Supervised and Traditional Term Weighting Methods for Automatic Text Categorization

Man Lan, Chew Lim Tan *Senior Member, IEEE*, Jian Su

Manuscript received July 25, 2007; revised September 25, 2007.

Abstract

In vector space model (VSM), text representation is the task of transforming the content of a textual document into a vector in the term space so that the document could be recognized and classified by a computer or a classifier. Different terms (i.e. words, phrases, or any other indexing units used to identify the contents of a text) have different importance in a text. The term weighting methods assign appropriate weights to the terms to improve the performance of text categorization. In this study, we investigate several widely-used unsupervised (traditional) and supervised term weighting methods on benchmark data collections in combination with SVM and *k*NN algorithms. In consideration of the distribution of relevant documents in the collection, we propose a new simple supervised term weighting method, i.e. $tf.rf$, to improve the terms' discriminating power for text categorization task. From the controlled experimental results, these supervised term weighting methods have mixed performance. Specifically, our proposed supervised term weighting method, $tf.rf$, has a consistently better performance than other term weighting methods while other supervised term weighting methods based on information theory or statistical metric perform the worst in all experiments. On the other hand, the popularly used $tf.idf$ method has not shown a uniformly good performance in terms of different data sets.

Index Terms

text categorization, text representation, term weighting, SVM, *k*NN

I. INTRODUCTION

Text categorization (TC) is the task of automatically classifying unlabelled natural language documents into a predefined set of semantic categories. As the first and a vital step, text representation converts the content of a textual document into a compact format so that the document can be recognized and classified by a computer or a classifier. In the vector space model (VSM), the content of a document is represented as a vector in the term space, i.e. $d = (w_1, ..., w_k)$, where k is the term (*feature*) set size. Terms can be at various levels, such as syllables, words, phrases, or any other complicated semantic and/or syntactic indexing units used to identify the contents of a text. Different terms have different importance in a text, thus an important indicator w_i (usually between 0 and 1) represents how much the term t_i contributes to the semantics of document d. The term weighting method is such an important step to improve the effectiveness of TC by assigning appropriate weights to terms. Although TC has been intensively studied for several decades, the term weighting methods for TC are usually borrowed from the

traditional information retrieval (IR) field, for example, the simplest *binary* representation, the most famous $tf.idf$ and its various variants.

Recently, the study of term weighting methods for TC has gained increasing attention. In contrast to IR, TC is a *supervised* learning task as it makes use of prior information on the membership of training documents in predefined categories. This known information is effective and has been widely used for the feature selection [1] and the construction of text classifier to improve the performance of the system. In this study, we group the term weighting methods into two categories according to whether the method involves this prior information, i.e. *supervised term weighting method* (if it uses this known membership information) and *unsupervised term weighting method* (if it does not use this information). For example, the traditional term weighting methods borrowed from IR, such as $binary$, tf (term frequency), $tf.idf$ and its various variants, belong to the unsupervised term weighting methods as the calculation of these weighting methods do not make use of the information on the category membership of training documents.

Generally, the supervised term weighting methods adopt this known information in several ways. One approach is to weight terms by using feature selection metrics, such as χ^2, *information gain*, *gain ratio*, *odds ratio* and so on (see [2] and [3]). Since these functions are effective in feature selection, they are naturally thought to be of great help to assign appropriate weights to terms in text categorization. Another approach is based on statistical confidence intervals [4] which rely on the prior knowledge of the statistical information in the labelled training data. Yet there is another approach that combines the term weighting method with a text classifier [5]. Similar to the idea of using feature selection scores, the scores used by the text classifier aim to distinguish the positive documents from negative documents are believed to be effective in assigning more appropriate weights to the terms. These approaches will be further discussed in subsection II-C.

Since these supervised term weighting methods take the document distribution into consideration, they are naturally expected to be superior to the unsupervised (traditional) term weighting methods. However, not much work has been done on their comprehensive comparison with unsupervised term weighting methods. Although there are partial comparisons in [3] and [4], these supervised term weighting methods have been shown to have mixed results. On the other hand, another work [2] has drawn conclusions quite contrary to our previous findings [6]. Therefore, the first fundamental question arises here, i.e. “Are supervised term weighting

methods based on known information able to lead to better performance than unsupervised ones for text categorization?"

At the same time, Leopold [7] pointed out that it is the text representation schemes which dominate the performance of text categorization rather than the kernel functions of SVM. That is, choosing an appropriate term weighting method is more important than choosing and tuning kernel functions of SVM for text categorization. Thus, a second question surfaces here : "Does the difference between supervised and unsupervised term weighting methods have any relationship with different learning algorithms?"

In addition, since the prior membership information of the training samples is quite helpful, we analyze the terms' discriminating power based on this known information. As a result, we have also proposed a new supervised term weighting method, i.e. $tf.rf$, which first appeared in [6] and [8] and recently in [9]. While our previous work in [6], [8] and [9] provided experimental evidence for the effectiveness of $tf.rf$ under particular experimental circumstances, we will now formally address how and why this new method is proposed using analytical explanation and empirical observation. Thus, a third question emerges here: " Why is the new supervised method, i.e. $tf.rf$, effective for text categorization?". Moreover, to further validate the effectiveness of $tf.rf$ and compare with others published results, we also replicate the experiments in [10].

Therefore, the purpose of this study is to address the above three questions. To address the first two questions, we compare various widely-used traditional and supervised term weighting methods (including our $tf.rf$) on two popular benchmark data corpora, i.e. Reuters-21578 and 20 Newsgroups, using SVM and *k*NN. Although these experiments provide evidence to validate the effectiveness of $tf.rf$, in order to make the comparison of our results with the published results meaningful, we also duplicate the experiments under almost the same circumstances as [10]. Regarding the third question, to understand the new method $tf.rf$ well, it is the first time that we give a detailed analysis to address how the new method is proposed and explained.

The rest of this paper is structured as follows. Section II reviews the popular traditional term weighting methods and the state-of-the-art supervised term weighting methods. Section III analyzes the terms' discriminating power and explains the newly proposed supervised term weighting method. Section IV introduces the experimental methodology. Section V reports experimental results and discussions. Section VI concludes this paper with future work.

II. TERM WEIGHTING METHODS: A BRIEF REVIEW

In text representation, terms are words, phrases, or any other indexing units used to identify the contents of a text. However, no matter which indexing unit in use, each term in a document vector must be associated with a value (weight) which measures the importance of this term and denotes how much this term contributes to the categorization task of the document. In this section, we review a number of traditional term weighting methods and the-state-of-art supervised term weighting methods.

A. Three Factors for Term Weighting Assignment

Salton [11] discussed three considerations of the assignment of appropriately weighted single term in IR field. First, term occurrences (*tf*) in one document appear to closely represent the content of the document. Second, term frequency alone may not have the discriminating power to pick up all the relevant documents from other irrelevant documents. Therefore, an *idf* factor has been proposed to increase the term's discriminating power for IR purpose. In general, the two factors, *tf* and *idf*, are combined by a multiplication operation and are thought to improve both *recall* and *precision* measures. Third, to take the effect of documents length into consideration, a *cosine* normalization factor is incorporated to equalize the length of the documents.

1) Term Frequency Factor: Table I summaries four commonly-used term frequency factors, including a binary weight, a normal raw term frequency, a logarithm of term frequency and an inverse term frequency. The simplest *binary* representation (i.e. 1 for presence and 0 for absence) ignores the occurrences of the term in the document and has been used in all learning algorithms, especially in Naive Bayes and decision tree where the real number format of term

TABLE I

TERM FREQUENCY FACTORS

Term Frequency Factor	Denoted by	Description
1.0	*binary*	Binary weight equal to 1 for terms present in a vector
term frequency alone	tf	Raw term frequency (number of times a term occurs in a document)
$log(1+tf)$	$\log tf$	Logarithm of the term frequency
$1-\frac{r}{r+tf}$	ITF	Inverse term frequency, usually $r=1$

TABLE II

COLLECTION FREQUENCY FACTORS

Collection Frequency Factor	Denoted by	Description
1.0	1.0	Use original term frequency factor
$log(\frac{N}{n_i})$	idf	Multiply tf by an inverse document frequency (idf) factor
$log(\frac{N-n_i}{n_i})$	*idf_prob*	Multiply tf by a *term relevance*, i.e. probabilistic idf
Chi-square	χ^2	Multiply tf by a χ^2 function
information gain	ig	Multiply tf by a *information gain* function
gain ratio	gr	Multiply tf by a *gain ratio* function
Odds Ratio	or	Multiply tf by a *Odds Ratio* function

weights cannot be used[1]. The most popular term frequency representation only adopts the raw term frequency (*tf*) in the document. Moreover, different variants of term frequency have been presented, for example $\log(1+tf)$ (see [13]), where the logarithmic operation is used to scale the effect of unfavorably high term frequency in one document. Inspired by the inverse document frequency, ITF (*inverse term frequency*) was presented by [7].

The term frequency alone factors could be used as term weights without other factors. In our previous study [6], we have investigated three term frequency alone methods, i.e. tf, $\log(1+tf)$ and ITF, using linear SVM in terms of micro-averaged break-even point measure on two benchmark data collections. The results showed that there is no significant difference among them. The most possible reason may be that although they are different transformations, they are derived from the same base – term occurrences, and thus there is no fundamental difference among them. As a result, we only include the *tf* alone and the *binary* method as representatives for further research in this study.

