International Journal of Computational Intelligence Systems, Vol. 5, No. 3 (June, 2012), 505-518

Utilizing Multi-Field Text Features for Efficient Email Spam Filtering

Wuying Liu
College of Computer, National University of Defense Technology
Changsha, Hunan 410073, China
wyliu@nudt.edu.cn

Department of Language Engineering, PLA University of Foreign Languages
Luoyang, Henan 471003, China

Ting Wang*
College of Computer, National University of Defense Technology
Changsha, Hunan 410073, China
tingwang@nudt.edu.cn

Received 12 December 2010
Accepted 24 January 2012

Abstract

Large-scale spam emails cause a serious waste of time and resources. This paper investigates the text features of email documents and the feature noises among multi-field texts, resulting in an observation of a power law distribution of feature strings within each text field. According to the observation, we propose an efficient filtering approach including a compound weight method and a lightweight field text classification algorithm. The compound weight method considers both the historical classifying ability of each field classifier and the classifying contribution of each text field in the current classified email. The lightweight field text classification algorithm straightforwardly calculates the arithmetical average of multiple conditional probabilities predicted from feature strings according to a string-frequency index for labeled emails storing. The string-frequency index structure has a random-sampling-based compressible property owing to the power law distribution and can largely reduce the storage space. The experimental results in the TREC spam track show that the proposed approach can complete the filtering task in low space cost and high speed, whose overall performance 1-ROCA exceeds the best one among the participators at the trec07p evaluation.

Keywords: Email Spam Filtering, Text Classification, Multi-Field Learning, Lightweight Field Classifier, Power Law, TREC Spam Track.

1. Introduction

Email spam is the bulk, promotional and unsolicited message. Currently, with the rapid development of computing and communicating technologies, spam emails increase exponentially and spread over the world, which causes a serious waste of time and resources. So it is crucial to filter spam emails efficiently.

Email spam filtering is normally defined as an online supervised binary text classification (TC) problem, which is simulated as an immediate full feedback task in the TREC evaluation. This online learning task [1] assumes that the messages are ordered chronologically. At the beginning of the task, the filter

* Corresponding author.

has no labeled message. Messages are processed in their chronological sequence and the user feedback (category label) for each message is communicated to the filter immediately following classification.

Up to now, many online TC algorithms [2] have been proposed for spam filtering. For instance: 1) the perceptron algorithm is an early online additive weight-updating algorithm [3], a multiplicative variant of the perceptron is the positive winnow algorithm, and a further variant of the positive winnow is the balanced winnow algorithm; 2) the online Bayesian algorithm [4] is based on the vector space model (VSM), and uses the joint probabilities of words and categories to estimate the probabilities of categories for a given document; 3) the kernel-based online support vector machines (SVM) algorithm [5] is a recent algorithm, and the relaxed online SVM algorithm [6] relaxes the maximum margin requirement and produces nearly equivalent results; and 4) the online fusion of dynamic markov compression (DMC) and logistic regression algorithm [7] is a fusion algorithm.

The online TC algorithms have shown a good effectiveness for email spam filtering [8]. To our knowledge, the online SVM algorithm indeed gives state of the art performance on email spam filtering, which has out-performed other single TC algorithms, including the perceptron algorithm, the Bayesian algorithm, and the logistic regression algorithm [9]. The relaxed online SVM algorithm has gained several best results in the TREC2007 spam track. The online fusion of DMC and logistic regression on character 4-gram algorithm is the winner at the trec07p evaluation of immediate full feedback.

The previous TC algorithms often pursued the high classification accuracy and the high overall performance 1-ROCA [8] of online supervised learning, without more claiming their low space-time complexity. For instance, specified in the TREC spam track, even the space-time limitation (total 1 GB RAM and 2 sec/email) is still unpractical and horrible in a real large-scale email system, where large-scale emails will form a round-the-clock data stream and there will be more than thousands of emails arriving during 2 seconds.

According to the practical requirement of the large-scale email system, both the classification accuracy and the space-time complexity all should be concerned. This paper addresses the practical email spam filtering and further defines it as a space-time limited online supervised binary TC problem. The main idea is breaking the space-time limited TC problem into multiple simple sub-problems according to the structural feature of email documents, and linearly combining multiple results from sub-problems to form the final decision. The optimal linear combination method and the efficient algorithm for solving sub-problems are two crucial points concerned in this paper.

The multi-field structural feature of email documents [10] can be used to optimize the linear combination performance. Our multi-field learning framework brings the statistical, computational and representational advantages [11] like ensemble learning methods. This paper propose a compound weight method to calculate the linear combination coefficients, considering both the historical classifying ability of each field classifier and the classifying contribution of each text field in the current classified email. This method is similar to the semi-supervised learning technique which uses the information of the current unlabeled email.

Within the multi-field learning framework, the total space-time cost depends on the one of each field classifier. Unfortunately, most previous TC algorithms often have a complex training process, in which the multi-pass scan, VSM representation, feature selection and other complex operations make the algorithm space-time-inefficient, and unsuitable to be implemented as the field classifier for the practical application of email spam filtering. So this paper explores a lightweight field text classification algorithm for the field classifier. This algorithm utilizes an efficient string-frequency index to calculate the arithmetical average of multiple conditional probabilities predicted from feature strings.

The rest of this paper is organized as follows. In section 2, we investigate the text features of email documents and find the feature noises among multi-field texts and the power law distribution of feature strings. In section 3, we describe the multi-field learning idea, and propose an efficient compound weight method. In section 4, we explore a string-frequency index data structure, and propose a lightweight field text classification algorithm. In section 5, the experiments and results are described. At last the conclusion and future work are given.

2. Text Features of Email Documents

In most statistical TC algorithms for email spam filtering, email is normally treated as a single plain-text document, and text feature is also extracted within this single document. Actually a full email (often including five natural text fields: Header, From, ToCcBcc, Subject, and Body) is a multi-field text document. Feature extraction from full email document makes many text features disturb each other, and text feature from one field is often noise to other fields.

2.1. *Multi-Field Text Feature Noises*

In statistical TC algorithms, a document is normally represented as a text feature vector. The dimension of feature vector space, the total number of text features, reflects the representational granularity of vector space model. Previous researches have shown that the overlapping word-level k-gram model can achieve promising results [12, 13]. For email documents, single plain-text model (SPTM) and multi-field model (MFM) are two different representations. The SPTM ignores the field information of text feature, regarding the same string occurrence in different fields as single text feature, while the MFM treats it as distinct text features. The dimensions of k-gram feature vector space for trec07p email set are shown in Table 1. For the two email representations, four overlapping word-level models are applied respectively. For MFM, the five natural text fields' information is considered.

