Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 7819626, 12 pages
http://dx.doi.org/10.1155/2016/7819626

*Research Article*

# Improved Feature Weight Algorithm and Its Application to Text Classification

**Songtao Shang,[1] Minyong Shi,[1] Wenqian Shang,[1] and Zhiguo Hong[2]**

[1]*School of Computer Science, Communication University of China, Beijing 100024, China*
[2]*School of Computer, Faculty of Science and Engineering, Communication University of China, Beijing 100024, China*

Correspondence should be addressed to Songtao Shang; songtao.shang@foxmail.com

Received 2 November 2015; Revised 1 February 2016; Accepted 3 March 2016

Academic Editor: Andrzej Swierniak



Text preprocessing is one of the key problems in pattern recognition and plays an important role in the process of text classification. Text preprocessing has two pivotal steps: feature selection and feature weighting. The preprocessing results can directly affect the classifiers' accuracy and performance. Therefore, choosing the appropriate algorithm for feature selection and feature weighting to preprocess the document can greatly improve the performance of classifiers. According to the Gini Index theory, this paper proposes an Improved Gini Index algorithm. This algorithm constructs a new feature selection and feature weighting function. The experimental results show that this algorithm can improve the classifiers' performance effectively. At the same time, this algorithm is applied to a sensitive information identification system and has achieved a good result. The algorithm's precision and recall are higher than those of traditional ones. It can identify sensitive information on the Internet effectively.

## 1. Introduction

The information in the real-world is always in disorder. Usually, we need to classify the disordered information for our cognition and learning. In the field of information processing, a text is the most basic form to express information, such as the news, websites, and online chat messages. Therefore, text classification is becoming an important work. Text classification (TC) [1] is the problem of automatically assigning predefined categories to free text documents. Vector Space Model (VSM) [2, 3] is widely used to express text information. That is, a text $D$ can be expressed as $V(d) = \{\langle T_1, W_1\rangle, \langle T_2, W_2\rangle, \langle T_3, W_3\rangle, \ldots, \langle T_n, W_n\rangle\}$, where $T_1, T_2, \ldots$, and $T_n$ are features of the text and $W_1, W_2, \ldots$, and $W_n$ are the weights of the features. Even a moderate-sized text collection consists of tens or hundreds of feature terms. This is prohibitively high for many machine learning algorithms, and only a few neutral network algorithms can handle such a large number of input features. However, many of these features are redundant, which will influence the precision of TC algorithms. Hence, the problem or difficulty in the research of TC is how to reduce the dimensionality of features.

Feature selection (FS) [4, 5] is to find the minimum feature subsets that can represent the original text; that is, FS can be used to reduce the redundancy of features, improve the comprehensibility of models, and identify the hidden structures in high-dimensional feature space. FS is the key step in the process of TC, since it is the preparation of classification algorithms, and the precision of FS has direct impact on the performance of classification. The common methods of FS often use machining learning algorithms [6], such as Information Gain, Expected Cross Entropy, Mutual Information, Odds Ratio, and CHI. Each algorithm has its own advantages and disadvantages. In the English data sets, Yang and Pedersen's experiments showed that the Information Gain, Expected Cross Entropy, and CHI algorithms get better performance [7]. In Chinese data sets, Liqing et al.'s experiments showed that the CHI and Information Gain algorithms get the best performance and the Mutual Information algorithm is the worst [8]. Many researchers employ other theories or algorithms on FS; references [9, 10] have adopted Gini Index into FS and achieved good performance. Gini Index is a measurement to evaluate the impurity of a set, which describes the importance of features in classification. It is widely used for splitting attributes in

decision tree algorithms [11]. That is to say, Gini Index can identify the importance of features.

On the other hand, feature weighting (FW) is another important aspect of FS. TF-IDF [12] is a popular FW algorithm. However, TF-IDF is unsuitable for text FW. The basic idea of TF-IDF is that a feature term is more important if it has a higher frequency in a text, known as Term Frequency (TF); and feature term is less important if it appears in different text documents in a training set, known as Inverse Document Frequency (IDF). In TC, feature frequency is one of the most important aspects regardless of its appearance in one or more texts. Many researchers have improved the TF-IDF algorithm by replacing the IDF part with FS algorithms, such as TF-IG and TF-CHI [13–15]. This paper mainly focuses on an Improved Gini Index algorithm and its application and proposes an FW algorithm using Improved Gini Index algorithm instead of the IDF part of TF-IDF algorithm, known as TF-Gini algorithm. The experiment results of the algorithm show that the TF-Gini algorithm is a promising algorithm. To test the performance of the Improved Gini Index and TF-Gini algorithms in this paper, we introduce $k$NN [16, 17], f$k$NN [18], and SVM [19, 20] as the TC algorithms standards.

As an application of TF-Gini algorithm, this paper designs and implements a sensitive information monitoring system. Sensitive information on the Internet is the most interesting phenomenon that information experts are eager to investigate. Since the application of the Internet is developing very fast, many new words and phrases emerge on the Internet every day. Therefore, new words and phrases pose challenges to sensitive information monitoring. At present, most of the network information monitoring and filtering algorithms are inflexible and inaccurate. By the previous research results, we use the TF-Gini as the core algorithm and design and implement a system for sensitive information monitoring. This system can update sensitive words and phrases in time and has better performance on monitoring the Internet information.

## 2. Text Classification

*2.1. Text Classification Process.* The text classification process is shown in Figure 1. The left part is the training process. The right part is the classification process. The function of each part can be described as follows.

*2.1.1. The Training Process.* The purpose of the training process is to build a classification model. The new samples are assigned to appropriate category by using this model.

*(1) Words Segmentation*. In English and other Western languages there is a space delimiter between words. In Oriental languages there are no space delimiters between words. Chinese word segmentation is the basis for all Chinese text processing. There are many Chinese word segmentation algorithms [21, 22]; most of them use machine learning algorithms. ICTCLAS (Institute of Computing Technology, Chinese Lexical Analysis System) is the best Chinese lexical analyzer, which is designed by ICT (Institute of Computing Technology, Chinese Academy of Science). In this paper, we use ICTCLAS for Chinese word segmentation.

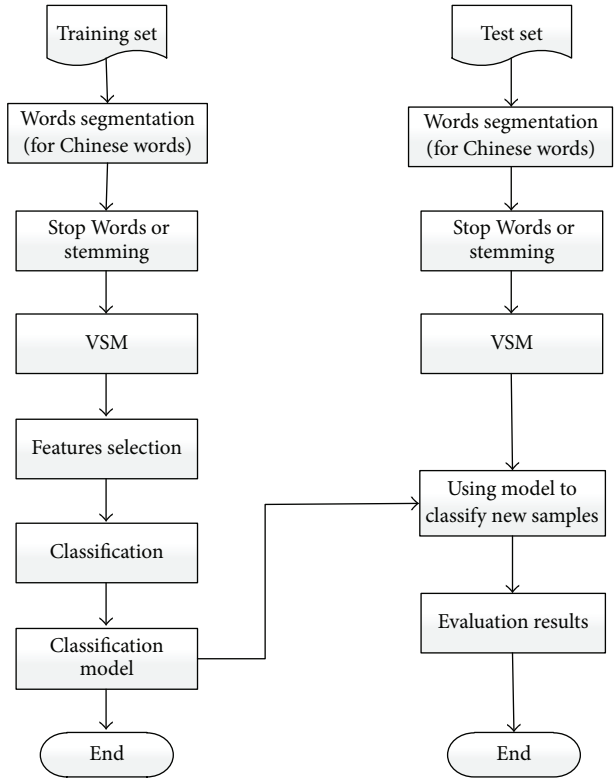


Figure 1: Text classification process.

*(2) Stop Words or Stemming*. A text document always contains hundreds or thousands of words. Many of the words appear in a very high frequency but are useless, such as "the," "a," nonsense articles, and adverbs. These words are called Stop Words [23]. Stop Words, which are language-specific functional words, are frequent words that carry no information. The first step during text process is to remove these Stop Words.