2) Collection Frequency Factor: Table II describes several different collection frequency factors, namely, the multipliers of 1 (that ignores the collection frequency factor), a conventional inverse collection frequency factor (idf), a probabilistic inverse collection frequency (*idf-prob*), a χ^2 factor (χ^2), an *information gain* factor (ig), a *gain ratio* factor (gr) and an *Odds Ratio* factor (or), respectively.

[1] [12] used a multinomial event model for Naive Bayes text classification which can relax the constraint that document vectors should be binary-valued.

Since term frequency alone may not have the discriminating power to pick up all relevant documents from other irrelevant documents, an *idf* (inverse document frequency) factor which takes the collection distribution into account has been proposed to help to improve the performance of IR. The *idf* factor varies inversely with the number of documents n_i which contain the term t_i in a collection of N documents and is typically computed as $log(N/n_i)$.

In the classical probabilistic model for IR, the relevance properties of the documents are considered and thus a *term relevance* weight is derived as the proportion of relevant documents in which a term occurs divided by the proportion of non-relevant documents in which the term occurs. Due to the lack of knowledge of the occurrences of the terms in the relevant and non-relevant documents in IR, this *term relevance* can be reduced to an inverse document frequency factor of the form $log((N - n_i)/n_i)$ [14] under an approximate circumstance. We will call this variant factor as *idf_prob* (i.e. probabilistic *idf*) because it was derived from the probabilistic model and has an *idf* form [14]. However, in TC, the *term relevance* factor can be estimated based on the available training data set in advance and actually it is a well-known statistical measure, i.e. *Odds Ratio*.

In our previous study [6], we have also investigated the term weighting methods related to idf factor, i.e. $tf.idf$, $\log(1+tf).idf$, *tf.idf_prob* and idf alone. The experimental results indicated that when compared with the term frequency factor alone, the idf factor gives no or even decreases the discriminating power of the terms when combined with the term frequency and there is no significant difference among them. Consequently, we adopt the most popular $tf.idf$ method as a baseline in this study.

In addition, some researchers have explored the term weighting methods by replacing the *idf* factor with the metrics that have been used in feature selection, such as *information gain*, *gain ratio*, χ^2 and *Odds Ratio* (see [2] and [3]). In contrast to the standard idf factor, these factors are supervised factors because they require known information on the category membership of the training documents before the weights can be accurately estimated.

3) Normalization Factor: To eliminate the length effect, we use the *cosine* normalization to limit the term weight range within $(0, 1)$. Specifically, the *binary* feature representation does not use any normalization since the original value is 0 or 1. Assuming that w_{ij} represents the weight of term t_i in document d_j, the final term weight w_{ij} may then be defined as $\frac{w_{ij}}{\sqrt{\sum_i (w_{ij}^2)}}$.

B. Traditional Term Weighting Methods

As we mentioned before, the traditional term weighting methods for TC are usually borrowed from IR and belong to the unsupervised term weighting methods. The simplest one is *binary* representation. The most popular one is $tf.idf$ proposed by Jones (first appearing in [15] and reprinted in [16]). Note that the tf here also has various variants, such as raw term frequency, $log(tf)$, $log(tf+1)$ or $log(tf)+1$. Besides, the idf factor (usually computed as $log(N/n_i)$) also has a number of variants, such as $log(N/n_i+1)$, $log(N/n_i)+1$ and $log(N/n_i-1)$ (i.e. *idf_prob*), etc.

Interestingly, $tf.idf$ has a variant known as BM25 (see [17]) that appears in IR literature. To understand BM25, let us first take a look at RSJ (**R**obertson and **S**parck **J**ones). RSJ (see [18]) is also known as the relevance weighting from the classical probabilistic model for IR. However, due to the unavailability of relevance information in advance (otherwise it makes a nonsense of the search process), the final term weight of term t_i is further simplified based on some reasonable assumptions (for details see [17]) leading to

$$\text{RSJ weight } w_i \;=\; log\frac{N}{n_i} \tag{1}$$

It is now apparent that we can regard idf as a simple version of the RSJ weight, applicable when we have no relevance information.

The BM25 weighting function is based on an analysis of the behavior of the full eliteness model under different values of the parameters. With respect to term t_i, its BM25 wight formula can be expressed as:

$$\text{BM25 weight } w_i \;=\; f(tf_i) * w_i^{(1)} \tag{2}$$

where $w_i^{(1)}$ is the usual RSJ weight, $f(tf_i) = \frac{(k_1+1)tf_i}{K+tf_i}$ (K and k_1 are global parameters which are in general unknown, but may be tuned on the basis of evaluation data; tf_i is the frequency of term t_i in a document). It is clear that the formula (2) expresses BM25 as a $tf*idf$ weighting scheme. The first component is a tf component and the second component is the RSJ weight, which reduces to an idf measure as discussed earlier (see [17]).

In this study, we use the typical format of $tf.idf$ as a multiplication of raw term frequency (tf) and $\log(N/n_i)$. There may be other variants which we do not cover in this study because they share the same idea of $tf.idf$ and their formats are basically similar to each other.

C. Supervised Term Weighting Methods

Text categorization is a supervised learning task because the category labels of training documents are available in advance. Generally, this known information has been used in the supervised term weighting methods in the following ways.

1) Combined with Information Theory Functions or Statistic Metrics: One approach is to weight terms by adopting feature selection metrics, such as χ^2, *information gain*, *gain ratio*, *Odds Ratio*, etc. The main purpose of feature selection is to reduce the high dimensionality of the term space by selecting the most relevant and discriminating features for the classification task. The terms with higher feature selection scores are deemed to have contributed more to the text categorization than those with lower scores.

For example, in [2], Deng et al replaced the idf factor with χ^2 factor to weight term and asserted that $tf.\chi^2$ is more effective than $tf.idf$ in their experiments with a SVM-based text categorization. Similarly, Debole [3] assigned weights to terms by replacing the idf factor with the metrics that have been used for feature selection process, namely, *information gain*, χ^2 and *gain ratio* and also called them "*supervised term weighting*". The difference between the two studies is that [2] used different metrics for feature selection and term weighting while [3] adopted the same metric for feature selection and term weighting. In [3] these supervised term weighting methods have not been shown to have a consistent superiority over the standard $tf.idf$-based term weighting. In most cases, the $tf.idf$ method is better than these complicated supervised term weighting approaches. Specifically, the observation in [3] that $tf.idf$ always outperforms $tf.\chi^2$ contradicts the conclusion in [2].

Besides the text categorization task, the idea of using feature selection metrics as term weighting methods has been adopted in other text mining tasks. For example, in document summarization, Mori adopted *gain ratio* as a term weighting method to compare with a *tf*-based summarization system and the result showed that this *gr*-based term weighting method is very effective in summarization [19].

2) Based on Statistical Confidence Intervals: In [4], the authors introduced a new term weighting method called `ConfWeight` based on statistical confidence intervals. They estimate the proportion of documents containing term t_k to be a Wilson proportion estimate $\tilde{p}$. For a given category, we can get $\tilde{p_+}$ and $\tilde{p_-}$ by applying $\tilde{p}$ to the positive and negative category in the training data set respectively. Then, they label *MinPos* and *MaxNeg* for the lower and higher

limits of the 95% confidence interval of $\tilde{p_+}$ and $\tilde{p_-}$ respectively. Now the strength of term t_i for category c_k is defined as:

$$str_{t_i,c_k} = \begin{cases} log_2(2 - \frac{MinPos}{MinPos+MaxNeg}) & \text{if } \textit{MinPos} > \textit{MaxNeg} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Then by using a global policy technique, the final ConfWeight of term t_i in document d_j is defined as:

$$ConfWeight_{t_i,d_j} = log(tf + 1) * \max_{c \in c_k}(str^2_{t_i,c}) \quad (4)$$

The first component in Equation (4) is actually a logarithmic operation of term frequency. The second component of `ConfWeight` determines the weight of a particular term. Thus, though the authors assigned weight to terms based on statistical confidence, to calculate the `ConfWeight` value for each term with respect to each category c_i, the prior knowledge of labelled training data set is required. The experimental results showed that *ConfWeight* generally outperformed $tf.idf$ and *gain ratio* on three benchmark data collections. Furthermore, these experiments failed to show that supervised weighting methods are generally higher than unsupervised ones.

3) Interaction with Text Classifier: This approach is to weight terms in the interaction with a text classifier. Similar to the idea of using feature selection scores, since the text classifier selects the positive test documents from negative test documents by assigning different scores to the test samples, these scores are believed to be effective in assigning more appropriate weights to the terms.

For example, in [5], terms are weighted using an iterative approach involving the *k*NN text classifier at each step. For each iteration, the weights are slightly modified and the categorization accuracy is measured using an evaluation set (a split from the training set). Convergence of weights should provide an optimal set of weights. However, this method is generally much too slow to be used, particularly for large problems (involving a huge vocabulary).

III. A New Supervised Term Weighting Method

A. Terms' Discriminating Power Analysis

For multi-label classification problem, the benchmark on each corpus is simplified into multiple independent binary classification problems. That is, in each experiment, a chosen category is tagged as the positive category and the other categories in the same corpus are combined together

as the negative category. Therefore all collection frequency factors are specified "locally" to a specific positive category c_i. We adopt this local policy to assign term weights in this study.

Term frequency represents a close relationship between the term and the content of the documents which contain this term. It is observed that if high frequency terms are spread widely over a large number of documents, we may not retrieve the relevant documents from the whole collection. Consequently, the traditional idf factor and its variants are introduced to improve the discriminating power of terms in IR. However, in TC, it may not be the case. To illustrate the difference between them, we take Figure 1 for example to analyze the terms' discriminating power to TC.

