PAPER

Intelligent Email Categorization Based on Textual Information and Metadata

Jihoon YANG[†], *Nonmember*, **Venkat CHALASANI**[††], *Nonmember*, *and* **Sung-Yong PARK**[†], *Nonmember*

SUMMARY A set of systematic experiments on intelligent email categorization has been conducted with different machine learning algorithms applied to different parts of data in order to achieve the most correct classification. The categorization is based on not only the body but also the header of an email message. The metadata (e.g. sender name, sender organization, etc.) provide additional information that can be exploited to improve the categorization capability. Results of experiments on real email data demonstrate the feasibility of our approach to find the best learning algorithm and the metadata to be used, which is a very significant contribution in email classification. It is also shown that categorization based only on the header information is comparable or superior to that based on all the information in a message for all the learning algorithms considered.

key words: *emails, categorization, classification, machine learning, metadata*

1. Introduction

With the proliferation of the Internet and numerous affordable gadgets (e.g. PDAs, cell phones), emails have become an indispensable medium for people to communicate with each other. People can send emails not only to desktop PCs or corporate machines but also to mobile devices (e.g. Research In Motion's Blackberry *), and thus they receive messages regardless of the time and place. This has caused a drastic increase in email correspondence and made people spend significant amount of time in reading their messages.

Unfortunately, as email communication becomes prevalent, all kinds of emails are generated. People have a tendency to make emails as their first choice when they need to talk to someone. A supervisor or leader of a group sends out a message to group members for meeting arrangement. The internal communications department of a company distributes an email message to all employees to remind the deadline of timecard submission. These depict the situations in which email communication is very efficient while traditional methods (e.g. phone calls) are time-consuming and expensive. Though emails brought us enormous convenience and fast delivery of message, it also caused us trouble of managing the huge influx of data everyday. It has become important to distinguish messages of interest from the huge amount of data we receive. For instance, a message from the boss asking for a document might be much more critical than a message from a friend suggesting lunch. To make matters worse, knowing the efficacy and ease of email communications, there exist a number of wicked people try to hoax innocent people with jokes or even viruses, and salespeople try to advertise their goods with unsolicited messages. Therefore, it is clearly of interest to design a system that automatically classifies (and, if necessary, filters) emails.

Against this background, we attempt to find the best method for automatic email classification (or categorization; both terms will be used interchangeably in the paper as they both appear in the literature). Simply, we can define a set of filtering rules for categorization. However, this approach is not complete in the sense that the rules are not perfect for a variety of cases and thus can not be fired or can be misfired in many cases. Machine learning is a general and very effective method for the task. A variety of machine learning algorithms in the literature have been used for email categorization task on different metadata (e.g. sender name, sender organization, sender domain, transmission time, subject line, and the like). However, no one approach has been shown to outperform others consistently, nor have been made any sound claims about the performance between learning algorithms and metadata. We therefore aim to solve this problem and figure out the best approach based on a set of systematic experiments for intelligent email categorization with different machine learning algorithms applied to different parts of data. Our domain of interest is email messages, however our approach can be applied to other types of text data as well (e.g. patents). An email can be simply categorized into spams and non-spams. Furthermore, it can be assorted into more detailed categories such as meetings, corporate announcements, and so on. As mentioned previously, additional information (e.g. sender) in addition to the text data in an email is considered for more precise classification. The RAINBOW text classification system [15] is adopted in our experiments. Some of the learning algorithms in RAINBOW are chosen and modified for our experimental studies.

Manuscript received April 1, 2002.
Manuscript revised October 12, 2002.

[†]Department of Computer Science, Sogang University; This work was supported by Korea Research Foundation Grant (KRF-2002-003-D00133) to Jihoon Yang.

[††]SRA International, Inc.

*http://www.blackberry.net

The rest of the paper is organized as follows: Section 2 introduces RAINBOW and describes the three learning algorithms we chose in RAINBOW. Section 3 explains the email corpus and features we used and presents the results of the experiments designed to evaluate the performance of different experimental settings. Section 4 describes previous work in text (including email) classification research. Section 5 concludes with a summary and discussion of some directions for future research.

2. Rainbow and Learning Algorithms

RAINBOW is a freely available program that performs statistical text classification written by Andrew McCallum and his group at Carnegie Mellon University [15]. It is based on the BOW library of C code. The library provides facilities for finding text files by recursively traversing directories, detecting document boundaries for files with multiple documents, tokenizing a text file (including N-grams), handling strings very efficiently, building sparse representation of documents and tokens, pruning vocabulary, setting word vector weights, smoothing word probabilities, scoring queries for retrieval or classification, writing and reading efficient data structures to and from the disk, performing data split into train and test sets, incorporating a variety of learning algorithms, classifying documents in both stand-alone or server mode, and so on [15].

RAINBOW operates in two steps: 1) read in documents, compute statistics, and write the statistics,"model", to the disk; and 2) perform classification using the model. A variety of machine learning algorithms are deployed in RAINBOW, among which the following three algorithms have been used in our work: TFIDF [10], Naïve Bayes [9], [17], and Support Vector Machines [7], [28], [29]. We explain each algorithm briefly. (Detailed descriptions can be found in the references.)

