A New Text Categorization Technique Using Distributional Clustering and Learning Logic

Hisham Al-Mubaid and Syed A. Umair

Abstract—Text categorization is continuing to be one of the most researched NLP problems due to the ever-increasing amounts of electronic documents and digital libraries. In this paper, we present a new text categorization method that combines the distributional clustering of words and a learning logic technique, called *Lsquare*, for constructing text classifiers. The high dimensionality of text in a document has not been fruitful for the task of categorization, for which reason, feature clustering has been proven to be an ideal alternative to feature selection for reducing the dimensionality. We, therefore, use distributional clustering method (IB) to generate an efficient representation of documents and apply *Lsquare* for training text classifiers. The method was extensively tested and evaluated. The proposed method achieves higher or comparable classification accuracy and F_1 results compared with SVM on exact experimental settings with a small number of training documents on three benchmark data sets *WebKB*, *20Newsgroup*, and *Reuters-21578*. The results prove that the method is a good choice for applications with a limited amount of labeled training data. We also demonstrate the effect of changing training size on the classification performance of the learners.

Index Terms—Text categorization, feature selection, machine learning.

1 Introduction

Text Categorization (TC) is the task of assigning a given text document to one or more predefined categories. This problem has received a special and increased attention from researchers in the past few decades due to many reasons like the gigantic amount of digital and online documents that are easily accessible and the increased demand to organize and retrieve these documents efficiently. The fast expansion of the *Internet* globally also has increased the need for more text categorization systems. Efficient text categorization systems are beneficial for many applications, for example, information retrieval, classification of news stories, text filtering, categorization of incoming e-mail messages and memos, and classification of Web pages [23].

A large number of machine learning, knowledge engineering, and probabilistic-based methods have been proposed for TC [7], [8], [12], [16], [23], [25], [28]. The most popular methods include Bayesian probabilistic methods, regression models, example-based classification, decision trees, decision rules, Rocchio method, neural networks, support vector machines (SVM), and association rules mining.

In this paper, we explore the application of a learning technique, called *Lsquare* [10], with the word-cluster representation computed by the *Information Bottleneck* (IB) method [4] into the text categorization problem. Lsquare is a logic-based supervised machine learning algorithm that generates classifiers from training data and the generated classifiers are DNF and CNF rules [10]. Lsquare extracts certain logic relationships from the training data and deduces those logic formulas that can correctly classify a given (*testing*) data set. It has achieved impressive performance in a number of classification-based NLP and data mining problems [1], [2], [9], [10]. In this paper, we explain the basic idea of Lsquare (Section 3.3) and how it is applied to text categorization. Then, we describe how text classifiers are generated and how test documents are classified.

The proposed TC method has been implemented and evaluated by a large number of experiments on three benchmark text collections: *WebKB*, *20Newsgroup*, and *Reuters-21578*. Our method achieves higher or comparable classification accuracies and F1 results than SVM on exact experimental settings with small amounts of training documents.

The main *contributions* of this paper are as follows: 1) The paper proposes a new TC technique based on an existing state-of-the-art feature clustering technique and a logic-based learning algorithm (Lsquare). The resulting TC system equally performs or outperforms one of the best performers in this task (i.e., SVM), as extensively verified through experiments. TC has been extensively researched with a large number of machine learning and rule-based techniques, but none of the previous work has explored the effectiveness of learning-logic in this problem. 2) The devised TC technique outperforms SVM across all three benchmark data sets on small training sizes where the training set contains *very* few examples. This is suitable for applications where *labeled* training data is very limited, like in classifying personal e-mail messages or memos or in classifying and organizing documents and files of a small organization. 3) The devised TC technique accepts only binary features (based on learning-logic), which makes it a perfect pick for applications in which features are essentially binary. Also, this aspect gives it an advantage in computational time complexity (i.e., it is faster to do

• *The authors are with the University of Houston-Clear Lake, 2700 Bay Area Blvd., Box 40, Houston, TX 77058.*
E-mail: hisham@uhcl.edu, syed.a.umair@gmail.com.

Manuscript received 20 Dec, 2004; revised 17 Nov. 2005; accepted 30 Mar. 2006; published online 19 July 2006.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0521-1204.

computations with binary values than with rational/real values).

The rest of this paper is organized as follows: Next, we describe prior related work. Section 3 describes the learning in Lsquare. Section 4 discusses text categorization by Lsquare. Then, in Section 5, the experiments and evaluation results are explained and discussed. Finally, Section 6 discusses the conclusions and future work.

