Text Classification Using Stochastic Keyword Generation

Cong Li I-CONGL@MICROSOFT.COM
Ji-Rong Wen JRWEN@MICROSOFT.COM
Hang Li HANGLI@MICROSOFT.COM
Microsoft Research Asia, 5F Sigma Center, No.49 Zhichun Road, Haidian, Beijing, China, 100080

Abstract

This paper considers improving the performance of text classification, when *summaries* of the texts, as well as the texts themselves, are available *during learning*. Summaries can be more accurately classified than texts, so the question is how to effectively use the summaries in learning. This paper proposes a new method for addressing the problem, using a technique referred to as 'stochastic keyword generation' (SKG). In the proposed method, the SKG model is trained using the texts and their associated summaries. In classification, a text is first mapped, with SKG, into a vector of probability values, each of which corresponds to a keyword. Text classification is then conducted on the mapped vector. This method has been applied to email classification for an automated help desk. Experimental results indicate that the proposed method based on SKG *significantly* outperforms other methods.

1. Introduction

We consider the problem of improving the performance of text classification, when other types of data are available during *learning*. Specifically, we consider the case in which the classified texts have *summaries*. We assume here that summaries can be more accurately classified than texts. The question is how we combine the use of classified texts and the use of summaries in learning. (Note that we assume that summaries are not available in classification.)

An example of this kind of text classification is the email classification at a help desk. For example, the help desk at Microsoft receives email messages from customers describing software errors. The goal is to automatically classify the messages into a number of predefined categories. The email messages are difficult to classify because customers often use different terms to describe the same problem. However, at the help desk, summaries to the messages made by engineers are available in addition to the manually classified email messages. The summaries are often more indicative to the categories.

We address the problem in this paper and propose a new method based on a technique which we refer to as 'stochastic keyword generation' (SKG for short). To the best of our knowledge, this problem has not been studied previously.

In SKG, we employ a probabilistic model to represent the relationship between the words in different texts and the keywords in the summaries of those texts. During the learning phase, we create an SKG model using texts and their summaries as training data. For a newly given text, we generate a list of keywords from the text using the SKG model. Each of the generated keywords has a probability value.

In text classification based on SKG, we first use texts and their summaries to train an SKG model. We then employ the obtained SKG model to map each of the classified texts into a probability vector. Next we use the obtained vectors and the corresponding classes to construct a classifier.

Our method maps each training example into a probability vector before constructing a classifier. This can be viewed as a sort of preprocessing, different from the traditional feature selection and feature transformation.

We conducted experiments on email classification at a help desk and found that our method based on SKG *significantly* outperforms other methods.

The approach presented in this paper may be extended to other classification problems in similar settings, by generalizing stochastic keyword generation into stochastic feature generation.

The rest of the paper is organized as follows. In Section 2 we describe related work and in Section 3 we describe our method of text classification using SKG. Next, we present our experimental results in Section 4. Finally, we make concluding remarks in Section 5.

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

2. Related Work

Text classification (or text categorization) is the process of automatically assigning texts into a number of predefined categories. Many methods for text classification based on supervised learning have been proposed. They include methods using regression models (e.g., Fuhr et al. 1991), naive Bayesian classifiers (e.g., Lewis & Ringuette 1994), nearest neighbor classifiers (e.g., Yang 1994), neural networks (e.g., Wiener et al. 1995), support vector machines (e.g., Joachims 1998), boosting (e.g., Schapire & Singer 2000), symbolic rules (e.g., Li & Yamanishi 2002), and the perceptron algorithm with margins (e.g., Li et al. 2002). Among these methods, the so-called margin classifiers, including support vector machines, boosting, and perceptrons, are reported to perform the best for many data sets. All of the methods above use only classified texts in training.

Feature selection and transformation are important preprocessing steps for text classification (Liu & Motoda 1998). Feature selection is the process of removing irrelevant features in order to enhance the efficiency of training and classification. The commonly used measures for feature selection include document frequency, information gain, mutual information, χ^2-test score, and term strength (Yang & Pederson 1997). Feature transformation is the process of mapping examples from the original feature space to a higher dimensional space in order to improve the classification ability of a classifier. For instance, the back-propagation algorithm (Rumelhart et al. 1986) automatically maps examples to a higher dimensional space represented by the hidden layer in a multilayer neural network. The support vector machine learning algorithm (Vapnik 1995) also attempts to use a kernel function to map examples from the original feature space to a higher dimensional space. Feature selection and feature transformation actually automatically construct new feature spaces.