2.1 TFIDF

TFIDF classifier (TFIDF) [10] is similar to the Rocchio relevance feedback algorithm [22] except that it does not incorporate relevance feedback. The TFIDF word weights described in section 3.2. First, a prototype vector $\vec{c}$ is generated for every class $c \in \mathcal{C}$ where $\mathcal{C}$ is the set of all classes, by combining all feature vectors in documents d of the class

$$\vec{c} = \sum_{d \in c} \vec{d}$$

Then, the classification is to find a prototype vector that yields the largest cosine of a document d (which we want to classify) and the prototype vector itself

$$\arg\max_{c \in \mathcal{C}} cos(\vec{d}, \vec{c}) = \arg\max_{c \in \mathcal{C}} \frac{\vec{d} \cdot \vec{c}}{\| \vec{d} \| \cdot \| \vec{c} \|}$$

This is a very simple yet powerful algorithm, and numerous variants have been proposed in the literature. (See [10] for detailed descriptions on TFIDF classifier and similar approaches.)

2.2 Naïve Bayes

In Naïve Bayes classifier (NB), it is assumed that a term's occurrence is independent of the other terms. We want to find a class that produces the highest conditional probability given a document d:

$$\arg\max_{c \in \mathcal{C}} P(c|d)$$

By Bayes rule [9],

$$P(c|d) = \frac{P(d|c) \cdot P(c)}{P(d)}$$

It is clear that

$$P(c) = \frac{|c|}{\sum_{c' \in \mathcal{C}} |c'|}$$

and $P(d)$ can be ignored since it is common to all classes.

There are two ways to compute $P(d|c)$ based on the representation: either binary or term frequency-based. We show how to compute $P(d|c)$ for the latter. (See [16] for binary case.) Let N_{it} be the number of occurrences word w_t in document d_i, and V the vocabulary size. Then $P(d_i|c)$ is the multinomial distribution:

$$P(d_i|c) = P(|d_i|)|d_i|! \prod_{t=1}^{|V|} \frac{P(w_t|c)^{N_{it}}}{N_{it}!}$$

$P(|d_i|)|d_i|!$ is also common to all classes and thus can be dropped. Finally, the probability of word w_t in class c can be estimated from the training data:

$$P(w_t|c) = \frac{1 + \sum_{i=1}^{|\mathcal{D}|} N_{it} P(c|d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|\mathcal{D}|} N_{is} P(c|d_i)}$$

where $\mathcal{D}$ is the training data set.

2.3 Support Vector Machines

The Support Vector Machine (SVM) is an approach to learning a two-class pattern classification problems considering the geometric distribution of patterns [28], [29]. SVM attempts to find the hyperplane that separates patterns of two classes and maximizes the "margin", sum of the distances between the closest patterns and the hyperplane in each class. The SVM finds the solution using quadratic programming techniques. For linearly non-separable datasets, SVM maps the original data into a higher dimensional space using an appropriate kernel function which is chosen to ensure that the new data becomes linearly separable. SVM is different

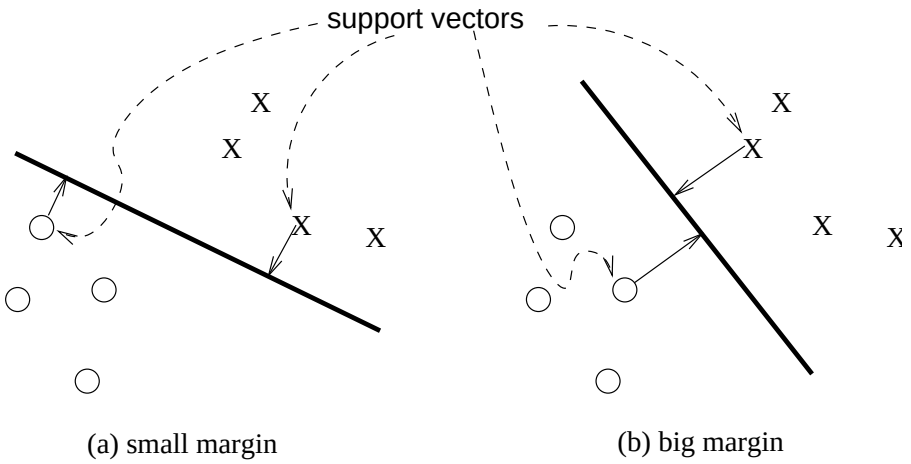


Fig. 1 SVM finds a hyperplane with bigger margin (b).

from other learning algorithms (including TFIDF, NB) in that it does not utilize *all* training patterns. Instead, it finds and uses only a subset of patterns that lie close to the decision boundary. These patterns are called the *support vectors*. Figure 1 depicts an example in two dimensional space where a hyperplane with a bigger margin (b) is preferred to the other with a smaller margin (a).

3. Experiments

We explain which categories and features have been considered for the emails collected for our experiments, and exhibit the results.