2 Related Work

In this section, we briefly discuss the related research in text categorization and, for more details, we will refer to a certain publication, e.g., [23]. *Support Vector Machines* (SVM) have been prominently used for text categorization (Joachims [15] and Dumais et al. [8]) using the bag-of-words model. Dumais et al. [8] showed that using Mutual Information (MI) for feature selection combined with SVM outperforms other learners such as *Rocchio*, *decision trees*, *Naive Bayes*, and *Bayesian Nets*. Their results show a 92.0 percent break-even point on the 10 largest categories of the Reuters-21578 data set [8]. Moreover, Yang and Liu [27] and Joachims [16] confirmed the suitability and performance of SVM for text categorization using bag-of-words.

Decision tree learners attempt to select from training data some informative words using an information gain criterion, then predict the category of a document based on the occurrence of word combinations. Among the most popular decision tree-based methods are ID-3 and C4.5 [15], [18]. Decision rule methods generate classifiers by inductive rule learning. Examples of decision rule methods include Charade, DL-ESC, and RIPPER [23].

While traditional machine learners employ attribute-value representations, the use of logic programming representations led to the establishment of Inductive Logic Programming (ILP) [11]. The effectiveness of ILP methods for TC lies in formulating classifiers based on word order. One way to represent ordering information is with logic. Labeled training examples of the target class C can be represented as labeled ground facts of the form +c(d) or -c(d), where d is a constant that identifies a document, together with facts of the form wi(d, p). The fact wi(d, p) indicates that word wi appears in the document d at position p. In a typical ILP system, the c(d) facts would be used as training examples and the wi(d, p) facts would be used as background relations.

Several techniques have been proposed for feature selection and dimensionality reduction; see, for example, [28], [23], [17]. In Distributional clustering of words, words are represented as distributions over categories of the documents where they appear [20]. Using a naive Bayes classifier, Baker and McCallum [3] applied the distributional clustering scheme of Pereira et al. [20] for text categorization. Furthermore, Slonim and Tishby [24] used the Information Bottleneck (IB) method for clustering words and showed that the distributional clustering of Pereira et al. [20] is a special case of general IB clustering framework [25]. Both Baker and McCallum [3] and Slonim and Tishby [24] used agglomerative clustering algorithms and using Naive Bayes showed that the feature size is greatly reduced by using distributional clustering without significant loss of categorization accuracy.

Other methods such as Latent Semantic indexing (e.g., [14]) have been applied and have been shown to be inferior to feature clustering [3]. Dhillon et al. [7] proposed an information-theoretic framework that captures the optimality of word clusters in terms of generalized Jensen-Shannon divergence between multiple probability distributions. Slonim and Tishby [24] showed that word-clusters representation computed using the IB method can significantly improve classification accuracy, especially on a small training size of data set. Moreover, Bekkerman et al. [4] showed that feature clustering when compared to bag-of-words representation on the 20Newsgroup data set improves performance.

3 The Proposed Method

We propose a new TC method based on a successful feature clustering technique and a logic-based learning algorithm Lsquare. Our approach depends on representing the text document as a projection on clusters formed from the input data set, then applying the Lsquare learner to build text classifiers. Next, we present the distributional clustering approach using the Information Bottleneck method.

3.1 Word Features and Feature Clustering

Most of the TC methods use the vector space or bag-of-words model for representing document vectors [23], [7]. Each word in the documents corresponds to a feature in the vector representation. This leads to high dimensionality of text documents. Selecting features from text documents is one of the key issues in TC and received a fair amount of research with various methods of feature selection have been proposed; see, for example, [28], [17]. Word clustering is used to counter the issue of high dimensionality where similar words are grouped into clusters. One of the most prominent methods of word clustering for TC is the distributional clustering of words [3], [4], [7], [20], [24]. Each feature is basically a word cluster, and such feature clustering has been proven to be more effective than feature selection with word features techniques [4].

Clustering approaches attempt to increase the performance of the classifiers and to alleviate some issues affecting TC like high dimensionality and data sparseness. High dimensionality may not be an issue for some learners (e.g., SVM [15]), but, for other learners, it could pose a problem in terms of "loss of information." Involving a large number of words in forming the clusters minimizes the need for aggressive feature selection, so there is no loss of words (information), which could have been discarded if (aggressive) feature selection is used. Clustering approaches also solve the problem of sparseness as many words are grouped into one cluster and, hence, the probability of the presence of a feature in the projection of a text document is increased.