Many problems in machine translation, information retrieval and text summarization can be formalized as one, based on a mapping between two spaces. The central issue of statistical machine translation is to construct a probabilistic model between the spaces of two languages. For example, Brown et al. (1993) proposed a series of statistical translation models for sentence translation. The essence of the models is to perform probabilistic mapping between the linguistic data in two languages. In information retrieval, many statistical methods (e.g., Berger & Lafferty 1999) have been proposed for effectively finding the mapping relationship between terms in the space of user queries and terms in the space of documents. This is because a major difficulty for document retrieval is that in many cases, terms in user queries and terms in documents do not match very well. In statistics-based summarization, for example, Knight and Marcu (2000) proposed using texts and the associated abstracts as data to conduct text summarization. Again, mapping between data in two spaces (texts and summaries) is a central issue.

When getting emails I get a notice that an email has been received but when I try to view the message it is blank. I have also tried to run the repair program off the install disk but that it did not take care of the problem.

(a)

receive emails; some emails have no subject and message body

(b)

Figure 1: A example of a text and its summary: (a) the text (b) the summary

3. Text Classification Using Stochastic Keyword Generation

3.1 Problem Setting

Suppose that we are to conduct text classification based on supervised learning. As usual, we have a number of classified texts and we can construct a classifier using these classified texts. Our new assumption is that in addition to the classified texts, we also have summaries associated to the texts (i.e., each of them has a summary). Furthermore, the summaries can be more accurately classified than the texts. Our question, then, is how we can use the summaries to improve the performance of text classification. Note that we assume that future texts *to be classified* do not have summaries.

As one example, we consider the case of email message classification at the help desk of Microsoft. The help desk receives emails from customers that describe software errors that the customers have encountered. Figure 1 (a) shows an example message. The engineers at the help desk assign labels to the messages, indicating the categories of the errors and the solutions, after they help resolve the problems. Thus, the email messages are hand-classified into categories. The categories include 'Cannot Open Word File', 'Empty Outlook Message', etc. The category of the message in Figure 1 (a) is 'Empty Outlook Message'. Usually, the engineers also add summaries to the messages. Figure 1 (b) shows the summary of the message in Figure 1 (a). We see that the customer's description is not as technical as the engineer's description. It is often the case that customers' descriptions vary depending on the person, while engineers' descriptions are much more consistent. Our goal here is to automatically classify *new* email messages, using both the classified messages and the summaries of the messages *as training data*. As indicated in Figure 1,

Table 1: Comparison between the conventional problem and the new problem on text classification

	Conventional	New
Classifier	$\mathcal{X} \to \mathcal{Y}$	$\mathcal{X} \to \mathcal{Y}$
Training Data	$\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)\}$	$\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)\}$ $\{(\mathbf{x}_1, \mathbf{s}_1), \cdots, (\mathbf{x}_l, \mathbf{s}_l)\}$
Test Data	$\{(\mathbf{x}'_1, y'_1), \cdots, (\mathbf{x}'_r, y'_r)\}$	$\{(\mathbf{x}'_1, y'_1), \cdots, (\mathbf{x}'_r, y'_r)\}$

summaries are more indicative, and thus should be more useful in classification.

Let us describe the problem more formally. Let $\mathcal{X} = \{0,1\}^n$ denote a space of features and $\mathcal{Y}$ a set of classes. Given a word list, let $\mathbf{x} = (x_1, x_2, \cdots, x_n) \in \mathcal{X}$ denote the value of a feature vector representing a text, in which the *i*-th element indicates the occurrence or non-occurrence of the *i*-th word of the word list in the text. Let $y \in \mathcal{Y}$ denote a class representing a category of texts. Let $\mathbf{X} = (X_1, X_2, \cdots, X_n)$ denote a random variable on $\mathcal{X}$, and Y a random variable on $\mathcal{Y}$.

Let $\mathcal{S} = \{0,1\}^m$ denote another space of features. Given a keyword list, let $\mathbf{s} = (s_1, s_2, \cdots, s_m) \in \mathcal{S}$ denote the value of a feature vector representing a summary, in which the *i*-th element indicates the occurrence or non-occurrence of the *i*-th keyword of the keyword list in the summary. Let $\mathbf{S} = (S_1, S_2, \cdots, S_m)$ denote a random variable on $\mathcal{S}$.