3.1 Email Corpus

A large number of real email messages have been collected and supplied by members in Advanced Technology Group of SRA International, Inc. in the following categories: *address* (i.e. contact points), *corporate announcement*, *finance* (e.g. stock info, market analysis, financial news), *meeting*, *purchase* (e.g. order status, membership enrollment), *spam*, *todo* (e.g. service requests, questions), and *travel* (e.g. itinerary, reservations, cancellations) categories. Though the total number of messages we received is very large, the total number of *usable* messages is much smaller. For instance, there were many identical messages broadcast to recipients. Also, some of the messages were too short and did not contain any body, and thus were not appropriate for our experiments. Among the categories described above, the following three are considered since a reasonable number of emails have been collected in the categories. 60% of the email corpus is used for training and the remaining 40% for testing which are shown in the parentheses respectively.

- *corporate announcement* (113, 76): all the company-wide announcements such as events, news, system outage notices, recruiting messages, call for papers, employee locations, virus warnings, holiday party info, company stock updates, memos, etc.
- *meeting* (435, 290): all kinds of meetings (e.g. research meeting, seminar, class, discussion, demo, talk, conference call, luncheon, birthday, special events) with different purposes (i.e. announcement, cancellation, change, reminder).
- *spam* (259, 171): all unsolicited messages including advertisements, coupons, etc.

There exist messages that belong to more than one categories. For instance, a message announcing a group meeting can belong to the *meeting* as well as the *corporate announcement* categories. For simplicity, we assume that each message belong to only one category. (e.g. We excluded *meeting* messages from *corporate announcement* even though they belong to the category as well.)

3.2 Representation

Classification of documents necessarily has to involve some analysis of the contents of a document. In the absence of a satisfactory solution to the natural language understanding problem, most current approaches to document retrieval (including RAINBOW) use a *bag of words* representation of documents [26]. Thus, A document is represented as a vector of weights for terms (or words) from a *vocabulary*. There are several possibilities for determining the weights: *binary values* can be assigned for each term to indicate its presence or absence in a document; or *term frequency* can be used to indicate the number of times that the term appears in a document; or *term frequency – inverse document frequency* can be used to measure the term frequency of a word in a document relative to the entire collection of documents [26]. A document can be processed using

Table 1 Features created for the body and the header of an email.

Description	*Name*
bag of words (body)	<word>
bag of words (subject)	<word>:Subject
number of ! in body	:NoExclamations
number of ! in subject	:SubjectNoExclamations
number of $ in body	:NoDollar
number of $ in subject	:SubjectNoDollar
number of upper case words in body	:NoUpper
number of upper case words in subject	:SubjectNoUpper
number of single character words in body	:NoSingle
number of single character words in subject	:SubjectNoSingle
number of phone numbers in body	:NoPhones
number of phone numbers in subject	:SubjectNoPhones
number of toll-free numbers in body	:NoTollFrees
number of toll-free numbers in subject	:SubjectNoTollFrees
number of email addresses in body	:NoEmails
number of email addresses in subject	:SubjectNoEmails
number of URLs in body	:NoURLs
number of URLs in subject	:SubjectNoURLs
number of time phrases in body	:NoTimes
number of time phrases in subject	:SubjectNoTimes
percentage of special characters in body	:SpecialPercent
percentage of special characters in subject	:SubjectSpecialPercent
sender name	<name>:SenderName
sender ID	<ID>:SenderID
sender domain name	<domain_name>:SenderDomainName
sender domain type	<domain_type>:SenderDomainType
recipient name	<name>:RecipientName
recipient ID	<ID>:RecipientID
recipient domain name	<domain_name>:RecipientDomainName
recipient domain type	<domain_type>:RecipientDomainType
CC name	<name>:CCName
CC ID	<ID>:CCID
CC domain name	<domain_name>:CCDomainName
CC domain type	<domain_type>:CCDomainType
BCC name	<name>:BCCName
BCC ID	<ID>:BCCID
BCC domain name	<domain_name>:BCCDomainName
BCC domain type	<domain_type>:BCCDomainType
number of recipients	:NoRecipients
day when the message was sent	<day>:Day
hour when the message was sent	:Hour
is this a replied message?	:Replied
is this a forwarded message?	:Forwarded

stopping and *stemming* procedures [11], [26] to obtain the bag of words. The stopping procedure eliminates all commonly used terms (e.g. *a*, *the*, *this*, *that*) and the stemming procedure [18] produces a list of representative (root) terms (e.g. *play* for *plays*, *played*, *playing*).

Let d be a document. Let w_i be the ith word in d. The *term frequency* of w_i, $TF(w_i, d)$, is the number of times w_i occurs in d. The *document frequency* of w_i, $DF(w_i)$, is the number of documents in which w_i occurs at least once. The *inverse document frequency* of w_i, $IDF(w_i)$, is defined as $IDF(w_i) = log(\frac{|D|}{DF(w_i)})$, where $|D|$ is the total number of documents. Then, the *term frequency – inverse document frequency* of w_i is defined as $TF(w_i, d) \cdot IDF(w_i)$ [26]. Either the binary values, term frequency or the term frequency – inverse document frequency is used in the classifiers chosen in this paper.