3.2 Distributional Clustering Using the IB Method

Clustering of words is an effective alternative to feature selection mechanisms where "*similar*" words are grouped into word-clusters [24]. The notion of distributional clustering was first introduced by Pereira et al. [20]; the method consists of finding the cluster hierarchy of one set (*nouns*) based on the similarity of their conditional distribution with regard to another set (*verbs*). We employ a method that uses the more general framework of Information Bottleneck to form the clusters [4]. The IB method provides an optimal solution to the question of similarity measure between the words, and provides a framework for clustering (more details in [25]). The IB method finds the efficient relevant coding or the compact representation of one variable X, given the joint distribution of two random variables $\mathrm{P(X, Y)}$, while the mutual information about the other variable Y is preserved as much as possible. Then, IB tries to solve the problem of

$$\mathrm{Maximize\ I(\tilde{\ }X, Y) - \beta\, I(\tilde{\ }X, X)\ over\ P(\tilde{\ }X|X)},$$

where X is a random variable from a data set given by i.i.d. Observations: ~X are the desired partition of X. $\mathrm{I(\tilde{\ }X; Y)}$ is the mutual Information between $\tilde{X}$ and a variable Y, and β determines the allowed reduction in formation the partition has. In the method employed in [4], X represents the input words and variable Y represents the class labels. Moreover, they provide a hierarchical top-down clustering procedure for producing the distributional IB clusters [4]. Starting with one cluster that contains all the input data, the clusters are split in each iteration with incrementing the annealing parameter β.

3.3 Lsquare

The classification technique employed for text classification is the *Lsquare* method of Felici et al. [9], [10]. The basic idea of *Lsquare* is as follows: *Lsquare* is a two-class classification technique that is based on learning logic. It views the training data as logic formulas and the resulting classifiers are logic formulas as well. The logic formulas are represented as vectors of $\{0, \pm1\}$ entries. Each vector represents one document, while each $\{0, \pm1\}$ entry in the vector represents a term/feature in that document.

The Lsquare system accepts as input two sets A and B of $\{0, \pm1\}$ vectors, all having the same length. Lsquare outputs (from the training data) a set of 20 disjunctive normal form (DNF) logic formulas and 20 conjunctive normal form (CNF) logic formulas. Each logic formula is capable of classifying any (new) vector to whether it belongs to class A or B. These logic formulas are called a *separating set* and this separating set will be our text classifier, as explained later. The following discussion explains how the separating sets are constructed and deduced from the training data.

3.3.1 *Generating Separating Sets*

We want to differentiate the vectors of a given set B from the vectors of a given set A by using a set S of $\{0, \pm1\}$ (*separating*) vectors. We need some definitions before we continue. A $\{0, \pm1\}$ vector f is nested in a $\{0, \pm1\}$ vector g if, for any entry fi of f equals to 1 or -1, the corresponding entry gi of g satisfies gi $=$ fi. Thus, f is not nested in g if and only if there is some fi $= \pm1$ of f for which gi $= -$fi or gi $= 0$. Let A and B be sets of $\{0, \pm1\}$ vectors of the same length say $\mathrm{n} \geq 1$. For any $\mathrm{b} \in \mathrm{B}$, a $\{0, \pm1\}$ vector s (separating vector) of length n separates b from A if:

1. s is not nested in any $\mathrm{a} \in \mathrm{A}$ and
2. s is nested in b.

Such a separation makes sense only if both A and B are nonempty and, if each record of A or B contains at least one $\{\pm1\}$, with this, item 1 implies that s is nonzero. A set S of $\{0, \pm1\}$ vectors separates B from A, called the separating set, if each $\mathrm{s} \in \mathrm{S}$ satisfies item 1 and if, for each $\mathrm{b} \in \mathrm{B}$, there is an $\mathrm{s} \in \mathrm{S}$ that satisfies item 2. Lsquare system converts this into a logic system and solves it to find S. This step is repeated recursively to find the maximum number of $\mathrm{b} \in \mathrm{B}$ vectors that can be separated from A. Then, the process is repeated by swapping the roles of the sets A and B. This way, two separating sets can be created to differentiate between A and B. Moreover, Lsquare partitions the set A (respectively, B) into d disjoint sets of essentially equal cardinality, say $\mathrm{A_1, A_2, \dots, A_d}$ (respectively, $\mathrm{B_1, B_2, \dots, B_d}$). Then, it selects an integer c satisfying $\mathrm{d/2 < c < d}$, then derive from $\mathrm{A_1, A_2, \dots, A_d}$ and $\mathrm{B_1, B_2, \dots, B_d}$ certain sets $\mathrm{A^k}$ and $\mathrm{B^k}$, where $\mathrm{k = 1, 2, \dots, d}$. Specifically, $\mathrm{A^k = A_k \cup A_{k+1} \cup \dots \cup A_{k+c-1}}$ and $\mathrm{B^k = B_k \cup B_{k+1} \cup \dots \cup B_{k+c-1}}$, the subscripted indices are interpreted in circular fashion. Thus, $2\mathrm{d}$ separating sets are created from A and B by repeating the same process of computing S with $\mathrm{A = A^k}$ and $\mathrm{B = B^k}$ [9], [10].

4 Text Categorization

Our text categorization approach depends on representing the text document as a projection on word clusters, then applying the Lsquare to build the text classifiers. Since we compare our approach with SVM on the same experimental settings, we include, in this section, some details about SVM and combining SVM with distributional clustering for TC.