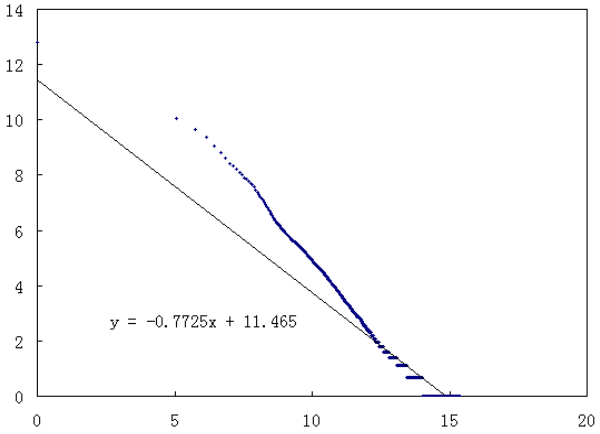
Table 1. Dimensions of *k*-gram Feature Vector Space under the Two Representations: SPTM and MFM.

	1-gram	2-gram	3-gram	4-gram
SPTM	1,037,395	4,189,054	9,447,962	13,869,560
MFM	1,258,491	4,906,594	10,390,571	14,880,647

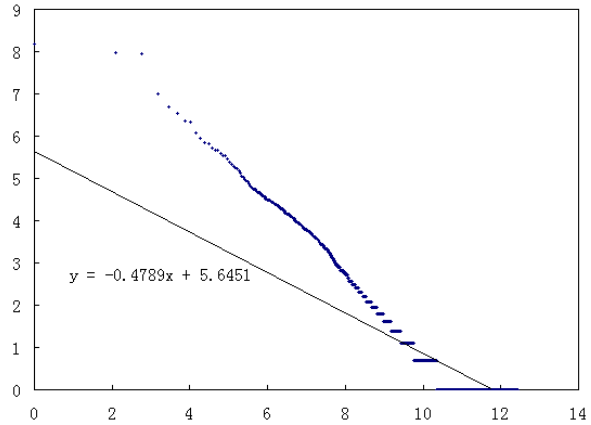
Table 1 shows the dimension of MFM is larger than that of SPTM for each k-gram model. For instance, the difference between the two representations is 1,011,087 for the 4-gram model. The result from Table 1 indicates that text feature noises exist indeed in SPTM. Because more finely granular text feature can reduce the noises and increase the TC accuracy, this paper proposes a multi-field learning (MFL) framework, which is an alignment technique of text feature sources. In the MFL framework, text features are enhanced by the field information, and the undesired influences among text features from different fields are expected to be reduced.

2.2. *Power Law Distribution of Feature Strings*

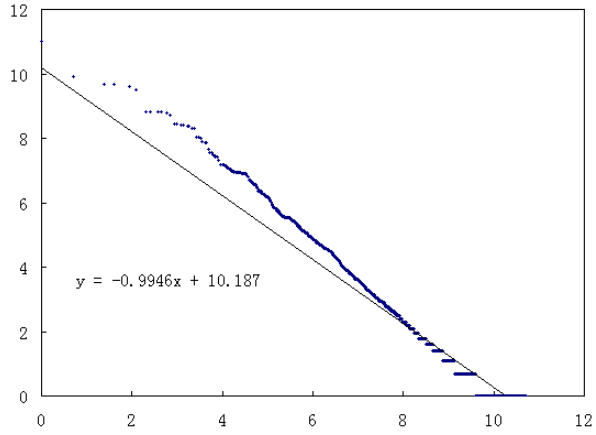
Within the MFL framework, the distribution of feature strings from each field text is crucial to develop an efficient field text classification algorithm. It has been proved that the number distribution of words follows the power law in most text documents. We can validate that the power law also exists in each field of email documents through the statistic of feature strings. We consider the five natural text fields of email documents separately, and for each text field we calculate the number of each feature string for trec07p email set. Here, the feature string also applies the four overlapping word-level models.



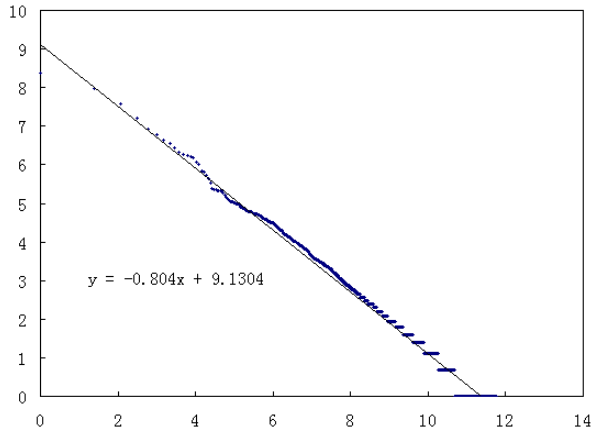
(a) Frequency Distribution of 4-gram Feature Strings and Trendline in the Header Field



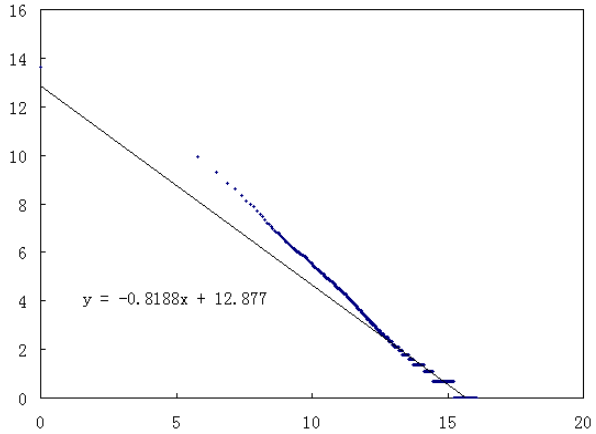
(b) Frequency Distribution of 4-gram Feature Strings and Trendline in the From Field



(c) Frequency Distribution of 4-gram Feature Strings and Trendline in the ToCcBcc Field



(d) Frequency Distribution of 4-gram Feature Strings and Trendline in the Subject Field



(e) Frequency Distribution of 4-gram Feature Strings and Trendline in the Body Field

Fig. 1. Frequency Distributions of 4-gram Feature Strings and Trendlines in the Five Natural Text Fields. In each sub-figure, the horizontal-axis indicates the frequency rank of feature string (log scale), and the vertical-axis indicates the frequency of feature string (log scale).