Figure 1 depicts the distributions of documents which contain six terms, t_1, t_2, t_3, t_4, t_5 and t_6, given one chosen positive category on one data collection. In this figure, each column represents

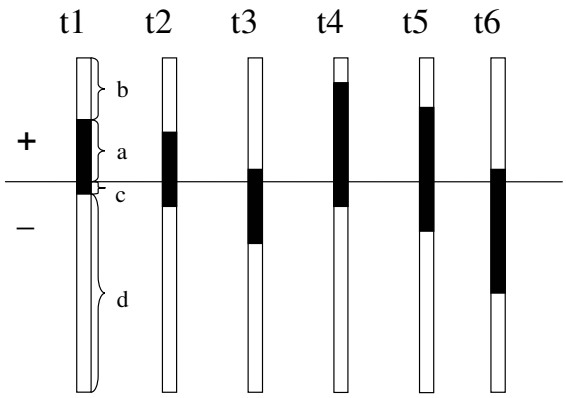

Fig. 1. Examples of different distributions of documents which contain six terms in the whole collection

the documents distribution in the corpus for each term. The height of one column is the number of documents in the corpus. The horizontal line divides these documents into two categories, the positive (above) and the negative (below). The heights of the columns above and below the horizontal line denote the number of documents in the positive and negative categories respectively. The height of the shaded part is the number of documents which contain this term. We use a, b, c and d to denote the number of different documents as below:

- a is the number of documents in the positive category which contain this term
- b is the number of documents in the positive category which do not contain this term

- c is the number of documents in the negative category which contain this term
- d is the number of documents in the negative category which do not contain this term

Thus, N, the number of documents in the whole collection, is the sum of a, b, c and d. In general, $d >> a, b, c$.

By using these notations, several widely-used collection frequency factors and information theory functions are represented as follows:

$$idf = log\frac{N}{a+c} \tag{5}$$

$$\textit{idf\_prob} = log\frac{b+d}{a+c} \tag{6}$$

$$\chi^2 = N * \frac{(a*d-b*c)^2}{(a+c)(b+d)(a+b)(c+d)} \tag{7}$$

$$\begin{aligned} ig = & \frac{a}{N} * log\frac{a*N}{(a+c)*(a+b)} + \frac{b}{N} * log\frac{b*N}{(b+d)*(a+b)} + \\ & \frac{c}{N} * log\frac{c*N}{(a+c)*(c+d)} + \frac{d}{N} * log\frac{d*N}{(b+d)*(c+d)} \end{aligned} \tag{8}$$

$$gr = \frac{ig}{-\frac{(a+b)}{N} * \log\frac{(a+b)}{N} - \frac{(c+d)}{N} * \log\frac{(c+d)}{N}} \tag{9}$$

$$or = \frac{a*d}{b*c} \tag{10}$$

We assume the six terms have the same term frequency (tf). The first three terms, i.e. t_1, t_2 and t_3, have the same idf, which we will call $idf1$, and the last three terms, i.e. t_4, t_5 and t_6, share the same idf, which we will call $idf2$. It is clear to find that $idf1 > idf2$ as the traditional idf factor gives more weights to the first three terms with less document frequency in the corpus than the last three terms.

The membership of a document in a category in TC is pretty much as the relevance of a document to an information need in IR [20]. In IR, the relevance information cannot be available in advance (otherwise it makes a nonsense of the search process), while in TC, the labelled training data set is available. Considering the examples in Figure 1, in IR we cannot know where this horizontal line is. However in TC, due to the availability of the training data set, we can draw out this line. This difference between IR and TC comes from the availability of training data sets.

It is quite clearly observed that these six terms contribute differently to the classification task. Intuitively, it is considered that the more concentrated the terms are in the positive category than

in the negative category, the more contribution they may make to separate the positive samples from the negative examples. Thus, in the above examples in Figure 1, term $t1$ contributes more than $t2$ and $t3$ to distinguish the positive from the other negative categories. Similarly, term $t4$ contributes more than term $t5$ and $t6$.

However, by using idf and *idf_prob* (listed in Equation 6) factors, we cannot see any difference among the first three terms. The similar observation can be found among the last three terms. Let us discuss the following two cases in which the idf and *idf_prob* factors fail to show their discriminating power in TC.

case 1 : Consider the first three terms, i.e. $t1$, $t2$ and $t3$. We can easily find that these three terms make different contributions to TC. Specifically, $t1$ contributes more power to discriminate the positive documents from the negative documents than $t2$ and $t3$. Thus, $t1$ should be weighted more than $t2$ and $t3$. However, by using idf or *idf_prob* factor, the three terms will equalize their discriminating power for text categorization. The same observation can be found for the last three terms. Therefore, in this case, the idf and *idf_prob* factors fail to assign appropriate weights to terms according to their different contributions to text classification tasks.

case 2 : Consider the two terms $t1$ and $t4$. Note the two have different idf values and $idf2$ ($t4$) $< idf1$ ($t1$). However, it is clearly observed from this figure that $t4$ contributes more than $t1$ and thus $t4$ should be weighted more than $t1$. Thus, in this case, the idf and *idf_prob* factors have reverse effects on expressing the terms' discriminating power for text categorization.

In the above two cases, the idf and *idf_prob* factors have no or even decrease the discriminating power for terms in TC. An obvious explanation for the failure of idf and *idf_prob* in TC is that they do not take the document distribution in the positive and negative category into consideration.

Consequently, the other supervised factors listed from Equations (7) to (10), i.e. χ^2, *Odds Ratio*, *information gain* and *gain ratio*, should, in theory, outperform idf and *idf_prob* as they make use of this prior information from the training data set. The basic idea of using these functions is that the best terms for category c_k are the ones distributed most differently in the positive and negative examples of c_k. However, these functions may have different interpretations of this principle. For instance, in the experimental sciences χ^2 is used to measure how the results of an observation differ (i.e. are independent) from the results expected according to an initial

hypothesis (here, the lower values means lower dependence). In TC, it is used to measure how independent term t_i and category c_k are. The term t_i with the lowest value for $\chi^2(t_i, c_k)$ are thus the most independent from c_k; since we are interested in the terms which are not, the terms for which $\chi^2(t_i, c_k)$ is highest are considered to contribute the most for classification task.

To discuss the difference between the above supervised factors and idf, we take the two factors χ^2 and or for example in the following cases.

case 3 : Consider the first three terms in Figure 1 again. The three are weighted differently in terms of idf, χ^2 and or factors. Specifically, in idf value, $t_1 = t_2 = t_3$; in χ^2 value, $t_1 = t_3 > t_2$; in or value, $t_1 > t_2 > t_3$. This observation indicates that the three show inconsistent discriminating power with respect to each other. Moreover, even though χ^2 and or both consider the document distribution in the positive and negative category, the two assign inconsistent weights for the same term. This difference arises from their different theoretical bases or rationales.

case 4 : Consider the two terms t_1 and t_4. Although the idf factor unreasonably results in $t_1 > t_4$, χ^2 and or correct this obvious mistake and appropriately makes $t_4 > t_1$. This case shows that χ^2 and or would assign more appropriate weights to terms than idf.

case 5 : Consider the two terms t_2 and t_5. Let $a = c$ for each term (i.e. the same number of documents which contain this term in the positive and negative category). Clearly, $t_2 > t_5$ in idf value while $t_5 > t_2$ in both χ^2 and or values. This is inconsistent with our intuitive consideration that t_2 and t_5 should contribute equally because each term occurs with the same frequency in the positive category as that in the negative and thus they both have the same discriminating power.

Apparently, these supervised factors might assign more reasonable weights to terms than idf and *idf_prob* as they consider the distribution of the documents in the positive and negative category from the training data set. However, we must note that in this section this is only a a qualitative analysis of terms' discriminating power. These supervised factors might show various degrees of discriminating power when weighting terms in the real world cases. To illustrate their different discriminating power from quantitative analysis aspect, we will take some real case examples in Section III-C.

B. A New Supervised Term Weighting Scheme – tf.rf

The basic idea of our intuitive consideration is quite simple: the more concentrated a high frequency term is in the positive category than in the negative category, the more contributions it makes in selecting the positive samples from the negative samples.

Although the above supervised term weighting factors take the document distribution into account, they are not always consistent with the above intuitive consideration. Specifically, several supervised term weighting factors discussed in the previous subsection are symmetric in terms of positive and negative categories. That is, given one category c_k, if the distributions of term t_1 in the positive and negative categories are the same as those of term t_3 in the negative and positive categories respectively, they will be weighted equally in χ^2, ig and gr value. But, as we mentioned in the previous subsection, our intuitive consideration is that t_1 contributes more than t_3 and thus t_1 should be weighted more than t_3. Why we would like to assign more weights to t_1 rather than t_3? For the multilabel multiclass TC task, when we build a text classifier for each category, we usually label this category as positive category and group all the other categories as negative category. Thus, the documents in the positive category concentrate on one topic or several topics close to each other while the documents in the negative category are spread over a wide range of topics since all non-positive categories are grouped together as negative. Thus, the high frequency terms concentrated in the positive category are good discriminators to select the positive samples from among the various negative samples. This idea of favoring positive terms can also be observed in [23] which tried to select such "positive" words via a metric (the square root of chi-square) as opposed to "negative" words indicative of negative documents. Therefore, unlike the above three supervised factors, we would like to assign more weight to term t_1 than t_3.