The conventional text classification problem can be described as follows. Given a set of classified texts $L_X = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_l, y_l)\}$ as training data, a supervised learning algorithm A creates a classifier $h_X = A(L_X)$ which is a function from $\mathcal{X}$ to $\mathcal{Y}$. Given a set of unclassified texts as test data, we use the classifier to classify them. The performance of the classifier can be measured in terms of the generalization error $E_{P(\mathbf{X},Y)}[h_X(\mathbf{X}) \neq Y]$.

In this paper, we employ *linear* classifiers which can be represented as

$$h_X(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} [\sum_{i=1}^{n} w_{i,y}^{(X)} x_i + b_y],$$

where $w_{i,y}^{(X)}$ represents the weight supporting y with the occurrence of the *i*-th word of the word list in the text, and b_y represents the prior weight supporting y. Naive Bayesian classifiers, perceptrons, and linear support vector machines are linear classifiers.

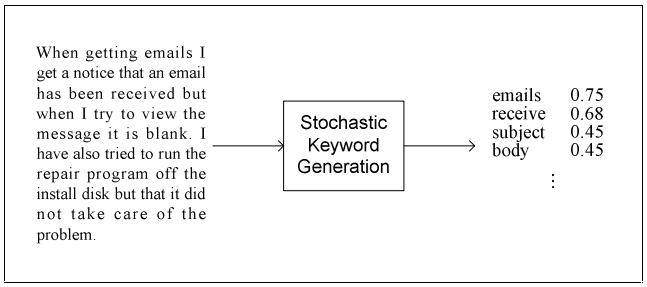


Figure 2: An example of stochastic keyword generation

In our new classification problem, we use as training data, not only $L_X = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_l, y_l)\}$, but also $\{(\mathbf{x}_1, \mathbf{s}_1), (\mathbf{x}_2, \mathbf{s}_2), \cdots, (\mathbf{x}_l, \mathbf{s}_l)\}$ which is a set of the texts and their associated summaries. We assume that the summaries can be more accurately classified than the texts themselves. More precisely, $E_{P(\mathbf{X},Y)}[h_X(\mathbf{X}) \neq Y] > E_{P(\mathbf{S},Y)}[h_S(\mathbf{S}) \neq Y]$, where $E_{P(\mathbf{S},Y)}[h_S(\mathbf{S}) \neq Y]$ denotes the generalization error of conducting classification on summaries, defined in a similar way as classification on texts.

Table 1 shows a comparison between the conventional text classification problem and the new text classification problem.

The question is whether we can construct a high performance classifier in our new problem setting.

3.2 Stochastic Keyword Generation

We consider the problem of automatically generating keywords for a given text. We propose a new approach to address this problem, which we call 'stochastic keyword generation' (SKG).

We assume that we have a number of texts, each having a summary. In addition, we have a list of keywords. One way to create the keyword list, which we employ in this paper, is to collect all the content words (non-stopwords) in the summaries and view them as keywords.

Thus, for each keyword, we can determine whether it occurs in the summary of a given text. For each keyword, we can create a binary classifier which judges how likely the keyword occurs (or does not occur) in the summary of a text, using the texts and the associated summaries. If we have *m* keywords then we can create *m* such classifiers.

Given a new text, we can use the classifiers to calculate the likelihood values of the keywords and output those keywords and their likelihood values. We call this process SKG. Figure 2 shows an example of SKG. Note that

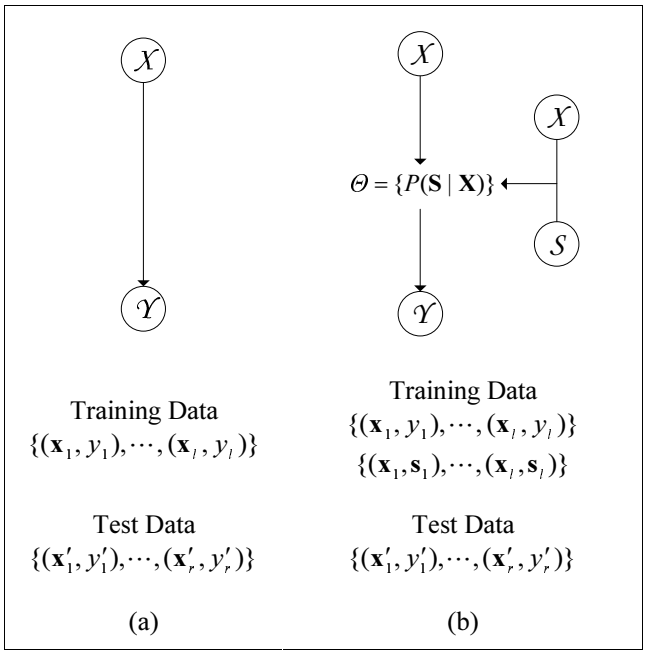


Figure 3: Differences between (a) the conventional approach and (b) our approach

although the word 'body' does not appear in the text, it can still be generated as a keyword.