Fig. 1 includes five sub-figures corresponding to the five natural text fields of email documents. In each sub-figure of Fig. 1, the horizontal-axis (x-axis) indicates the frequency rank of feature string (log scale), and the vertical-axis (y-axis) indicates the frequency of feature string (log scale). We can draw a trendline (y = ***a*** x + ***b***) for each sub-figure of Fig. 1, which indicates that the number distribution of 4-gram feature strings approximately follows the power law.

Our statistic shows that not only the number distribution of 4-gram feature strings follows the power law in every text field, but also that of 1-gram, 2-gram, and 3-gram feature strings follows the power law. Table 2 shows the detailed trendline coefficients ***a*** and ***b***. The ubiquitous power law brings a new opportunity to reduce storage space via deleting the low frequency feature strings.

Table 2. Trendline (y = ***a*** x + ***b***) Coefficients ***a*** and ***b*** for *k*-gram Feature Strings in the Five Natural Text Fields.

	1-gram		2-gram		3-gram		4-gram	
	a	***b***	***a***	***b***	***a***	***B***	***a***	***b***
Header	-0.8707	11.364	-0.8543	12	-0.8076	11.73	-0.7725	11.465
From	-0.8405	9.0269	-0.6313	7.3073	-0.5118	6.0107	-0.4789	5.6451
ToCcBcc	-1.2365	11.196	-1.1375	11.165	-1.0443	10.525	-0.9946	10.187
Subject	-1.3741	13.416	-0.9913	11.023	-0.8566	9.7211	-0.804	9.1304
Body	-1.4133	17.804	-1.1895	17.245	-0.9327	14.355	-0.8188	12.877

3. Compound Weight within the MFL Framework

A full email is a typical multi-field text document. Applying the divide-and-conquer strategy, multi-field learning breaks a complex TC problem into multiple simple sub-problems according to the structural feature of multi-field text documents. Each sub-problem may have its specific text features, and the combined multiple classifying results will be expected to improve the final classification accuracy.

3.1. *Multi-Field Learning Framework*

Fig. 2 shows the multi-field learning framework for the binary TC of multi-field documents, including a splitter, several field classifiers, a combiner, and an immediate learner.

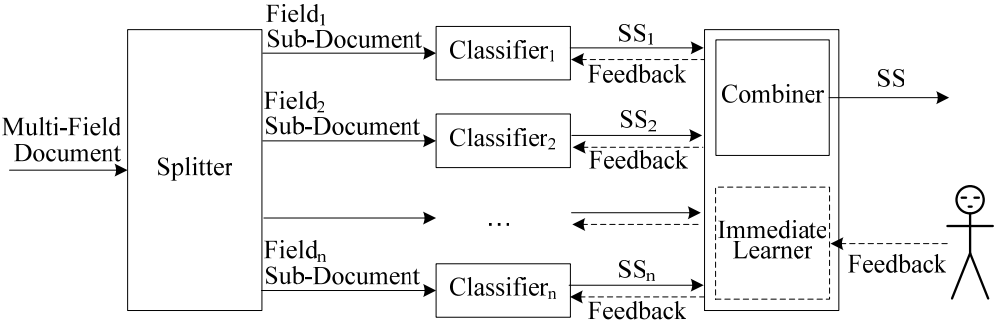


Fig. 2. Multi-Field Learning Framework.

The splitter analyses a multi-field document, and splits it into several sub-documents. There are two kinds of sub-documents: one is the natural field sub-document, and the other is the artificial sub-document. There are some explicit classifiable texts hard to be pretended by spammers in email documents. For instance, spammers try to camouflage the spam text, but they never conceal the email addresses with expectation to be called back from the spam receivers. So, by some regular expression rules, we can extract these specific texts to form artificial sub-documents which do not really exist in actual multi-field documents. The artificial sub-document construction is equivalent to increasing the statistical weight for some specific texts. This paper implements a MFL framework of seven sub-documents for email documents, in which the splitter extracts five natural sub-documents (Header, From, ToCcBcc, Subject, and Body) according to the natural field structure and extracts two artificial sub-documents (H.IP, H.EmailBox) by regular expression rules. The H.IP sub-document contains IP address texts and the H.EmailBox sub-document contains email address texts within the Header field of email documents.

Each field classifier is obligated to process its related field sub-documents. Text feature extracting, TC model training or updating, and sub-document predicting made by each field classifier are only localized in its related sub-documents. The output of each field classifier is not the traditional binary result but a spamminess score (SS), which is a real number reflecting the likelihood that the classified document is spam. The traditional Bayesian conditional probability $P(spam|doc)$, shown in Eq. (1), reflects this likelihood.

$$P(spam \mid doc) = \frac{P(doc \mid spam)P(spam)}{P(doc \mid spam)P(spam) + P(doc \mid ham)P(ham)} \quad (1)$$

If the $P(spam|doc)$ is applied to estimate the SS, then the SS threshold T, shown in Eq. (2), can be used to make a binary judgment. But the values of both SS and threshold are affected by the number distribution of labeled spams and hams, and the number of two categories labeled data is not fixed during the time of online filtering.

$$T = \frac{P(spam)}{P(spam) + P(ham)} \quad (2)$$

In order to eliminate this number distribution influence and make the same SS value has the equivalent likelihood during the whole online filtering, this paper scales up the number of two categories

labeled data to make $P(spam)=P(ham)$, and uses the scaled Bayesian conditional probability $P(spam|doc)$, shown in Eq. (3), to represent the SS, then the SS threshold $T=0.5$ will be a fixed point.

$$P(spam \mid doc) = \frac{P(doc \mid spam)}{P(doc \mid spam) + P(doc \mid ham)} \tag{3}$$

The MFL framework is designed for a general purpose, easily to be applied to previous TC algorithms, because previous TC algorithms can be used to implement the field classifier by changing a binary result output to a continuous SS output. From the perspective of machine learning, the MFL framework adds a document-level category label to each sub-document. Each field classifier can develop more sophisticate features and train a TC model in its own feature space, which reduces the feature disturbance between the sub-documents and makes the TC model more precise.

The combiner combines multi-classifier's scores to form the final SS. If the final $SS \in [0, T]$, then the document will be predicted as a ham; otherwise, if the final $SS \in (T, 1]$, it will be predicted as a spam. Immediately after the classification decision, the immediate learner sends the user feedback (category label) to each field classifier for its TC model updating. The linear combination strategy in the combiner is the key point to guarantee good effectiveness within the MFL framework.