On first sight, the or factor seems to be consistent with our consideration. But we may recall *case 5* listed in the previous subsection where two terms t_2 and t_5 have the same ratio of a and c. In our consideration, they would have the same contribution to TC no matter what value of b and d. That means, slightly increasing or decreasing the value of b and d (i.e. whether adding or deleting the documents in the positive or negative category which do not contain this term) should not have an impact of terms' discriminating power. However, it is not the case in or value. The difference between or and our consideration is that we consider the terms'

discriminating power to be imposed by the number of relevant documents which contain this term only, i.e. a and c. Since $d >> a, b, c$ in general case, the or value would vary drastically because it involves the d value.

Based on these intuitive considerations and the analysis in the previous subsection, we propose a new supervised weight factor rf (*relevance frequency*) to capture this basic idea. Its formula is expressed as:

$$rf = log(2 + \frac{a}{c}) \tag{11}$$

When combined with tf by a multiplication operation, the weight of term t_i is defined as:

$$tf.rf = tf * log(2 + \frac{a}{c}) \tag{12}$$

We assign the constant value 2 in the rf formula because the base of this logarithmic operation is 2. Without the constant $2+$, the formula (11) would have the effect of giving a number of other terms zero weight. Let us consider two extreme cases in formula (12). In one extreme case, if $a = 0$ and then $rf = 1$, the final weight of a term in a document coincides with term frequency alone. In another extreme case, if $c = 0$, to avoid zero divisor, we set the minimal denominator as 1, which is reasonable whatever the value of a is. Thus the rf formula is replaced by:

$$rf = log(2 + \frac{a}{max(1, c)}) \tag{13}$$

and, the final term weight is replaced by:

$$tf.rf = tf * log(2 + \frac{a}{max(1, c)}) \tag{14}$$

Compared with the other supervised factors, the rf factor does not involve the d value. This is based on the observation that $d >> a, b, c$ and thus d will lessen the significance of a and c in expressing the terms' discriminating power for text categorization.

We name it rf (**r**elevance **f**requency) because only the frequency of relevant documents (i.e. those which contain this term) are considered in this formula. That is, only the ratio of a and c exerts an influence on the formula. Clearly, rf function captures the idea that the best discriminators for c_k are the ones distributed more in the positive samples of c_k than in the negative ones. Therefore, by using rf, we weight t_1 more than t_2 and t_3 in the above examples. Similarly, t_4 is weighted more than t_5 and t_6 by using rf factor.

C. Empirical Observation of Term's Discriminating Power in Real World Cases

The two previous subsections presented an analytical explanation of the six widely-used term weighting factors (Equations (5) to (10)) and our newly proposed rf factor. To evaluate their discriminating power from a quantitative aspect, we apply them to real world cases. To accomplish it, we choose four terms from the Reuters News Corpus. Table III and IV list the six different factors' values for the selected four terms in terms of two categories, namely, *00_acq* and *03_earn*, respectively.

TABLE III

COMPARISON OF THE WEIGHTING VALUES OF FOUR FEATURES IN CATEGORY *00_acq*

Feature	Category: *00_acq*					
	idf	rf	χ^2	or	ig	gr
acquir	3.553	4.368	850.66	30.668	0.125	0.161
stake	4.201	2.975	303.94	24.427	0.074	0.096
payout	4.999	1	10.87	0.014	0.011	0.014
dividend	3.567	1.033	46.63	0.142	0.017	0.022

TABLE IV

COMPARISON OF THE WEIGHTING VALUES OF FOUR FEATURES IN CATEGORY *03_earn*

Feature	Category: *03_earn*					
	idf	rf	χ^2	or	ig	gr
acquir	3.553	1.074	81.50	0.139	0.031	0.032
stake	4.201	1.082	31.26	0.164	0.018	0.018
payout	4.999	7.820	44.68	364.327	0.041	0.042
dividend	3.567	4.408	295.46	35.841	0.092	0.095

Based on the literal meaning, the first two terms, i.e. *acquir*[2] and *stake*, are closely related to the content of category *00_acq* while the last two terms, i.e. *payout* and *dividend*, are closely related to the content of category *03_earn*. However, as the idf factor neglects the category

[2]Here the *acquir* is a common root of several morphological forms of words and the original words can be *acquire, acquired, acquires, acquiring, acquirement,* etc.

information of the training set, each of these four terms is weighted equally by the idf factor in terms of the two categories. On the contrary, the supervised term weighting factors assign different weights to different terms for different categories.

Let us recall the two factors composing a term's weight in a document, i.e. term frequency factor and collection frequency factor. They both are considered to be equally important in contributing to text classification. Therefore, de-emphasizing or emphasizing either factor may result in inappropriate weights to terms. Usually, although the frequency of a term in a document varies according to the natural property of documents, it is quite a common case that a term occurs in zero, or several times rather than tens or even hundreds of times in a document.

Consider the χ^2 factor first. In Table III, the χ^2 values of the first two terms are much larger than the last two terms, which seems reasonable since the first two terms are more relevant to the content of category *00_acq* than the last two. However, the dynamic range of these four χ^2 values vary more considerably than that of tf, therefore it results in suppressing the impact of the term frequency factor in contributing the discriminating power to this term. On the other hand, in Table IV, the χ^2 factor assigns unreasonable values to these four terms, for example, the term *payout* is expected to be weighted much more than the first two terms as it is more closely related to the content of category *03_earn*. However, it is not the case. This observation indicates that χ^2 might not always assign appropriate weights to terms in real world cases.

Compared with the values of the χ^2 factor, the absolute values of the ig and gr factors are very small. However, after the normalization operation on these values, the two similar observations can be found for the above four terms with respect to the two categories, i.e. one is the large dynamic range of the two factors' values suppress the contributions of the term frequency factor for text classification, another is the term might not be weighted appropriately, for example, the term *payout* for category *03_earn*.

Unlike the above three factors, the or factor seems assign reasonable values for these four terms with respect to the two categories. However, the dynamic range of the or values for the four terms vary even more enormously than the above three factors and the term frequency factor and thus the or factor may suppress the term frequency factor's contribution power for text classification even more.

Finally, it is interesting to find that the rf factor has a comparable impact to term frequency factor for each of the four terms. Thus the $tf.rf$ method will be able to balance the impact of

both the term frequency factor and collection frequency factor on expressing their discriminating power.

So far, we address the third question based on the analytical explanation and the empirical observation from several selected samples. To experimentally compare the performance of these factors and validate the effectiveness of the $tf.rf$ method, we will conduct two series of experiments under various circumstances.

IV. Benchmark Methodology

A. Combined Term Weighting Methods

Eight different supervised and unsupervised term weighting schemes listed in Table V are selected in this study. These methods are chosen due to their reported superior classification

TABLE V

SUMMARY OF EIGHT SUPERVISED AND UNSUPERVISED TERM WEIGHTING METHODS

Methods	Denoted by	Description
Unsupervised term Weighting	$binary$	0 for absence or 1 for presence
	tf	term frequency alone
	$tf.idf$	classic *tf.idf*
Supervised term weighting	$tf.rf$	our newly proposed scheme
	rf	rf factor alone, i.e. $binary * rf$
	$tf.\chi^2$	$tf.chi^2$
	$tf.ig$	tf.*information gain*
	$tf.logor$	tf.log(*Odds Ratio*)

results or their typical representation in text categorization. The first three are unsupervised term weighting methods. The *binary* representation is the simplest method. The tf (term frequency) alone method has been shown to give satisfactory results in our previous study [6]. The most popular $tf.idf$ serves as a standard baseline. The last five are supervised term weighting schemes. The $tf.rf$ is our newly proposed term weighting method. The rf alone method, which is actually the product of $binary$ (term frequency) and rf, can be used to explore the effects of rf alone on the classification task. The $tf.\chi^2$ and $tf.ig$ methods are two typical representatives which are based on information-theory and the two functions have been widely used for feature selection. We also investigate the $tf.gr$ (*gain ratio*) method. Given the local policy for term weighting,

the result of *gain ratio* is identical with that of *information gain* since they only differ in a constant multiplicative factor. The or factor is derived from the probabilistic model and it was actually the *term relevance* factor in IR. Its common format is shown in Equation (10). However, since sometimes it seems to overstate relative positions, we would take the logarithm format to ameliorate this effect. Furthermore, to make the comparison between or and rf more meaningful, we adopted $log(2 + ad/bc)$ in this study, which has the similar format of rf, where the base of the logarithm operation is 2 as well. The $tf.logor$ method has a rather better performance than $tf.or$, thus we report the result of $tf.logor$ in this paper.

B. Inductive Learning Algorithms

We choose two state-of-the-art algorithms, i.e. *k*NN and SVM, for several reasons. Firstly, the two achieve top-notch performance among the widely-used algorithms (see [10], [21], [22]). Secondly, although other algorithms such as Decision Tree and Naive Bayes are also widely used, they are not included because the real number format of term weights could not be used except for the *binary* representation (see an exception in [12]). Finally, the two algorithms scale to large classification problems with thousands of features and examples.

1) Support Vector Machines: As a relatively new machine learning algorithm, SVM shows better performance than other methods due to its ability to efficiently handle relatively high dimensional and large-scale data sets without decreasing classification accuracy (see [7], [10], [21], [22]). Its excellent performance comes from the structural risk minimization principle on which the SVM is based. Generally, according to the different kernel functions from computational learning theory, SVMs are classified into two categories, i.e. linear and nonlinear (such as polynomial, radial based function (RBF), etc).

Specifically, in this study we adopt the linear SVM rather than non-linear SVM for several reasons. First, linear SVM is simple and fast [21]. Second, our preliminary experimental results has shown that the linear SVM performs better than the non-linear models. Even at the preliminary optimal tuning level, the accuracy achieved with the RBF kernel is lower than that of the linear kernel. This observation is also consistent with the findings in [21] and [22]. Thus a simple linear SVM will serve our purpose as the first benchmark classifier algorithm. The SVM software we use is LIBSVM-2.8 [24].

