# Email Categorization Using Fast Machine Learning Algorithms

Jihoon Yang* and Sung-Yong Park

Department of Computer Science, Sogang University
1 Shinsoo-Dong, Mapo-Ku, Seoul 121-742, Korea
{jhyang, parksy}@ccs.sogang.ac.kr

**Abstract.** An approach to intelligent email categorization has been proposed using fast machine learning algorithms. The categorization is based on not only the body but also the header of an email message. The metadata (e.g. sender name, organization, etc.) provide additional information that can be exploited and improve the categorization capability. Results of experiments on real email data demonstrate the feasibility of our approach. In particular, it is shown that categorization based only on the header information is comparable or superior to that based on all the information in a message.

## 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 nowadays. People can send emails not only to the desktop PCs or corporate machines but also to the mobile devices, and thus they receive messages regardless of the time and place. This has caused a drastic increase of 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 got 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 messages, 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 a document might be much more critical than a message from a friend suggesting a lunch. To make matters worse, knowing the efficacy and ease of email communications, there exist a number of wicked people trying

* This research was supported by the Sogang University Research Grants in 2002.

S. Lange, K. Satoh, and C.H. Smith (Eds.): DS 2002, LNCS 2534, pp. 316–323, 2002.

to hoax innocent people with jokes or even viruses, and salespeople trying to advertise their goods with unsolicited messages. Therefore, it is clearly of interest to design a system that automatically classifies emails.

Against this background, we propose an approach to automatic text classification (or categorization; both terms will be used interchangeably in the paper) using machine learning algorithms. We are interested in *fast* learning algorithms to deal with large amounts of data swiftly. 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 [1] is adopted in our experiments. Among the learning algorithms in RAINBOW, two fast algorithms are chosen and modified for our experimental studies.

## 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 [1]. 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 two algorithms have been used in our work: TFIDF [2] and Naïve Bayes [3,4]. These algorithms are chosen considering their fast learning speed. We explain each algorithm briefly. (Detailed descriptions can be found in the references.)

### 2.1 TFIDF

TFIDF classifier (TFIDF) is similar to the Rocchio relevance feedback algorithm [5] and TFIDF word weights described in section 3.2. First, a prototype vector $\boldsymbol{c}$ 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

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

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

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

This is a very simple yet powerful algorithm, and numerous variants have been proposed in the literature. (See [2] 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 gives the highest conditional probability given a document $d$:

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

By Bayes rule [3],

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

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 [6] 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.

## 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

Email messages have been collected and manually categorized. Among the various categories we defined, *corporate announcement*, *meeting*, and *spam* categories were considered in our experiments. 189, 725, and 430 email messages were collected for such categories respectively, among which 60% was used for training and the remaining 40% was used for testing. 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 (though can be unrealistic), 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 [7]. 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 [7]. A document can be processed using *stopping* and *stemming* procedures [7,8] to obtain the bag of words. The stopping procedure eliminates all commonly used terms (e.g. *a*, *the*, *this*, *that*) and the stemming procedure [9] produces a list of representative (root) terms (e.g. *play* for *plays*, *played*, *playing*).

Let $d$ be a document. Let $w_i$ be the $i$th 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 $TF(w_i, d) \cdot IDF(w_i)$ [7]. 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 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 mentioned 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. email addresses, phone numbers). Overall, 43 features have been defined and used in our work.

### 3.3 Experimental Results

First, the learning algorithms (TFIDF and NB) 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.

One of our goals was to undertake comparative studies between the two learning algorithms. In addition, we intended to figure out how different parts of email messages make difference 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 performance 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. Furthermore, we compared the performance of the algorithms with all the features from different parts of an email (i.e. $A$, $H$, $S$, $BS$, $HS$) with only 50 features that had the highest information gain [10].

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 (i.e. ten-fold cross-validation). Table 2 exhibits our experimental results. 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 two 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 a header without the subject line.

We observed the following from Table 2.