More formally, given a data set containing texts and their summaries $\{(\mathbf{x}_1, \mathbf{s}_1), (\mathbf{x}_2, \mathbf{s}_2), \cdots, (\mathbf{x}_l, \mathbf{s}_l)\}$, we can construct a classifier for each of the keywords $(S_1, S_2, \cdots, S_m)$. For example, we can create a naive Bayesian classifier for each keyword. Given a new text $\mathbf{x}$, we can use the naive Bayesian classifiers to calculate the conditional probabilities of the keywords: $(P(S_1 = 1 \mid \mathbf{x}), P(S_2 = 1 \mid \mathbf{x}), \cdots, P(S_m = 1 \mid \mathbf{x}))$. Note that we use the naïve Bayesian classifiers as a density estimator here for assigning keyword probabilities, not for classification.

Hereafter, we will use Θ to denote the space of conditional probability values of keywords, and $\boldsymbol{\theta}(\mathbf{x}) = (\theta_1(\mathbf{x}), \theta_2(\mathbf{x}), \cdots, \theta_m(\mathbf{x}))$ a vector in Θ, where $\theta_j(\mathbf{x}) = P(S_j = 1 \mid \mathbf{x})$ $(j = 1, \cdots, m)$. We call it 'stochastic keyword generation model'.

We note that SKG can be used for both keyword generation and for other applications like text classification, as will be made clear below.

3.3 Algorithm

We propose to employ SKG in the text classification problem described in Section 3.1. Figure 3 illustrates the differences between the conventional approach (a) and our approach (b).

Input: $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_l, y_l)\}$
and $\{(\mathbf{x}_1, \mathbf{s}_1), (\mathbf{x}_2, \mathbf{s}_2), \cdots, (\mathbf{x}_l, \mathbf{s}_l)\}$

1. Train SKG model $\boldsymbol{\theta}(\mathbf{x})$ using $\{(\mathbf{x}_1, \mathbf{s}_1), (\mathbf{x}_2, \mathbf{s}_2), \cdots, (\mathbf{x}_l, \mathbf{s}_l)\}$
2. Map $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_l\}$ into $\{\boldsymbol{\theta}(\mathbf{x}_1), \boldsymbol{\theta}(\mathbf{x}_2), \cdots, \boldsymbol{\theta}(\mathbf{x}_l)\}$
3. Train classifier h_Θ using $L_\Theta = \{(\boldsymbol{\theta}(\mathbf{x}_1), y_1), (\boldsymbol{\theta}(\mathbf{x}_2), y_2), \cdots, (\boldsymbol{\theta}(\mathbf{x}_l), y_l)\}$:

(Let $\mathbf{w}_y^{(S)}$ denote $(w_{y,1}^{(S)}, w_{y,2}^{(S)}, \cdots, w_{y,m}^{(S)})$)
Select a confidence threshold τ and a learning speed η
for each ($y \in \mathcal{Y}$) $\mathbf{w}_y^{(S)}(0) \leftarrow \mathbf{0}, b_y(0) \leftarrow 0$
$t \leftarrow 0, R = \max_{\mathbf{x}_i} \|\boldsymbol{\theta}(\mathbf{x}_i)\|$
repeat
  for $i = 1, 2, \cdots, l$
    for each ($y \in \mathcal{Y}$)
      if $y = y_i$ **then** $z \leftarrow 1$
        else $z \leftarrow -1$
      if $z(\boldsymbol{\theta}(\mathbf{x}_i) \cdot \mathbf{w}_y^{(S)}(t) + b_y(t)) \le \tau$ **then**
        $\mathbf{w}_y^{(S)}(t+1) \leftarrow \mathbf{w}_y^{(S)}(t) + \eta z \boldsymbol{\theta}(\mathbf{x}_i)$
        $b_{t+1} \leftarrow b_{t+1} + \eta z R^2$
        $t \leftarrow t + 1$
      end if
    end for
  end for
until no updates made within the **for** loops

Output: classifier
$h_\Theta(\boldsymbol{\theta}(\mathbf{x})) = \arg\max_{y \in \mathcal{Y}} [\mathbf{w}_y^{(S)} \cdot \boldsymbol{\theta}(\mathbf{x}) + b_y]$

Figure 4: Learning algorithm

Figure 4 delineates three steps of the algorithm..