3.2. *Compound Weight*

The linear combination of spamminess scores from field classifiers is a simple and efficient method, which is defined in Eq. (4). Here SS denotes the final spamminess score, n denotes the number of field classifiers, and SS_i denotes the spamminess score predicted by the ith field classifier. The coefficient α_i (real number) can be set by different weighted strategies. The straightforward weighted strategy is arithmetical average calculating method: $\alpha_i=1/n$, abbreviated as **strat1**.

$$SS = \sum_{i=1}^{n} \alpha_i SS_i \tag{4}$$

Email spam filtering is an online supervised learning process, so the normalized historical classification accuracy rates of field classifiers can be used to estimate the linear combination coefficients. Within the MFL framework, each field classifier's historical SS values can be plotted to a receiver operating characteristic (ROC) curve. The percentage of the area below the ROC curve, abbreviated as ROCA, indicates the historical classifying ability. So each ROCA is reasonable to estimate the classification accuracy rate of each field classifier. This historical performance weighted strategy was proposed in our previous research [10], abbreviated as **strat2**, where the normalized current n values of ROCA were used to set the coefficient α_i before an email was classified. Our research has also proved that the overall performance of strat2 overcomes that of strat1.

Furthermore, the information amount of the current classified email will also influence the classification accuracy at the time of online predicting. The length of the text in each field sub-document can be used as the measure of the information for each field, which formed the current classifying contribution weighted strategy, abbreviated as **strat3** combining strategy. In this strategy, the normalized number of characters in field sub-documents is used to set the coefficient α_i.

In fact, the strat2 and strat3 strategies are two sides of the same coin. The two strategies, the historical performance weighted strategy and the current classifying contribution weighted strategy, will affect the classification accuracy together. This paper presents a compound weight considering the strat2 and strat3 strategies on the assumption that the two strategies contribute equally to a correct classification. Let α_i^{strat2} and α_i^{strat3} denote separately the coefficient of the strat2 and strat3, then a compound weight, shown in Eq. (5), is used as the coefficient α_i. This compound weighted strategy is abbreviated as **strat4**.

$$\alpha_i = \frac{\alpha_i^{strat2} + \alpha_i^{strat3}}{2} \tag{5}$$

The total space-time cost within the MFL framework depends on the space-time complexity of each field classifier. Unfortunately, most TC algorithms often have a complex training process, which is unsuitable to be implemented as the field classifier for the practical application. So this paper next explores a lightweight field text classification algorithm to implement the field classifier.

4. Lightweight Field Text Classification

Previous TC algorithms normally use a VSM representation to train a TC model, which has to align vector dimensions, select features, and often leads to high dimensional sparseness and time-consuming problems. The online TC algorithm faces an open incremental text feature space, and cannot foreknow the vector space dimension. This paper presents a data structure of string-frequency index (SFI), based on which the proposed lightweight field text classification algorithm (named as SFITC-R algorithm) converts the supervised online training and classifying processes into index incremental updating and retrieving processes. The SFITC-R algorithm smoothly solves the online open text feature space problem, and is space-time-efficient owing to the SFI data structure and suitable to implement the field classifier.

4.1. *String-Frequency Index*

The feature string frequency of historical labeled documents, the key of online supervised machine learning, gives rich classification information and must be stored effectively. This paper applies the overlapping word-level 4-gram model to define feature strings, and lets a sub-document **D** be represented as a sequence of feature strings in the form **D**=S_j, (j=*1*, *2*, ..., *N*). The string-frequency index is a data structure to store the feature string information of labeled data, from which we can conveniently calculate spamminess score of each feature string according to the scaled Bayesian conditional probability $P(spam|S_j)$, and straightforwardly combine the scores to form the sub-document's final score.

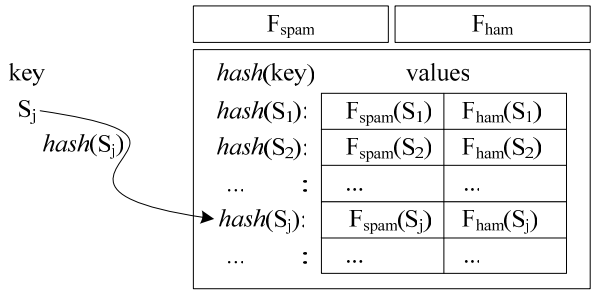


Fig. 3. String-Frequency Index.

Fig. 3 shows the SFI including two integers and a hash table. The integral F_{spam} and F_{ham} denote separately the total number of labeled spams and hams, which are then scaled up in order to make $P(spam)=P(ham)$. Each table entry is a key-value pair <Key, Value>, where each key is a feature string and each value consists of two integers. Similarly, the integral $F_{spam}(S_j)$ and $F_{ham}(S_j)$ denote separately the number of occurrences of feature string S_j in labeled spams and hams, and the S_j denotes the jth feature string. The hash function maps the feature string S_j to the address of two integers $F_{spam}(S_j)$ and $F_{ham}(S_j)$.

