Stemming via Distribution-Based Word Segregation for Classification and Retrieval

Narayan L. Bhamidipati and Sankar K. Pal, *Fellow, IEEE*

***Abstract*—A novel corpus-based method for stemmer refinement, which can provide improvement in both classification and retrieval, is described. The method models the given words as generated from a multinomial distribution over the topics available in the corpus and includes a procedurelike sequential hypothesis testing that enables grouping together distributionally similar words. The system can refine any stemmer, and its strength can be controlled with parameters that reflect the amount of tolerance to be allowed in computing the similarity between the distributions of two words. Although obtaining the morphological roots of the given words is not the primary objective, the algorithm automatically does that to some extent. Despite a huge reduction in dictionary size, classification accuracies are seen to improve significantly when the proposed system is applied on some existing stemmers for classifying 20 Newsgroups and WebKB data. The refinements obtained are also suitable for cross-corpus stemming. Regarding retrieval, its superiority is extensively demonstrated with respect to four existing methods.**

***Index Terms*—Precision and recall, prototype selection, stemming, text categorization.**

I. Introduction

STEMMING is the process of clubbing together words that are similar in nature. Generally, morphologically similar words are grouped together under the assumption that they are also semantically similar. Stemming is frequently used in the field of information retrieval [1], [2] because it results in an increase in recall, as documents that do not contain the exact query terms are also retrieved. In particular, all documents that contain words with the same stem as the query term are considered relevant. Stemming also reduces the size of the feature set (when words are viewed as the features of documents). For the purpose of classification, this means that the models involved are far less complex than what would have been if the original set of words were used. This also means that it would lead to better generalization, in the sense that a small training error would also imply a small test error. It has been observed that the classification performance does not decrease much due to the application of some of the standard stemmers. In addition, this would lead to a reduction in the size of the index that needs to be stored. One may note that stemming in text mining may also be viewed as feature or prototype selection/reduction or clustering in pattern recognition and as case selection in case-based reasoning problems, where the basic objective is to select the most representative features or dimensions, or cases of a class or a concept based on some similarity measure or grouping.

Several standard techniques that perform stemming are available in the literature [3]. The strength of a stemmer is the amount of reduction in the size of the dictionary it obtained [3]. Strong (or aggressive) stemmers may reduce the size of the index for a given corpus drastically. However, stemming is afflicted with two kinds of errors: understemming and overstemming [4]. Understemming is the case where words that should have been grouped into the same class are not so, and the performance is suboptimal. When too many unrelated words are merged together, then it is the case of overstemming. This leads to a reduction in precision during retrieval and an increase in the error rate for classification. One may also note that with increase in the strength of a stemmer, recall is increased, but either retrieval precision or classification accuracy (or both) is degraded. This may be concluded by observing that increase in stemming strength gradually leads to a reduced number of stem classes, and in an extreme case, all the words belong to a single-stem class, which provides none of the information required for classification or retrieval.

Thus, the general objective in designing a stemmer is to ensure that classification accuracy and retrieval precision are maintained. In this paper, we describe the design of such a stemmer. We make use of the classification information of the corpus and model words as they arise from a multinomial distribution [5]. A segregation method based on this can be employed on any existing-rule-based stemmer to refine its equivalence classes for improvement. The proposed methodology is found to improve both classification accuracy and retrieval precision when applied on the Porter [6] and Truncate(3) [7] stemmers and compared with some existing ones, including co-occurrence-based refinement [7] and a distributional clustering-based stemmer [8]. The classification performance is measured on the 20 Newsgroups (20NG) and WebKB data sets, both in terms of the accuracy and precision-recall plots of naive Bayes (NB), support vector machine (SVM), and maximum entropy (MaxEnt)-based classifiers. The retrieval efficiency has been tested on the Wall Street Journal (WSJ) data set, and precision-recall values have been displayed graphically. All the results have been tested for statistically significant improvement by the proposed methodology over some of the related methods.

The paper is organized as follows. The background on stemming and related work is provided in Section II. Then, we describe the proposed stemming technique in Sections III and IV

Manuscript received November 8, 2005; revised May 22, 2006. This paper was recommended by Associate Editor E. Santos.

The authors are with the Machine Intelligence Unit, Indian Statistical Institute, Kolkata 700108, India (e-mail: bln_r@isical.ac.in; sankar@isical.ac.in).

Digital Object Identifier 10.1109/TSMCB.2006.885307

and present the experimental results in Section V. We draw our conclusions in Section VI.

II. Stemming and Related Work

Documents are generally represented in terms of the words they contain, as in the vector-space model [9]. Many of these words are similar to each other in the sense that they denote the same concept(s), i.e., they are semantically similar. Generally, morphologically similar words have similar semantic interpretations, although there are several exceptions to this, and may be considered equivalent. The construction of such equivalence classes is known as stemming. A number of stemming algorithms or stemmers, which attempt to reduce a word to its stem or root form, have been developed. Thus, the document may now be represented by the stems rather than by the original words. As the variants of a term are now conflated to a single representative form, it also reduces the dictionary size, which is the number of distinct terms needed for representing a set of documents. A smaller dictionary size results in savings in storage space and processing time.

Stemming is often used in information retrieval because of the various advantages it provides [2]. The literature is divided on this aspect, with some authors finding stemming helpful for retrieval tasks [2], while others did not find any advantage [10]. However, they are all unanimous regarding the other advantages of stemming. Not only is the storage space for the corpus and retrieval times reduced but recall is also increased without much loss of precision. Moreover, the system has the option for query expansion to help a user refine his/her query.

A. Different Stemming Algorithms

Various stemmers are available for several languages, including English. The most prominent ones are those introduced by Lovins, Dawson, Porter, Krovetz, Paice/Husk and Xu, and Croft. We now provide a brief description of some of these algorithms.

1) Truncate(n): This is a trivial stemmer that stems any word to the first n letters. It is also referred to as n-gram stemmer [7]. This is a very strong stemmer. However, when n is small, e.g., one or two, the number of overstemming errors is huge. For this reason, it is mainly of academic interest only. In this paper, we have chosen n to be 3, 4, and 5 and refer to them as *trunc3*, *trunc4* and *trunc5*, respectively.

2) Lovins Stemmer: The Lovins stemmer [11] was developed by Lovins and is a single-pass longest match stemmer. It performs a lookup on a table of 294 endings, which have been arranged on a longest match principle. The Lovins stemmer removes the longest suffix from a word. Once the ending is removed, the word is recoded using a different table that makes various adjustments to convert these stems into valid words. However, it is highly unreliable and frequently fails to form words from the stems or to match the stems of like-meaning words.

3) Dawson Stemmer: The Dawson stemmer [12], which was developed by Dawson, extends the Lovins stemmer. This is also a single-pass longest match algorithm, but it uses a much more comprehensive list of around 1200 suffixes, which were organized as a set of branched character trees for rapid access. In this case, there is no recoding stage, which had been found to be unreliable.

4) Porter Stemmer: Porter proposed the Porter stemmer [6], which is based on the idea that the suffixes in the English language (approximately 1200) are mostly made up of a combination of smaller and simpler suffixes. It has five steps, and within each step, rules are applied until one of them passes the conditions. If a rule is accepted, the suffix is removed accordingly, and the next step is performed. The resultant stem at the end of the fifth step is returned.