4.1 Representing Text Documents

We transform text documents into a representation suitable for our learning algorithm *Lsquare*. We convert each document to a vector where the entries of the vector are features of word clusters. We apply the clustering algorithm [4] on each data set to obtain an effective representation of input documents. The input words are represented as distributions over all the categories of the whole data set. The output of the algorithm includes assignment of input words to a number of clusters with association weight of cluster to word. The total number of clusters required is prespecified to the algorithm. In our approach, we use *hard clustering* where we assign only one cluster to each input word. (*In this work, we used 120 clusters in all experiments; this was chosen after an extensive testing; we found 120 is the most effective in terms of performance and computational time.*) The resulting clusters include almost all the words in the input data except the stopwords (40,558 stopwords for WebKB, 199,150 stopwords for 20NG, and 22,597 stopwords for Reuters-21578). The clustering approach automatically gives a word stemming effect as words with the same stem are placed into same cluster.

The training and testing documents are represented as vectors of dimension k, (*k is the number of clusters, in the evaluations, we chose k = 120*). Since Lsquare accepts only binary data ($\{0, \pm 1\}$ vectors), features are assigned binary values with $+1$ to indicate the presence of any of the words of the cluster in the document, and -1 represents the case when none of the words of the cluster occurs in the document. For SVM, each feature in the vector space represents the word cluster count, i.e., the number of words of the cluster that appeared in the document; which is the same as in previous similar work [4], [7].

4.2 SVM with Distributional Clustering

Support vector machines (SVM) [26], [5] method was applied to text categorization and achieved excellent results [8], [15]. SVM is an inductive learning technique for two-class classification. A considerable amount of theoretical justifications is present in the literature to support SVM. Furthermore, SVM was extensively applied into other areas and achieved remarkable results. It has been proven, in TC research, that SVM is one of the best learning algorithms.

SVM attempts to determine a linear decision surface (or hyperplane) which separates the positive and negative training examples with a maximum margin in case of linearly separable data. For nonlinearly separable data, two techniques are used, soft-margin hyperplanes, and mapping the input data vectors into a higher dimensional space where a maximum margin hyperplane can be computed [26].

We use a linear SVM in all our experiments as most of the related work. The implementation we used is the linear SVM-light (by Joachims, available from http://svmlight.joachims.org) with the default parameters. As with *Lsquare*, we construct with SVM a classifier for each category and this is identical to the settings and procedure applied for *Lsquare*. We use the IB distributional clustering [4] to compute the word clusters. Then, each word cluster is a feature/entry in the document vector.

4.3 Lsquare with Distributional Clustering

Lsquare is a binary classifier that operates on a vector space model of documents. The training data are modeled as a collection of vectors, one for each document. Then, applying Lsquare to the training vectors will generate a classifier L. In the experiments, we construct a classifier for each category. For a given category C, the training data consists of two sets of vectors, one set for positive examples and an other for negative examples of that category. And, the sequence of steps of the training phase is outlined in Algorithm 1. We always select an equal number of positive and negative training documents. For example, when the training size is 40 documents, we randomly select 20 positive and 20 negative documents to train the classifiers.

```
Algorithm 1: Text Classification with Lsquare
                The training phase
Input: training dataset $D$ of text documents.
Output: a classifier $L_i$ for each category $C_i$.
For each category $C_i$, do the following:
   1. Extract from $D$ the documents representing positive
      examples on $C_i$ call this set $PT_i$ (positive examples are
      documents labeled with category $C_i$)
   2. Randomly select $|PT_i|$ documents from $D$
      representing negative examples on $C_i$, call this set $NT_i$
   3. Convert set of documents $PT_i$ to set of vectors $PV_i$
      using word clusters as features.
   4. Convert set of documents $NT_i$ to set of vectors $NV_i$
   5. Create the training data set $T_i$ for category $C_i$:
            $T_i \leftarrow \{PV_i, NV_i\}$
   6. Apply Lsquare to $T_i$ to generate classifier $L_i$
```

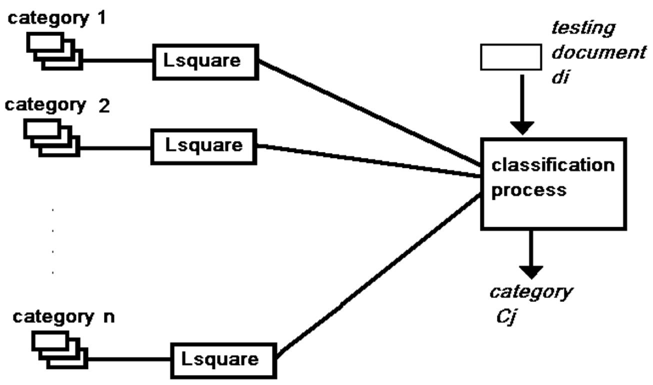


Fig. 1. Text categorization with Lsquare.