The data structure of SFI is a little similar to the widely used word-frequency matrix (WFM). The main difference between SFI and WFM is their design motivations: the SFI is designed to represent a frequency distribution of strings within two categories, and the WFM is designed to represent a frequency distribution of words within documents. Due to different motivations, the SFI has more advantages comparing to the WFM for binary TC applications. The WFM storage space is proportional to the number of documents, while the SFI storage space is proportional to the number of categories. In the binary TC, the number of documents is usually larger than two, so the SFI is more efficient than the WFM considering the storage space.

```
// SFITC-R: String-frequency index text classification algorithm (with random sampling).
// D: Sub-Document; L: Binary Category Label; SFI: String-Frequency Index; R: Training Feature Loss Rate.
SFITC-R (D; L; SFI; R)
(1) If (L = null) Then: PREDICT (D; SFI);
(2) Else: UPDATE (D; L; SFI; R).

// PREDICT: Online classifying procedure.
PREDICT (D; SFI)
(1) String[] S := FEATURE(D);
(2) Integer I_s := SFI.F_spam;
(3) Integer I_h := SFI.F_ham;
(4) New ArrayList<Float> F;
(5) If (I_s = 0) Or (I_h = 0) Then: Float SS_i := 0.5;
```

```
(6) Else:
   (6.1) Loop: For Each S_j∈S Do:
      (6.1.1) If (SFI.containKey(S_j)) Then:
         (6.1.1.1) Integer I_sj := SFI.F_spam(S_j);
         (6.1.1.2) Integer I_hj := SFI.F_ham(S_j);
         (6.1.1.3) Float SS_j := (I_sj/I_s)/(I_sj/I_s+I_hj/I_h);
         (6.1.1.4) F.add(SS_j);
   (6.2) Integer N := F.length;
   (6.3) If (N = 0) Then: Float SS_i := 0.5;
   (6.4) Else: Float SS_i := (1/N)∑SS_j; // SS_j∈F
(7) If (SS_i > 0.5) Then: Label L := spam;
(8) Else: Label L := ham;
(9) Output: SS_i and L.

// UPDATE: Online training procedure.
UPDATE (D; L; SFI; R)
(1) String[] S := FEATURE(D);
(2) String[] SRS := RANDOMSAMPLING(S; R);
(3) If (L = spam) Then:
   (3.1) SFI.F_spam := SFI.F_spam + 1;
   (3.2) Loop: For Each S_j∈SRS Do:
      (3.2.1) If SFI.containKey(S_j) Then: SFI.F_spam(S_j) := SFI.F_spam(S_j) + 1;
      (3.2.2) Else: SFI.putKey(S_j), And SFI.F_spam(S_j) := 1, SFI.F_ham(S_j) := 0;
(4) Else If (L = ham) Then:
   (4.1) SFI.F_ham := SFI.F_ham + 1;
   (4.2) Loop: For Each S_j∈SRS Do:
      (4.2.1) If (SFI.containKey(S_j)) Then: SFI.F_ham(S_j) := SFI.F_ham(S_j) + 1;
      (4.2.2) Else: SFI.putKey(S_j), And SFI.F_spam(S_j) := 0, SFI.F_ham(S_j) := 1.

FEATURE (D) //Extract the feature string sequence from D based on overlapping word-level 4-gram model.
RANDOMSAMPLING(S; R) //Random sample the feature string sequence based on the training feature loss rate R.
```

Fig. 4. Pseudo-Code for the SFITC-R Algorithm.

Supported by the SFI, the SFITC-R algorithm takes the online classifying process of a sub-document as an index retrieving process, and also takes the supervised online training process as an incremental updating process of index. Fig. 4 gives the pseudo-code for the SFITC-R algorithm consisting of two main procedures: PREDICT and UPDATE.

When a new (Label=null) sub-document arrives, the PREDICT procedure is triggered: 1) it extracts the feature string sequence from the sub-document based on the overlapping word-level 4-gram model; 2) it retrieves the current SFI and calculates each feature string's SS according to the Eq. (6) scaled Bayesian conditional probability; and 3) it assumes that each feature string's contribution to the final SS is equivalent and uses the arithmetical average to calculate the final SS as Eq. (7).

$$SS_j = P(spam \mid S_j) = \frac{F_{spam}(S_j)/F_{spam}}{F_{spam}(S_j)/F_{spam} + F_{ham}(S_j)/F_{ham}} \quad (6)$$

$$SS_i = \frac{1}{N}\sum_{j=1}^{N} SS_j \quad (7)$$

When a new labeled sub-document arrives, it is only required that the sub-document's feature strings are put into the SFI. The UPDATE procedure firstly extracts the feature string sequence, and then randomly samples the feature string sequence to form a new compressed sequence based on a preset training feature loss rate **R**; and finally, updates the frequency or adds a new index entry to the SFI according to the feature strings within the compressed sequence.

4.2. *Space-Time Complexity*

The SFITC-R algorithm mainly makes up of PREDICT and UPDATE procedures, whose space-time complexity depends on the SFI storage space and the loops in the two procedures.

The SFI storage space is efficient owing to two reasons: the inherent compressible property of index

files and the random-sampling-based compressible property at the time of incremental updating.

The SFI is an improved version of traditional inverted files [14], which simplifies the position and document ID information to two integers, only reflecting the occurrence frequency of feature strings. This hash list structure, prevailingly employed in Information Retrieval, has a lower compression ratio of raw texts. Though the training sub-documents will mount in the wake of the increasing of online feedbacks, the SFI storage space will only increase slowly. Theoretically, the inherent compressible property of index files ensures that the SFI storage space is proportional to the total number of feature strings, and is independent of the number of training sub-documents.

The random-sampling-based compressible property of SFI is caused by the number distribution of feature strings in email documents and our SFI-based calculating method. During the whole filtering, if a feature string has less frequency (≤2) in the SFI, it can be believed that the feature string is useless for classification. As an extreme instance, if a feature string occurs only once all the time, it is useless because it will never be used in the future. The SFITC-R algorithm only utilizes the ratio of a feature string's frequency in spams to that in hams. So the SFI's classifying ability will not change after removing the useless feature strings.

As the online spam filtering faces an open text space, you can not foreknow the feature string's occurrence in the future. It has been found that the number distribution of feature strings follows the power law in email sub-documents. According to this finding, we define the uselessness rate as the ratio of the number of feature strings with less frequency to the total number of feature strings in the SFI, and show the number of feature strings and related uselessness rate of trec07p email set in Table 3.

Table 3. Number of Feature Strings and Uselessness Rate.

	Number of Feature Strings (num)			Uselessness Rate (%)	
	$\boldsymbol{N}(1)$	$\boldsymbol{N}(2)$	$\boldsymbol{N}(*)$	$\boldsymbol{U}(\leq 1)$	$\boldsymbol{U}(\leq 2)$
Body	5636342	2212571	9739679	58	81
From	217976	14123	249322	87	93
Header	3516042	505070	4720386	75	85
H.EmailBox	406052	86012	641531	63	77
H.IP	97120	47151	232521	42	62
Subject	82711	15606	126788	65	78
ToCcBcc	29755	5384	44472	67	79
Multi-Field	9985998	2885917	15754699	63	82

The $\boldsymbol{N}(1)$ and $\boldsymbol{N}(2)$ separately denote the number of feature string which only occurs once and twice in the trec07p email set. The $\boldsymbol{N}(*)$ denotes the total number of feature strings. The $\boldsymbol{U}(\leq 1)$ and $\boldsymbol{U}(\leq 2)$ are defined in Eq. (8) and Eq. (9) separately.

$$U(\leq 1) = \frac{N(1)}{N(*)} \tag{8}$$

$$U(\leq 2) = \frac{N(1) + N(2)}{N(*)} \tag{9}$$

Table 3 shows that the uselessness rates in seven fields are all higher, and the total uselessness rate is between 63% and 82%, which is the theoretical tolerant range of index entry loss rate. It indicates if we complete the whole trec07p filtering of immediate full feedback, the SFI will include 63% to 82% useless index entry. Lots of useless feature strings form the "long tail", which confirm our conjecture.