Emails have additional information in the header in addition to the text message (i.e. email body). For instance, an email header includes the *sender*, *receivers*, *subject*, and the like. A set of additional features can be derived from the header and be used with the body. These additional features have a potential for more accurate classification. For example, we can define additional features of *sender name*, *sender ID*, *sender domain name*, and *sender domain type*. If we had a sender "Jihoon Yang <jihoon_yang@sra.com>", we can define additional features of "JihoonYang:SenderName", "jihoon_yang:SenderID", "sra:SenderDomainName", and "com:SenderDomainType". (Note that we define header features in the form of "<feature_value> :<feature_name>" in order to construct unique features that do not appear in the body.)

Another set of features can be defined for both

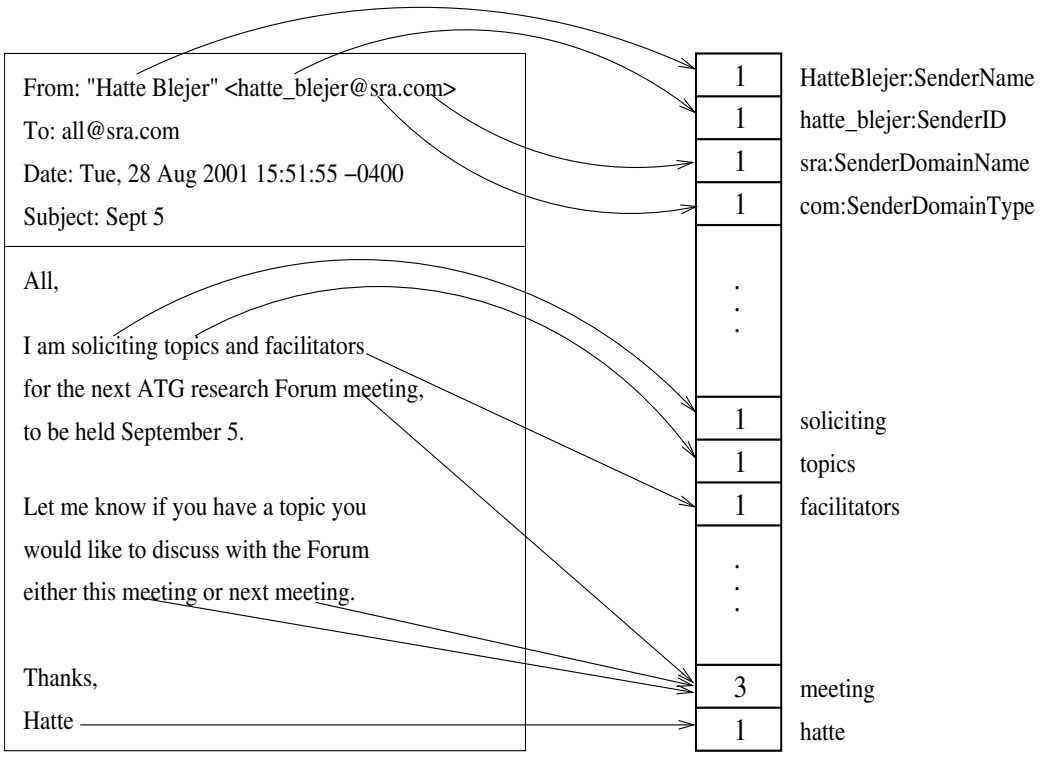


Fig. 2 Bag of words representation of a sample email.

email body and the header, especially in the subject line. For instance, we can count how many $ characters are included in the message, which might insinuate the message is a spam. We can also count how many special characters appear in the message. We define several such features and represent them as described above. For instance, if we had 10 and 5 $'s appearing in the text and the subject line, we can store those values for two features ":NoDollars" and ":SubjectNoDollar", respectively. While some of the features can be obtained by simple syntactic analysis, some other features require application of information extraction techniques (e.g. emails, phone numbers).

Table 1 summarizes the features we defined and used. Since we incorporate all the features in bag of word representation, all the features are either binary (including boolean), integer, or real values. Figure 2 shows the term frequency-based representation for a sample email message where stopping procedure is applied but stemming procedure is not, with some of the additional features.

3.3 Experimental Setting and Results

First, the three learning algorithms (TFIDF, NB, SVM) are trained in RAINBOW. (The model generation part in RAINBOW is modified to include all the features in Table 1.) After training is done, the "scores" (i.e. similarity or probability) of the training patterns with respect to classes are computed in RAINBOW. Then the *thresholds* are computed by scanning the sorted list of scores once so that the trained classifier yields the maximum classification accuracy. In other words, a threshold that is less than most of the emails in the class and greater than most of the emails in other classes is determined. These thresholds computed for each class are used in testing. This is an *independent* training in contrast to the *winner-take-all* strategy originally included in RAINBOW. Independent training is necessary since a message can belong to more than one classes or it may not belong to any. For simplicity in experiments, we limited the classification of each message to only one category as explained in Section 3.1.