One of the issues facing the learning in TC is the imbalanced data problem, that is, the availability of a much larger negative set compared to a positive set size. This occurs when considering all noncategory documents as the negative set, especially when there are a large number of categories [29]. A number of techniques have been proposed to address this imbalanced data problem (Hearst et al. [13] and Ruiz and Srinivasan [22]). In our method evaluation, we simply select equal amounts of positive and negative training documents, and we repeat each experiment 10 times with different randomly selected positive and negative documents. During the testing phase, all classifiers generated for all categories constitute the testing process and will be used to determine the category label of a given testing document, see Fig. 1. In our experiments and for each generated classifier, we use all the remaining documents which were not used for training as a testing set to evaluate the accuracy of that classifier.

4.4 Classifying New Documents with *Lsquare*

Now, for a given testing document d, we apply d to each classifier L_i of each category C_i. Then, if d actually belongs to category C_i and L_i classifies it so, we count this as correct; otherwise, it is counted as an error in category C_i. In our experiments and for each classifier L_i of category C_i, we record the a, b, c, and d values as follows:

- a = # of C_i documents that L_i classifies into C_i.
- b = # of non-C_i documents that L_i classifies into C_i.
- c = # of C_i documents that L_i classifies as non-C_i.
- d = # of C_i documents that L_i classifies as non-C_i.

TABLE 1
Categorization Performance in Terms of Accuracy, Break-Even Point, and F1 for Lsquare and SVM Using WebKB on Different Training Sizes

Training Size *# of docs*	**Accuracy**		**Break-Even Point**		**F1**	
	Lsquare	**SVM**	**Lsquare**	**SVM**	**Lsquare**	**SVM**
10	70.72	58.67	69.40	67.38	66.02	64.95
20	93.61	78.38	87.21	75.12	86.99	74.33
30	93.40	91.25	87.92	84.24	87.67	84.00
40	94.34	91.27	88.68	85.23	88.35	85.20
50	95.19	91.27	90.40	85.23	90.19	85.20
60	95.55	91.44	91.36	85.63	91.21	85.62
80	97.27	93.65	93.95	88.30	93.93	88.30
100	96.80	94.59	93.20	88.69	93.14	88.69
120	97.32	94.33	94.10	89.34	94.06	89.27
150	97.82	96.17	94.69	91.93	94.67	91.93
200	97.88	96.05	94.79	91.42	94.78	91.41
Macroaverage	**93.63**	**87.99**	**89.61**	**84.16**	**89.18**	**83.80**
Microaverage	**96.55**	**93.18**	**92.92**	**88.34**	**92.81**	**88.25**

5 Experiments and Results

The TC approach proposed in this paper has been fully implemented and evaluated with extensive experimentation; this section presents the details of implementation, data sets, and test results. To evaluate our approach, we compare it fairly against SVM under the same experimental settings to allow for direct comparison.

5.1 Evaluation Methodology

A number of the metrics used in TC are evaluated and measured for categorization effectiveness. We use the well-known *precision* and *recall* metrics. *Precision* is defined simply as the ratio of correctly assigned category C documents to the total number of documents classified as category C. *Recall* is the ratio of correctly assigned category C documents to the total number of documents actually in category C. We also can define precision and recall in terms of a, b, c, and d values defined above (in Section 4.4) as:

$$Precision = a/(a+b) \quad Recall = a/(a+c).$$

Some combination of precision and recall can be more effective in measuring classifier performance. Such measures include *F-measure* and precision-recall *Break Even Point* (BEP). F-measure is known by calculating the harmonic mean of precision (P) and recall (R) and F1 is computed as:

$$\mathrm{F_1 = 2PR/(P+R).}$$

The precision-recall breakeven point is the point where precision is equal to recall and is often determined by calculating the arithmetic mean of precision and recall. We also use the accuracy in this paper as a measure and the accuracy is computed as the ratio of correctly classified testing documents to the total number of testing documents. Of course, all these performance metrics are computed for each category separately (*per category*), that is, we apply all the testing documents to each classifier Li (*classifier Li is for category Ci*) to computer P, R, F1, and accuracy for that category Ci. Then, we calculate the average of all categories. For example, in Table 1, the first line indicates that, using Lsquare, the average accuracy of the four classifiers (*of WebKB data set*) when training on only 10 documents and testing the remaining documents (4,189 documents) of the WebKB data set is 72.72 percent. Further, Table 2 and Table 3 contain the accuracy of each WebKB category/classifier separately. The reported performance results are the average of 10 trials in each experiment. The overall performance of an approach is known with the help of microaverage or macroaverage of the accuracies, F1, or BEPs for all categories. In microaverage, each document is given an equal weight, while macroaverage gives each category equal weight. That is, in Microaverage, a category with a large number of documents has more influence on the result than one with a smaller number of documents.