2) k Nearest Neighbor: The *k*NN algorithm is very simple and effective. However, the most important drawback is its inefficiency in the case of high dimensional and large-scale data sets. This drawback comes from the nature of this "lazy" learning algorithm as it actually does not have a true learning phase and thus incurs a high computational cost at the classification time.

[22] set *k* as $30 - 45$ since they found in that range the *k*NN yields stable effectiveness. Similarly, Joachims [10] tried different $k \in \{1, 15, 30, 45, 60\}$. Following the above two attempts, we explored the *k* values where $k \in \{1, 15, 30, 45\}$ for the *k*NN classifier and the results for the k with the best performance on the test samples are reported in subsection V-A.

C. Data Corpora

1) The Reuters News Corpus: The documents from the top ten largest categories of the Reuters-21578 document collection are used. According to the ModApte split, 9980 news stories have been partitioned into a training set of 7193 documents and a test set of 2787 documents. Stop words (292 stop words), punctuation and numbers are removed. The Porter's stemming [25] is done to reduce words to their base forms. The resulting vocabulary has 15937 words. By using χ^2 metric, the top $p \in \{25, 50, 75, 150, 300, 600, 900, 1200, 1800, 2400\}$ features per category are tried. Since SVMs have the capability to deal with high dimensional features, the previous studies showed that feature selection does not improve or even slightly degrades the SVM performance in [7] and [26]. We also conduct experiments by inputting the full words (after removing stop words, stemming and setting minimal term length as 4) without feature selection.

One issue of the Reuters corpus is the skewed category distribution problem. Among the top ten categories which have 7193 training documents, the most common category (*earn*) has a training set frequency of 2877 (40%), but 80% of the categories have less than 7.5% instances.

2) The 20 *Newsgroups Corpus:* The 20 Newsgroups corpus[3] is a collection of approximate $20,000$ newsgroup documents nearly evenly divided among 20 discussion groups. Some newsgroups are very closely related to each other, for example, the category of *comp.sys.ibm.pc.hardware* and category of *comp.sys.mac.hardware*. However, others are highly unrelated, for example, the category of *misc.forsale* and category of *soc.religion.christian*. After removing duplicates and

[3]The 20 Newsgroups corpus can be freely downloaded from http://people.csail.mit.edu/jrennie/20Newsgroups/.

headers, the remaining 18846 documents are sorted and partitioned by date into 11314 training documents (about 60%) and 7532 test documents (about 40%). Therefore, compared with the skewed category distribution in the Reuters corpus, the 20 categories in the 20 Newsgroups corpus are of approximate uniform distribution.

The resulting vocabulary, after removing stop words (513 stop words) and words that occur less than 3 and 6 times in the positive and negative categories respectively, has 50088 words. According to the χ^2 statistics metric, the top $p \in \{5, 25, 50, 75, 100, 150, 200, 250, 300, 400, 500\}$ features are tried.

3) The Ohsumed Corpus: The Ohsumed collection is a subset of clinically-oriented MEDLINE from year 1987 to year 1991, consisting of $348,566$ references from 270 medical journals over a five-year period. The Ohsumed in year 1991 includes 74337 documents but only 50216 have abstracts. Among these 50216 documents, Joachims [10] used the first 10000 documents for training and the second 10000 documents for testing. Specifically, we also use the Ohsumed corpus adopted by Joachims in [10] because this makes the comparison between our experiments and other published experiments reliable.

This corpus including medical abstracts from the MeSH categories are related to 23 cardiovascular diseases and each disease corresponds to one category label. After selecting such category subset, the unique abstract number becomes $13,929$ ($6,286$ for training and $7,643$ for testing). The resulting vocabulary, after removing stop words (513 stop words) and words that occur less than 3 and 6 times in the positive and negative categories respectively, has 19501 words. According to the χ^2 metric, the top $p \in \{10, 25, 50, 75, 100, 200, 300, 400, 600, 800, 1000, 1200\}$ features are tried.

D. Performance Evaluation

Precision and *recall* are two popular performance measures for TC. However, neither *precision* nor *recall* makes sense in isolation from each other as it is well known from the IR practice that higher levels of *precision* may be obtained at the price of low values of *recall*. To combine *precision* and *recall*, the two most widely-used measures, i.e. F_1 and *breakeven point*, have been proposed.

F_1 function attributes equal importance to *precision* (*p*) and *recall* (*r*) and it is computed as:

$$F_1 = \frac{2 * p * r}{p + r} \tag{15}$$

Usually, F_1 function is estimated from two ways, i.e. micro-averaged and macro-averaged. The two methods may give quite different results, i.e. the ability of a classifier to behave well also on categories with few examples will be emphasized by macro-averaged and much less so by micro-averaged.

The *breakeven point* is the value at which *precision* equals *recall*. To obtain the *breakeven point*, a plot of *precision* as a function of *recall* is computed by repeatedly varying the decision threshold ρ. As we known, new instances are classified by computing a score based on the classifier function and comparing the score with a decision threshold ρ, that is, new instances exceeding this decision threshold are said to belong to the category. Therefore different values of decision threshold ρ can be set to favor *precision* or *recall* depending on the application. If for no values of the threshold ρ *precision* and *recall* are exactly equal, the threshold ρ is set to the value for which *precision* and *recall* are closest. We must note that there may be no parameter setting that yields the *breakeven*.

E. Significance Tests

To verify the impact of the difference on the performance variation of these term weighting methods, we employ the McNemar's significance tests [27]. It is a χ^2-based significance test for goodness of fit that compares the distribution of counts expected under the null hypothesis to the observed counts. The McNemar's test can be summarized as follow.

Two classifiers f_A and f_B based on two different term weighting methods are performed on the test set. For each example in test set, we record how it is classified and construct the contingency table as shown in Table VI. The null hypothesis for the significance test states that on the test set, two classifiers f_A and f_B will have the same error rate, which means that $n_{10} = n_{01}$. Then the statistic χ is defined as

$$\chi = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}} \tag{16}$$

where n_{01} and n_{10} are defined in Table VI.

Dietterich showed that under the null hypothesis, χ is approximately distributed as χ^2 distribution with 1 degree of freedom, where the significance levels 0.01 and 0.001 correspond to the

TABLE VI

MCNEMAR'S TEST CONTINGENCY TABLE

n_{00} : Number of examples misclassified by both classifiers f_A and f_B	n_{01} : Number of examples misclassified by f_A but not by f_B
n_{10} : Number of examples misclassified by classifiers f_B but not by f_A	n_{11} : Number of examples misclassified by neither f_A nor f_B

two thresholds $\chi_0 = 6.64$ and $\chi_1 = 10.83$ respectively. Given a χ score computed based on the performance of a pair of classifiers f_A and f_B, we compare χ with threshold values χ_0 and χ_1 to determine if f_A is superior to f_B at significance levels of 0.01 and 0.001 respectively. If the null hypothesis is correct, then the probability that this quantity is greater than 6.64 is less than 0.01. Otherwise we may reject the null hypothesis in favor of the hypothesis that the two term weighting schemes have different performance when trained on the particular training set.

V. EXPERIMENTS

In this study, we conduct two series of experiments under various experimental circumstances.

The main purpose of the first series of experiments is to address the first two questions, i.e. to explore the superiority of supervised term weighting methods and the relationship between term weighting methods and algorithms. To accomplish this, we compare eight unsupervised and supervised (including our $tf.rf$) term weighting methods on two popular benchmark data corpora, i.e. Reuters-21578 and 20 Newsgroups, using SVM and *k*NN in terms of micro- and macro-averaged F_1 measure. The experimental results and discussion on these two corpora are in subsection V-A, V-B and V-C.

To further validate the effectiveness of $tf.rf$, we conduct the second series of experiments. The second series of experiments is to duplicate the experiment of [10] and thus to make the comparison more reliable. To do this, we conduct experiments under almost the same circumstances as [10], i.e. the same Ohsumed corpus, SVM algorithm and breakeven point measure. The comparative experimental results and discussion are in subsection V-D.

A. Results on the Reuters and 20 Newsgroups Corpora

Figures 2 to 5 report the results of the first series of experiments on the Reuters corpus and the 20 Newsgroups corpus using SVM and *k*NN. Each curve in these figures represents a different term weighting method.

Figure 2 depicts the micro-averaged and macro-averaged F_1 performance of different term weighting methods on the Reuters corpus using SVM. The performance of different term weighting methods at a small vocabulary size cannot be summarized in one sentence but the trends are distinctive in that the micro-averaged F_1 points of different term weighting methods generally increase as the number of features grows. All term weighting methods reach a maximum micro-averaged F_1 point at the full vocabulary. Among these, the best four micro-averaged F_1 points 0.9272, 0.9232, 0.9219 and 0.9191 are reached at the full vocabulary, using $tf.rf$, tf, $tf.logor$ and rf, respectively. The $tf.rf$ method has always been shown to perform better than others. In contrast to the performance in terms of micro-averaged F_1 measure, the performance of these methods in terms of macro-averaged F_1 measure does not increase significantly as the number of features grows. However, these methods show consistent performance in macro-averaged F_1 and the $tf.rf$ method is still the best among these methods.