There are couple of aspects we need to clarify in SVM: First, SVM is basically a classifier for two categories. In order to handle more than two categories, we chose *one-against-all* approach. In other words, each class was compared with all the other classes in two-class categorization. Second, there can be a variety of kernel functions adoptable in SVM. The linear function was chosen in our experiments.

One of our goals was to undertake comparative studies between the three learning algorithms. In addition, we intended to figure out how different parts of email messages make differences in classification. For instance, classification based on subject lines might produce comparable results to classification with all data in the message. For this purpose, we compared the per-

formance of classifiers trained with different parts of the message. We considered the following five combinations: all (A), header (H), subject line (S), body and subject line (BS), and header without subject line (HS). We also considered both cases when the stemming procedure was applied or not. Table 2 shows the number of features generated in each case. (Note that the stopping procedure was applied in all cases.) Furthermore, we compared the performance of the algorithms with all the features from different parts of an email (i.e. A, H, S, BS, HS) to that with only 50 features of the highest information gain [6].

Table 2 Number of features in the combinations of different parts in the email.

	A	BS	H	HS	S
without stemming	14409	11795	3077	2061	1002
with stemming	11016	8402	2949	2061	874

For each experimental setting, we ran each algorithm ten times with different combinations of training and test messages, but maintained the same sizes as mentioned in section 3.1. Table 3 exhibits our experimental results in terms of *accuracy*, *precision*, and *recall*. Consider the following four different cases of relationships between the desired class and the predicted class (in terms of the number of messages):

	in predicted	*not in predicted*
in desired	a	c
not in desired	b	d

Accuracy concerns the percentage of correct classification:

$$Accuracy = \frac{a+d}{a+b+c+d}$$

$$= \frac{\#\ of\ messages\ correctly\ categorized}{total\ \#\ of\ messages}$$

Precision and recall are slightly different and are defined as:

$$Precision = \frac{a}{a+b}$$

$$= \frac{\#\ of\ messages\ desired\ to\ be\ in\ a\ category}{total\ \#\ of\ messages\ predicted}$$

$$Recall = \frac{a}{a+c}$$

$$= \frac{\#\ of\ messages\ predicted\ to\ be\ in\ a\ category}{total\ \#\ of\ messages\ desired}$$

Based on above formulas, accuracy, precision, and recall are computed for *overall* categories. Note that test cases postfixed by MI and T are the ones with mutual information-based feature selection and stemming, respectively. The entries in the table correspond to means and standard deviations and are shown in the form *mean± standard deviation*. The best accuracy, precision, and recall among three algorithms and five (or four with stemming) feature sets are shown in bold face. Note that there is no row for HS when stemming procedure is applied since the procedure does not make any difference for headers without subject lines.

We observed the following from Table 3.

1. *Poor performance of* TFIDF *classifier*: TFIDF performed poorer than NB and SVM in all cases. This is because TFIDF yields very poor *accuracy* and *recall* despite comparable *precision*.
2. *Good performance of* NB *without feature subset selection*: NB performed reasonably consistent and good in different experimental settings. However, for incomplete data (i.e. H, HS, and S), it worked better without feature subset selection by mutual information. It might be surmised that the more features considered (i.e. without feature subset selection by mutual information and with all features in both the body and the header of an email) the better NB performs.
3. *Better accuracy and recall of* SVM *with feature subset selection*: SVM always performed significantly better in terms of accuracy and recall with small feature subsets than with all the features. This show that SVM finds better decision boundary defined by the feature subset with high mutual information than the entire feature set.
4. *No effect of stemming*: Stemming did not make any significant difference for all algorithms in performance, though it decreased the size of the feature set.
5. *Good performance with headers*: In NB, the performance with header information (H) was almost as good as that with all features (A) or body and subject line features (BS). In SVM, H always outperformed other cases. In TFIDF, subject line was good enough for high precision. This means we can get good performance by considering only some parts (e.g. header, subject line) of the message.
6. *Superiority of* SVM: The best result was obtained when SVM was applied to the header with feature subset selection (with or without stemming). While NB performed reasonably well when all features (A) or body and subject line features (BS) were used, SVM worked better using the header features (H) only. Though may people reported similar results of SVM's good performance, this is yet another significant result that we can rely on and deploy in practical email, and other classification tasks.

Table 3 Perforamnce of learning algorithms in different experiments.