5) Paice/Husk Stemmer: The Paice/Husk stemmer [13] is a simple iterative stemmer and uses just one table of rules; each rule may specify either deletion or replacement of an ending. The rules are grouped into sections that correspond to the final letter of the suffix, making the access to the rule table quicker. Within each section, the order of the rules is significant. Some rules are restricted to words from which no ending has yet been removed. After a rule has been applied, processing may be allowed to continue iteratively or may be terminated.

6) Krovetz Stemmer: The Krovetz stemmer [14] was developed by Krovetz and makes use of inflectional linguistic morphology. It effectively and accurately removes inflectional suffixes in three steps: the conversion of a plural to its singular form, the conversion of past to present tense, and the removal of *-ing*. The conversion process first removes the suffix and then through the process of checking in a dictionary for any recoding, returns the stem to a word. It is a light stemmer in comparison to the Porter and Paice/Husk stemmers.

7) Co-Occurrence-Based Stemmer by Xu and Croft: Xu and Croft [7] observed that most stemmers perform understemming or overstemming, or even both. Strong stemmers generally perform overstemming only. Xu and Croft came up with an algorithm that would refine the stemming performed by a strong stemmer. To this end, they computed the co-occurrences of pairs of words that belong to the same equivalence class. For each pair, they also computed the expected number of co-occurrences, which would account for words that occur together randomly. Thus, they obtained a measure that is similar to the mutual information measure defined as

$$em(w_i, w_j) = \max\left(\frac{n(i,j) - En(i,j)}{n_i + n_j}, 0\right)$$

where n_i and n_j are the frequencies of w_i and w_j, respectively, and $n(i,j)$ is the number of times the two words co-occur. E denotes the expected value. This measure ignores any co-occurrences that may be attributed to pure chance. If $em(w_i, w_j)$ is significantly greater than zero, they conclude that in the given corpus, the two words indeed appear together and may be retained in the same equivalence class.

Splitting the equivalence classes in an optimal way however is computationally very expensive. When the equivalence classes are large, Xu and Croft opt for a suboptimal solution obtained by a connected component-labeling algorithm applied after achieving the thresholds of the em scores.

8) Dictionary-Based Stemmers: There have also been dictionary-based stemmers [2], [15], [16] that improve on an existing stemmer by employing knowledge obtained from a dictionary. Word co-occurrences in a dictionary are considered to imply the relations between words.

9) Probabilistic Stemmers: Given a word in a corpus, the most likely suffix–prefix pair that constitutes the word is computed [17]. Each word is assumed to be made up of a stem (suffix) and a derivation (prefix), and the joint probability of the (stem, derivation) pair is maximized over all possible pairs constituting the word. The suffix and prefix are chosen to be nonempty substrings of the given word, and it is not clear what should be done in the case when a word should be stemmed to itself.

10) Refinement of an Existing Stemmer: In some cases, errors produced by a stemmer are manually rectified by providing an exception list [14]. The stemmer would first look up the exception list, and if the word is found there, it returns the stem found there. Otherwise, it uses the usual stemmer. The aforementioned co-occurrence-based stemmer is also one such algorithm where the exceptions are obtained automatically.

11) Distributional Clustering as Stemming: Distributional clustering [8], [18]–[20] joins (distributionally) similar words into a group if the words have similar probability distributions among the target features that co-occur with them. In [18]–[20], the distributions are estimated by observing the grammatical relationships between words and their contexts, whereas in [8], the distributions are obtained from the frequency of words in each category of the corpus. In their work on document classification, Baker and McCallum had chosen the class labels as the target features. The root forms of the words are not taken into consideration while grouping them. This algorithm described in [8] is given as follows. The mutual information of each word in the corpus with the class variable is computed, and the words are sorted in descending order. The number of desired clusters is fixed beforehand, e.g., to M. The first M words are initialized to form M singleton clusters. The two most similar (of the M) clusters are merged. This similarity is measured in terms of the Kullback–Leibler divergence of the distributions of the two clusters. The next word in the sorted list forms a new singleton cluster. Thus, the number of clusters remains M each time. In this paper, we refer to Baker and McCallum's method as *baker*. In our implementation, we have fixed M to the number of stems obtained by refining the *trunc3* stemmer using our model.

B. Stemming and Classification

There are several works on text classification (see, e.g., [21]) where stemming has been employed in a routine manner. However, there are differences in opinions of researchers regarding the effectiveness of stemming for the purpose of classification. While Riloff [22] and Spitters [23] had concluded that stemming may not help increase classification accuracy, Buseman had observed that morphological analysis increases the performance for a series of classification algorithms applied to German e-mail classification. In a recent work, Gustad and Bouma [16] observed that stemming does not consistently improve classification accuracy. More recently, however, Cohen *et al.* [24] found stemming advantageous while classifying medical documents.

Perhaps, the reasons for such varied observations lie in the different characteristics of the document collections involved. On the one hand, stemming would increase the number of instances per feature (by reducing the number of features), which is a favorable situation for classification. On the other hand, stemming may merge words regardless of the class information that they hold, thereby confusing a classifier that is presented with such mixed instances.

In the next section, we propose a novel stemming technique, whereby even very strong stemming does not reduce the classification accuracy.

III. Proposed Stemming Technique

A. Criteria

We try to improve on the existing stemmers previously discussed on the following aspects.

1) Substitute Words: These words, although very similar in meaning and/or usage, often do not tend to appear with each other. These are often the result of varying author styles, where a particular author uses just one of the substitute words all the time. Examples include words that have different spellings under British and American usage (e.g., colour and color, respectively). To infer that they should be stemmed to the same word, one would need to analyze their co-occurrence with other words to find such relations.

2) Words With Many Senses: When a word has many senses, there might be words that are semantically similar to it in just one sense. Merging them would lead to loss of information [14]. It is desirable that only those words that match in all the given senses are merged.

3) Creation of New Words: Rule-based stemmers occasionally create new words while stripping suffixes. For example, the Porter stemmer stems both *change* and *changing* to *chang*. Note that this is inevitable, and if a rule were to modify the final stem by adding an *e* to it, it would lead to yet other problems such as *hang* and *hanging* both stemming to *hange*. The creation of such words may also increase ambiguity when dissimilar classes of words are merged. For example, the Porter stemmer stems *range*, *ranged*, *ranges*, *ranging*, and *rang* to *rang*, even though *rang* is unrelated to the rest.

4) Simplicity and Speed: Rule-based stemmers only need to step through a sequence of predefined rules and are very efficient, albeit at the cost of stemming errors. Corpus-based refinements are computationally expensive, as seen in the case of co-occurrence-based stemmers, where the process of refining the stems involves computing the co-occurrences of each pair of words that map to the same stem. If an equivalence class (set of words mapping to the same stem) is "large," splitting it optimally becomes an arduous task.

5) Cross-Corpus Stemming: It is desirable to perform the stemming operation only once. In addition, additional information such as categories may not be available for all corpora. However, one would like the stemmer to perform reasonably

well in that situation also. The stemmer may be built based on a single corpus, and the same set of stems is also employed for other corpora. The challenge is to come up with a stemmer that does well even when the two corpora are very different in nature.

We now describe a stemming algorithm that incorporates all the aforementioned desiderata.

B. Stemmer Refinement by Distribution-Based Segregation

The objective at hand is that given an equivalence class of words, it is to be split in such a way that the resulting equivalence classes reflect improved stemming in terms of classification and retrieval. The primary objective is not to group morphological or semantically similar words as a human linguist would do, although such a feature would be an added attraction. We utilize the information available in a classified text corpus to perform the splitting. We assume that we have a corpus whose documents are distributed into several groups or categories, with each one consisting of documents that pertain to some topic. The primary assumption behind the proposed methodology is that two words may be stemmed to the same stem if they are similar in their distribution across various categories.

