as the data used for evaluating the measures in Section 4. If spam is posted in these forums, the spam is manually deleted by the forum administrator. We regard the deleted posts as spam.

[2] `http://quote.yahoo.co.jp`

Table 2. The results of each measure for YahooJapanFinance forum data

forum	# of spam	# of nonspam	measure	Recall	Precision	F-score	nonspam(%)
4314	291	1424	Length	0.45	0.57	0.50	**2.67**
			Size	**0.89**	0.60	0.72	7.51
			Maximin	0.80	**0.80**	**0.80**	6.18
4974	331	1315	Length	0.39	**0.77**	0.52	**4.33**
			Size	**0.63**	0.69	0.66	11.56
			Maximin	0.60	0.75	**0.67**	9.81
6830	317	1613	Length	0.40	**0.72**	0.52	**3.47**
			Size	**0.74**	0.57	0.64	17.73
			Maximin	0.69	0.69	**0.69**	13.95
8473	264	1597	Length	0.57	**0.76**	0.65	**4.32**
			Size	**0.72**	0.63	0.67	8.14
			Maximin	0.67	0.69	**0.68**	9.39

We selected and collected data from four forums: 4314[3], 4974[4], 6830[5], 8473[6]. We detect spams from these four data sets using the three measures proposed in Section 4. The results are shown in Table 2, where the values Recall, Precision, F-score in the table are defined as follows:

$$Recall = \frac{\#\text{ of detected spam documents}}{\#\text{ of spam documents}}$$

$$Precision = \frac{\#\text{ of detected spam documents}}{\#\text{ of detected documents}}$$

$$F\text{-}score = \frac{2 * Recall * Precision}{Recall + Precision}$$

As shown in Table 2, the measure Maximin has the highest F-scores, which vary from 68% to 80%, among three measures for all of the data sets. On the other hand, as for recall values, the measure Size outperforms others, while its F-scores remain close to those of Maximin. We also evaluate the three measures when the inputs are nonspam documents only. The percentages of documents judged as spams in this setting are summarized in the column "nonspam" of Table 2. Note that the value nonspam should be 0% ideally. The measure Length has the lowest nonspam values over all the data sets. In summary, none of the three measures completely outperforms the others.

We examined the false positive strings (nonspams which our method judges to be spams) and the false negative strings (spams which our method judges to be nonspams). In the former case, most of the false positive strings were difficult to distinguish from spams by their contents, even for human. In the latter case, many false negative strings contained abusive language, which are not spams in

[3] http://messages.yahoo.co.jp/?action=q&board=4314
[4] http://messages.yahoo.co.jp/?action=q&board=4974
[5] http://messages.yahoo.co.jp/?action=q&board=6830
[6] http://messages.yahoo.co.jp/?action=q&board=8473

general but are deleted by the forum administrator. So, in a practical point of view, our method detects spams well.

6 Conclusions and Discussions

We proposed a new, unsupervised, language independent method for spam detection based on the alienness of strings. We provided three alienness measures, namely Length, Size and Maximin. We observed that spam documents seem to give rise to equivalence classes with measurements larger than nonspam documents, and developed a method for finding a threshold value for discriminating between spam and nonspam. To the best of our knowledge, our method is the only method that is truly unsupervised, and requires no tuning of parameters.

Since our method depends on the redundant information contained in spam documents, most existing benchmark datasets that remove this could not be used. The data used in our experiments was the only data readily available to us that consisted of manually annoated positive and negative examples taken from an unprocessed, "natural" distribution of documents. It should also be noted that Japanese is a very popular language on the Web, and according to [18], it is the most frequently used language for blogs. We are currently collecting more data in order to further evaluate and improve our method.

References

1. Jansen, B.J.: Adversarial information retrieval aspects of sponsored search. In: Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb) (2006), http://airweb.cse.lehigh.edu/2006/
2. CNETNEWS.COM: Tempted by blogs, spam becomes 'splog' (2005), http://news.com.com/2100-1032_3-5903409.html
3. CNETNEWS.COM: Spim, splog on the rise (2006), http://news.com.com/2100-7349_3-6091123.html
4. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. Stanford Digital Library Working Paper (1998)
5. Takeda, M., Matsumoto, T., Fukuda, T., Nanri, I.: Discovering characteristic expressions in literary works. Theoretical Computer Science 292(2), 525–546 (2003)
6. Narisawa, K., Yamada, Y., Ikeda, D., Takeda, M.: Detecting blog spams using the vocabulary size of all substrings in their copies. In: Proceedings of the 3rd Annual Workshop on Weblogging Ecosystem(at the 15th World Wide Web Conference) (2006)
7. Benczú, A.A., Csalogány, K., Sarlós, T.: Link-based similarity search to fight web spam. In: Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb) (2006), http://airweb.cse.lehigh.edu/2006/
8. Becchetti, L., Castillo, C., Donato, D., Leonardi, S., Baeza-Yates, R.: Link-based characterization and detection of web spam. In: Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb) (2006), http://airweb.cse.lehigh.edu/2006/

9. da Costa Carvalho, A.L., Chirita, P.A., de Moura, E.S., Calado, P., Nejdl, W.: Site level noise removal for search engines. In: Proceedings of the 15th World Wide Web Conference (2006)
10. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A Bayesian approach to filtering junk e-mail. In:AAAI Workshop on Learning for Text Categorization (1998), ftp://ftp.research.microsoft.com/pub/ejh/junkfilter.pdf
11. Yoshida, K., Adachi, F., Washio, T., Motoda, H., Homma, T., Nakashima, A., Fujikawa, H., Yamazaki, K.: Density-based spam detector. In: KDD, pp. 486–493 (2004)
12. Zipf, G.K.: Human Behavior and the Principle of Least Effort. Addison-Wesley, Reading (1949)
13. Zipf, G.K.: The Psycho-Biology of Language: An Introduction to Dynamic Philology. Houghton Mifflin (1935)
14. Blumer, A., Blumer, J., Haussler, D., Ehrenfeucht, A., Chen, M.T., Seiferas, J.I.: The smallest automaton recognizing the subwords of a text. Theoretical Computer Science 40, 31–55 (1985)
15. Manber, U., Myers, G.: Suffix arrays: a new method for on-line string searches. SIAM Journal on Computing 22(5), 935–948 (1993)
16. Narisawa, K., Inenaga, S., Bannai, H., Takeda, M.: Efficient computation of substring equivalence classes with suffix arrays. In: Ma, B., Zhang, K. (eds.) CPM 2007. LNCS, vol. 4580, pp. 340–351. Springer, Heidelberg (2007)
17. Kasai, T., Lee, G., Arimura, H., Arikawa, S., Park, K.: Linear-time longest-common-prefix computation in suffix arrays and its applications. In: Amir, A., Landau, G.M. (eds.) CPM 2001. LNCS, vol. 2089, pp. 181–192. Springer, Heidelberg (2001)
18. Technorati: The state of the live web (2007), http://technorati.com/weblog/2007/04/328.html