	TFIDF			NB			SVM		
features	*accuracy*	*precision*	*recall*	*accuracy*	*precision*	*recall*	*accuracy*	*precision*	*recall*
A	77.3±0.3	88.2±0.6	71.2±0.6	94.0±0.2	95.3±0.3	92.1±0.3	81.1±0.5	94.2±0.8	83.2±0.5
BS	71.8±0.3	85.8±0.6	66.1±0.3	**94.5**±0.2	94.2±0.4	**94.3**±0.2	75.3±0.6	**97.4**±0.3	69.8±0.8
H	73.3±0.5	84.8±0.7	69.4±0.9	93.7±0.3	93.1±0.3	93.3±0.3	79.0±0.7	88.6±2.0	79.1±1.1
HS	76.2±0.4	85.4±0.8	73.6±0.5	89.5±0.4	88.9±0.6	90.5±0.3	84.1±2.2	92.2±1.9	82.7±2.2
S	78.7±0.7	90.0±0.4	75.0±0.7	86.6±0.4	89.4±0.4	83.4±0.5	86.7±0.4	96.4±0.3	81.3±0.4
A-MI	85.7±0.4	95.0±0.9	81.2±0.7	93.8±0.4	93.8±0.4	93.4±0.5	94.9±0.5	96.6±0.5	93.1±0.9
BS-MI	79.4±0.4	94.3±0.5	74.4±0.7	90.4±0.3	90.5±0.3	89.9±0.4	89.9±0.4	90.5±0.5	87.9±0.7
H-MI	79.1±0.4	89.4±0.7	77.6±0.5	84.6±0.9	85.3±0.8	85.3±0.9	96.6±0.2	**97.1**±0.4	95.5±0.4
HS-MI	83.4±0.4	91.9±0.4	83.8±0.5	66.3±4.8	68.5±5.5	61.3±5.8	**96.7**±0.2	96.8±0.3	**96.5**±0.2
S-MI	71.0±0.5	96.9±0.6	66.0±0.7	75.0±0.4	82.6±0.3	70.8±0.5	91.7±0.5	93.4±0.3	88.3±0.6
A-T	76.0±0.4	86.0±0.5	69.7±0.5	**94.6**±0.3	95.1±0.2	92.7±0.4	80.0±0.9	95.1±1.0	81.1±1.7
BS-T	68.5±0.3	83.7±0.6	63.0±0.4	94.3±0.3	93.2±0.4	**94.5**±0.4	76.9±0.9	**98.0**±0.2	70.8±1.1
H-T	73.0±0.7	83.5±1.2	68.3±0.8	94.0±0.4	93.3±0.5	93.9±0.4	78.4±0.7	89.7±1.3	79.6±0.7
S-T	79.0±0.5	89.0±0.6	75.7±0.6	87.7±0.2	89.8±0.3	84.7±0.4	85.1±0.5	95.8±0.5	79.4±0.6
A-T-MI	85.3±0.4	94.4±0.8	81.3±0.6	93.2±0.3	92.7±0.4	92.7±0.4	94.6±0.4	96.3±0.3	93.3±0.6
BS-T-MI	78.7±0.5	92.6±1.0	74.3±0.6	90.7±0.3	90.6±0.3	90.2±0.4	90.6±0.4	90.7±0.6	88.8±0.6
H-T-MI	78.4±0.3	89.6±0.5	77.0±0.7	83.1±0.5	84.1±0.5	83.1±0.8	**96.4**±0.3	**96.4**±0.4	**95.6**±0.3
S-T-MI	72.0±0.6	96.0±0.5	67.3±0.8	76.1±0.4	82.8±0.3	72.0±0.7	91.5±0.3	93.4±0.4	87.7±0.7

4. Related Work

A large number of approaches to text classification have been proposed in the literature. Basically, text is a kind of data and thus any machine learning algorithms for pattern classification (e.g. decision tree [19], [20], Bayesian classifier [9], artificial neural networks [23], [24], [30], etc.) can be used. (Of course, there are biases and limitations in each algorithm which make it infeasible to apply the algorithm due to the sparseness and the large scale of text data. See [17] for an introduction to machine learning algorithms.) Yang compared the performance of different learning algorithms in text classification [31]. Numerous learning algorithms have been used for classification (or clustering or retrieval) of a variety of text data (e.g. Web pages, TV closed captions[4], scientific papers [2], patents [12]). Here we review some of the approaches to email classification.

An email filtering system, IFILE, is developed and claimed to be accurate and fast [21]. This is very similar to RAINBOW using Naïve classifier. IFILE selects features based on *age* which is exactly the same as occurrence count in RAINBOW. MAILCAT is proposed as an email organizing agent [27]. It predicts the most likely three folders into which an email should be categorized. Simple TFIDF is used in MAILCAT. [1] compared the performance between NB and TiMBL (a memory-based classifier) in spam filtering. Mutual information is used for feature subset selection. They claim both algorithms outperform traditional keyword-based filtering. An approach to filtering junk emails by [25] is very interesting and most similar to our approach. They defined features from the text as well as 35 phrases (e.g. "only $", "FREE!", etc.) and 20 domain specific non-textual features (e.g. domain type of the sender). The Naïve Bayes is used considering different costs of misclassification. Feature subset selection is also practiced based on mutual information. Another similar approach of interest has been proposed in [3]. They also chose three algorithms: SVM, TFIDF, and Unigram (which is very similar to Naïve Bayes). They compared the performance of learning algorithms with entire message or with header only, and claim that there is no significant difference between using all content or only the header. It is noticeable that all the three algorithms were comparable in their experiments. Similar to Brutlag's approach, [8] proposed an approach slightly different to ours assorting features into three types: *structured* (e.g. sender domain), *textual* (i.e. words), and *handcrafted* (e.g. number of dollar signs). Their experiments exhibit that careful selection among these features enhance performance. Yet another similar interesting system is EMAILVALET [14] which predicts whether a message should be forwarded to a user's wireless device. EMAILVALET makes use of five learning algorithms (NB, TFIDF, probabilistic TFIDF [10], Ripper [5], Winnow [13]). It tokenizes both the body and email addresses in the header. While we define additional features considering the structure of an email, they simply tokenize address strings (e.g. sender name, sender ID, etc.). They also performed comparative studies for different combinations of body, date, and time, and conclude that including the body does not harm learning.