TABLE 2
Break-Even Point for SVM and Lsquare Using 200 Training Examples of Each Category of *WebKB* (Using Word Clusters)

Category	**BEP Lsquare**	**BEP SVM**
course	94.53	92.82
faculty	97.13	92.90
student	97.50	93.76
project	90.02	86.21
Average	**94.79**	**91.42**

TABLE 3
Break-Even Point for SVM on *WebKB* from [15] (Using Word Features)

Category	**BEP SVM**
course	94.2
faculty	79.0
student	53.3
project	89.9
Average	**79.1**

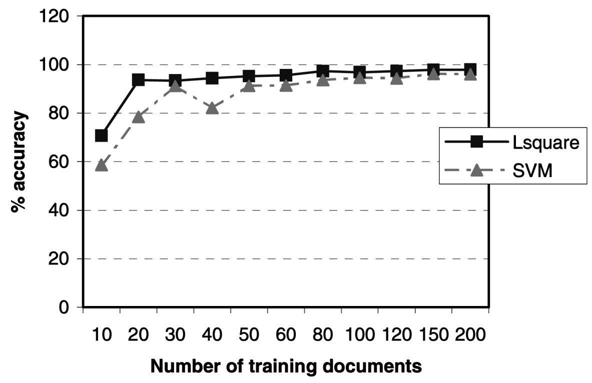


Fig. 2. Accuracy result on the *WebKB* data set.

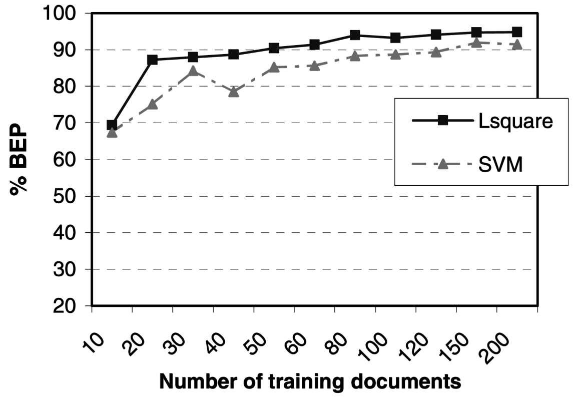


Fig. 3. Break-even point results on *WebKB*.

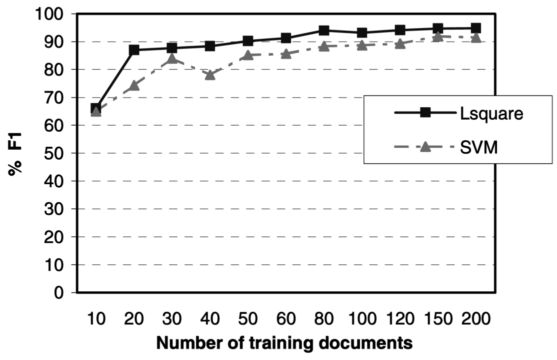


Fig. 4. F1 results on *WebKB*.

5.2 Data Sets

The experimental evaluation was performed on three well-known data sets in text categorization research WebKB, 20 Newsgroup (20NG), and Reuters-21578.

The *WebKB* (World Wide Knowledge Base) data set is a collection Web documents extracted from four academic domains. WebKB was collected by Craven et al. [6]. This collection contains 8,282 Web pages and seven categories in total. However, following an earlier study using this data set (e.g., [19]), we used only four categories in our experiments: course, faculty, project, and student, with a total of 4,199 documents. This is the only data set in our experiments that is purely unilabeled, i.e., each document belongs to only one category.

The *20 Newsgroups* (20NG) corpus contains almost 20,000 articles taken from the Usenet newsgroups [30]. These articles are evenly distributed on 20 categories; actually, each category is a discussion group in this newsgroup. Furthermore, each article is assigned into one or more categories. Only less than 5 percent of the articles in this set belong to more than one category.

The *Reuters-21578* data set contains 21,578 news articles from the Reuters newswire [21]. Each article belongs to one or more categories. The *ModApt* split of this collection contains 9,603 training documents and 3,299 test documents in 135 categories. Of the 135 categories, only 90 categories have at least one training and one testing documents. In our experiments, following other TC projects, we ran a test on the 10 most populated categories (top 10), which are the 10 categories having highest number of documents.

These three data sets differ in the context such that only a few keywords can efficiently categorize the Reuters-21578 and WebKB data sets, while, for more complex data sets such as 20NG, even low frequency words have an effect on the categorization results [4].