(1) Given the set of texts and their summaries $\{(\mathbf{x}_1, \mathbf{s}_1), (\mathbf{x}_2, \mathbf{s}_2), \cdots, (\mathbf{x}_l, \mathbf{s}_l)\}$, it constructs an SKG model $\boldsymbol{\theta}(\mathbf{x}) = (\theta_1(\mathbf{x}), \theta_2(\mathbf{x}), \cdots, \theta_m(\mathbf{x}))$. We employ the naïve Bayesian classifiers for creating the SKG model in this paper.

(2) Map the set of classified texts $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_l\}$ into a set of probability vectors $\{\boldsymbol{\theta}(\mathbf{x}_1), \boldsymbol{\theta}(\mathbf{x}_2), \cdots, \boldsymbol{\theta}(\mathbf{x}_l)\}$ using the constructed SKG model.

(3) Construct a classifier with the probability vectors and the associated classes $\{(\boldsymbol{\theta}(\mathbf{x}_1), y_1), (\boldsymbol{\theta}(\mathbf{x}_2), y_2), \cdots, (\boldsymbol{\theta}(\mathbf{x}_l), y_l)\}$. A classifier here is represented as $h_\Theta(\boldsymbol{\theta}(\mathbf{x})) = \arg\max_y [\sum_{i=1}^{m} w_{i,y}^{(S)} \theta_i(\mathbf{x}) + b_y]$.

In this paper, we employed the perceptron algorithm with margins (PAM) to train a perceptron classifier. Figure 4 shows the details of its training process. We also used the linear support vector machine (SVM) as a classifier.

In text classification, given a new text $\mathbf{x} \in \mathcal{X}$, our method first creates its corresponding $\boldsymbol{\theta}(\mathbf{x})$, and then uses the classifier $h_{\Theta}(\boldsymbol{\theta}(\mathbf{x}))$ to classify the text.

Note that in our method, stochastic keyword generation turns out to be a preprocessing step for text classification.

4. Experimental Results

4.1 Email Classification

We conducted experiments on email auto-classification at the help desk of Microsoft.

We used email messages with summaries to train stochastic keyword models as described in Section 3.2, and used the obtained stochastic keyword models and classified email messages to train classifiers as described in Section 3.3. Next, we used the obtained classifiers to classify other email messages.

There were 2,517 email messages belonging to 52 categories. The largest category contained more than 800 email messages, while the smallest category contained only 10. The average length of a message was about 80 words, and there were about 10,000 unique words in the email messages. All of the 2,517 messages were assigned summaries. The average length of a summary was 14 words, and there were about 1,500 unique words in the summaries. We removed stopwords (we defined 459 stopwords) from both the texts and summaries, but did not perform stemming on them.

We randomly split the data into five parts and conducted 5-fold cross validation. In each trial of the cross validation, we used the training examples for the construction of both SKG models and classifiers, and we discarded the summaries of the test examples. All of the experimental results reported below are those *averaged* over the five trials.

In each trial, we collected all the words whose total frequencies were larger than 2 from the summaries of the training examples and created a set of keywords $S_j, j = 1,2,\cdots,m$. There were on average 510 keywords in the five trials. For each keyword S_j, we selected from the email messages 20 words having the largest information gain values (cf. Yang & Pederson 1997). We denote the set of words related to S_j as $\mathcal{X}_j$. There were on average about 2,400 words in the set of the selected words $\mathcal{X} = \bigcup_{j=1}^{m} \mathcal{X}_j$.

In each trial, we constructed an SKG model with the texts and summaries in the training examples. For a given text **x**, we calculated the conditional probability of each keyword S_j given the text **x**, using only the corresponding set of words $\mathcal{X}_j$. This is because we found that this kind of 'feature selection' helps to improve performance. (We set $|\mathcal{X}_j| = 20$ based on a preliminary experimental result using a random 3:1 split of the training data prior to the first trial of the cross validation. See Section 4.2 for details on further experiments.).