Though such a number of approaches have been proposed, there does not exist any single approach that exhibits solid results of comparative studies that can be used generally in real-world applications. On the contrary, we exhibited all preprocessing steps in detail and derived a sound and practical conclusion that would help us greatly in text (including email) classification tasks.

5. Summary and Discussion

A set of systematic experiments on intelligent email categorization has been conducted with different machine learning algorithms applied to different parts of data in order to achieve the most correct classification. Three machine learning algorithms (TFIDF, NB, and SVM) were used and their performance was compared. We also studied how different parts of email structure affect the classification capability. Experimental results demonstrate that NB and SVM outperform TFIDF, and NB yields better performance without feature subset selection (especially when small number of parts of emails were used) and SVM works well with feature subsets based on mutual information. It was also found, at least with our current corpus, that classification with the header was as accurate as that with entire message, with even fewer features. In particular, the best result was obtained when SVM was applied to the header with feature subset selection (with or without stemming), which can be a valuable guideline in solving real-world text classification tasks.

Some avenues for future work include:

- *Maintenance of quality data*: First, we can collect more data. Our experiments were with less than 1,500 messages in three categories, and more than half of the messages belong to *spam* class. In order to get more accurate, credible statistics, we need to keep collecting messages. Also, the three categories can be extended to include interesting domains. Current emails belong to only one category, but we can extend this to include messages belonging to multiple categories since there exist many such emails in the real world. Our current system is already capable of handling this multiple-class membership problem, and we plan to experiment with this kind of data. Furthermore, categories can be organized into a concept hierarchy, which can be used in classification (e.g. Yahoo). In addition, we can eliminate junks from messages which are usually unnecessary in classification and can possibly cause confusion and misclassification. For instance, we can remove signatures in messages. Of course, this kind of information can be useful in detecting the *authors* of the messages, but they are usually not useful in categorization.
- *Extensive experimental study*: There can be different parameter settings in each algorithm. Our experimental studies are based on the default parameter settings. We can perform extensive experiments with different settings available in RAINBOW. Furthermore, we can try different techniques which are not included in RAINBOW (e.g. different methods for feature subset selection). We can also build ensemble of classifiers (e.g. voting, boosting, bagging) and compare their performance. In addition, a fair comparison between our approach and other approaches (e.g. in Section 4) will be of interest. Without accurate information on the data and parameters of their approaches, it is not feasible to accomplish a fair comparison. We might choose a few approaches, duplicate them, and then make a comparison to verify and fortify our claim.
- *Combination with NLP-based information extraction*: Categorization can be bolstered with extracted information by natural language processing. For example, if we extracted date, time, place, and duration of a meeting from a message, we know the message is about a meeting even without going through the classification routine. This additional information can boost the performance of text-based classification. Similar to [25], additional domain-dependent phrases can be defined for each category and extracted as well. RAINBOW can be mixed with a rule-based system using such additional features.
- *Consideration of attachments*: Attachments, if exist, can be exploited. Attachments can simply be processed with email messages in classification or can be classified separately (with respect to the original categories or new categories). Moreover, attachments can be classified with respect to a different set of categories. For instance, we can define categories with the types of attachments (e.g. Word document, presentation slides, spreadsheets, postscripts, etc.). The information about the number of attachments and their types can be used in classification.
- *Extension to prioritization*: A simple approach to prioritization would be doing it based on the categories. This can be extended using the extracted information. That is, we can prioritize a message based on the information we extract. For instance, if we extracted an activity or an event with its time and place of occurrence, we can determine the priority of the message. The additional information we defined from the header (e.g. sender) can be also exploited to adjust the priority. Also, machine learning can be applied to this problem.
- *Action learning*: There can be typical actions associated with each category. For instance, a person might forward meeting messages to someone else, in general. This kind of information can be used for learning people's "actions" (or behavior).

Acknowledgments

The authors wish to thank members of the wireless group (Scott Bennett, Andrew Graham, Raj Lingam, Brian Michl, Steve Pichney, and Shyam Sundaram) at SRA International, Inc. for their comments and help

on this paper. In particular, we are grateful to Andrew Graham for his mailtool which made it possible for us to generate large number of labeled examples so easily. We also thank people in Advanced Technology Group of SRA International, Inc. for submitting email messages for our work.