Each word is assumed to have a multinomial distribution [5] over the set of categories of the given corpus. In a multinomial distribution, n events are observed; each of which has k possible outcomes, with the ith outcome having a probability of p_i. The binomial distribution is a special case where $k = 2$. Words deemed to be arising from the same multinomial distribution are kept in the same equivalence class, whereas those that are significantly different from each other are separated out. Here, differences in the total number of appearances of the words (denoted by n) are ignored, just as in the vector-space (or the bag of words) model. The distribution of each word is estimated from its frequencies in the various categories. Formally, the proposed methodology is described here.

Let $\{w_1, w_2, \ldots, w_n\}$ be the set of words belonging to an equivalence class, i.e., they all stem to the same stem. Let K be the number of categories of the given text corpus. For each word w_i, we compute the occurrence vector $n_{i1}, n_{i2}, \ldots, n_{iK}$, where n_{ik} is the number of occurrences of w_i under the kth category. We assume that each w_i arises from a multinomial distribution whose parameters are $p_{i1}, p_{i2}, \ldots, p_{iK}$ and that $n_i = \sum_{k=1}^{K} n_{ik}$. Here, each p_{ik} denotes the probability of w_i appearing under the kth category and is estimated as the corresponding proportion of occurrences in the corpus n_{ik}/n_i. The aim is to partition this set of words into non empty subsets such that each subset consists of words whose estimated distributions do not significantly differ from each other.

Moreover, this needs to be done without prior knowledge of the size of the partition.

We employ a procedure similar to sequential hypothesis testing [25] to attain this goal. Two thresholds/cutoffs, e.g., t_1 and t_2 ($t_1 \leq t_2$), are chosen for this purpose. The words are sorted in descending order based on their frequencies. Without loss of generality, we shall now denote this sorted list of words by $\{w_1, w_2, \ldots, w_n\}$. The most frequent word w_1 is chosen and is considered to stem to itself. We denote this as $stem(w_1) = s_1$. Let S be the current set of stems. Thus, initially, $S = \{s_1\}$. We shall also denote the equivalence class of stem s_j by S_j, which is defined as $S_j = \{w_k : stem(w_k) = s_j\}$.

For each subsequent word, we compute the distance between its distribution function and that of each stem in S $d_{ij} = d(w_i, s_j)$. This distance represents the dissimilarity between a word and the candidate stem, with a small distance value indicating that the word is very similar to the stem and may be merged with it. The distance function may be chosen to reflect the manner in which we want to measure the dissimilarity between a word and a stem and may be any of those discussed in Section III-C. A more comprehensive list of (dis)similarity measures is available in [20]. If each of these distances is greater than the bigger cutoff, i.e., $d_{ij} > t_2 \forall j$, we shall call the current word a new stem and add it to set S. On the other hand, if any of the distances, e.g., d_{ij}, is smaller than the smaller cutoff t_1, we shall add the current word to the equivalence class of s_j so that $stem(w_i) = s_j$.

This procedure is iterated with the two thresholds modified such that the new lower threshold is greater than t_1 and the larger one is smaller than t_2. It may be noted that since the proposed algorithm depends on the accurate estimation of word distributions, the larger the number of words or documents per class, the better the expected performance.

For the purpose of cross-corpus stemming, the stems are first constructed based on one corpus. Then, the words of the other corpus are stemmed using the proposed method whenever they are available in the first one. For all other words, we rely on a standard stemmer such as *porter* or *trunc3*.

C. Choice of Distance Function and Thresholds

The aforementioned description provides a general form of the corpus-based stemmer. For implementation, one needs to have a proper choice of distance function and thresholds. These are described here. The term "distance" can be defined in various ways to produce a variety of (mostly similar) stemmers. For example, the distance between a candidate word w_i and a stem s_j may be defined in one of the following ways:

1) the distance between w_i and a "prototype" (or a representative) of the set S_j;
2) the minimum distance between w_i and an element of S_j;
3) the maximum distance between w_i and an element of S_j.

For each of the preceding options, one may compute the Euclidean distance, cosine distance (derived from the cosine similarity metric) [26], or the Kullback–Leibler distance [27] between the two distributions. Alternatively, one may compute a test statistic that would be used for testing the equality of the two distributions. The distance function may also take into consideration the size of the longest common prefix so that words with a longer common prefix would be more likely to be stemmed to the same stem.

We deduce the computation time of our algorithm by looking at the operations performed for refining each of the initial equivalence classes. Suppose that a stem class consists of n words and m concept groups. Thus, the objective is to split the given stem class into m concept classes. The words are sorted in

descending order based on their frequencies in $O(n \log n)$ time. Then, each word would be compared with at most m prototypes (one for each concept class). Thus, splitting a stem group is an $O(mn)$ operation [assuming that $m < \log n$; otherwise, it would be $O(n \log n)$]. It may be noted that at this stage, co-occurrence-based refinement would need to compute co-occurrences between all pairs of words, thereby becoming an $O(n^2)$ algorithm.

Now, if there are M stem classes initially, the complexity of our method is $O(Mmn)$. In addition, n is expected to be N/M, where N is the total number of words, i.e., $Mn = N$. Hence, the average complexity of the proposed method is $O(mN)$, with m being interpreted as the average number of concept groups per initial stem class. We note that in the preceding derivation, m depends on M because as M decreases, the size of the initial equivalence classes and, consequently, m are expected to increase.

This is an added advantage for the proposed method over the co-occurrence-based method as it would not require a prior stemming result to start with the refinement process [equivalent to using the Truncate(0) stemmer]. However, for the purpose of stemming, this would necessitate the incorporation of longest-common-prefix-based modification during distance calculations.

There are no strict guidelines for choosing t_1 and t_2, except that a high t_1 would result in more words getting a stem class of their own (understemming) and a low t_2 would lead to large equivalence classes (overstemming). Thus, all that we are doing by choosing t_1 and t_2 is to fix a level of permissible understemming and overstemming errors. However, it is not possible to directly compute the exact number/proportion of such errors (since the exact distributions of the words are not known beforehand). If $t_1 = t_2$, then we do not need multiple iterations in the given procedure. This would result in a reduction in computing time. However, it may miss out on some simple mergers of equivalence classes. This is so because once a word is called a new stem, it cannot be merged with any of the existing stems at a later stage. Choosing $t_1 < t_2$ allows us to do just that. In this case, whenever one is sure of neither merging the current word with an existing stem nor assigning it to a new class of its own, this decision may be put off for later. In a following iteration, due to the change in the structure of the classes or the values of the chosen thresholds, the decision may become clearer. The strength of the stemmer would be proportional to the size of thresholds t_1 and t_2.

IV. Implementation

For implementation of the proposed methodology, we preprocess the given corpus first and then refine a given stemmer by splitting the equivalence classes generated by that stemmer. We briefly describe these tasks here.

Any text corpus would contain several noisy terms. To clean such noise, some standard preprocessing tasks are performed on the given corpora. The headers of the documents are ignored altogether, and only those words that appear in at least two documents are retained. HTML tags and stopwords are removed before building the model. All words are converted to lower case.