We classified the test data using our method, namely classification based on SKG. When the classifier is trained with the perceptron algorithm with margins, we denote the method as 'SKG (PAM)'. When the classifier is the support vector machine, we denote the method as 'SKG (SVM)'. For PAM, we used a tool that we developed ourselves. For SVM, we used a tool developed by Platt (1998), which was also used in (Dumais et al. 1998) for text classification.

For comparison, we also tested other methods.

First, we used only email messages to train classifiers. (That is to say, we did not use the summaries). As classifiers, we employed SVM and PAM. Our use of SVM was exactly the same as that in (Dumais et al. 1998). That is, for each class, we conducted binary classification in which we selected 300 features based on information gain. PAM was also run on the same feature set. We denote the methods as 'Text (SVM)' and 'Text (PAM)' respectively.

Second, we also used all the words in the texts as features when using SVM and PAM. We denote them as 'Text (all)(SVM)' and 'Text (all)(PAM)' respectively.

Third, we used only summaries to train SVM and PAM classifiers. Next, we used them to classify email messages. The methods are denoted as 'Sum (SVM)' and 'Sum (PAM)' respectively.

Fourth, we merged each email message with its associated summary and viewed the merged results as peudo-texts to train classifiers. Again, we employed SVM and PAM. We denote the methods as 'Text+Sum (SVM)' and 'Text+Sum (PAM)' respectively.

Fifth, we treated the keywords extracted from the summaries as features of the email messages and conducted training and classification with the email messages on the basis of the features. This can be viewed as feature selection for text classification using summaries.

Table 2: Experimental results of email classification

Method	Top 1 Accuracy (%)	Top 3 Accuracy (%)
Prior	34.1	47.8
Text (PAM)	58.7	73.7
Text (all)(PAM)	58.3	73.5
Sum (PAM)	50.0	69.1
Text+Sum (PAM)	56.2	70.4
Text (KW)(PAM)	58.2	73.1
Text-$\mathcal{X}$(PAM)	58.3	73.8
SKG (PAM)	**63.6**	**78.9**
Text (SVM)	57.3	76.7
Text (all)(SVM)	56.7	76.9
Sum (SVM)	47.4	71.2
Text+Sum (SVM)	55.5	73.7
Text (KW)(SVM)	54.6	75.4
Text-$\mathcal{X}$(SVM)	57.6	77.0
SKG (SVM)	**61.5**	**81.5**

We also employed SVM and PAM as classifiers. We denote the methods as 'Text (KW)(SVM)' and 'Text (KW)(PAM)' respectively.

Sixth, we used PAM and SVM only on $\mathcal{X}$, the set of words used for creating the stochastic keywords generation models. We denote them as 'Text-$\mathcal{X}$ (PAM)' and 'Text-$\mathcal{X}$(SVM)' respectively.

We also tested the baseline method, denoted as 'Prior', in which we always chose the most frequently occurring classes.

Table 2 shows the results in terms of top 1 accuracy and top 3 accuracy.

From Table 2, we see that SKG (PAM) and SKG (SVM) significantly outperform their corresponding methods that do not use SKG. The results of the *sign test* (cf., Yang & Liu 1999) show that the improvements are *statistically significant* (p-value < 0.005).

The experimental results of Text (PAM), Text (all)(PAM) Text (SVM), and Text (all)(SVM) indicate that feature selection based on information gain do not affect the performances very much.

The performances of Sum (PAM) and Sum (SVM) are not as good as those of Text (PAM) and Text (SVM), suggesting that a classifier trained from the space of summaries cannot work well in the space of email messages. This is because the word usage in the email messages are quite different from the word usage in the summaries. For the same reason, Text (KW)(PAM) and Text (KW)(SVM) do not perform well either.

Table 3: Performance of SKG (PAM) when using different numbers of features $|\mathcal{X}_j|$ to model the conditional probability of each keyword S_j

$\lvert\mathcal{X}_j\rvert$	Top 1 Accuracy (%)	Top 3 Accuracy (%)
10	61.5	78.3
20	**63.6**	78.9
30	63.3	**79.3**
40	62.3	78.5
50	61.2	77.8

The performances of Text+Sum (PAM) and Text+Sum (SVM) are not better than those of Text (PAM) and Text (SVM), suggesting that simply merging texts with their summaries for training cannot help improve performance.