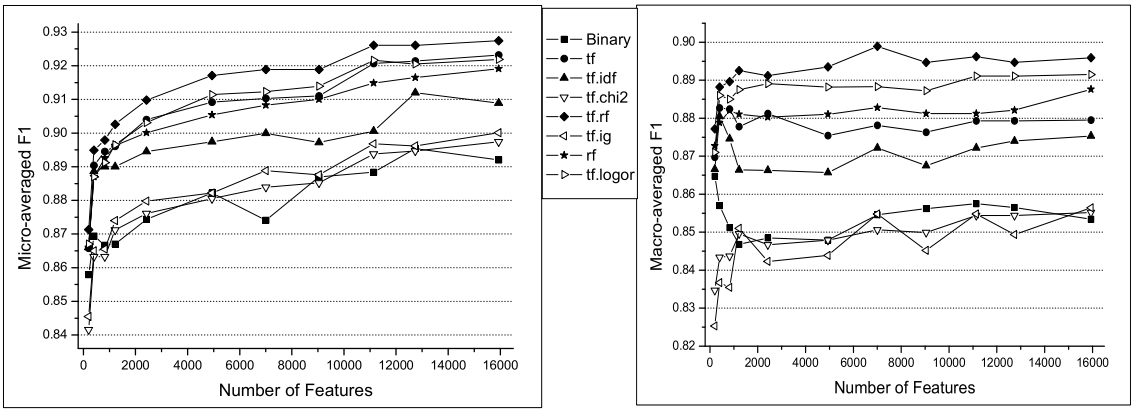

Fig. 2. Micro-averaged (left) and Macro-averaged (right) F_1 measure of the eight unsupervised and supervised term weighting approaches on the Reuters-21578 top ten categories using linear SVM algorithm with different numbers of features

Figure 3 depicts the micro-averaged and macro-averaged F_1 performance of different term weighting methods on the 20 Newsgroups corpus using SVM. The trends of the curves are similar to those in Figure 2. That is, the micro-averaged F_1 points of different term weighting

methods show a tendency to increase as the number of the features grows. However, these curves approach a plateau when the number of features exceeds 5000. Finally, almost all the term weighting schemes reach a maximum of micro-averaged F_1 point at the full vocabulary. Among them, the best three micro-averaged F_1 points 0.8081, 0.8038 and 0.8012 are reached at the full vocabulary, using rf, $tf.rf$ and $tf.idf$, respectively. Moreover, the performance of these methods in macro-averaged F_1 measure is quite similar to that in micro-averaged F_1 measure on this nearly uniform category distribution corpus.

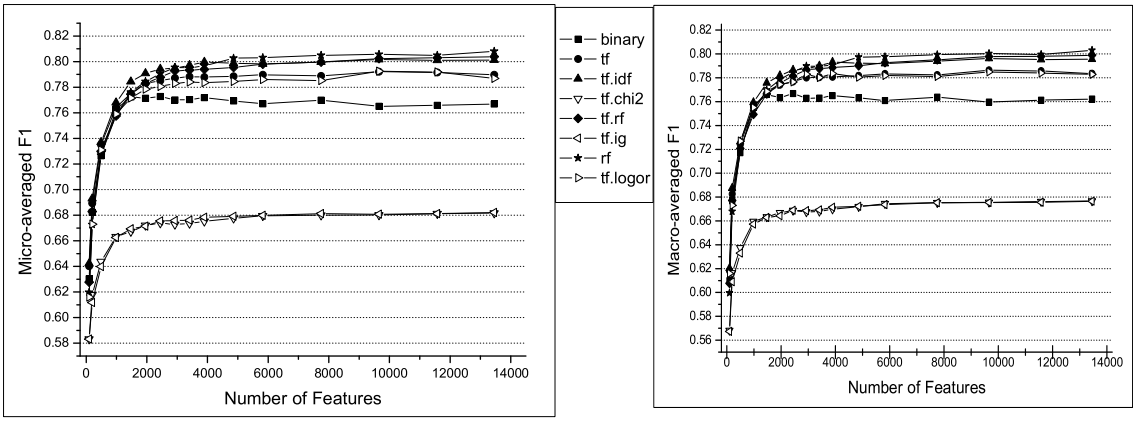

Fig. 3. Micro-averaged (left) and Macro-averaged (right) F_1 measure of the eight unsupervised and supervised term weighting approaches on 20 Newsgroups Corpus using linear SVM algorithm with different numbers of features

Figure 4 depicts the micro-averaged and macro-averaged F_1 performance of different term weighting methods on the Reuters corpus using *k*NN. The best k value for *k*NN is 30 and this is consistent with [10]. Unlike the shape of curves on the same corpus (Reuters) for the SVM algorithm, the performance of each term weighting method reaches a peak at a small feature set size around 400 in terms of micro-averaged F_1 performance and around 200 in terms of macro-averaged F_1 performance respectively. As the number of features grows, the performance of all methods declines except for $tf.ig$ and $tf.\chi^2$. The best two micro-averaged F_1 points 0.8404 and 0.8399 are achieved by using $binary$ and $tf.rf$ methods at the feature set size of 405. Similarly, the best three macro-averaged F_1 points 0.8259, 0.8219 and 0.8218 are achieved at the feature set size of 203 by using $tf.rf$, rf and $binary$ methods, respectively. When the number of features is more than 1000, the computation inefficiency of *k*NN algorithm at the classification time is the most important drawback which prevented us from conducting further experiments.

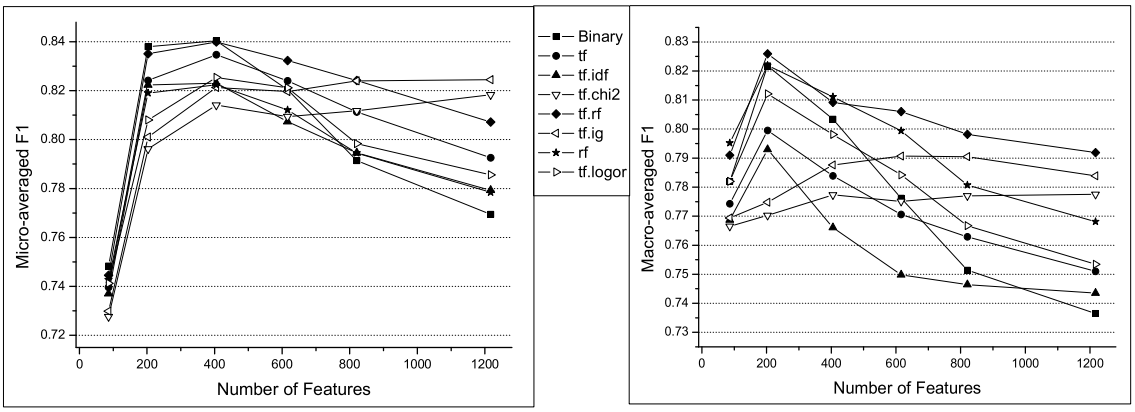

Fig. 4. Micro-averaged (left) and Macro-averaged (right) F_1 measure of the eight unsupervised and supervised term weighting approaches on the Reuters-21578 top ten categories using *k*NN ($k = 30$) algorithm with different numbers of features

Figure 5 depicts the micro-averaged and macro-averaged F_1 performance of different term weighting methods on the 20 Newsgroups corpus using *k*NN. The best k value for *k*NN is 15. The trends of curves are generally similar to those based on the Reuters and *k*NN (Figure 4). Almost all the curves reach a peak at a small features size around 500 except for $tf.rf$, rf and $tf.logor$. The curves of these three methods show a tendency to increase slowly as the number of features grows. The best two micro-averaged F_1 points 0.6913 and 0.6879 are achieved by using $tf.rf$ and rf when the number of features is around 2000, respectively. Similarly, we did not conduct experiments at a larger vocabulary size due to the built-in inefficiency problem of the *k*NN algorithm. Moreover, the performance of these methods in macro-averaged F_1 measure is quite similar to the micro-averaged F_1 measure on this nearly uniform category distribution corpus.

B. Discussion on the Reuters and 20 Newsgroups Corpora

We stated that the performance of different term weighting methods is closely related to the learning algorithm (SVM or *k*NN) and the property of the data corpus (skewed or uniform category distribution). That is, the comparison of supervised and unsupervised term weighting methods should be studied in conjunction with the text classifier and the property of the data corpus.

We use the McNemar's tests [27] to validate if there is significant difference between two term

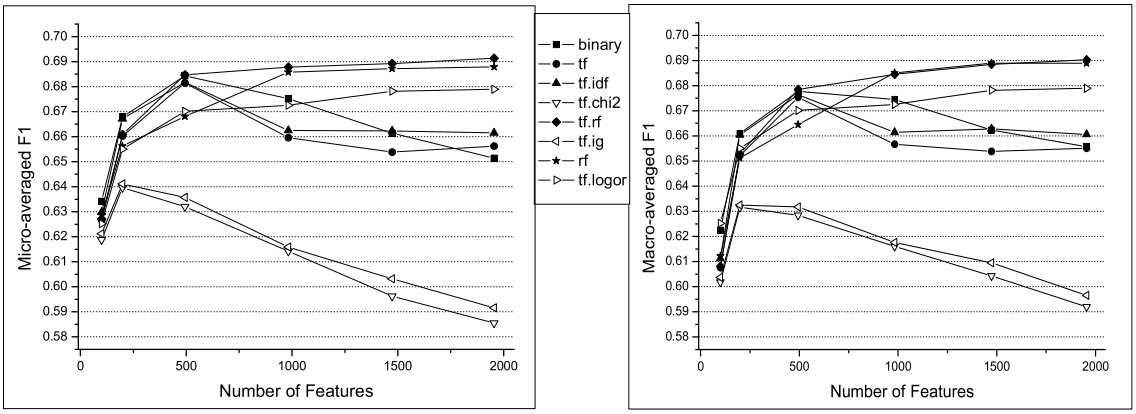

Fig. 5. Micro-averaged (left) and Macro-averaged (right) F_1 measure of the eight unsupervised and supervised term weighting approaches on 20 Newsgroups Corpus using *k*NN ($k = 15$) algorithm with different numbers of features

weighting methods in terms of the micro-averaged F_1 performance analysis. Table VII summarizes the statistical significance tests results on the two data sets and two learning algorithms at a certain feature set size where most of the methods reach their best performance. The term weighting methods with insignificant performance differences are grouped into one set and "$>$" and "$>>$" denote better than at significance level 0.01 and 0.001 respectively.