Then, to decide if a given word w_i may be merged with a stem class S_j, we test the difference between the estimated distributions of w_i and S_j. The distributions are estimated as the corresponding frequencies of words appearing under the K topics of the given corpus. For testing the difference between the two distributions, Pearson's statistic [28] is computed as described here. Let $(n_{i1}, n_{i2}, \ldots, n_{iK})$ be the topic vector of w_i. Define m_{jk} to be $\sum_{w_i \in S_j} n_{ik}$. In addition, let m_j denote the total $\sum_{k=1}^{K} m_{jk}$. It is assumed that the estimated distribution of S_j is the actual one. To test if $(n_{i1}, n_{i2}, \ldots, n_{iK})$ has arisen from the distribution of S_j, Pearson's statistic is computed as

$$\frac{m_j}{n_i} \sum_{k=1}^{K} \frac{n_{ik}^2}{m_{jk}} - n_i$$

where n_i and m_j are the totals, as defined previously. When n_i is large, this statistic is known to approximately follow a χ^2 distribution with $K - 1$ degrees of freedom.

Since some of the n_{ik} values may be zero, we replace them by

$$n'_{ik} = 0.9 n_{ik} + 0.1 \frac{n_i}{K} = n_{ik} + 0.1 \left(\frac{n_i}{K} - n_{ik} \right). \quad (1)$$

This is done for each term in the dictionary. What we do here, essentially, is to perform a smoothing operation. Now, none of the cells is empty, and moreover, the total remains the same. In the remainder of this paper, we shall refer to n'_{ik} as n_{ik} itself. If w_i is merged with S_j, the m_{jk}s are updated by adding n_{ik} [after modifying as in (1)] for each j.

Since we have sorted the words in descending order based on their frequencies, the $\chi^2_{(K-1)}$ assumption is satisfied initially. In addition, when the frequency of a word is very low, it would not matter too much as the word itself might not have much effect during classification. For this reason, in the initial stages, we merge words into a stem class only if their distributional similarity is very high. Later on, this criterion is relaxed a little by allowing more distant words to be merged to a stem class. Thus, each equivalence class is split into two iterations.

During the first iteration, the values of t_1 and t_2 are set to $\chi^2_{(K-1),\alpha}$ and $4\chi^2_{(K-1),\alpha}$, respectively. Here, $\chi^2_{(K-1),\alpha}$ is the upper α cutoff of the $\chi^2_{(K-1)}$ distribution, i.e., the value of the $\chi^2_{(K-1)}$ distribution function at the chosen cutoff is $1 - \alpha$. t_1 and t_2 are both set to $2\chi^2_{(K-1),\alpha}$ during the second iteration. In our implementation, we had chosen α to be 0.05.

It may be noted that this paper differs from others based on distributional similarity primarily because the null hypothesis is chosen as the word and a stem group are similar. In other words, unless there is strong evidence that the word does not belong to a stem group, it is not separated from that group. Since the estimate of the distribution of the stem itself depends on the choice of words already included in the stem class, this choice needs to be made carefully at each step. As mentioned in [19], estimation of distribution is better for words with higher frequencies. The proposed methodology tries to merge

the most frequent words first, resulting in better estimates for the distribution of each stem class.

Although our methodology does not create any new words of its own, during cross-corpus stemming, when the words encountered are not in the dictionary, a standard stemmer is used, and that may introduce new words into the system.

V. Experimental Results and Comparison

A. Data Sets Used

We evaluated the performance of the proposed methodology on three data sets, namely: 1) 20NG, 2) WebKB, and 3) WSJ. They are described as follows.

1) 20NG [29]: This collection is a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering, and it consists of 19 997 newsgroup documents, which was partitioned evenly across 20 different newsgroups.

2) WebKB [30]: This data set, which was collected from computer science departments of various universities in January 1997 by the World Wide Web Knowledge Base project of the Carnegie Mellon University text learning group, which consists of 8282 web pages manually classified into seven categories.

3) WSJ: This data set is a part of the Text Retrieval Conference collection [31] and consists of more than 170 000 records, which appeared from 1987 to 1992 in the WSJ. The queries (also called topics) and query relevance scores (in the form of qrel files) are available at http://trec.nist.gov/data/test_coll.html.

B. Evaluation Procedure

The performance of distribution-based stemmer refinement has been evaluated in two ways. First, a direct evaluation in terms of linguistic analysis has been performed. This would reveal how similar the system is to a human who groups together morphologically and semantically related words. The second evaluation is an indirect evaluation that observes the effects of stemming on classification accuracy and retrieval performance.

For performing the direct evaluation through a linguistic analysis, we followed the procedure described in [32]. A generalization of this procedure is provided in [33] and is useful for automatic evaluation of stemmers, but this is not employed here due to lack of resources on the authors' part. There are 13 621 words in the intersection of the vocabularies of the 20NG and the WebKB data sets and a UNIX word list (located at /usr/share/dict/words). Of these words, we chose all the words starting with the alphabets a, b, c, p, q, and r, which comprised a total of 5235 words. These words were manually grouped into 2069 classes, on the basis that all and only those words that were judged to be semantically and morphologically related were kept in the same group. As in [32], words with at least the first two letters in common were considered for grouping together. Thus, words such as *ran* and *run* or *buy* and *bought* were not stemmed to each other, but *bring* and *brought* were kept in the same group.

Paice has defined the following indexes for quantifying overstemming and understemming. Let W be the size of the given word sample, and let N_G and N_S denote the number of concept groups (denoted by g) and stem classes (denoted by s), respectively. In addition, let n_g and n_s denote the number of words in g and s, respectively. Now, suppose that g consists of words from k_g distinct stem classes, with u_{gi} instances from ith such class, and that s consists of words from l_s distinct concept groups, with v_{sj} instances from jth such group; the understemming index (UI) and the overstemming index (OI) are defined as follows:

$$\mathrm{UI} = \frac{\frac{1}{2}\sum_{g=1}^{N_G}\sum_{i=1}^{k_g} u_i(n_g - u_i)}{\frac{1}{2}\sum_{g=1}^{N_G} n_g(n_g - 1)} \tag{2}$$

$$\mathrm{OI} = \frac{\frac{1}{2}\sum_{s=1}^{N_S}\sum_{j=1}^{l_s} v_j(n_s - v_j)}{\frac{1}{2}\sum_{g=1}^{N_G} n_g(W - n_g)}. \tag{3}$$

Assuming that the concept groups are known, the denominators in the definitions of UI and OI are constant. The numerators of UI and OI are called "global unachieved merge total" and "global wrongly merged total," respectively, [32] and these undesired quantities should be zero for an ideal stemmer. Thus, a linguistic analysis of stemmers involves comparing how close both UI and OI are to zero.

We now describe the procedure for indirect evaluation of stemmers by studying the resulting classification accuracy and retrieval precision. For evaluating the performance of our system in refining the classification accuracy of a stemmer, we used the bow toolkit [34] and conducted the following experiments for each data set. The document collections are preprocessed as mentioned in Section IV, and the equivalence classes are split accordingly.

For training and testing, the data set is split randomly into two parts, and the same split is used for each stemmer. The proportion of documents chosen for training is first taken to be 60. The training and testing phases are repeated five times for each choice of the proportion.

The text-classification algorithms employed to compute the classification accuracy were NB [35], SVMs [36], [37], and MaxEnt [38]. Each document in the test set was given a classification score for each of the available categories. If a single category was to be assigned to a document, the one with the maximum score for that document was chosen. We computed the classification accuracy, which is the proportion of test documents assigned to the correct class. This value was also computed for each of the individual categories.