The performances of Text-$\mathcal{X}$ (PAM) and Text-$\mathcal{X}$ (SVM) are inferior to those of SKG (PAM) and SKG (SVM), indicating that the better performances of SKG were not due to the use of the selected feature set $\mathcal{X}$.

4.2 Discussions

We investigated the effects of using different numbers of words (features) from email messages in the construction of the SKG models. Table 3 shows the performances of SKG (PAM) in different settings. We see that all the results are better than those of non-SKG methods in Table 2, and among them the results achieve the best when $|\mathcal{X}_j| = 20$ and $|\mathcal{X}_j| = 30$. The results indicate that 20 to 30 words (features) will usually suffice to identify a keyword.

We also investigated the performance of *deterministic keyword generation*, in which we mapped examples from $\mathcal{X}$ to $\mathcal{S}$ instead of $\mathcal{X}$ to Θ in training and testing. More precisely, we first used the email messages and their summaries to train the naïve Bayesian classifiers, as we did before. However, we did not use the naïve Bayesian classifiers for probability assignment, but used them for *classification*. For each text, we used the classifiers to generate a 1-0 vector. The value was one, if the corresponding keyword has a probability value larger than 0.5, and zero otherwise. Next, we converted all of the classified texts into 1-0 vectors and used them for the construction of classifiers. We employed PAM and SVM as before. The experimental results, denoted as 'DKG (PAM)' and 'DKG (SVM)' respectively, are shown in Table 4. We see that DKG (PAM) slightly improves upon Text (PAM). However, the result from the sign test is *not* statistically significant. SKG (PAM) performs much

Table 4: Results of email classification with deterministic keyword generation (results with stochastic keyword generation are also listed)

Method	Top 1 Accuracy (%)	Top 3 Accuracy (%)
DKG (PAM)	59.8	73.9
SKG (PAM)	**63.6**	**78.9**
DKG (SVM)	57.6	76.9
SKG (SVM)	**61.5**	**81.5**

Table 5: Results of classification on summaries (results of classification on email messages are also listed)

Method	Top 1 Accuracy (%)	Top 3 Accuracy (%)
Text (PAM)	58.7	73.7
Sum-Sum (PAM)	87.0	96.8
Text (SVM)	57.3	76.7
Sum-Sum (SVM)	86.9	97.2

better than DKG (PAM), suggesting that the mapping of examples from $\mathcal{X}$ to Θ is more effective. The results with SVM have similar tendencies. Thus it is safe to say that probabilistic assignment of keywords is more reliable than deterministic assignment of keywords for text classification.

Table 5 shows the performances of classification on summaries (i.e., we used the summaries for both training and classification), which we denoted as Sum-Sum (PAM) and Sum-Sum (SVM). The results indicate that the summaries made by the engineers can be more accurately classified than the texts written by the customers. This has verified that our assumption (summaries can be more accurately classified than texts) holds for the data in the experiments.

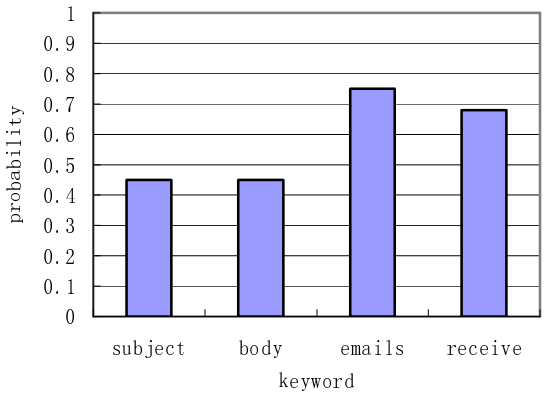
Figure 5 gives an example from the category 'Empty Outlook Message', which was incorrectly classified by Text (PAM) and Text (SVM), but was correctly classified by SKG (PAM) and SKG (SVM). The customer described the problem as 'it is blank'. In contrast, the engineer described the problem as 'no subject and message body'. It appears that the email messages written by a large group of customers tend to use varying and redundant expressions, while the summaries written by a small group of engineers tend to use uniform and concise expressions. This is the reason that summaries can be more accurately classified than texts. Figure 5(c) shows the result of the top four keywords and their probability values based on stochastic keyword generation. We see that SKG can 'recover' the summaries quite well. As a result, SKG can indeed effectively utilize the summary

When getting emails I get a notice that an email has been received but when I try to view the message it is blank. I have also tried to run the repair program off the install disk but that it did not take care of the problem.