TABLE VII

STATISTICAL SIGNIFICANCE TESTS RESULTS ON THE TWO DATA CORPORA AND TWO LEARNING ALGORITHMS AT CERTAIN NUMBERS OF FEATURES IN TERMS OF THE MICRO-AVERAGED F_1 MEASURE.

Algorithm	Data Corpus	#_Features	McNemar's Test Results
SVM	Reuters	15937	$(tf.rf, tf, rf, tf.logor) > (tf.idf) > (tf.ig, tf.\chi^2, binary)$
SVM	20 Newsgroups	13456	$(rf, tf.rf, tf.idf) > (tf, tf.logor) >> binary >> (tf.ig, tf.\chi^2)$
*k*NN	Reuters	405	$(binary, tf.rf) > tf >> (tf.idf, rf, tf.ig, , tf.logor) > tf.\chi^2$
*k*NN	20 Newsgroups	494	$(tf.rf, binary, tf.idf, tf) >> (rf, tf.logor) >> (tf.ig, tf.\chi^2)$

Although these methods achieve their best performance at different numbers of features, the results in Table VII shows approximate ranks of these methods since most of them show consistent performance with respect to each other as the number of features grows given specific data corpus and learning algorithm. Moreover, from Figure 2 to Figure 5, several findings can be found as follows.

Generally, these supervised and unsupervised term weighting methods have not been shown a universally consistent performance on the two corpora and the two different algorithms.

Actually, these supervised term weighting methods have been shown two extremes results in all experiments. On the one hand, our $tf.rf$ method consistently shows the best performance with respect to the two different algorithms and two text corpora. Moreover, the rf alone method which ignores the term frequency also shows a comparable performance to $tf.rf$ in most experiments except for on Reuters data set using the *k*NN algorithm (Figure 4). On the other hand, the two typical supervised methods based on the information theory, i.e. $tf.\chi^2$ and $tf.ig$, are the worst methods among these eight methods in most cases. Specifically, they are inferior to the unsupervised ones, i.e. tf, $tf.idf$ and $binary$, and also inferior to the two newly presented special supervised methods, i.e. $tf.rf$ and rf. Exceptionally, in Figure 2, $tf.ig$ and $tf.\chi^2$ outperform $binary$ representation. For another example, in Figure 4, $tf.ig$ is comparable to rf and $tf.idf$, and again $tf.\chi^2$ is the worst method of all. Another supervised term weighting method, $tf.logor$ performs rather better than $tf.ig$ and $tf.\chi^2$ and the best performance of $tf.logor$ is comparable to that of tf alone in most cases but still a bit worse than that of $tf.rf$. These findings indicate that these sophisticated methods based on information theoretic functions have no superiority over the simpler unsupervised ones. This is contrary to our original expectation that the supervised term weighting methods which consider the document distribution in the training documents should always be better than the unsupervised ones.

Moreover, the performance of the unsupervised term weighting methods, i.e. tf, $tf.idf$ and $binary$, is dependent on the special data corpus and the learning algorithm in use. That is, the three have no consistent performance with respect to each other. For example, for the SVM-based text classifier (Figure 2 and Figure 3), $binary$ is the worst among these three methods. However, in Figure 2, tf is better than $tf.idf$ and but it is the other way round in Figure 3. On the other hand, different behaviors are noted for the *k*NN-based text classifier. For example, in Figure 4, $binary$ is the best and $tf.idf$ is the worst among the three unsupervised methods while in Figure 5 the performance of the three methods are comparable to each other. Therefore, several general observations can be made here. First, the popularly-used $tf.idf$ method performs well or even comparable to $tf.rf$ on the 20 Newsgroups corpus using SVM and *k*NN. This could be explained by the observation that on the skewed corpus the idf factor has lost its discriminating power for terms while on uniform corpus it has retained such power. This indicates that the property

of data corpus has a great impact on idf. Second, *binary* performs well or even comparable to $tf.rf$ using *k*NN on both corpora. However, on SVM-based model, *binary* has rather bad performance. This indicates that *k*NN favors *binary* while SVM does not. Third, one advantage of tf is its good robustness. Although tf does not have a comparable performance to $tf.rf$ in all experiments, it outperforms many other methods consistently and significantly. This can be seen from Figure 2 to Figure 5.

C. Discussion on the Effects of Feature Set Size on Different Algorithms

We stated that the performance of term weighting methods is closely related to the learning algorithms and data collections. However, the size of the feature set at which each term weighting method reaches the peak is closely dependent upon the learning algorithm in use rather than the term weighting method itself and the benchmark data collection.

Specifically, for the SVM-based text classifier, almost all term weighting methods achieve their best performance when inputting the full vocabulary. The findings in Figure 2 and Figure 3 indicate this. Since SVM also has the capability to handle thousands or even tens of thousands of features, the traditional feature selection can be omitted. These findings are entirely consistent with those reported in previous studies (see [3] and [10]).

For the *k*NN-based text classifier, all term weighting methods achieve their best performance at a small feature set size. For example, in Figure 4, most methods reach a peak at the feature set size of 400 or so, except for $tf.ig$ and $tf.\chi^2$. Similarly, in Figure 5, most methods attain their maximum performance at the feature set size of 500 except for $tf.rf$, rf and $tf.logor$ which increase slowly with increasing feature set size.

The possible explanation for this difference lies in their different theoretical rationales. When the size of the input feature set increases, the number of features with noise also increases. SVM algorithm is resilient to noise because only the *support vectors* are effective for the classification performance [22]; if all other examples (vectors) are removed, the model learned will not change. This property makes SVM theoretically different from other algorithms. However, *k*NN is an example-based learning algorithm, thus all the examples with noise have impact on the classification performance.

D. Results and Discussion on The Ohsumed Corpus

To validate the effectiveness of $tf.rf$ under the same experimental circumstances as [10], we conduct the second series of experiments on the Ohsumed corpus using SVM in terms of micro-averaged breakeven point measure. Besides $tf.rf$ and $tf.idf$, we also include other two methods with good performance reported in the previous subsection, i.e. $binary$ and tf.

Figure 6 depicts the results on the Ohsumed corpus in terms of the micro-averaged breakeven point value. In addition, Table VIII summarizes the best results (as shown in bold font) of four

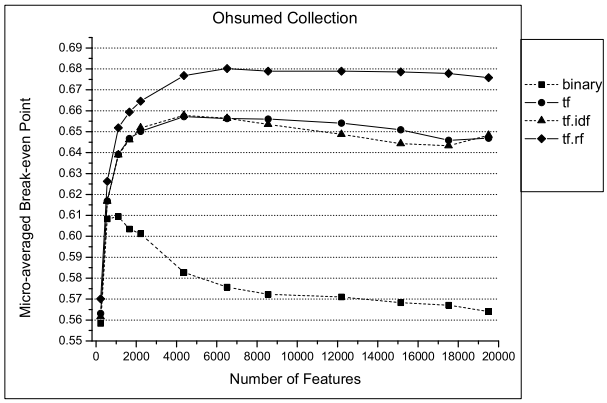

Fig. 6. Micro-averaged break-even points results for the Ohsumed Data Collection by using four term weighting schemes at different numbers of features.

different methods. It is clearly observed that $tf.rf$ performs consistently and significantly better

TABLE VIII

THE BEST PERFORMANCE OF SVM WITH FOUR TERM WEIGHTING SCHEMES ON THE OHSUMED CORPUS

Method	# of Features	micro-R	micro-P	micro-F1	macro-F1
$binary$	1111	0.6091	0.6097	0.6094	0.5757
tf	4363	0.6578	0.6566	0.6572	0.6335
$tf.idf$	4363	0.6567	0.6588	0.6578	0.6407
$tf.rf$	6511	**0.6810**	**0.6800**	**0.6805**	**0.6604**

than other term weighting methods and achieves the best performance in all experiments in terms of micro-averaged breakeven point, i.e. 0.6805. On the other hand, the $binary$ representation performs consistently the worst among these four methods, which is consistent with our previous

observation that SVM does not favor *binary*. The tf and $tf.idf$ perform comparable to each other and are better than $binary$ all along. These observations are not surprising and are consistent with the previous findings on the Reuters and 20 Newsgroups corpora.

In [10], Joachims used $tf.idf$ to represent the text[4] and conducted experiments on the same data corpus using SVM in terms of micro-averaged breakeven point. Table IX shows the results reported by Joachims [10] in which SVMs learn polynomial classifiers and radial basis function (rbf) classifiers based on the following two kinds of kernel functions:

$$K_{poly}(\vec{d_1}, \vec{d_2}) = (\vec{d_1}.\vec{d_2} + 1)^d \tag{17}$$

$$K_{rbf}(\vec{d_1}, \vec{d_2}) = exp(\gamma(\vec{d_1} - \vec{d_2})^2) \tag{18}$$

Note that when $d = 1$ in polynomial kernel functions, the polynomial SVM actually is actually a linear SVM.

TABLE IX

EXPERIMENTAL RESULTS REPORTED BY JOACHIMS [10] ON THE OHSUMED CORPUS.