Classification accuracy measures the total number of correctly classified documents. However, when documents are misclassified, it does not distinguish between them on the basis of their classification scores. To take this into account, we have adopted the following precision-recall method. The documents are first sorted in descending order based on the classification scores. Only the largest score was considered for each document. Now, at any value of recall, the precision (or classification accuracy) is computed. A higher precision-recall curve is preferable. It may be noted that classification accuracy can be obtained from this curve as the precision when recall is set

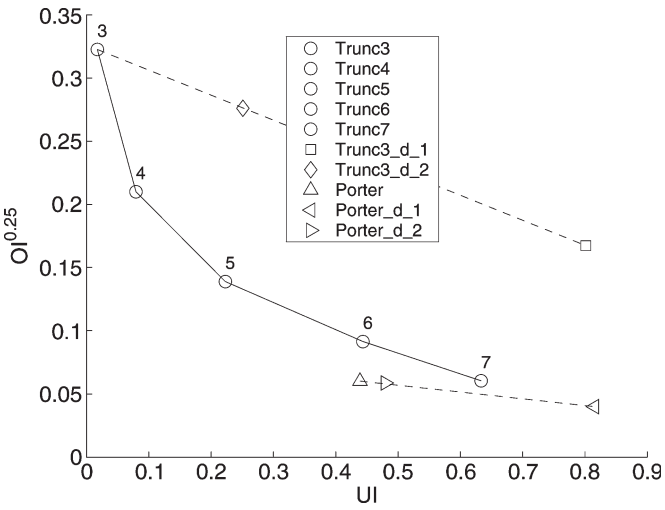


Fig. 1. Plot of OI versus UI for stemmers and their refinements based on the 20NG and WebKB data sets (OI raised to the power 0.25 for clarity).

to 100%. Experiments are also performed where the precision-recall curves are obtained for each individual category.

To evaluate the retrieval capabilities of our algorithm, we conducted the succeeding experiments on the WSJ data set using the SMART system. Topics 101–150 were chosen as the given queries. The word vector weighting was set to term frequency–inverse document frequency (TF–IDF). For each query, documents are retrieved, and the precision is noted at recall values set to 10%, 20%, ..., 100%. These precision values are averaged over all queries and are presented in the form of precision-recall plots in Fig. 4.

C. Error-Counting-Based Evaluation

Here, we perform a linguistic analysis of the stemmers under consideration. The UI and OI values were computed for all the stemmers based on both the 20NG and the WebKB data sets. These values are displayed graphically in Fig. 1. Dashed lines are drawn from the UI–OI values of *trunc3* and *porter* to those of their refinements (*trunc3_d*) to show how the errors changed on refinement. It may wrongly appear that while refining the *trunc3* stemmer, too many understemming errors are introduced at the cost of reducing a few overstemming errors. This impression is due to the fact that the denominator in (3) is much larger than that in (2). More insight may be gathered by looking at the *ang* example provided here.

We provide some examples of the equivalence classes produced by the proposed methodology by refining the Porter and Truncate(3) stemmers. The Porter stem class containing (*abort, aborts, aborted, abortion*) was split into two classes, whereby *abortion* was separated from the rest. Similarly, (*circularity*) was segregated from (*circular, circulars*). The stem group corresponding to *close* was split into three groups (*close, closing, closes*), (*closed*), and (*closely, closeness*). The preceding splits resulted from the differences in usage of the words in the collection. For example, even though semantically and morphologically, *circularity* is related to *circular*, this term and its plural have a different meaning in general and arise in a different context. Similarly, although the verb and adjective forms of *close* have been separated out, *closed* was made into a new group of itself, resulting in an understemming error.

Next, we present an example of refining the Truncate(3) stemmer. The stem class *ang* is split into the following eight classes (the first word of each group is the stem): 1) *(angel, angelic, anglican, anglo, angling, anguish, anglicans)*; 2) *(angeles, angelo)*; 3) *(angelino)*; 4) *(angels)*; 5) *(anger, angry, angola, angered, angelos, ang, angrier, angering)*; 6) *(angers)*; 7) *(angle, angles, angular, angst, angus, angled, angulated, angstrom)*; and 8) *(angmar)*. It may be noted that despite some overstemming (e.g., *angling* mixed with *angel*) and understemming (e.g., *angels* and *angers* are left out as singletons instead of being merged with *angel* and *anger*, respectively) errors, the splitting is largely successful as most of the related words appear in the same group, especially because no linguistic analysis is performed. In particular, *angstrom* has been retained in the same group as *angle*, perhaps as a consequence of both appearing in similar contexts.

The number of understemming and overstemming errors in this case, with the chosen sample being the set of words beginning with *ang*, are 13 and 54, respectively, while UI and OI turn out to be 0.35 and 0.15, respectively. A quick inspection reveals that there is more overstemming than understemming—had *angers*, *angling*, and *angels* been merged with the appropriate classes, the understemming would have been just 1.

D. Comparison of Classification Accuracy and Retrieval Performance

As described in Section III-B, the proposed distribution-based segregation methodology can be employed to refine the equivalence classes generated by any existing stemmer. In the present investigation, we used it to obtain new stemmers based on the Porter and Truncate(3) stemming for both the corpora. Let these new stemmers be denoted as *porter_d_i* and *trunc3_d_i*, where i is 1 for the 20NG data set and 2 for the WebKB data set. Similarly, the stemmers derived using Baker and McCallum's distributional clustering are denoted as *baker_1* and *baker_2*.

The reason for choosing Porter and Truncate(3) stemmers is described here. Porter's stemmer is one of the most standard stemmers as evident by its use in the literature. Truncate(3) is a stemmer that is used mostly for academic purposes. We have used that because it is a very strong stemmer and results in several overstemming errors, thereby providing significant scope for refinement.

The comparison process has four parts. In the first part, we compare the performance, in terms of the classification accuracies, of the refined new stemmers *porter_d_i* and *trunc3_d_i* with that of the original ones (i.e., *porter* and *trunc3*) and *no stemming* and *baker_i*. The objective is to demonstrate both the effectiveness of refinement by our method and improvement over the baseline performance where the original words are used.

In the second part, we compare these new stemmers with the co-occurrence-based modified Porter and Trunc3 stemming of Xu and Croft [7]. These may be denoted, in short, as *porter_c_i* and *trunc3_c_i*, with i being the same as in

TABLE I
CLASSIFICATION ACCURACIES FOR 20NG

Stemming Method	Classification Method		
	NaiveBayes	SVM	MaxEnt
no stemming	80.44	80.06	77.08
porter	79.37	79.73	77.40
trunc3	72.88	71.11	68.24
trunc4	78.68	77.41	73.64
trunc5	79.73	78.15	74.24
porter_c_1	79.59	79.57	77.46
trunc3_c_1	74.28	74.13	75.03
porter_d_1	80.34	80.21	77.53
trunc3_d_1	81.98	81.19	77.85
baker_1	81.38	81.19	77.15
trunc3_c_2	74.82	73.38	71.01
trunc3_d_2	74.73	73.51	70.59

TABLE II
CLASSIFICATION ACCURACIES FOR WebKB

Stemming Method	Classification Method		
	NaiveBayes	SVM	MaxEnt
no stemming	63.02	72.35	64.09
porter	61.86	70.14	62.76
trunc3	57.55	63.77	58.07
trunc4	60.85	66.56	50.67
trunc5	61.27	64.57	51.13
porter_c_2	62.00	70.16	62.64
trunc3_c_2	59.90	65.11	58.11
porter_d_2	63.08	71.38	63.91
trunc3_d_2	63.95	69.53	64.21
baker_2	61.19	69.51	64.43
trunc3_c_1	59.10	64.77	58.60
trunc3_d_1	61.39	69.51	64.13

the first part. Here, the objective is to compare the performance of our distribution-based refinement process with the co-occurrence-based refinement process. These stemmers are also compared with *baker_i*.