The number distribution of feature strings follows a power law, and our SFITC-R algorithm only requires the relative frequency cause that the random-sampling-based feature strings selection can cut the "long tail" useless feature strings in the online situation.

Both two compressible properties of SFI make that the online labeled email stream can be incrementally stored as the SFI space-efficiently.

The incremental updating or retrieving of SFI has constant time complexity according to a hash function. The major time cost of the online classifying procedure is the cost for $3N+1$ divisions in the loop (see 6.1 of Fig.

4). The online training procedure is lazy, requiring no retraining when a new labeled sub-document is added. From Fig. 4, it is found that the time cost of per updating is only proportional to the total number of feature strings in the sub-document. Except the only loop (see 3.2 and 4.2 of Fig. 4) according to the number of feature strings, there is no time-consuming operations. Above time complexity is acceptable in the practical online spam filtering application.

5. Experiments

The proposed work has been evaluated on a widely-used email spam filtering task of immediate full feedback, defined in the TREC spam track [15]. The hardware environment for running experiments is a PC with 1 GB memory and 2.80 GHz Pentium D CPU. Experimental corpus is trec07p email set, which contains total 75,419 emails (25,220 hams and 50,199 spams). The TREC spam filter evaluation toolkit and the associated evaluation methodology [15] are applied.

5.1. *Implementation and Evaluation*

In the experiments described below, a seven fields MFL framework of email documents was implemented (see section 3.1). Within this MFL framework, each field classifier is implemented based on the SFITC-R algorithm and applies mechanical overlapping word-level 4-gram model to define feature strings. According to the four different weighted strategies: the strat1, strat2, strat3 and strat4 strategy of the MFL framework (see section 3.2), we implement separately four email spam filters: the sfitc1, sfitc2, sfitc3 and sfitc4 filter.

Three other filters are chosen as baselines: 1) the *bogo* filter (bogo-0.93.4) [16, 17] is a classical implementation of VSM-based online Bayesian algorithm; 2) the *tftS3F* filter [18] is based on the relaxed online SVM algorithm and has achieved several best results at the TREC2007 spam track; and 3) the *wat3* filter [7], the winner at the trec07p immediate full feedback spam track, is based on the online fusion of DMC and logistic regression with overall performance 1-ROCA at 0.0055. These two filters can be run in the same environment with our filters, and compared on running time.

Here reports the overall performance measurement 1-ROCA, the area above the ROC curve percentage, where 0 is optimal, and the total running time to evaluate the filter's performance. We also report two measurements: the spam misclassification percentage (Misspam) and the ham misclassification percentage (Misham) to show the validity of our strat4 strategy. All the above measurements are automatically computed by the TREC spam filter evaluation toolkit, which can also plot the ROC curve and the ROC learning curve for ROC analysis. The ROC curve is the graphical representation of Misham and Misspam. The area under the ROC curve is a cumulative measure of the effectiveness of the filter over all possible values. The ROC learning curve is the graphical representation of the filter's behavior and the user's expectation evolve during filter use. The ROC learning curve is that cumulative 1-ROCA is given as a function of the number of messages processed, which indicates that the filter has reached steady-state performance.

5.2. *Results and Discussion*

The experiments include two parts, one evaluates that the SFITC-R algorithm is time-efficient and can also achieve the best overall performance within the MFL framework; the other verifies that the SFI data structure has the random-sampling-based compressible property and the proposed approach is space-efficient.

Table 4. Experimental Results: Time, 1-ROCA, Misspam, Misham Performance and TREC07 Rank.

	Time (sec)	1-ROCA (%)	Misspam (%)	Misham (%)	TREC07 Rank
sfitc4	2834	0.0055	0.21	0.11	
wat3		0.0055			*1*
sfitc2	2776	0.0067	0.16	0.15	
sfitc3	1910	0.0070	0.40	0.08	
sfitc1	1863	0.0074			
tftS3F	62554	0.0093			*2*
bogo	25100	0.1558			

In the first part of experiments, the *bogo*, *tftS3F*, and our four filters run on immediate full feedback task on the trec07p corpus separately, and our four filters set their training feature loss rate **R**=0. The detailed experimental results are shown in Table 4. The results show that the sfitc4 filter can complete filtering task in high speed (2834 sec), whose overall performance 1-ROCA is comparable to the best *wat3* filter's (0.0055) among the participators at the trec07p evaluation. The time and 1-ROCA performances of our four filters exceed the *bogo*'s and the *tftS3F*'s more. Comparing the sfitc2 and the sfitc3 in the percent of misclassified spams and hams, we find that the strat2 strategy optimizes spam's decision (0.16<0.40) and the strat3 strategy optimizes ham's decision (0.08<0.15). The Misspam and Misham of sfitc4 shows that compound weight can consider the both two aspects.

Fig. 5 shows the ROC curves and Fig. 6 shows the ROC learning curves of the *bogo*, *tftS3F*, *wat3*, and our best sfitc4 filter respectively. In Fig. 5, the area surrounded by the left border, the top border and the sfitc4 curve is relatively small, which means that the overall filtering performance of sfitc4 filter is promising. Fig. 5 also shows that the overall performance is comparable among the *tftS3F*, *wat3*, and sfitc4 filters. In Fig. 6, around 7,000 training samples, the sfitc4 curve achieves the ideal 1-ROCA performance (0.01). Comparing the *sfitc4*, *tftS3F* and *wat3* learning curves, we find that all the performances decrease near 20,000 training samples. However, when close to 40,000 training samples, the sfitc4 can quickly return the ideal steady-state, and the average overall performance 1-ROCA can reach 0.0055. This indicates that the SFITC-R algorithm applying strat4 strategy of the MFL framework has strong online learning ability.