(a)

receive emails; some emails have no subject and message body

(b)



(c)

Figure 5: An example of classification (a) the email message (b) the summary (c) top 4 keywords and their probability values

information for improving the performances of text classification.

5. Conclusions and Future Work

This paper has presented a new text classification method using stochastic keyword generation for preprocessing when summaries of texts are available during learning. Stochastic keyword generation is the technique of generating keywords from a text on the basis of conditional probability models. The models are trained based on supervised learning, with texts and their summaries as training data. In text classification based on SKG, a text is first mapped, with SKG, into a vector of probability values, each of which corresponds to a keyword. Text classification is then conducted on the mapped probability vector. Experimental results indicate that the performance of the new method in the email classification task for an automated help desk is significantly better than those of the baseline methods.

We note that SKG is not a technique limited only to text classification. It is potentially useful in other text processing, such as text summarization and text clustering.

We also note that the classification problem we have raised in this paper is not limited to the classification of

texts. Our approach may be applicable to other classification problems if we generalize stochastic keyword generation to stochastic feature generation.

There are several issues remaining for future work. First, theoretical analysis on the SKG method can be performed. Second, experiments with different settings can be conducted. For example, we assume that the texts which have summaries and the texts which have been classified belong to the same data set. In principle, the SKG method can be applied even when they do not.

Acknowledgements

We thank Yunbo Cao and the anonymous reviewers for their valuable comments on the earlier versions of this paper. We acknowledge Susan Dumais and John Platt for making the SVM tools available to us. We thank Ben Wellington and Tim Paek for checking the English of this paper.

References

Berger, A. & Lafferty, J. (1999). Information Retrieval as Statistical Translation. In *Proceedings of the Twenty-Second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 222-229.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., & Mercer, R. L. (1993). The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, vol. 19, no. 2, pp. 263-312.

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive Learning Algorithms and Representations for Text Classification. In *Proceedings of the Seventh ACM International Conference on Information and Knowledge Management*, pp. 148-155.

Fuhr, N., Hartmanna, S., Lustig, G., Schwantner, M., & Tzeras, K. (1991). Air/X – A Rule-based Multi-stage Indexing System for Large Subject Fields. In *Proceedings of the Third International Conference on* "*Recherche d'Information Assistee par Ordinateur*", pp. 606-623.

Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the Tenth European Conference on Machine Learning*, pp. 137-142.

Knight, K. & Marcu, D. (2000). Statistics-Based Summarization - Step One: Sentence Compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence*, pp. 703-710.

Krauth, W. & Mezard, M. (1987). Learning Algorithms with Optimal Stability in Neural Networks. *Journal of Physics A - Mathematical General*, vol. 20, no. 11, pp. 745-752.

Lewis, D. D. & Ringuette, M. (1994). A Comparison of Two Learning Algorithms for Text Categorization. In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*, pp. 81-93.

Li, H. & Yamanishi, K. (2002). Text Classification Using ESC-based Stochastic Decision Lists. *Information Processing and Management*, vol. 38, no. 3, pp. 343-361.

Li, Y., Zaragoza, H., Herbrich, R., Shawe-Taylor, J., & Kandola, J. (2002). The Perceptron Algorithm with Uneven Margin. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 379-386.

Liu, H. & Motoda, H. (1998). Feature Transformation and Subset Selection. *IEEE Intelligent System*, vol. 13, no. 2, pp. 26-28.

Platt, J. (1998). Fast Training of Support Vector Machines Using Sequential Minimal Optimization. In Schölkopf, B., Burges, C., & Smola, A. (Eds.), *Advances in Kernel Methods - Support Vector Learnin*g, pp. 185-208, MIT Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning Internal Representations by Error Propagation. In Rumelhart, D. E. & McClelland, J. L. (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, pp. 318-362, MIT Press.

Schapire, R. & Singer, Y. (2000). BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning*, vol. 39, no. 2/3, pp. 135—168.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*, Springer Verlag.

Wiener, E., Pedersen, J. O., & Weigend, A. S. (1995). A Neural Network Approach to Topic Spotting. In *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval*, pp. 317-322.

Yang, Y. (1994) Expert Network: Effective and Efficient Learning from Human Decisions in Text Categorization and Retrieval. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 13-22.

Yang, Y. & Pedereson, J. O. (1997). A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 412-420.

Yang, Y. & Liu, X. (1999). A Re-examination of Text Categorization Methods. In *Proceedings of the Twenty-Second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 42-49.