Stemming techniques are used to find out the root or stem of a word. Stemming converts words to their stems, which incorporates a great deal of language-dependent linguistic knowledge. Behind stemming, the hypothesis is that words with the same stem or root mostly describe the same or relatively close concepts in the document. Hence, words can be conflated by using stems. For example, the words, *user, users, used,* and *using* all can be stemmed to the word "USE." In Chinese text classification, we only need Stop Words because there is no conception of stemming in Chinese.

*(3) VSM (Vector Space Model)*. VSM is very popular in natural language processing, especially in computing the similarity between documents. VSM is mapping a document to a vector. For example, if we view each feature word as a dimension, and the word frequency as its value, the document can be represented as an $n$-dimensional vector; that is, $T = \{\langle t_1, w_1\rangle, \langle t_2, w_2\rangle, \ldots, \langle t_n, w_n\rangle\}$, where $t_i$ $(1 \leq i \leq n)$ is the

Table 1: Three documents represented in the VSM.

| | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | $t_6$ | $t_7$ | $t_8$ | $t_9$ | $t_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $\text{doc}_1$ | 1 | 2 | | 5 | | 7 | | 9 | | |
| $\text{doc}_2$ | | 3 | | 4 | | 6 | 8 | | | |
| $\text{doc}_3$ | 10 | | 11 | | 12 | | | 13 | 14 | 15 |

$i$th feature and $w_i$ $(1 \le i \le n)$ is the feature's frequency which is also called weights. Table 1 is an example of a VSM of three documents with ten feature words. $\text{doc}_1$, $\text{doc}_2$, and $\text{doc}_3$ are documents; $t_1, t_2, \ldots, t_{10}$ are feature words.

Therefore, we can represent $\text{doc}_1$, $\text{doc}_2$, and $\text{doc}_3$ in VSM as $T_1 = \{\langle t_1, 1\rangle, \langle t_2, 2\rangle, \langle t_4, 5\rangle, \langle t_6, 7\rangle, \langle t_8, 9\rangle\}$, $T_2 = \{\langle t_2, 3\rangle, \langle t_4, 4\rangle, \langle t_6, 6\rangle, \langle t_7, 8\rangle\}$, and $T_3 = \{\langle t_1, 10\rangle, \langle t_3, 11\rangle, \langle t_5, 12\rangle, \langle t_8, 13\rangle, \langle t_9, 14\rangle, \langle t_{10}, 15\rangle\}$.

*(4) Feature Words Selection.* After Stop Words and stemming, there are still tens of thousands of words in a training set. It is impossible for a classifier to handle so many words. The purpose of feature words selection is to reduce the dimension of features words. It is a very important step for text classification. Choosing the appropriate representative words can improve classifiers' performance. This paper mainly focuses on this stage.

*(5) Classification and Building Classification Model.* After the above steps, all the training set texts have been mapped to the VSM. This step uses the classification algorithms on VSM and builds classification model which can assign an appropriate category to new input samples.

*2.1.2. The Classification Process.* The steps of word segmentation, Stop Words or stemming, and VSM are similar to the classification process. The classification process will assign an appropriate category to new samples. When a new sample arrives, using the classification model calculates the most likely class label of the new sample and assigns it to the sample. The results evaluation is a very important step in the text classification process. It indicates whether the algorithm is good or not.

*2.2. Feature Selection.* Although the Stop Words and stemming step get rid of some useless words, the VSM still has many insignificant words which even have a negative effect on classification. It is difficult for many classification algorithms to handle high-dimensional data sets. Hence, we need to reduce the dimensional space and improve the classifiers' performance. FS uses machine learning algorithms to choose appropriate words that can represent the original text, which can reduce the dimensions of the feature space.

The definition of FS is as follows.

Select $m$ features from the original $n$ features $(m \le n)$. The $m$ features can be more concise and more efficient to represent the contents of the text. The commonly used FS algorithm can be described as follows.

*2.2.1. Information Gain (IG).* IG is widely used in the machine learning field, which is a criterion of the importance of the feature to a class. The value of IG equals the difference of information entropy before and after the appearance of a feature in a class. The greater the IG, the greater the amount of information the feature contains, the more important in the TC. Therefore, we can filter the best feature subset according to the feature's IG. The computing formula can be described as follows:

$$\begin{aligned}\text{InfGain}(w) = &-\sum_{i=1}^{|C|} P(c_i)\log P(c_i) \\ &+ P(w)\sum_{i=1}^{|C|} P(w \mid c_i)\log P(w \mid c_i) \\ &+ P(\overline{w})\sum_{i=1}^{|C|} P(\overline{w}_i \mid c_i)\log P(\overline{w} \mid c_i),\end{aligned} \tag{1}$$

where $w$ is the feature word, $P(w)$ is the probability of text that contains $w$ appearing in the training set, $P(\overline{w})$ is the probability of text that does not contain $w$ appearing in the training set, $P(c_i)$ is the probability of $c_i$ class in the training set, $P(w \mid c_i)$ is the probability of text that contains $w$ in $c_i$, $P(\overline{w} \mid c_i)$ is the probability of text that does not contain $w$ in $c_i$, and $|C|$ is the total amount of classes in training set.

The shortage of IG algorithm is that it takes into account features which do not appear although in some circumstance, the nonappearing features could contribute to the classification or the contribution is far less than that of features appearing in the training set. However, if the training set is unbalanced and $P(\overline{w}) \gg P(w)$, the value of IG will be decided by $P(\overline{w})\sum_{i=1}^{|C|} P(\overline{w}_i \mid c)\log P(\overline{w} \mid c_i)$, and the performance of IG can be significantly reduced.

*2.2.2. Expected Cross Entropy (ECE).* ECE is well known as KL distance, and it reflects the distance between the probability of the theme class and the probability of the theme class under the condition of a specific feature. The computing formula can be described as follows:

$$\begin{aligned}&\text{CrossEntryText}(w) \\ &= P(w)\sum_{i=1}^{|C|} P(w \mid c_i)\log\frac{P(w \mid c_i)}{P(c_i)},\end{aligned} \tag{2}$$

where $w$ is the feature word, $P(w)$ is the probability of text that contains $w$ appearing in the training set, $P(c_i)$ is the probability of class $c_i$ in the training set, $P(w \mid c_i)$ is the probability of text that contains $w$ in $c_i$, and $|C|$ is the total amount of classes in training set.

ECE has been widely used in feature selection of TC, and it also achieves good performance. Compared with the IG algorithm, ECE does not consider the case that feature does not appear, which reduced the interference of rare feature which does not occur often, and improves the performance of classification. However, it also has some limitations.

(1) If $P(w \mid c_i) > P(c_i)$, then $\log(P(w \mid c_i)/P(c_i)) > 0$. When $P(w \mid c_i)$ is larger and $P(c_i)$ is smaller, the logarithmic value is larger, which indicates that feature $w$ has strong association with class $c_i$.

(2) If $P(w \mid c_i) < P(c_i)$, then $\log(P(w \mid c_i)/P(c_i)) < 0$. When $P(w \mid c_i)$ is smaller and $P(c_i)$ is larger, the logarithmic value is smaller, which indicates that feature $w$ has weaker association with class $c_i$.

(3) When $P(w \mid c_i) = 0$, there is no association between feature $w$ and the class $c_i$. In this case, the logarithmic expression has no significance. Introducing a small parameter, for example, $\varepsilon = 0.0001$, that is, $\log((P(w \mid c_i) + \varepsilon)/P(c_i))$, the logarithmic will have significance.