The third part deals with comparison in terms of cross-corpus performance, where a stemmer refined using information from one data set is applied to another data set. The objective is to study the dependence of a stemming algorithm on the data set based on which is derived and its applicability to a dissimilar data. Here, we consider *trunc3_d_1* and *trunc3_c_1* applied to the WebKB data set and *trunc3_d_2* and *trunc3_c_2* applied to the 20NG data set.

The fourth part involves comparison with respect to retrieval performance, where the objective is to study the effects of the stemmers on the retrieval at various levels of recall. The stemmers being considered here are *porter_c_1*, *porter_d_1*, *trunc3_d_1*, and *baker_1*, and they are applied to the WSJ data set. The stemmers refined on the 20NG data set are chosen for retrieval on the WSJ data set in order to evaluate how the refinements generalize to other data sets in case of retrieval.

E. Results

Tables I and II report the classification accuracies obtained by different stemming algorithms for the 20NG data set and the WebKB data sets, respectively. All abbreviations used in the result tables are described in Section V-D. We make the following observations from Tables I and II.

1) Both *porter_d* and *trunc3_d* fare better than *porter* and *trunc3*, respectively, in all cases for both data sets.
2) *porter_d* shows better performance than *porter_c* in all cases, although the number of stems obtained by *porter_d_1* is slightly more than that by *porter_c_1*.
3) *trunc3_d* provides better classification accuracy compared to *no stemming* and *baker*, with the case of SVM for the WebKB data set being the only exception. The same observation holds when *trunc3_d* is compared to *porter_d*.
4) The number of words common to both data sets is 20 782, which is about 64% and 37% of the dictionary sizes of the WebKB and 20NG data sets, respectively (see Table III). The classification accuracy obtained by *trunc3_d_1* is significantly better than *trunc3_c_1* when applied to the WebKB data set. This confirms that the refinement procedure performed by employing the classification information from a different corpus works better when the number of common words is large.
5) The statistical significance of the improvement gained by using the proposed stemmers over that of existing stemmers is tested using a t statistic. Table IV contains the t statistic values for the case of the NB classifier applied to the 20NG data set when the test set size is chosen to be 40% (Fig. 2). Table V shows the corresponding values for the WebKB data set. We observe in Table IV, which corresponds to Fig. 2, that there is a significant improvement, at 95% confidence level, over existing stemmers when *trunc3_d_1* is employed. Similarly, *trunc3_d_2* has significantly, at 95% confidence level, outperformed existing stemmers as can be seen in Table V, which corresponds to Fig. 3.

Apart from the classification accuracies, we also provide results in the form of precision-recall plots (Figs. 2 and 3). It may be noted that the proposed algorithm consistently outperforms the rest, especially when considered along with the number of stems obtained (Table III). When based on *trunc3*, the proposed method (i.e., *trunc3_d*) consistently improves the classification precision for all values of recall. This is possibly because the stems obtained by *trunc3_d* satisfy the NB independence assumption better than the original set of words. In that respect, our methodology may be considered as a "modification of feature sets to make the independence assumption more true," as mentioned in [35].

TABLE III
STEM COUNTS AND THE LARGEST EQUIVALENCE CLASSES OBTAINED BY VARIOUS STEMMING ALGORITHMS

Stemming Method	Data Set					
	20NG			WebKB		
	Stem Count	Largest Class	Index Compression	Stem Count	Largest Class	Index Compression
no stemming	56436			32299		
porter	40821	gener: 24	13.6%	23446	gener: 25	15.3%
trunc3	8158	con: 675	76.6%	5053	con: 432	81.6%
trunc4	21252	inte: 236	42.2%	13283	inte: 192	50.4%
trunc5	33187	inter: 174	24.4%	20051	inter: 147	31.6%
porter_c	47441	generic: 18	11.0%	26249	general: 24	25.6%
trunc3_c	30710	considered: 652	24.2%	14921	convex: 430	44.8%
porter_d	48502	general: 13	10.7%	23971	general: 25	27.0%
trunc3_d	22643	discussion: 212	41.2%	6163	contents: 261	72.5%
trunc3_c (cross)	13771	convex: 360	47.3%	11036	considered: 358	56.9%
trunc3_d (cross)	10708	con: 314	52.2%	13857	int: 85	49.5%

TABLE IV
STATISTICAL SIGNIFICANCE VALUES FOR 20NG USING NB

Existing Stemmers	Proposed Stemmers		
	porter_d_1	trunc3_d_1	trunc3_d_2
no stemming	0.17 (0.43)	2.45 (0.02)	0.27 (0.40)
porter	1.73 (0.06)	4.38 (0.00)	2.53 (0.02)
trunc3	11.43 (0.00)	13.70 (0.00)	10.40 (0.00)
trunc4	9.23 (0.00)	9.91 (0.00)	8.89 (0.00)
trunc5	7.17 (0.00)	7.83 (0.00)	7.43 (0.00)
porter_c_1	2.53 (0.02)	4.21 (0.00)	2.29 (0.02)
trunc3_c_1	4.91 (0.00)	7.87 (0.00)	6.51 (0.00)
baker_1	1.31 (0.11)	2.17 (0.03)	1.21 (0.13)
trunc3_c_2	7.72 (0.00)	9.16 (0.00)	6.28 (0.00)

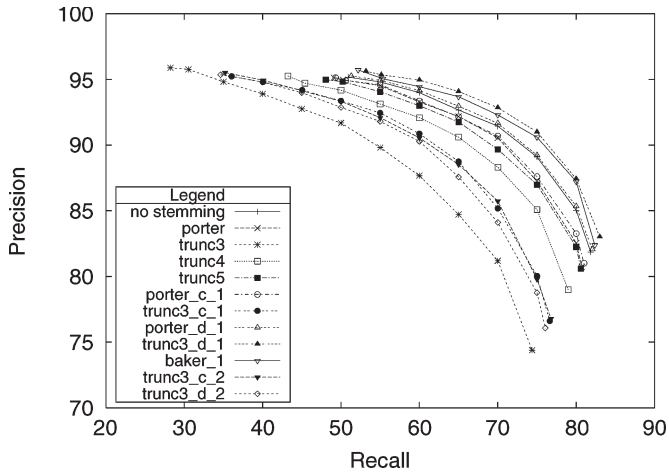


Fig. 2. Classification results for 20NG (test set size = 40, method = NB).