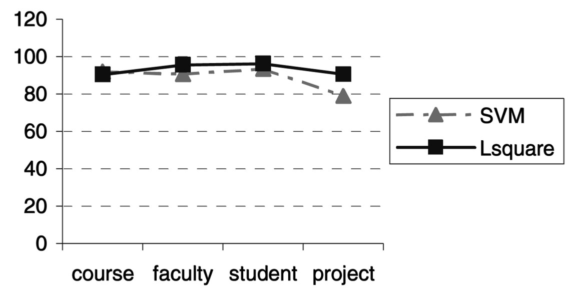
We used different training sizes ranging from 10 examples to 200 examples to train text classifiers. For training size n, to generate a classifier for category C, n/2 positive examples (documents from the category C) and an equal number ($\sim n/2$) of negative examples from categories other than C are used. Then, we test the classifier on the remaining documents not used for training from all categories. Reuters-21578 is already split into training and testing sets, so we select the training documents from the documents marked for training in the ModApt split. For 20NG and WebKB, there is no such split; the training documents are first selected randomly and the remaining documents are used as testing documents. Bekkerman et al. [4] used 50 and 300 clusters using a limited number of words in each cluster following Dumais et al.'s [8] settings for number of features used.

TABLE 4
Accuracy, Break-Even Point, and F1 Results of Lsquare and SVM Using 100 Training Examples of Each Category of WebKB

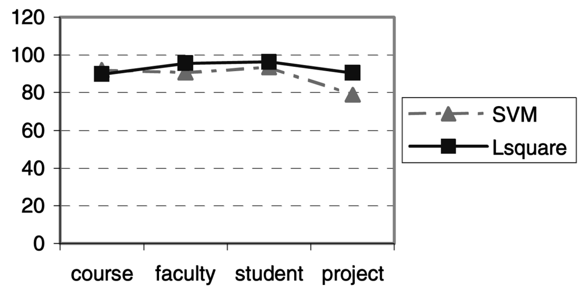
Training Size=100 doc	Lsquare			SVM		
Category	**Acc**	**BEP**	**F1**	**Acc**	**BEP**	**F1**
course	94.92	90.41	89.73	96.49	91.87	91.65
faculty	97.59	95.58	95.58	94.59	90.59	90.55
student	96.98	96.20	96.16	94.50	93.35	93.34
project	97.72	90.59	90.50	92.78	78.97	78.52
Average	**96.80**	**93.20**	**92.99**	**94.59**	**88.69**	**88.52**

TABLE 5
Accuracy, Break-Even Point, and F1 Results of Lsquare and SVM Using 200 Training Examples of Each Category of WebKB

Training Size=200 doc	Lsquare			SVM		
Category	**Acc**	**BEP**	**F1**	**Acc**	**BEP**	**F1**
course	97.33	94.53	94.38	96.92	92.82	92.70
faculty	98.45	97.13	97.12	95.95	92.90	92.84
student	97.98	97.50	97.46	94.44	93.76	93.65
project	97.76	90.02	89.91	96.89	86.21	86.19
Average	**97.88**	**94.79**	**94.73**	**96.05**	**91.42**	**91.36**



(a)



(b)

Fig. 5. Illustration of the performance results of Lsquare and SVM using 100 training examples for each category of the WebKB data Set: (a) BEP and (b) F1.

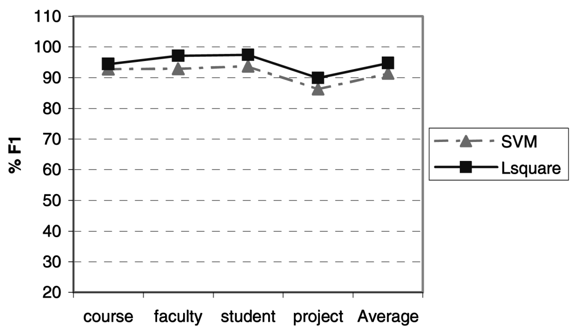


Fig. 6. Illustration of F1 results of Lsquare and SVM using 200 training examples for each category of WebKB.

TABLE 6
Performance Results on Different Training Size of *Reuters*

Training Size (*# of documents*)	**Accuracy**	
	Lsquare	**SVM**
10	87.48	84.97
20	92.67	94.23
30	93.72	95.37
40	94.63	96.09
50	95.32	96.46
60	95.36	96.81
80	95.87	97.19
100	95.96	97.09
120	96.16	97.51
150	96.39	97.52
200	96.75	97.61
Macroaverage	**94.57**	**95.53**

5.3 Experimental Results and Discussion

5.3.1 WebKB

Numerous sets of experiments were conducted on the WebKB data set. Table 1 shows the accuracy, break-even point, and F1 results of Lsquare and SVM on different training sizes of this data set. These results also are illustrated graphically in Figs. 2, Fig. 3, and Fig. 4. (*Recall that, in each line in Table 1, all the remaining documents after training (not used for training) are used for testing*.) Moreover, each test is done 10 times and, in every time, we change the randomly selected training examples. And, the results shown in the tables are the average of the 10 trials. These results show clearly that Lsquare outperforms SVM across all training sizes with significant differences in the accuracy and F1 metrics. Using the same clustering approach, Bekkerman et al. achieved a macroaveraged accuracy of 89.5 percent over all categories on 75 percent of training data and testing on 25 percent of data [4]. We notice that Table 1