(4) If $P(w \mid c_i)$ and $P(c_i)$ are close to each other, then $\log(P(w \mid c_i)/P(c_i))$ is close to zero. When $P(w \mid c_i)$ is large, this indicates that feature $w$ has a strong association with class $c_i$ and should be retained. However, when $\log(P(w \mid c_i)/P(c_i))$ is close to zero, feature $w$ will be removed. In this case, we employ information entropy to add to the ECE algorithm. The formula of information entropy is as follows:

$$\mathrm{IE}(w) = -\sum_{i=0}^{|C|} P(w \mid c_i) \log\left(P(w \mid c_i)\right). \tag{3}$$

In a summary, we combine (2) and (3) together; the ECE formula is as follows:

$$\mathrm{ECE}(w) = \frac{P(w)}{\mathrm{IE}(w) + \varepsilon} \sum_{i=1}^{|C|} \left(P(w \mid c_i) + \varepsilon\right) \log \frac{P(w \mid c_i) + \varepsilon}{P(c_i)}. \tag{4}$$

If feature $w$ exists only in one class, the value of information entropy is zero; that is, $\mathrm{IE}(w) = 0$. Hence we should introduce a small parameter in the denominator as a regulator.

*2.2.3. Mutual Information (MI).* In TC, MI expresses the association between feature words and classes. The MI between $w$ and $c_i$ is defined as

$$\mathrm{MutualInfo}(w) = \log \frac{P(w \mid c_i)}{P(w)} = \log \frac{P(w, c_i)}{P(w)\,P(c_i)}, \tag{5}$$

where $w$ is the feature word, $P(w, c_i)$ is the probability of text that contains $w$ in class $c_i$, $P(w)$ is the probability of text that contains $w$ in the training set, and $P(c_i)$ is the probability of class $c_i$ in the training set.

It is necessary to calculate the MI between features and each category in the training set. The following two formulas can be used to calculate the MI:

$$\begin{aligned} \mathrm{MI}_{\max}(w) &= \max_{i=1}^{|C|} \mathrm{MutualInfo}(w), \\ \mathrm{MI}_{\mathrm{avg}}(w) &= \sum_{i=1}^{|C|} P(c_i)\,\mathrm{MutualInfo}(w), \end{aligned} \tag{6}$$

where $P(c_i)$ is the probability of class $c_i$ in the training set and $|C|$ is the total amount of classes in the training set. Generally, $\mathrm{MI}_{\max}(w)$ is used commonly.

*2.2.4. Odds Ratio (OR).* The computing formula can be described as follows:

$$\mathrm{OddsRatio}(w, c_i) = \log \frac{P(w \mid c_i)\left(1 - P(w \mid \bar{c}_i)\right)}{\left(1 - P(w \mid c_i)\right) P(w \mid c_i)}, \tag{7}$$

where $w$ is the feature word, $c_i$ represents the positive class and $\bar{c}_i$ represents the negative class, $P(w \mid c_i)$ is the probability of text that contains $w$ in class $c_i$, and $P(w \mid \bar{c}_i)$ is the conditional probability of text that contains $w$ in all the classes except $c_i$.

OR metric measures the membership and nonmembership to a specific class with its numerator and denominator, respectively. Therefore, the numerator must be maximized and the denominator must be minimized to get the highest score according to the formula. The formula is a one-sided metric because the logarithm function produces negative scores while the value of the fraction is between 0 and 1. In this case, the features have negative values pointed to negative features. Thus, if we only want to identify the positive class and do not care about the negative class, Odds Ratio will have a great advantage. Odds Ratio is suitable for binary classifiers.

*2.2.5. $\chi^2$ (CHI).* $\chi^2$ statistical algorithm measures the relevance between the feature $w$ and the class $c_i$. The higher the score, the stronger the relevance between the feature $w$ and the class $c_i$. It means that the feature has a greater contribution to the class. The computing formula can be described as follows:

$$\chi^2(w, c_i) = \frac{|C|\left(A_1 \times A_4 - A_2 \times A_3\right)^2}{\left(A_1 + A_3\right)\left(A_2 + A_4\right)\left(A_1 + A_2\right)\left(A_3 + A_4\right)}, \tag{8}$$

where $w$ is the feature word, $A_1$ is the frequency containing feature $w$ and class $c_i$, $A_2$ is the frequency containing feature $w$ but not belonging to class $c_i$, $A_3$ is the frequency belonging to class $c_i$ but not containing the feature $w$, $A_4$ is the frequency not containing the feature $w$ and not belonging to class $c_i$, and $|C|$ is the total number of text documents in class $c_i$.

When the feature $w$ and class $c_i$ are independent, that is, $\chi^2(w, c_i) = 0$, this means that the feature $w$ is not containing identification information. We calculate values for each category and then use formula (9) to compute the value of the entire training set:

$$\chi^2_{\max}(w) = \max_{1 \le i \le |C|} \left\{\chi^2(w, c_i)\right\}. \tag{9}$$

## 3. The Gini Feature Selection Algorithm

### 3.1. The Improved Gini Index Algorithm

*3.1.1. Gini Index (GI).* The GI is a kind of impurity attribute splitting method. It is widely used in the CART algorithm [24], the SLIQ algorithm [25], and the SPRINT algorithm [26]. It chooses the splitting attribute and obtains very good classification accuracy. The GI algorithm can be described as follows.

Suppose that $Q$ is a set of $s$ samples, and these samples have $k$ different classes ($c_i$, $i = 1, 2, 3, \ldots, k$). According to the differences of classes, we can divide $Q$ into $k$ subsets ($Q_i$, $i = 1, 2, 3, \ldots, k$). Suppose that $Q_i$ is a sample set that belongs to class $c_i$ and that $s_i$ is the total number of $Q_i$; then the GI of set $Q$ is

$$\text{Gini}(Q) = 1 - \sum_{i=1}^{|C|} P_i^2, \tag{10}$$

where $P_i$ is the probability of $Q_i$ in $c_i$ and calculated by $s_i/s$.

When Gini($Q$) is 0 that is the minimum, all the members in the set belong to the same class; that is, the maximum useful information can be obtained. When all of the samples in the set distribute equally for each class, Gini($Q$) is the maximum; that is, the minimum useful information can be obtained.

For an attribute $A$ with $l$ distinct values, $Q$ is partitioned into $l$ subsets $Q_1, Q_2, \ldots, Q_l$. The GI of $Q$ with respect to the attribute $A$ is defined as

$$\text{Gini}_A(Q) = 1 - \sum_{i=1}^{l} \frac{s_i}{s} \text{Gini}(Q). \tag{11}$$

The main idea of GI algorithm is that the attribute with the minimum value of GI is the best attribute which is chosen to split.

*3.1.2. The Improved Gini Index (IGI) Algorithm.* The original form of the GI algorithm was used to measure the impurity of attributes towards classification. The smaller the impurity is, the better the attribute is. However, many studies on GI theory have employed the measure of purity: the larger the value of the purity is, the better the attribute is. Purity is more suitable for text classification. The formula is as follows:

$$\text{Gini}(Q) = \sum_{i=1}^{|C|} P_i^2. \tag{12}$$

This formula measures the purity of attributes towards categorization. The larger the value of purity is, the better the attribute is. However, we always emphasize the high-frequency words in TC because the high-frequency words have more contributions to judge the class of texts. But when the distribution of the training set is highly unbalanced, the lower-frequency feature words still have some contribution to judge the class of texts, although the contribution is far less significant than that the high-frequency feature words have. Therefore, we define the IGI algorithm as

$$\text{IGI}(w) = \sum_{i=1}^{|C|} P(w \mid c_i)^2 \, P(c_i \mid w)^2, \tag{13}$$

where $w$ is the feature word, $|C|$ is the total number of classes in the training set, $P(w \mid c_i)$ is the probability of text that contains $w$ in class $c_i$, and $P(c_i \mid w)$ is the posterior probability of text that contains $w$ in class $c_i$.

This formula overcomes the shortcoming of the original GI, which considers the features' conditional probability, combining the conditional probability and posterior probability. Hence, the IGI algorithm can depress the affection when the training set is unbalanced.

*3.2. TF-Gini Algorithm.* The purpose of FS is to choose the most suitable representative features to represent the original text. In fact, the features' distinguishability is different. Some features can effectively distinguish a text from others, but other features cannot. Therefore, we use weights to identify different features' distinguishability. The higher distinguishability of features will get higher weights.