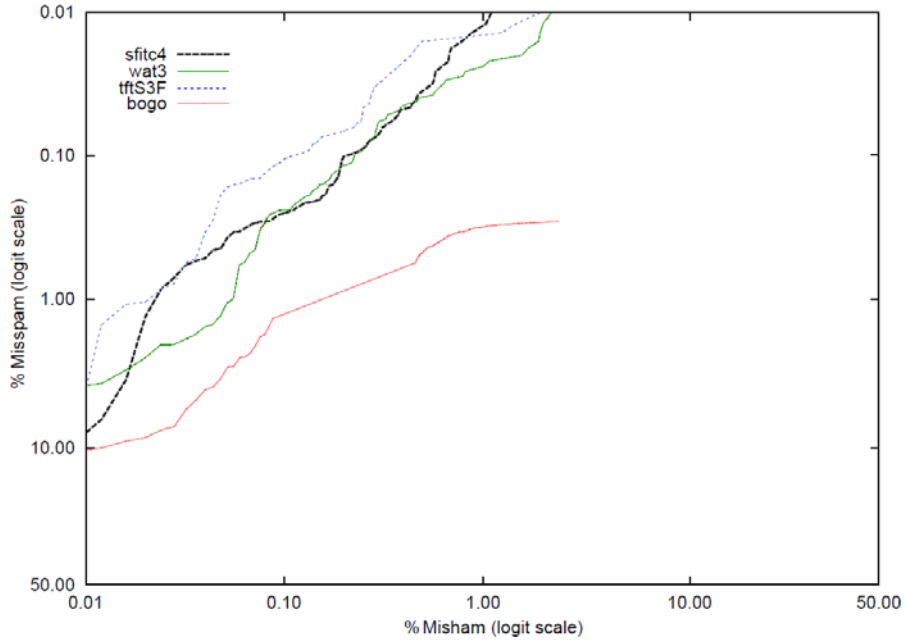


Fig. 5. ROC Curves: Immediate Full Feedback on the Trec07p Corpus.

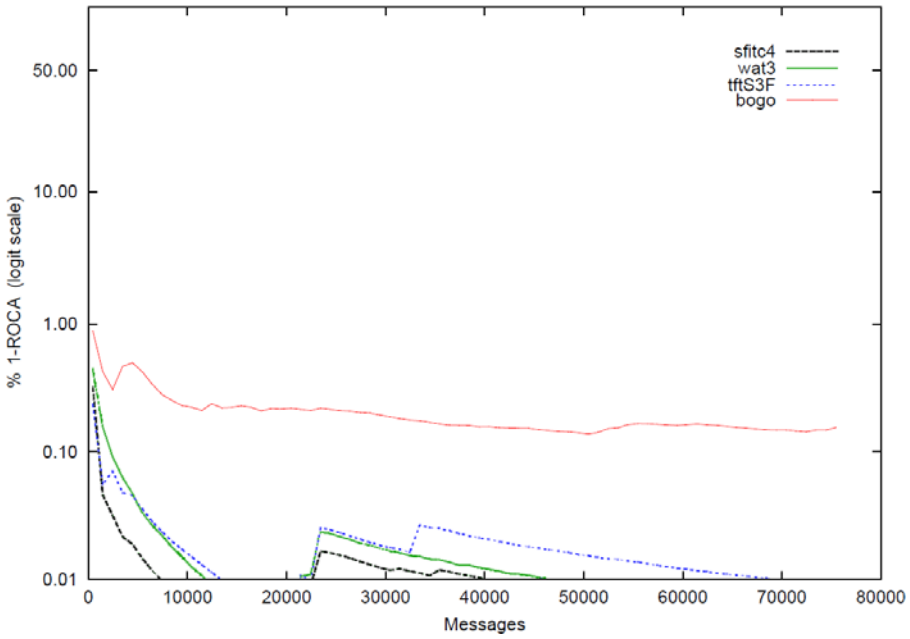


Fig. 6. ROC Learning Curves: Immediate Full Feedback on the Trec07p Corpus.

In the second part of experiments, we run the *sfitc4* filter under different training feature loss rate **R** from 10% to 90%. The *sfitc4* filter repeatedly runs 30 times for each training feature loss rate, and here reports the mean performance among the 30 results for each training feature loss rate.

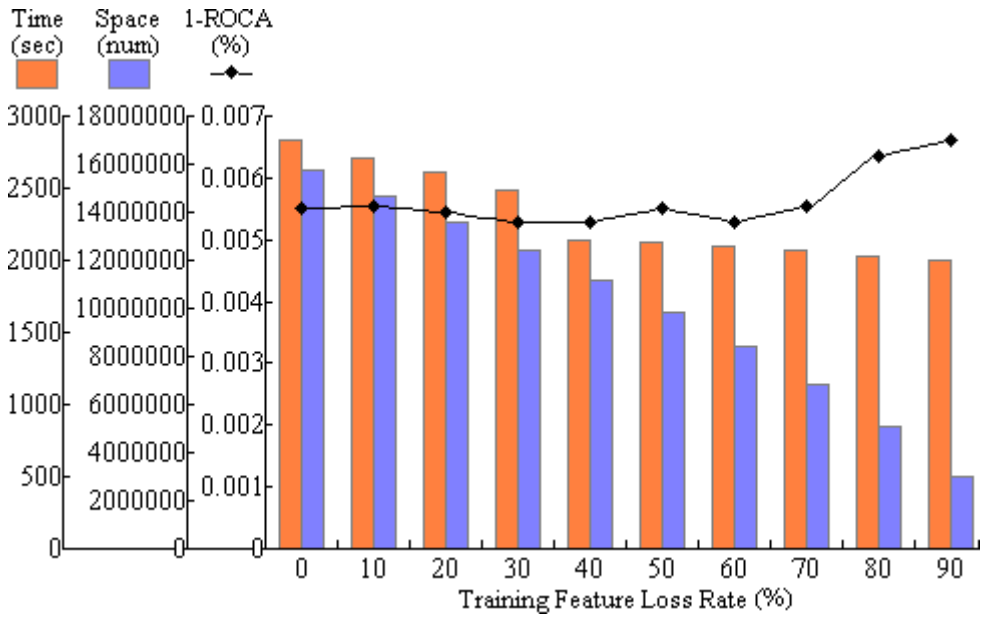


Fig. 7. Time, Space and 1-ROCA Performance under Different Training Feature Loss Rates.

Fig. 7 shows the time, space and 1-ROCA performance, where the space is the number of index entry in the final SFI storage. From Fig. 7, we find that the 1-ROCA performance is almost a constant (≈0.0055) while **R** varing from 0% to 70%, which indicates if we randomly delete up to 70% feature strings at the time of

online training, the 1-ROCA performance will not be influenced obviously. This result confirms the random-sampling-based compressible property of SFI in our SFITC-R algorithm within the MFL framework for email spam filtering.