TABLE V
STATISTICAL SIGNIFICANCE VALUES FOR WebKB USING NB

Existing Stemmers	Proposed Stemmers		
	porter_d_2	trunc3_d_2	trunc3_d_1
no stemming	0.87 (0.20)	2.43 (0.02)	-0.36 (0.64)
porter	4.13 (0.00)	5.70 (0.00)	-0.15 (0.56)
trunc3	13.21 (0.00)	15.18 (0.00)	5.64 (0.00)
trunc4	9.27 (0.00)	12.91 (0.00)	5.14 (0.00)
trunc5	8.41 (0.00)	11.58 (0.00)	4.73 (0.00)
porter_c_2	3.93 (0.02)	5.81 (0.00)	-0.26 (0.60)
trunc3_c_2	7.80 (0.00)	9.37 (0.00)	2.11 (0.03)
baker_2	0.71 (0.25)	1.97 (0.04)	-0.61 (0.72)
trunc3_c_1	9.45 (0.00)	10.96 (0.00)	2.97 (0.01)

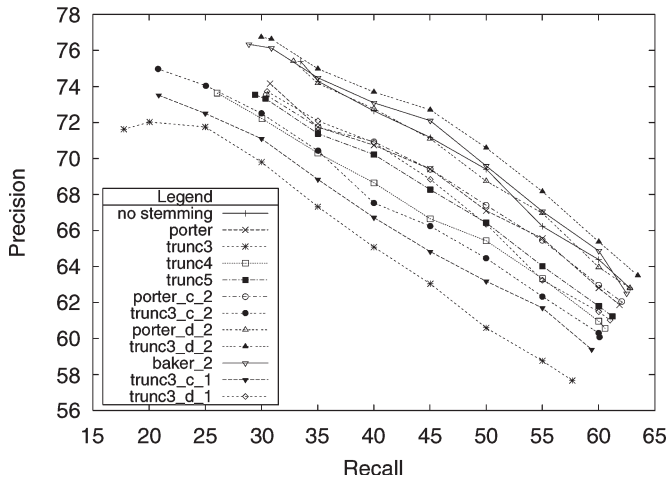


Fig. 3. Classification results for WebKB (test set size = 40, method = NB).

TABLE VI
TIME (IN SECONDS) TAKEN FOR CREATING THE STEM HASH TABLES

Method	Data Set	
	20NG	WebKB
porter_d	3.2	2.6
trunc3_d	5.5	5.1
porter_c	3.6	2.2
trunc3_c	9.2	4.7
baker	11.2	9.4

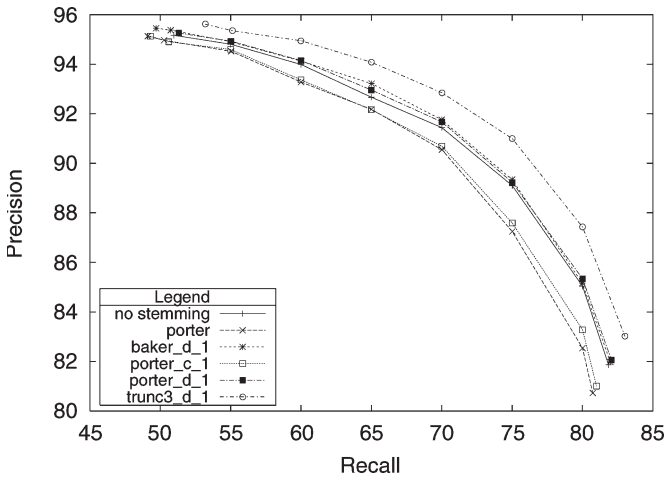


Fig. 4. Retrieval results for WSJ. The similarity measure is TFIDF.

Note here that the time taken for stemming words is the same for any of the stemmers, as the words and their stems can be stored in a hash table. It is the time taken for creating this hash table that may differ. While this is not significant for any of the rule-based stemmers, it is comparatively quite high for the co-occurrence-based stemmer, the proposed one, and *baker* (Table VI). The steep increase in the computation time for *trunc3_c* is a consequence of the large equivalence classes formed by *trunc3*.

For testing the retrieval efficiency of the proposed methodology, we have chosen a part of the WSJ data set that consists of WSJ articles published from 1987 to 1992. The queries used were topics 101–150. The retrieval experiments have been performed using the SMART system with the word vector weighting set to TF–IDF [39]. As mentioned earlier in Section V-B, we have used the refinements obtained from the 20NG data set. We note that *trunc3_d_1* outperforms *baker_1*, as well as the remaining methods, as shown in Fig. 4, with *trunc3_d* providing more than 2% improvement over *baker_1* until the recall is above 90%. The improvement has been found to be statistically significant at a 95% confidence level.

VI. CONCLUSION

We have described the design of a stemming algorithm that uses the classification information of a corpus to refine a given stemmer. The main advantage over other stemmers such as co-occurrence-based stemmers is its ability to drastically reduce the dictionary size while maintaining both classification accuracy and retrieval precision. Experiments that were conducted on the 20NG and WebKB data sets confirm the superiority of the proposed methodology for the task of text categorization when classifiers such as NB, SVMs, and MaxEnt are used. This is also supported by precision-recall-based evaluation. Another set of experiments performed on the WSJ data set demonstrates the enhancement in retrieval precision when refined stemmers are employed instead of existing stemmers. The performance of the refinement done by employing the classification information from a different corpus increases as the number of common words increases.

ACKNOWLEDGMENT

The authors would like to thank M. Mitra for providing the WSJ data set and for his help with the SMART system, and the anonymous reviewers for several insightful comments. N. L. Bhamidipati would like to thank the ISI-INSEAD (France) Fellowship for carrying out his doctoral research.

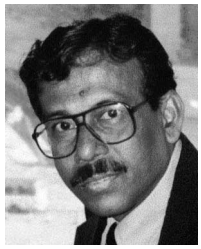
REFERENCES

[1] G. Salton, "Developments in automatic text retrieval," *Science*, vol. 253, no. 5023, pp. 974–980, Aug. 1991.
[2] W. Kraaij and R. Pohlmann, "Viewing stemming as recall enhancement," in *Proc. 17th ACM SIGIR Conf.*, Zurich, Switzerland, Aug. 1996, pp. 40–48.
[3] W. B. Frakes and C. J. Fox, "Strength and similarity of affix removal stemming algorithms," *ACM SIGIR Forum*, vol. 37, no. 1, pp. 26–30, 2003.
[4] F. Yamout, R. Demachkieh, G. Hamdan, and R. Sabra, "Further enhancement to Porter algorithm," in *Proc. Workshop Mach. Learn. and Interact. Text-Based Inf. Retrieval*, Ulm, Germany, Sep. 2004, pp. 7–24.
[5] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Discrete Multivariate Distributions*. Hoboken, NJ: Wiley-Interscience, 1997.
[6] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.
[7] J. Xu and W. B. Croft, "Corpus-based stemming using coocurrence of word variants," *ACM Trans. Inf. Syst.*, vol. 16, no. 1, pp. 61–81, 1998.
[8] L. D. Baker and A. K. McCallum, "Distributional clustering of words for text classification," in *Proc. 21st ACM SIGIR Conf.*, Melbourne, Australia, 1998, pp. 96–103.
[9] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, Nov. 1975.
[10] D. Harman, "How effective is suffixing?" *J. Amer. Soc. Inf. Sci.*, vol. 42, no. 1, pp. 7–15.
[11] J. B. Lovins, "Development of a stemming algorithm," *Mech. Transl. Comput. Linguist.*, vol. 11, no. 1/2, pp. 22–31, 1968.
[12] J. L. Dawson, "Suffix removal for word conflation," *Bull. Assoc. Lit. Linguist. Comput.*, vol. 2, no. 3, pp. 33–46, 1974.
[13] C. D. Paice, "Another stemmer," *SIGIR Forum*, vol. 24, no. 3, pp. 56–61, 1990.
[14] R. Krovetz, "Viewing morphology as an inference process," in *Proc. 16th ACM SIGIR Conf.*, Pittsburgh, PA, 1993, pp. 191–202.
[15] M. Kantrowitz, B. Mohit, and V. Mittal, "Stemming and its effects on TFIDF ranking," in *Proc. 23rd Annu. SIGIR Conf.*, Athens, Greece, 2000, pp. 357–359.
[16] T. Gustad and G. Bouma, "Accurate stemming of Dutch for text classification," *Lang. Comput.*, vol. 45, no. 1, pp. 104–117, 2002.
[17] M. Bacchin, N. Ferro, and M. Melucci, "A probabilistic model for stemmer generation," *Inf. Process. Manage.*, vol. 41, no. 1, pp. 121–137, 2005.
[18] F. Pereira, N. Tishby, and L. Lee, "Distributional clustering of English words," in *Proc. 31st Annu. Meeting Assoc. Comput. Linguist.*, 1993, pp. 183–190.
[19] D. Lin, "Automatic retrieval and clustering of similar words," in *Proc. 17th Int. Conf. Comput. Linguist.*, 1998, pp. 768–774.
[20] L. Lee, "Measures of distributional similarity," in *Proc. 37th Annu. Meeting Assoc. Comput. Linguist.*, 1999, pp. 25–32.
[21] Y. Yang, "A re-examination of text categorization methods," in *Proc. 22nd ACM SIGIR Conf.*, Berkeley, CA, 1999, pp. 42–49.