The TF-IDF algorithm is a classical algorithm to calculate the features' weights. For a word $w$ in text $d$, the weights of $w$ in $d$ are as follows:

$$\begin{aligned} \text{TF-IDF}(w, d) &= \text{TF}(w, d) \cdot \text{IDF}(w, d) \\ &= \text{TF}(w, d) \cdot \log\left(\frac{|D|}{\text{DF}(w)}\right), \end{aligned} \tag{14}$$

where TF($w, d$) is the frequency of $w$ appearance in text $d$, $|D|$ is the number of text documents in the training set, and DF($w$) is the total number of texts containing $w$ in the training set.

The TF-IDF algorithm is not suitable for TC, mainly because of the shortcoming of TF-IDF in the section IDF. The TF-IDF considers that a word with a higher frequency in different texts in the training set is not important. This consideration is not appropriate for TC. Therefore, we use purity Gini to replace the IDF part in the TF-IDF formula. The purity Gini is as follows:

$$\text{GiniTxt}(w) = \sum_{i=1}^{|C|} P_i (w \mid c_i)^{\delta}, \tag{15}$$

where $w$ is the feature word and $\delta$ is a nonzero value. $P_i(w \mid c_i)$ is the probability of text that contains $w$ in class $c_i$, and $|C|$ is the total number of classes in the training set. When $\delta = 2$, the following formula is the original Gini formula:

$$\text{GiniTxt}(w) = \sum_{i=1}^{|C|} P (w \mid c_i)^2. \tag{16}$$

However, in TC, the experimental results indicate that when $\delta = -1/2$, the classification performance is the best:

$$\text{GiniTxt}(w) = \sum_{i=1}^{|C|} \frac{\sqrt{P(w \mid c_i)}}{P(w \mid c_i)}. \tag{17}$$

Therefore, the new feature weighting algorithm, known as TF-Gini, can be represented as

$$\text{TF-Gini} = \text{TF} \cdot \text{GiniTxt}(w). \tag{18}$$

*3.3. Experimental and Analysis*

*3.3.1. Data Sets Preparation.* In this paper, we choose English and Chinese data sets to verify the performance of the FS and FW algorithms. The English data set which we choose

is Reuters-21578. We choose ten big categories among them. The Chinese data set is Fudan University Chinese text data set (19,637 texts). We use 3,522 pieces as the training texts and 2,148 pieces as the testing texts.

*3.3.2. Classifiers Selection.* This paper is mainly focused on text preprocessing stage. The IGI and TF-Gini algorithms are the main research area in this paper. To validate the above two algorithms' performance, we introduce some commonly used text classifiers, such as *k*NN classifier, f*k*NN classifier, and SVM classifier. All the experiments are based on these classifiers.

*(1) The kNN Classifier*. The $k$ nearest neighbors ($k$NN) algorithm is a very popular, simple, and nonparameter text classification algorithm. The idea of $k$NN algorithm is to choose the category for new input sample that appears most often amongst its $k$ nearest neighbors. Its decision rule can be described as follows:

$$\mu_j(d) = \sum_{i=1}^{k} \mu_j(d_i)\,\mathrm{sim}(d, d_i), \tag{19}$$

where $j$ ($j = 1, 2, 3, \ldots$) is the $j$th text in the training set, $d$ is a new input sample with unknown category, and $d_i$ is a testing text whose category is clear. $\mu_j(d_i)$ is equal to 1 if sample $d_i$ belongs to class $j$; otherwise it is 0. $\mathrm{sim}(d, d_i)$ indicates the similarity between the new input sample and the testing text. The decision rule is

$$\begin{aligned} &\text{if} \quad \mu_j(d) = \max_i \mu_i(d), \\ &\text{then} \quad d \in c_j. \end{aligned} \tag{20}$$

*(2) The fkNN Classifier*. The $k$NN classifier is a lazy classification algorithm. It does not need the training set. The classifier's performance is not ideal, especially when the distribution of the training set is unbalanced. Therefore, we improve the algorithm by using the fuzzy logic inference system. Its decision rule can be described as follows:

$$\mu_j(d) = \frac{\sum_{i=1}^{k} \mu_j(d_i)\,\mathrm{sim}(d, d_i)\left(1 - \mathrm{sim}(d, d_i)\right)^{2/(1-b)}}{\sum_{i=1}^{k} \left(1 - \mathrm{sim}(d, d_i)\right)^{2/(1-b)}}, \tag{21}$$

where $j$ ($j = 1, 2, 3, \ldots$) is the $j$th text in the training set and $\mu_j(d_i)\mathrm{sim}(d, d_i)$ is the value of memberships for sample $d$ belonging to class $j$. $\mu_j(d_i)$ is equal to 1 if sample $d_i$ belongs to class $j$; otherwise it is 0. From the formula, it can be seen that f$k$NN is based on $k$NN algorithm, which endows a distance weight on $k$NN formula. The parameter $b$ adjusts the degree of the weight, $b \in [0, 1]$. The f$k$NN decision rule is as follows:

$$\begin{aligned} &\text{if} \quad \mu_j(d) = \max_i \mu_i(d), \\ &\text{then} \quad d \in c_j. \end{aligned} \tag{22}$$

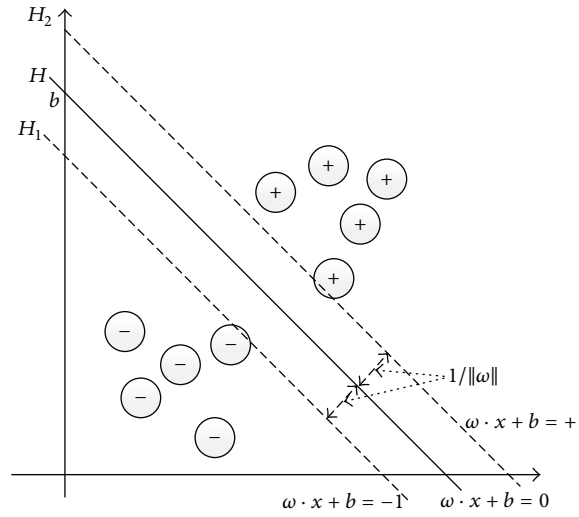


Figure 2: Optimal hyperplane for SVM classifier.

*(3) The SVM Classifier*. Support vector machine (SVM) is a potential classification algorithm, which is based on statistical learning theory. SVM is highly accurate, owing to its ability to model complex nonlinear decision boundary. It uses a nonlinear mapping to transform the original set into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane, that is, decision boundary, with an appropriate nonlinear mapping to a sufficiently high dimension; data from two classes can always be separated by a hyperplane. Originally, SVM mainly solves the problem of two-class classification. The multiclass problem can be addressed using multiple binary classifications.

Suppose that in the feature space a set of training vectors $x_i$ ($i = 1, 2, 3, \ldots$) belong to different classes $y_i \in \{-1, 1\}$. We wish to separate this training set with a hyperplane:

$$\omega \cdot x + b = 0. \tag{23}$$

Obviously, there are an infinite number of hyperplanes that are able to partition the data set into sets. However, according to the SVM theory, there is only one optimal hyperplane, which is lying half-way within the maximal margin as the sum of distances of the hyperplane to the closest training sample of each class. As shown in Figure 2, the solid line represents the optimal separating hyperplane.