References

[1] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. Spyropoulos, and P. Stamatopoulos. Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. In *Machine Learning and Textual Information Access Workshop, The Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 1–13, Lyon, France, September 2000.

[2] K. Bollacker, S. Lawrence, and L. Giles. Discovering relevant scientific literature on the web. *IEEE Intelligent Systems*, pages 42–47, March/April 2000.

[3] J. Brutlag and C. Meek. Challenges of the email domain for text classification. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 103–110, 2000.

[4] W. Chuang and J. Yang. A fast algorithm for hierarchical text classification. In *Proceedings of the Second International Conference on Data Warehousing and Knowledge Discovery*, pages 409–418, 2000.

[5] W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann, 1995.

[6] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley, 1991.

[7] N. Cristianini and J. Shawe-Taylor. *Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, UK, 2000.

[8] Y. Diao, H. Lu, and D. Wu. A comparative study of classification based personal e-mail filtering. In *Proceedings of the Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 408–419, Kyoto, Japan, 2000.

[9] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[10] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical Report CMU-CS-96-118, Carnegie Mellon University, Pittsburgh, PA, 1996.

[11] R. Korfhage. *Information Storage and Retrieval*. Wiley, New York, 1997.

[12] L. Larkey. A patent search and classification system. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, pages 79–87. ACM Press, 1999.

[13] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1987.

[14] S. Macskassy, A. Dayanik, and H. Hirsh. Emailvalet: Learning user preferences for wireless email. In *Learning About Users and Machine Learning for Information Filtering Workshop, International Joint Conference on Artificial Intelligence*, 1999.

[15] A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/∼mccallum/bow, 1996.

[16] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classificaiton. In *Learning for Text Categorization Workshop, National Conference on Artificial Intelligence*, 1998.

[17] T. Mitchell. *Machine Learning*. McGraw Hill, New York, 1997.

[18] M Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[19] R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[20] R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[21] J. Rennie. ifile: An application of machine learning to e-mail filtering. In *Text Mining Workshop, The Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000. IFILE is available at http://www.ai.mit.edu/∼jrennie/ifile.

[22] J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 14, pages 313–323. Prentice-Hall, Inc., 1971.

[23] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[24] D. Rumelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations into the Microstructure of Cognition*, volume 1 (Foundations). MIT Press, Cambridge, Massachusetts, 1986.

[25] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization Workshop, National Conference on Artificial Intelligence*, 1998.

[26] G. Salton. *Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, Massachusetts, 1989.

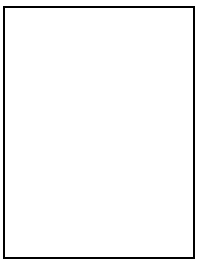
[27] R. Segal and J. Kephart. Mailcat: An intelligent assistant for organizing e-mail. In *Proceedings of the Third International Conference on Autonomous Agents*, 1999.

[28] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.

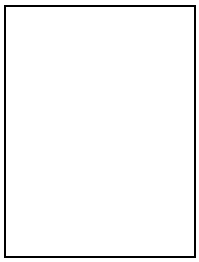
[29] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[30] J. Yang, R. Parekh, and V. Honavar. DistAI: An inter-pattern distance-based constructive learning algorithm. *Intelligent Data Analysis*, 3:55–73, 1999.

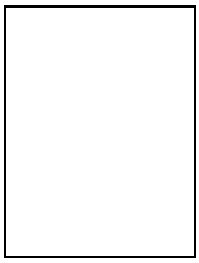
[31] Y. Yang. A re-examination of text categorization methods. In *Proceedings of the 22nd ACM SIGIR Conference*, pages 42–49, 1999.



Jihoon Yang is Assistant Professor of Computer Science at Sogang University. His research interests include machine learning, data mining and knowledge discovery, artificial intelligence, and bioinformatics. Dr. Yang holds a B.S. in Computer Science from Sogang University, and M.S. and Ph.D. degrees in Computer Science from Iowa State University. Contact him at Department of Computer Science, Sogang University, 1 Shinsoo-Dong, Mapo-Ku, Seoul 121-742, Korea; jhyang@ccs.sogang.ac.kr.



Venkat Chalasani is a researcher at SRA International where he develops data mining software. He received his Ph.D. in systems engineering from University of Virginia in 2000. His research interests include pattern clustering, classification and visualization. Contact him at SRA International, 4300 Fair Lakes Ct., Fairfax, VA 22033, U.S.A.; Venkat_Chalasani@sra.com.



Sung-Yong Park is Assistant Professor of Computer Science at Sogang University. He is interested in high performance cluster computing and system, Grid computing, middleware technologies, and operating systems. Dr. Park holds a B.S. in Computer Science from Sogang University, and M.S. and Ph.D. degrees in Computer and Information Science from Syracuse University. Contact him at Department of Computer Science, Sogang University, 1 Shinsoo-Dong, Mapo-Ku, Seoul 121-742, Korea; parksy@ccs.sogang.ac.kr.