[22] E. Riloff, "Little words can make a big difference for text classification," in *Proc. 18th ACM SIGIR Conf.*, Seattle, WA, 1995, pp. 130–136.
[23] M. Spitters, "Comparing feature sets for learning text categorization," in *Proc. RIAO*, Paris, France, 2000, pp. 1124–1135.
[24] A. M. Cohen, J. Yang, and W. R. Hersh, "A comparison of techniques for classification and ad hoc retrieval of biomedical documents," in *Proc. 14th Annu. Text REtrieval Conf.*, Gaithersburg, MD, 2005. [Online]. Available: http://trec.nist.gov/pubs/trec14/papers/ohsu-geo.pdf
[25] A. Wald, *Sequential Analysis*. New York: Wiley, 1947.
[26] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York: McGraw-Hill, 1983.
[27] S. Kullback, *Information Theory and Statistics*. New York: Wiley, 1959.
[28] E. L. Lehmann, *Testing Statistical Hypotheses*. New York: Springer-Verlag, 1997.
[29] *The 20 News Groups (20NG) data set.* [Online]. Available: http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html
[30] *The WebKB data set.* [Online]. Available: http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/
[31] D. Harman, "Overview of the Third Text Retrieval Conference," in *Proc. 3rd TREC-3*, 1995, pp. 1–20.
[32] C. D. Paice, "Method for evaluation of stemming algorithms based on error counting," *J. Amer. Soc. Inf. Sci.*, vol. 47, no. 8, pp. 632–649, 1996.
[33] R. S. de Madariaga, J. R. F. del Castillo, and J. R. Hilera, "A generalization of the method for evaluation of stemming algorithms based on error counting," in *Proc. 12th Int. Conf. String Process. and Inf. Retrieval*, Buenos Aires, Argentina, 2005, pp. 228–233.
[34] A. K. McCallum, *Bow: A Toolkit for Statistical Language Modeling, Text Retrieval, Classification and Clustering*, 1996. [Online]. Available: http://www.cs.cmu.edu/~mccallum/bow
[35] D. D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval," in *Proc. 10th Eur. Conf. Mach. Learn.*, Chemnitz, Germany, 1998, pp. 4–15.
[36] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.
[37] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proc. 10th Eur. Conf. Mach. Learn.*, Chemnitz, Germany, 1998, pp. 137–142.
[38] K. Nigam, J. Lafferty, and A. McCallum, "Using maximum entropy for text classification," in *Proc. IJCAI Workshop Mach. Learn. Inf. Filtering*, 1999, pp. 61–67.
[39] S. Chakrabarti, *Mining the Web: Discovering Knowledge From Hypertext Data*. San Mateo, CA: Morgan Kaufmann, 2002.

Narayan L. Bhamidipati (previously known as B. Lakshmi Narayan) received the B.Stat. (with honors) and M.Stat. degrees from the Indian Statistical Institute, Kolkata, India, in 1998 and 2000, respectively. He is currently working toward the Ph.D. degree in computer science at the Indian Statistical Institute.

His research interests include data, web, and text mining; graph theory; information theory; and soft computing.



Sankar K. Pal (M'81–SM'84–F'93) received the Ph.D. degree in radio physics and electronics from the University of Calcutta, Kolkata, India, in 1979, and the Ph.D. degree in electrical engineering along with DIC from Imperial College, University of London, London, U.K., in 1982.

He was with the University of California, Berkeley and the University of Maryland, College Park during 1986–1987; the NASA Johnson Space Center, Houston, TX, during 1990–1992 and 1994; and the U.S. Naval Research Laboratory, Washington, DC, in 2004. Since 1997, he has been serving as a Distinguished Visitor of the IEEE Computer Society (USA) for the Asia-Pacific Region and has held several visiting positions in Hong Kong and Australian universities. He is also currently the Director and a Distinguished Scientist of the Indian Statistical Institute, Kolkata. He founded the Machine Intelligence Unit and the Center for Soft Computing Research: A National Facility in the Institute in Calcutta. He is a coauthor of 13 books and about 300 research publications in the areas of pattern recognition and machine learning, image processing, data mining and web intelligence, soft computing, neural nets, genetic algorithms, fuzzy sets, rough sets, and bioinformatics. He is currently an Associate Editor of Pattern Recognition Letters, Neurocomputing (1995–2005), Applied Intelligence, Information Sciences, Fuzzy Sets and Systems, Fundamenta Informaticae, the LNCS Transactions on Rough Sets, the *International Journal on Computational Intelligence and Applications*, the Proceedings of INSA-A, and the *International Journal on Image and Graphics, and the International Journal of Approximate Reasoning.*

Dr. Pal was a recipient of the 1990 S. S. Bhatnagar Prize (which is the most coveted award for a scientist in India) and many prestigious awards in India and abroad including the 1999 G. D. Birla Award, the 1998 Om Bhasin Award, the 1993 Jawaharlal Nehru Fellowship, the 2000 Khwarizmi International Award from the Islamic Republic of Iran, the 2000–2001 FICCI Award, the 1993 Vikram Sarabhai Research Award, the 1993 NASA Tech Brief Award (USA), the 1994 IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award (USA), the 1995 NASA Patent Application Award (USA), the 1997 IETE-R.L. Wadhwa Gold Medal, the 2001 INSA-S.H. Zaheer Medal, and the 2005-2006 ISC-P.C. Mahalanobis Birth Centenary Award (Gold Medal) for Lifetime Achievement. He is currently an Associate Editor of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the IEEE TRANSACTIONS ON NEURAL NETWORKS [1994–1998 and 2003–2006]. He is also a member of the Executive Advisory Editorial Board of the IEEE TRANSACTIONS ON FUZZY SYSTEMS. He is also a Guest Editor of the IEEE Computer. He is a Fellow of the Academy of Sciences for the Developing World (Italy), the International Association for Pattern Recognition (USA), and all the four National Academies for Science/Engineering in India.