One of the problems of SVM is to find the maximum separating hyperplane, that is, the one maximum distance between the nearest training samples. In Figure 2, $H$ is the optimal hyperplane; $H_1$ and $H_2$ are the parallel hyperplanes to $H$. The task of SVM is making the margin distance between $H_1$ and $H_2$ maximum. The maximum margin can be written as follows:

$$\left.\begin{aligned} \omega \cdot x_i + b \geq +1 \quad y_i = +1 \\ \omega \cdot x_i + b \leq -1 \quad y_i = -1 \end{aligned}\right\} \Longrightarrow \quad y_i\left(\omega \cdot x_i + b\right) - 1 \geq 0. \tag{24}$$

Therefore, from Figure 2, we can get that the process of solving SVM is making the margin between $H_1$ and $H_2$ maximum:

$$\min_{\{x_i|y_i=1\}} \frac{\omega \cdot x_i + b}{\|\omega\|} - \max_{\{x_i y_i=-1\}} \frac{\omega \cdot x_i + b}{\|\omega\|} = \frac{2}{\|\omega\|}. \quad (25)$$

The equation is also the same as minimizing $\|\omega\|^2/2$. Therefore, it becomes the following optimization question:

$$\begin{aligned} \min \quad & \left( \frac{\|\omega\|^2}{2} \right) \\ \text{st.} \quad & y_i \left( \omega \cdot x_i + b \right) \geq 1. \end{aligned} \quad (26)$$

The formula means the points which satisfy $y_i(\omega \cdot x_i + b) = 1$, called support vectors, namely, the points on $H_1$ and $H_2$ in Figure 2, are all support vectors. In fact, many samples are not classified correctly by the hyperplane. Hence, we introduce slack variables. Then, the above formula becomes

$$\begin{aligned} \min \quad & \frac{1}{2} \|\omega\|^2 - \eta \cdot \sum_{i=1}^{|C|} \xi_i \\ \text{st.} \quad & y_i \left( \omega \cdot x_i + b \right) \geq 1 - \xi_i, \end{aligned} \quad (27)$$

where $\eta$ is a positive constant to balance the experience and confidence and $|C|$ is the total number of classes in the training set.

*3.3.3. Performance Evaluation.* Confusion matrix [27], also known as a contingency table or an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning algorithm. It is a table with two rows and two columns that reports the number of *false positives, false negatives, true positives,* and *true negatives*. Table 2 shows the confusion matrix.

True positive (TP) is the number of correct predictions that an instance is positive. False positive (FP) is the number of incorrect predictions that an instance is positive. False negative (FN) is the number of incorrect predictions that an instance is positive. True negative (TN) is the number of incorrect predictions that an instance is negative.

Precision is the proportion of the predicted positive cases that were correct, as calculated using the equation

$$\text{Precision } (P) = \frac{\text{TP}}{\text{TP} + \text{FP}}. \quad (28)$$

Recall is the proportion of the actual positive cases that were correct, as calculated using the equation

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (29)$$

We hope that it is better when the precision and recall have higher values. However, both of them are contradictory. For example, if the result of prediction is only one and accurate, the precision is 100% and the recall is very low because TP equals 1, FP equals 0, and FN is huge. On the other hand, if the prediction as positive contains all the results (i.e., FN equals 0), therefore the recall is 100%. The precision is low because FP is large.

Table 2: Confusion matrix.

| | Prediction as positive | Prediction as negative |
|---|---|---|
| Actual positive | TP | FN |
| Actual negative | FP | TN |

$F1$ [28] is the average of precision and recall. If $F1$ is high, the algorithm and experimental methods are ideal. $F1$ is to measure the performance of a classifier considering the above two methods and was proposed by Van Rijsbergen [29] in 1979. It can be described as follows:

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (30)$$

Precision and recall only evaluate the performance of classification algorithms in a certain category. It is often necessary to evaluate classification performance in all the categories. When dealing with multiple categories there are two possible ways of averaging these measures, namely, macroaverage and microaverage. The macroaverage weights equally all classes, regardless of how many documents belong to it. The microaverage weights equally all the documents, thus favoring the performance in common classes. The formulas of macroaverage and microaverage are as follows, and $|C|$ is the number of classes in the training set:

$$\begin{aligned} \text{Macro\_}p &= \frac{\sum_{i=1}^{|C|} \text{Precision}_i}{|C|}, \\ \text{Macro\_}r &= \frac{\sum_{i=1}^{|C|} \text{Recall}_i}{|C|}, \\ \text{Macro\_}F1 &= \frac{\sum_{i=1}^{|C|} F1_i}{|C|}, \\ \text{Micro\_}p &= \frac{\sum_{i=1}^{|C|} \text{TP}_i}{\sum_{i=1}^{|C|} \left( \text{TP}_i + \text{FP}_i \right)}, \\ \text{Micro\_}r &= \frac{\sum_{i=1}^{|C|} \text{TP}_i}{\sum_{i=1}^{|C|} \left( \text{TP}_i + \text{FN}_i \right)}, \\ \text{Micro\_}F1 &= \frac{2 \times \text{Micro\_}p \times \text{Micro\_}r}{\text{Micro\_}p + \text{Micro\_}r}. \end{aligned} \quad (31)$$

*3.3.4. Experimental Results and Analysis.* Figures 3 and 4 are the results on the English data set. The performance of classical GI is the lowest, and the IGI is the highest. Therefore, the IGI algorithm is helpful to improve the classifiers' performance.

From Figures 5 and 6, it can be seen that, in Chinese data sets, the classical GI still has poor performance. The best performance belongs to ECE, and then the second is IG. The performance of IGI is not the best. Since the Chinese language is different from the English language, at the processing of

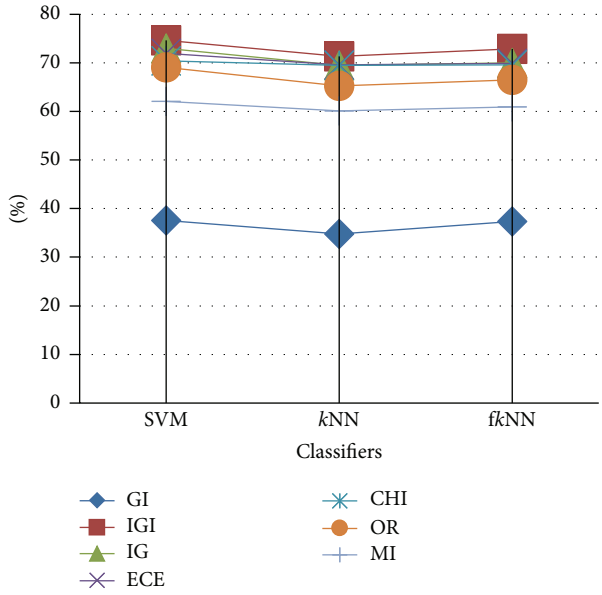


Figure 3: Macro_$F1$ results on Reuters-21578 data set.

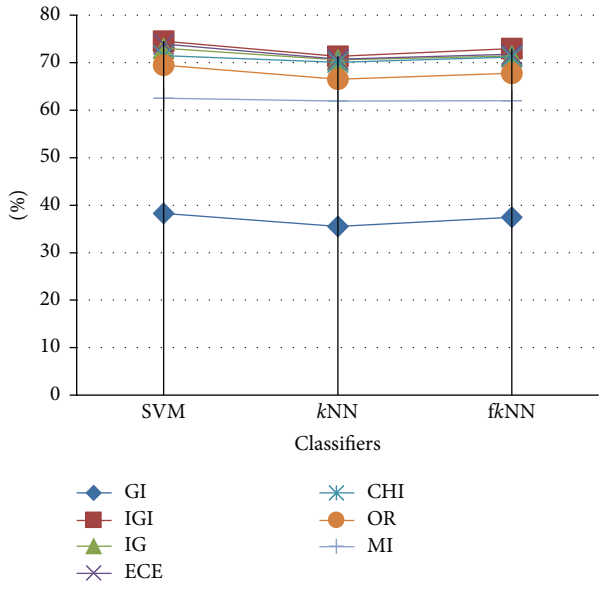


Figure 4: Macro_$F1$ results on Reuters-21578 data set.