	SVM (poly) $d =$				SVM (rbf) $\gamma =$		
	1	2	3	4	0.6	0.8	1.0
micro-averaged breakeven point (%)	60.7	64.7	65.9	65.9	65.7	66.0	66.1
	combined: **65.9**				combined: **66.0**		

Several observations from the comparison between these two studies are worth discussion.

- Our linear SVM results are more accurate, i.e. 68.05% for our linear SVM vs 60.7% for Joachims' linear SVM and 66.1% for his radial basis function with $\gamma = 0.8$. The observation that linear SVM outperforms other non-linear SVMs has already been supported by many researchers ([21] and [22]) and our previous study in [6]. There is no clear explanation for Joachims's different result.
- The performances of $tf.idf$ in two experiments are almost identical, i.e. 65.78% for our linear SVM and 66.1% for his radial basis function SVM. This difference is not significant.

[4]After stemming and stop-word removal, the resulting training corpus has 15561 terms which occur in at least three documents. Moreover, the author used *information gain* measure to select the most discriminating features.

- Last but not least, our proposed $tf.rf$ performs significantly the best among these methods in our experiments. Moreover, it outperforms the $tf.idf$ in Joachims' experiments whether using linear SVM or non-linear SVMs.

Note that although our experiments use the same corpus and same evaluation measure as Joachims', there are minor differences in data preparation, such as the stemming or stop words lists for text preprocessing, and the different feature selection measures. Joachims used *information gain* for feature selection while we used χ^2 instead. This difference is not significant and thus the comparison between the two experiments is reasonable.

VI. Concluding Remarks and Future Work

The following conclusions with empirical evidence address the three questions raised in Section I.

The answer to the first question is: not always. That is, not all supervised term weighting methods are superior to unsupervised methods. Actually, these supervised term weighting methods are the two extremes in terms of performance. On the one hand, the two supervised methods with solid theoretical bases, i.e. $tf.\chi^2$ and $tf.ig$, have the worst performance in all experiments. On the other hand, our proposed supervised method $tf.rf$, consistently achieves the best performance and outperforms other methods substantially and significantly.

The answer to the second question is, the performance of the term weighting methods, especially, the three unsupervised methods, has close relationships with the learning algorithms and data corpora. Their relationships with algorithms and data corpora can be summarized as follows:

- $tf.rf$ performs consistently the best in all experiments.
- $tf.\chi^2$ and $tf.ig$ perform consistently the worst in all experiments.
- $tf.logor$ outperforms $tf.\chi^2$ and $tf.ig$ and its performance is comparable to tf alone in most cases.
- $tf.idf$ performs well or even comparable to $tf.rf$ on the uniform category corpus (i.e. 20 Newsgroups corpus) either using SVM or *k*NN. This indicates that the property of data corpus has a great impact on idf.
- *binary* performs well or even comparable to $tf.rf$ on both corpora using *k*NN while rather bad on SVM-based text classifier. This shows that *k*NN favors *binary* and SVM does not.

- *tf* has no clear relationship with algorithm or data corpus. But although tf does not perform as well as $tf.rf$, it performs consistently well in all experiments.

Regarding the third question, the best performance of $tf.rf$ has been analyzed and explained from the qualitative aspect and confirmed by all experimental evidence. Given the cross-method comparison (various supervised and unsupervised), and cross-classifier (SVM and *k*NN) and cross-corpus validation (the Reuters, 20 Newsgroups and Ohsumed corpora), we are convinced that the observed consistently best performance of $tf.rf$ is general rather than corpus-dependent or classifier-dependent. We suggest that $tf.rf$ should be used as the term weighting method for text categorization task because this superior advantage over other methods is robustness, i.e. it works consistently the best either cross-classifier or cross-corpus.

We should point out that the observations above are made in combination with linear SVM and/or *k*NN algorithms in terms of micro-averaged F_1 or breakeven measure and other controlled experimental settings. It will be interesting to see in our future work if we can observe the similar results on a more general learning algorithm, such as centroid-based method, or by using other performance measure. In addition, since the term weighting is the most basic component of text preprocessing methods, we expect that our weighting method can be integrated into various text mining tasks, such as information retrieval, text summarization and so on.

Another interesting future work is to experiment on a new RCV1 benchmark corpus [26], which consists of about $850,000$ documents assigned to 103 categories. Due to its huge size, it is a more reliable but a much challenging task for learning models even the top-notch algorithms, such as SVM and *k*NN. They could not scale up very well to the huge text collections because of their polynomial complexity. For this reason, even though the recent work [4] and [28] used RCV1 as benchmark corpus, only a subset of RCV1 was used.

REFERENCES

[1] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 412–420.

[2] Z.-H. Deng, S.-W. Tang, D.-Q. Yang, M. Zhang, L.-Y. Li, and K. Q. Xie, "A comparative study on feature weight in text categorization," in *APWeb*, vol. 3007. Springer-Verlag Heidelberg, March 2004, pp. 588 – 597.

[3] F. Debole and F. Sebastiani, "Supervised term weighting for automated text categorization," in *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*. New York, NY, USA: ACM Press, 2003, pp. 784–788.

[4] P. Soucy and G. W. Mineau, "Beyond tfidf weighting for text categorization in the vector space model." in *IJCAI*, 2005, pp. 1130–1135.

[5] E.-H. Han, G. Karypis, and V. Kumar, "Text categorization using weight adjusted k-nearest neighbor classification," in *PAKDD '01: Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. London, UK: Springer-Verlag, 2001, pp. 53–65.

[6] M. Lan, S. Y. Sung, H. B. Low, and C. L. Tan, "A comparative study on term weighting schemes for text categorization," in *Proceedings of International Joint Conference on Neural Networks(IJCNN-05)*, vol. 1, Montreal, Canada., 31 July - 4 Aug, 2005, pp. 546–551.

[7] E. Leopold and J. Kindermann, "Text categorization with support vector machines. how to represent texts in input space?" *Machine Learning*, vol. 46, no. 1-3, pp. 423 – 444, January - February - March 2002.

[8] M. Lan, C. L. Tan, H. B. Low, and S. Y. Sung, "A comprehensive comparative study on term weighting schemes for text categorization with support vector machines," in *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*. New York, NY, USA: ACM Press, 2005, pp. 1032–1033.

[9] M. Lan, C. L. Tan, and H. B. Low, "Proposing a new term weighting scheme for text categorization," in *Proceedings of the Twenty-First National Conference on Artifical Intelligence(AAAI-06)*, Boston, Massachusetts., July, 2006, pp. 763–768.

[10] T. Joachims, "Text categorization with support vector machines: learning with many relevant features," in *Proceedings of ECML-98, 10th European Conference on Machine Learning*, C. Nédellec and C. Rouveirol, Eds., no. 1398. Chemnitz, DE: Springer Verlag, Heidelberg, DE, 1998, pp. 137–142. [Online]. Available: citeseer.ist.psu.edu/joachims97text.html

[11] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, no. 5, pp. 513–523, 1988.

[12] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *In AAAI-98 Workshop on Learning for Text Categorization*, 1998. [Online]. Available: citeseer.ist.psu.edu/mccallum98comparison.html

[13] C. Buckley, G. Salton, J. Allan, and A. Singhal, "Automatic query expansion using SMART: TREC 3," in *In Proc. of the Third Text REtrieval Conference*, 1994, pp. 69–80. [Online]. Available: citeseer.ist.psu.edu/37681.html

[14] H. Wu and G. Salton, "A comparison of search term weighting: term relevance vs. inverse document frequency," in *SIGIR '81: Proceedings of the 4th annual international ACM SIGIR conference on Information storage and retrieval*. New York, NY, USA: ACM Press, 1981, pp. 30–39.

[15] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[16] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 60, no. 5, pp. 493–502, 2004.

[17] S. E. Robertson, "Understanding inverse document frequency: on theoretical arguments for idf," *Journal of Documentation*, vol. 60, no. 5, pp. 503–520, 2004.

[18] S. E. Robertson and K. S. Jones, "Relevance weighting of search terms," *Journal of American Society for Information Science*, vol. 27, pp. 129–146, 1976.

[19] T. Mori, "Information gain ratio as term weight: the case of summarization of ir results," in *Proceedings of the 19th international conference on Computational linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 2002, pp. 1–7.

[20] T. Saracevic, "Relevance: A review of and a framework for the thinking on the notion in information science." *Journal of the American Society for Information Science*, vol. 26, pp. 321–343, 1975.

[21] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, "Inductive learning algorithms and representations for text categorization," in *Proceedings of the seventh international conference on Information and knowledge management*. ACM Press, 1998, pp. 148–155.

[22] Y. Yang and X. Liu, "A re-examination of text categorization methods," in *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM Press, 1999, pp. 42–49.

[23] Hwee Tou Ng, Wei Boon Goh, and Kok Leong Low. "Feature selection, perceptron learning, and a usability case study for text categorization." In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 67–73, New York, NY, USA, 1997. ACM Press.

[24] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at. [Online]. Available: http://www.csie.ntu.edu.tw/ cjlin/libsvm

[25] M. Porter, "An algorithm for suffix stripping." *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[26] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "Rcv1: A new benchmark collection for text categorization research," *J. Mach. Learn. Res.*, vol. 5, pp. 361–397, 2004.

[27] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 10, no. 7, pp. 1895–1923, 1998.

[28] Y.-S. Dong and K.-S. Han, "Text classification based on data partitioning and parameter varying ensembles," in *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*. New York, NY, USA: ACM Press, 2005, pp. 1044–1048.

PLACE
PHOTO
HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.