Table 5. Training Feature Loss Rate, Index Entry Loss Rate and Performance.

Training Feature Loss Rate (%)	Index Entry Loss Rate (%)	Time (sec)	Space (num)	1-ROCA (%)
0	0	2834	15754699	0.0055
10	6	2715	14763087	0.0055
20	13	2607	13660951	0.0054
30	21	2481	12511131	0.0053
40	29	2139	11257499	0.0053
50	37	2130	9895697	0.0055
60	46	2094	8467245	0.0053
70	56	2066	6860210	0.0055
80	68	2028	5071819	0.0064
90	81	2006	2984139	0.0066

The details of loss rate and performance are shown in Table 5. On average of 30 results, there are four 1-ROCA performances exceed the best one (0.0055) among the participators at the trec07p evaluation. From Table 5, we find that the index entry loss rate approximates a direct ratio of the training feature loss rate, which proves that random-sampling-based feature strings selection according to theoretical index entry loss rate between 63% and 82% is effective in the online situation.

Above experiments show that 1) the SFITC-R algorithm and the compound weight method can achieve high classification accuracy in low time complexity within the MFL framework; and 2) the SFI data structure, with two compressible properties, can largely reduce the space complexity of our approach.

6. Conclusions

For large-scale email spam filtering, a practical TC algorithm must manage to obtain high classification accuracy under the restriction of limited time and space. Compared with some advanced machine learning TC algorithms, our proposed approach makes use of the structural feature of email documents and obtains a comparable performance. The experimental results show that, applying the compound weighted strategy within the MFL framework and using the SFI data structure to store previous labeled emails, the SFITC-R algorithm can achieve the state-of-the-art performance at greatly reduced space-time cost, which adapts to practical requirements of large-scale email spam filtering. Based on the above researches, we can draw following conclusions:

- The multi-field structural feature can support the divide-and-conquer strategy. Using an optimal linear combination strategy of multi-field learning, the straightforward occurrence counting of string features may obtain promising classification performance, even beyond that of some advanced algorithms. This straightforward counting will also bring time reducing.
- The index data structure has the inherent compressible property of raw texts, by which the text retrieval approach can be used to treat the text classification problem. Each incremental updating or retrieving of index has constant time complexity, which may satisfy the space-limited and real-time requirements of online applications.
- The ubiquitous power law is a very important distribution feature of random events. This paper elucidates that the number distribution of feature strings follows the power law in email documents, according to which we succeed in random-sampling-based feature strings selection at the time of online training.

With the development of mobile computing and network communicating, the spam concept is generalized to email spam, instant messaging spam, short message service spam, and so on. This paper proposed filtering approach is more general and can be easily transferred to other spam filtering. Further research will concern online semi-supervised learning, active learning, and personal learning for spam filtering.

We will apply large-scale unlabeled data, select effective samples for training by mining differences among multiple field classifiers of the MFL framework, and improve the SFI structure for both global and personal labeled text storage.

Acknowledgements

The authors thank the anonymous reviewers for helping to greatly improve the paper. This research is supported by the National Natural Science Foundation of China (No. 61170156, No. 60933005), and the National High Technology Research and Development Program of China (No. 2010AA012505). We thank Dr. D. Sculley for his tftS3F filter code.

References

1. Gordon V. Cormack, A. Bratko, Batch and On-line Spam Filter Evaluation, In *CEAS2006: Proceedings of the 3rd Conference on Email and Anti-Spam* (2006).
2. Thiago S. Guzella, Walmir M. Caminhas, A Review of Machine Learning Approaches to Spam Filtering, *Expert Systems with Applications*. 36(7) (2009) 10206–10222.
3. Fabrizio Sebastiani, Machine Learning in Automated Text Categorization, *ACM Computing Surveys*. 34(1) (2002) 1–47.
4. Kian Ming Adam Chai, Hai Leong Chieu, Hwee Tou, Bayesian Online Classifiers for Text Classification and Filtering, In *SIGIR'02: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (2002), pp. 97–104.
5. J. Kivinen, A. Smola, R. Williamson, Online Learning with Kernels, *Advances in Neural Information Processing Systems*. 14 (2002) 785–793.
6. D. Sculley, Gabriel M. Wachman, Relaxed Online SVMs for Spam Filtering, In *SIGIR'07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (2007), pp. 415–422.
7. Gordon V. Cormack, University of Waterloo Participation in the TREC 2007 Spam Track, In *TREC2007: Notebook of the 16th Text REtrieval Conference* (National Institute of Standards and Technology, 2007).
8. Gordon V. Cormack, Email Spam Filtering: A Systematic Review, *Foundations and Trends in Information Retrieval*. 1(4) (2008) 335–455.
9. J. Goodman, W. Yin, Online Discriminative Spam Filter Training, In *CEAS2006: Proceedings of the 3rd Conference on Email and Anti-Spam* (2006).
10. Wuying Liu, Ting Wang, Multi-Field Learning for Email Spam Filtering, In *SIGIR'10: Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (2010), pp. 745–746.
11. Thomas G. Dietterich, Ensemble Methods in Machine Learning, In *MCS2000: Proceedings of the Multiple Classifier Systems* (2000), pp. 1–15.
12. H. Drucker, D. Wu, V. N. Vapnik, Support vector machines for spam categorization, *IEEE Transactions on Neural Networks*. 10(5) (1999) 1048–1054.
13. Zhihua Wei, Duoqian Miao, Jean-Hugues Chauchat, Rui Zhao, Wen Li, N-grams based Feature Selection and Text Representation for Chinese Text Classification, *International Journal of Computational Intelligence Systems*. 2(4) (2009) 365–374.
14. Justin Zobel, Alistair Moffat, Inverted Files for Text Search Engines, *ACM Computing Surveys*. 38(2) (2006) Article 6.
15. Gordon V. Cormack, TREC 2007 Spam Track Overview, In *TREC2007: Proceedings of the 16th Text REtrieval Conference* (National Institute of Standards and Technology, Special Publication 500-274, 2007).
16. Paul Graham, A Plan for Spam. (August 2002) http://www.paulgraham.com/spam.html.
17. Paul Graham, Better Bayesian Filtering. In the 2003 Spam Conference. (January 2003) http://www.paulgraham.com/better.html.
18. D. Sculley, Gabriel M. Wachman, Relaxed Online SVMs in the TREC Spam Filtering Track, In *TREC2007: Proceedings of the 16th Text REtrieval Conference* (National Institute of Standards and Technology, Special Publication 500-274, 2007).