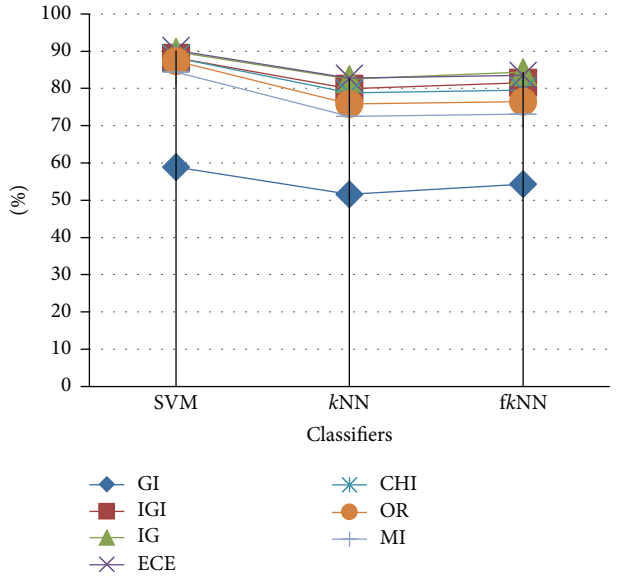


Figure 5: Macro_$F1$ results on Chinese data set.

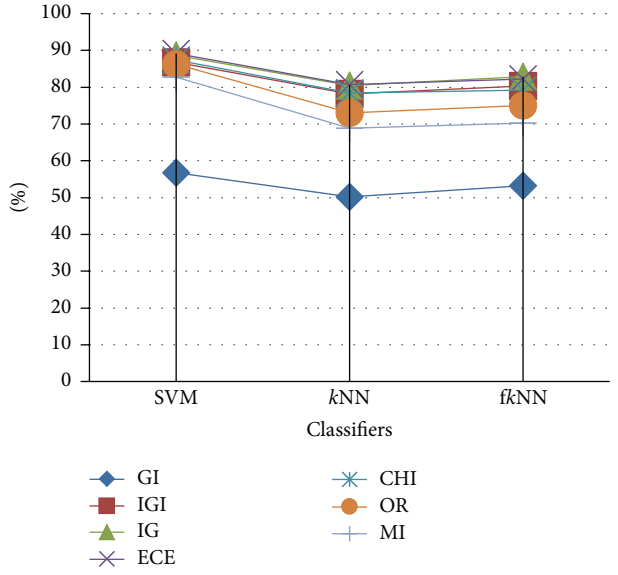


Figure 6: Micro_$F1$ results on Chinese data set.

Stop Words, the results of Chinese word segmentation algorithm will influence the classification algorithm's accuracy. Meanwhile, the current experiments do not consider the feature words' weight. The following TF-Gini algorithm has a good solution to this problem.

Although the IGI algorithm does not have the best performance in the Chinese data set, its performance is very close to the highest. Hence, the IGI feature selection algorithm is suitable and acceptable in the classification algorithm. It promotes the classifiers' performance effectively.

Figures 7, 8, 9, and 10 are results for the English and Chinese date sets. To determine which algorithm's performance is the best, we use different TF weights algorithms, such as TF-IDF and TF-Gini.

From Figures 7 and 8, we can see that the TF-Gini weights algorithm's performance is good and acceptable for the English data set; its performance is very close to the highest. From Figures 9 and 10, we can see that on the Chinese data set the TF-Gini algorithm overcomes the deficiencies of the IGI algorithm and gets the best performance. These results show that the TF-Gini algorithm is a very promising feature weight algorithm.

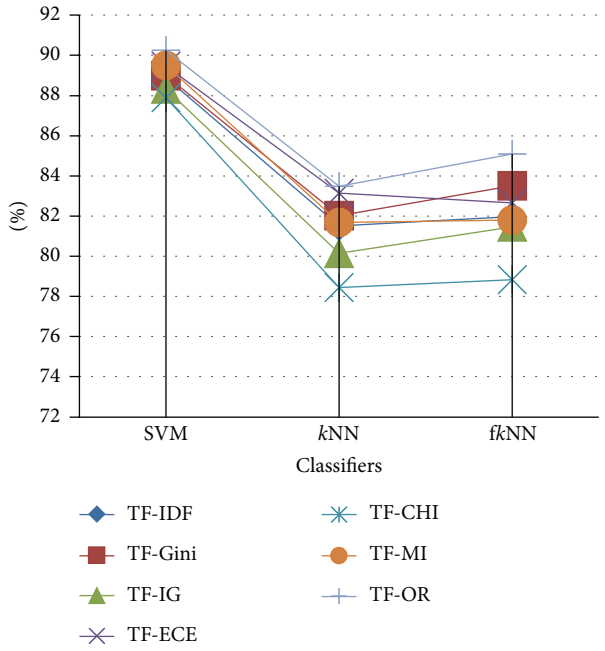


Figure 7: Feature weights Macro_$F1$ results on Reuters-21578 data set.

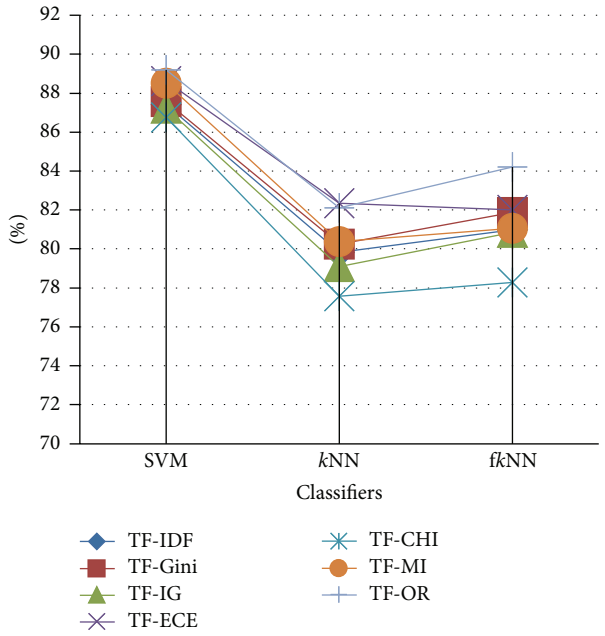


Figure 8: Feature weights Micro_$F1$ results on Reuters-21578 data set.

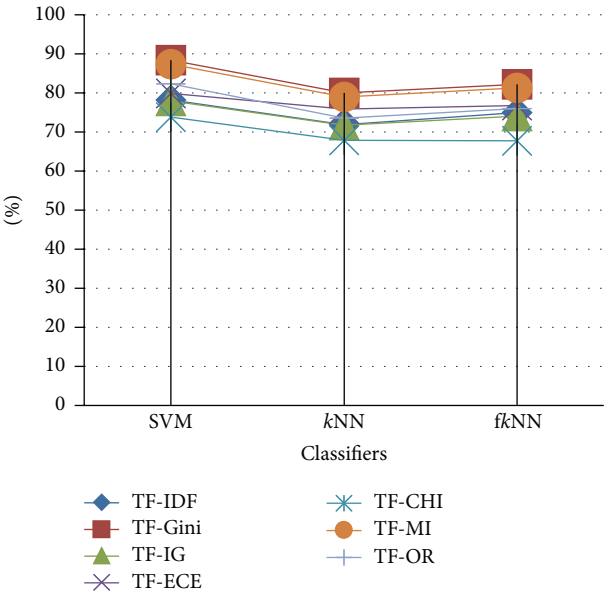


Figure 9: Feature weights Macro_$F1$ results on Chinese data set.

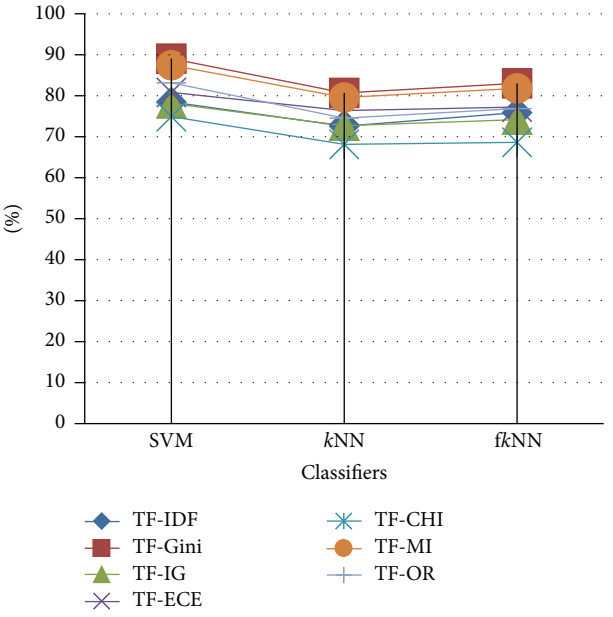


Figure 10: Feature weights Micro_$F1$ results on Chinese data set.