1. *Better performance of* NB *than* TFIDF: NB outperformed TFIDF in almost all cases. This is because TFIDF generally yielded very poor *accuracy* and *recall* despite comparable *precision*.
2. *Better performance of* TFIDF *with feature subset selection*: TFIDF produced better performance with feature subset selection by mutual information in all cases except $S$. In case of $S$, the precision with feature subset selection was higher than that without feature selection, but the accuracy and recall were lower. We surmise that the reduced feature set was good to determine

**Table 1.** Performance of learning algorithms in different experiments.

| | TFIDF | | | NB | | |
|---|---|---|---|---|---|---|
| *features* | *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 |
| *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 |
| *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 |
| *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 |
| *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 |
| *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 |
| *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 |
| *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 |
| *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 |
| *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 |
| *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 |
| *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 |
| *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 |
| *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 |
| *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 |
| *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 |
| *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 |
| *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 |

the categories precisely for specific emails while it did not include enough terms to cover all the messages in each category.

3. *Better 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.
4. *No effect of stemming*: Stemming did not make a significant difference for all algorithms in performance, though it decreased the size of the feature set.
5. *Good performance with headers*: In both algorithms, the performance with *H* was comparable to that with *A* or *BS*. In particular, TFIDF produced high precision (but low recall and accuracy) with only the subject line as explained above. This means we can get reasonable performance by considering only the header information (or even only the subject line) instead of the entire email message.

## 4 Summary and Discussion

An approach to intelligent email categorization has been proposed in this paper in order to cope with the immense influx of information these days. Two machine learning algorithms (TFIDF and NB) 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 outperforms

TFIDF and yields better performance without feature subset selection (especially when small number of parts of an email were used), and TFIDF 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 less number of features.

Some avenues for future work include:

- *Experiments with additional algorithms*: A number of machine learning algorithms have been proposed and compared in the literature [4,11,12]. For instance, the support vector machines [11], though slow, were claimed to be powerful and have been applied to text classification [13,12]. Experimental studies of SVM and its comparison with NB and TFIDF will give us additional knowledge on email classification.
- *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. 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.
- *Combination with NLP-based information extraction*: Categorization can be bolstered with extracted information by natural language processing. For example, if we extract 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 [14], 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 extract 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 knowledge can be used for learning people's "actions" (or behavior).

## References


1. McCallum, A.: Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/∼mccallum/bow (1996)
2. Joachims, T.: A probabilistic analysis of the roccihio algorithm with tfidf for text categorization. Technical Report CMU-CS-96-118, Carnegie Mellon University, Pittsburgh, PA (1996)
3. Duda, R., Hart, P.: Pattern Classification and Scene Analysis. Wiley, New York (1973)
4. Mitchell, T.: Machine Learning. McGraw Hill, New York (1997)
5. Rocchio, J.: Relevance feedback in information retrieval. In: The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice-Hall, Inc. (1971) 313–323
6. McCallum, A., Nigam, K.: A comparison of event models for naive bayes text classificaiton. In: Learning for Text Categorization Workshop, National Conference on Artificial Intelligence. (1998)
7. Salton, G.: Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley, Reading, Massachusetts (1989)
8. Korfhage, R.: Information Storage and Retrieval. Wiley, New York (1997)
9. Porter, M.: An algorithm for suffix stripping. Program **14** (1980) 130–137
10. Cover, T., Thomas, J.: Elements of Information Theory. John Wiley (1991)
11. Vapnik, V.: The Nature of Statistical Learning Theory. Springer Verlag (1995)
12. Yang, Y.: A re-examination of text categorization methods. In: Proceedings of the 22nd ACM SIGIR Conference. (1999) 42–49
13. Brutlag, J., Meek, C.: Challenges of the email domain for text classification. In: Proceedings of the Seventeenth International Conference on Machine Learning. (2000) 103–110
14. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A bayesian approach to filtering junk e-mail. In: Learning for Text Categorization Workshop, National Conference on Artificial Intelligence. (1998)