## 4. Sensitive Information Identification System Based on TF-Gini Algorithm

*4.1. Introduction.* With the rapid development of network technologies, the Internet has become a huge repository of information. At the same time, much garbage information is poured into the Internet, such as a virus, violence, and gambling. All these are called sensitive information that will produce extremely adverse impacts and serious consequences.

The commonly used methods of Internet information filtering and monitoring are string matching. There are a large number of synonyms and ambiguous words in natural texts. Meanwhile, a lot of new words and expressions appear on the Internet every day. Therefore, the traditional sensitive information filtering and monitoring are not effective. In this paper, we propose a new sensitive information filtering system based on machine learning. The TF-Gini text feature selection algorithm is used in this system. This system can

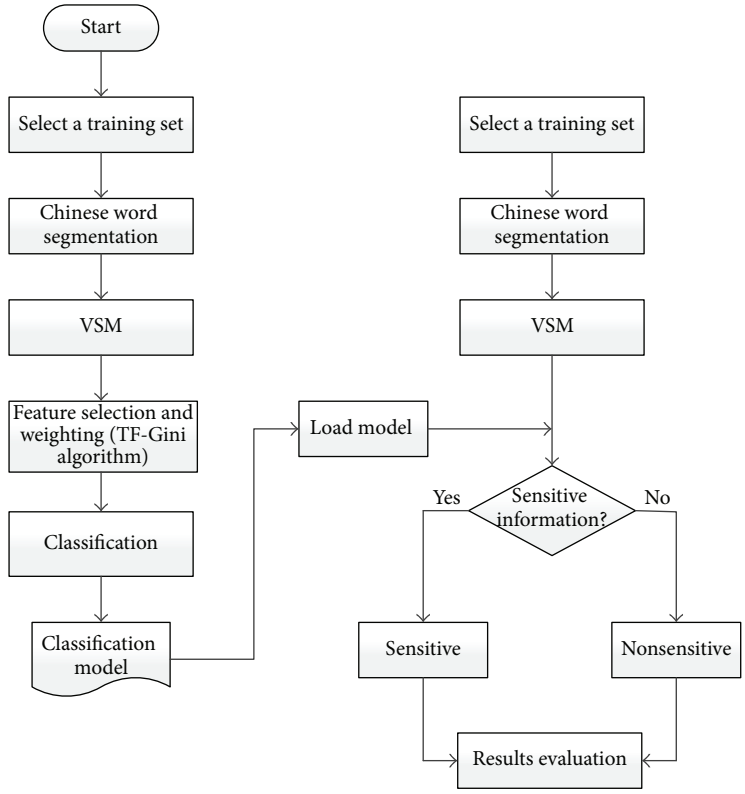


Figure 11: Sensitive system architecture.

discover the sensitive information in a fast, accurate, and intelligent way.

*4.2. System Architecture.* The sensitive system includes the text preprocessing, feature selection, text representation, text classification, and result evaluation modules. Figure 11 shows the sensitive system architecture.

*4.2.1. The Chinese Word Segmentation.* The function of this module is segmentation of Chinese words and getting rid of Stop Words. In this module, we use IKanalyzer to complete Chinese words segmentation. IKanalyzer is a lightweight Chinese words segmentation tool based on an open source package.

(1) IKanalyzer uses a multiprocessor module and supports English letters, Chinese vocabulary, and individual words processing.

(2) It optimizes the dictionary storage and supports the user dictionary definition.

*4.2.2. The Feature Selection.* At the stage of feature selection, the TF-Gini algorithm has a good performance on Chinese data sets. Feature words selection is very important preparation for classification. In this step, we use TF-Gini algorithm as the core algorithm of feature selection.

*4.2.3. The Text Classification.* We use the Naïve Bayes algorithm [30] as the main classifier. In order to verify the performance of the Naïve Bayes algorithm proposed in this paper, we use the $k$NN algorithm and the improved Naïve Bayes [31] algorithm to compare with it.

The $k$NN algorithm is a typical algorithm in text classification and does not need the training data set. In order to calculate the similarity between the test sample $q$ and the training samples $d_j$, we adopt the cosine similarity. It can be described as follows:

$$\cos\left(q, d_j\right) = \frac{\sum_{i=1}^{|V|} w_{ij} \times w_{iq}}{\sqrt{\sum_{i=1}^{|V|} w_{ij}^2} \times \sqrt{\sum_{i=1}^{|V|} w_{iq}^2}}, \tag{32}$$

where $w_{ij}$ is the $i$th feature word's weights in $d_j$ and $w_{iq}$ is the $i$th feature word's weights in sample $q$. $|V|$ is the number of feature spaces.

Naïve Bayes classifier is based on the principle of Bayes. Assume $m$ classes $c_1, c_2, \ldots, c_m$ and a given unknown text $q$ with no class label. The classification algorithm will predict if $q$ belongs to the class with a higher posterior probability in the condition of $q$. It can be described as follows:

$$P\left(c_i \mid q\right) = \frac{P\left(q \mid c_i\right) P\left(c_i\right)}{P\left(q\right)}. \tag{33}$$

Naïve Bayes classification is based on a simple assumption: in a given sample class label, the attributes are conditionally independent. Then, the probability $P(q)$ is constant,

Table 3: The result of Experiment 1.

| Classification | Precision (%) | Recall (%) | $F1$ (%) |
|---|---|---|---|
| $k$NN | 69.2 | 72 | 70.6 |
| Naïve Bayes | 80.77 | 84 | 82.35 |
| Improved Naïve Bayes | 71.88 | 92 | 80.7 |

Table 4: The result of Experiment 2.

| Classification | Precision (%) | Recall (%) | $F1$ (%) |
|---|---|---|---|
| $k$NN | 96.76 | 55.41 | 70.44 |
| Naïve Bayes | 96.6 | 96.6 | 96.6 |
| Improved Naïve Bayes | 83.61 | 100 | 91.08 |

and the probability of $P(c_i)$ is also constant, so the calculation of $P(q \mid c_i)$ can be simplified as follows:

$$P(c_i \mid q) = \prod_{j=1}^{n} P(x_j \mid c_i), \tag{34}$$

where $x_j$ is the $j$th feature word's weights of text $q$ in class $c_i$ and $n$ is the number of features in $q$.

*4.3. Experimental Results and Analysis.* In order to verify the performance of the system, we use two experiments to measure the sensitive information system. We use tenfold cross validation and threefold cross validation in different data sets to verify the system.

*Experiment 1.* The Chinese data set has 1,000 articles, including 500 sensitive texts and 500 nonsensitive texts. The 1,000 texts are randomly divided into 10 portions. One of the portions is selected as test set and the other 9 samples are used as a training set. The process is repeated 10 times. The final result is the average values of 10 classification results. Table 3 is the result of Experiment 1.

*Experiment 2.* The Chinese data set has 1,500 articles, including 750 sensitive texts and 750 nonsensitive texts. All the texts are randomly divided into 3 parts. One of the parts is test set; the others are training sets. The final result is the average values of 3 classification results. Table 4 is the result of Experiment 2.

From the results of Experiments 1 and 2, we can see that all the three classifiers have a better performance. The Naïve Bayes and Improved Naïve Bayes algorithms get a better performance. That is, the sensitive information system could satisfy the demand of the Internet sensitive information recognition.

## 5. Conclusion

Feature selection is an important aspect in text preprocessing. Its performance directly affects the classifiers' accuracy. In this paper, we employ a feature selection algorithm, that is, the IGI algorithm. We compare its performance with IG, CHI, MI, and OR algorithms. The experiment results show that the IGI algorithm has the best performance on the English data sets and is very close to the best performance on the Chinese data sets. Therefore, the IGI algorithm is a promising algorithm in the text preprocessing.

Feature weighting is another important aspect in text preprocessing, since the importance degree of features is different in different categories. We endow weights on different feature words; the more important the feature word is, the bigger the weights the feature word has. According to the TF-IDF theory, we construct a feature weighting algorithm, that is, TF-Gini algorithm. We replace the IDF part of TF-IDF algorithm with the IGI algorithm. To test the performance of TF-Gini, we also construct TF-IG, TF-ECE, TF-CHI, TF-MI, and TF-OR algorithms in the same way. The experiment results show that the TF-Gini performance is the best in the Chinese data sets and is very close to the best performance in the English data sets. Hence, TF-Gini algorithm can improve the classifier's performance, especially in Chinese data sets.

This paper also introduces a sensitive information identification system, which can monitor and filter sensitive information on the Internet. Considering the performance of TF-Gini on Chinese data sets, we choose the algorithm as text preprocessing algorithm in the system. The core classifier which we choose is Naïve Bayes classifier. The experiment results show that the system achieves good performance.

## Competing Interests

The authors declare that they have no competing interests.

## Acknowledgments

This paper is partly supported by the engineering planning project Research on Technology of Big Data-Based High Efficient Classification of News funded by Communication University of China (3132015XNG1504), partly supported by The Comprehensive Reform Project of Computer Science and Technology funded by Ministry of Education of China (ZL140103 and ZL1503), and partly supported by The Excellent Young Teachers Training Project funded by Communication University of China (YXJS201508 and XNG1526).

## References

[1] S. Jinshu, Z. Bofeng, and X. Xin, "Advances in machine learning based text categorization," *Journal of Software*, vol. 17, no. 9, pp. 1848–1859, 2006.

[2] G. Salton and C. Yang, "On the specification of term values in automatic indexing," *Journal of Documentation*, vol. 29, no. 4, pp. 351–372, 1973.

[3] Z. Cheng, L. Qing, and L. Fujun, "Improved VSM algorithm and its application in FAQ," *Computer Engineering*, vol. 38, no. 17, pp. 201–204, 2012.

[4] X. Junling, Z. Yuming, C. Lin, and X. Baowen, "An unsupervised feature selection approach based on mutual information," *Journal of Computer Research and Development*, vol. 49, no. 2, pp. 372–382, 2012.

[5] Z. Zhenhai, L. Shining, and L. Zhigang, "Multi-label feature selection algorithm based on information entropy," *Journal of*

*Computer Research and Development*, vol. 50, no. 6, pp. 1177–1184, 2013.

[6] L. Kousu and S. Caiqing, "Research on feature-selection in Chinese text classification," *Computer Simulation*, vol. 24, no. 3, pp. 289–291, 2007.

[7] Y. Yang and J. Q. Pedersen, "A comparative study on feature selection in text categorization," in *Proceedings of the 14th International Conference on Machine Learning*, pp. 412–420, Nashville, Tenn, USA, July 1997.

[8] Q. Liqing, Z. Ruyi, Z. Gang et al., "An extensive empirical study of feature selection for text categorization," in *Proceedings of the 7th IEEE/ACIS International Conference on Computer and Information Science*, pp. 312–315, IEEE, Washington, DC, USA, May 2008.

[9] S. Wenqian, D. Hongbin, Z. Haibin, and W. Yongbin, "A novel feature weight algorithm for text categorization," in *Proceedings of the International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE '08)*, pp. 1–7, Beijing, China, October 2008.

[10] L. Yongmin and Z. Weidong, "Using Gini-index for feature selection in text categorization," *Computer Applications*, vol. 27, no. 10, pp. 66–69, 2007.

[11] L. Xuejun, "Design and research of decision tree classification algorithm based on minimum Gini index," *Software Guide*, vol. 8, no. 5, pp. 56–57, 2009.

[12] R. Guofeng, L. Dehua, and P. Ying, "An improved algorithm for feature weight based on Gini index," *Computer & Digital Engineering*, vol. 38, no. 12, pp. 8–13, 2010.

[13] L. Yongmin, L. Zhengyu, and Z. Shuang, "Analysis and improvement of feature weighting method TF-IDF in text categorization," *Computer Engineering and Design*, vol. 29, no. 11, pp. 2923–2925, 2929, 2008.

[14] Q. Shian and L. Fayun, "Improved TF-IDF method in text classification," *New Technology of Library and Information Service*, vol. 238, no. 10, pp. 27–30, 2013.

[15] L. Yonghe and L. Yanfeng, "Improvement of text feature weighting method based on TF-IDF algorithm," *Library and Information Service*, vol. 57, no. 3, pp. 90–95, 2013.

[16] Y. Xu, Z. Li, and J. Chen, "Parallel recognition of illegal Web pages based on improved KNN classification algorithm," *Journal of Computer Applications*, vol. 33, no. 12, pp. 3368–3371, 2013.

[17] Y. Liu, Y. Jian, and J. Liping, "An adaptive large margin nearest neighbor classification algorithm," *Journal of Computer Research and Development*, vol. 50, no. 11, pp. 2269–2277, 2013.

[18] M. Wan, G. Yang, Z. Lai, and Z. Jin, "Local discriminant embedding with applications to face recognition," *IET Computer Vision*, vol. 5, no. 5, pp. 301–308, 2011.

[19] U. Maulik and D. Chakraborty, "Fuzzy preference based feature selection and semi-supervised SVM for cancer classification," *IEEE Transactions on Nanobioscience*, vol. 13, no. 2, pp. 152–160, 2014.

[20] X. Liang, "An effective method of pruning support vector machine classifiers," *IEEE Transactions on Neural Networks*, vol. 21, no. 1, pp. 26–38, 2010.

[21] P. Baomao and S. Haoshan, "Research on improved algorithm for Chinese word segmentation based on Markov chain," in *Proceedings of the 5th International Conference on Information Assurance and Security (IAS '09)*, pp. 236–238, Xi'an, China, September 2009.

[22] S. Yan, C. Dongfeng, Z. Guiping, and Z. Hai, "Approach to Chinese word segmentation based on character-word joint decoding," *Journal of Software*, vol. 20, no. 9, pp. 2366–2375, 2009.

[23] J. Savoy, "A stemming procedure and stopword list for general French corpora," *Journal of the American Society for Information Science*, vol. 50, no. 10, pp. 944–952, 1999.

[24] L. Rutkowski, L. Pietruczuk, P. Duda, and M. Jaworski, "Decision trees for mining data streams based on the McDiarmid's bound," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1272–1279, 2013.

[25] N. Prasad, P. Kumar, and M. M. Naidu, "An approach to prediction of precipitation using gini index in SLIQ decision tree," in *Proceedings of the 4th International Conference on Intelligent Systems, Modelling and Simulation (ISMS '13)*, pp. 56–60, Bangkok, Thailand, January 2013.

[26] S. S. Sundhari, "A knowledge discovery using decision tree by Gini coefficient," in *Proceedings of the International Conference on Business, Engineering and Industrial Applications (ICBEIA '11)*, pp. 232–235, IEEE, Kuala Lumpur, Malaysia, June 2011.

[27] S. V. Stehman, "Selecting and interpreting measures of thematic classification accuracy," *Remote Sensing of Environment*, vol. 62, no. 1, pp. 77–89, 1997.

[28] F. Guohe, "Review of performance of text classification," *Journal of Intelligence*, vol. 30, no. 8, pp. 66–69, 2011.

[29] C. J. Van Rijsbergen, *Information Retrieval*, Butter-Worths, London, UK, 1979.

[30] L. Jianghao, Y. Aimin, Y. Yongmei et al., "Classification of microblog sentiment based on Naïve Byaesian," *Computer Engineering & Science*, vol. 34, no. 9, pp. 160–165, 2012.

[31] Z. Kenan, Y. Baolin, M. Yaming, and H. Yingnan, "Malware classification approach based on valid window and Naïve bayes," *Journal of Computer Research and Development*, vol. 51, no. 2, pp. 373–381, 2014.