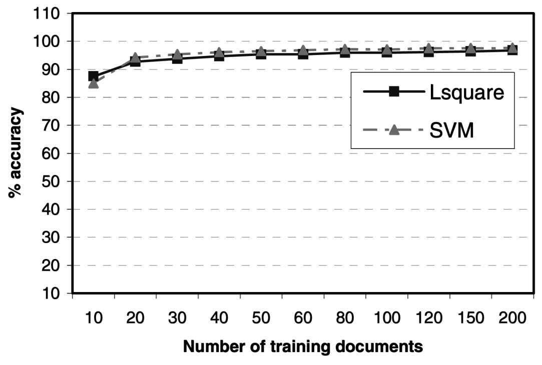


Fig. 7. Performance results of *Reuters*.

TABLE 7
Performance Results of Lsquare and SVM Using 20NG on Different Training Sizes

Training Size (*# of documents*)	**Accuracy**		**Break-Even Point**		**F1**	
	Lsquare	**SVM**	**Lsquare**	**SVM**	**Lsquare**	**SVM**
10	86.83	78.73	66.09	56.87	57.33	52.04
20	96.14	92.22	81.51	67.28	79.10	66.12
30	96.98	92.11	85.00	72.05	83.15	71.55
40	97.92	94.79	87.68	76.86	86.44	76.19
50	97.33	96.26	86.12	79.70	84.43	79.43
60	98.11	97.37	87.55	80.07	86.32	80.02
80	98.18	96.88	87.99	83.96	86.80	83.80
100	98.22	98.43	87.97	85.78	86.82	85.76
120	98.27	98.33	88.47	86.67	87.26	86.61
150	98.44	98.60	89.23	88.37	88.19	88.28
200	98.35	98.77	88.83	90.26	87.68	90.26
Macroaverage	**96.80**	**94.77**	**85.13**	**78.90**	**83.05**	**78.19**
Microaverage	**98.00**	**97.42**	**87.80**	**84.82**	**86.45**	**84.62**

also shows the stability of the method performance such that the performance increases proportionally with the training size. The results in Table 1 further prove that Lsquare is good for application with a limited/small number of labeled training data. The detailed results decomposed into four categories for a training size of 200 documents (of Table 1) are included in Table 2. Furthermore, we report from the literature the results of SVM on the WebKB data set in Table 3 (taken from [15]) to allow for more comparisons. Again, the Lsquare outperforms SVM in all categories (Table 2).

When considering the results for each category of WebKB using 100 and 200 training examples (Table 4 and Table 5, respectively), we observe that Lsquare shows better performance than SVM on most of these categories with 100 documents training size and on all categories when the training size is 200 documents. Table 4 and Table 5 show these results (*part of the results in Table 5 are already mentioned in Table 2*). Table 4 and Table 5 demonstrate, further, that the performance of the method is stable across all the categories and is not leaned/biased toward one or a group of categories. Fig. 5 shows the BEP and F1 results of SVM and Lsquare achieved on training size of 100 examples on WebKB, while Fig. 6 illustrates the F1 results using 200 training examples for each category of WebKB.

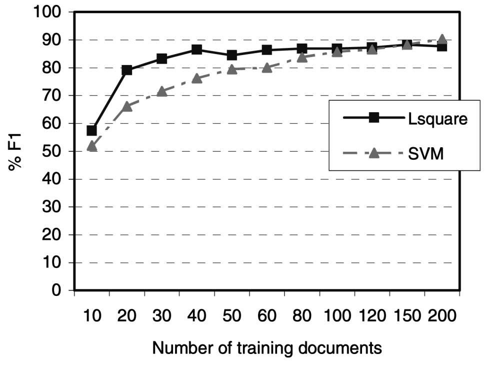


Fig. 8. F1 results of *Lsquare* and SVM on the 20NG.

5.3.2 Reuters-21578

The results from experiments on the Reuters-21578 data set are in Table 6 and illustrated in Fig. 7. We notice that, in this particular data set, SVM performs a little better than Lsquare, and the reason for this is the high sparseness of this data set, and that it is not as organized as the other data sets.

5.3.3 20NG

We also used the 20NG data set to evaluate Lsquare and SVM on the same numbers of training examples. Table 7 shows the Accuracy, Break-Even Point (BEP), and F1 results of both Lsquare and SVM. The graphical illustration of F1 results is in Fig. 8. These results show that Lsquare can perform better than SVM on most training sizes. The macroaveraged and microaveraged accuracy, BEP, and F1 of *Lsquare* are significantly better than those of SVM. We also notice that Lsquare outperforms SVM on smaller training sizes and both have comparable performances on larger training sizes. This suggests that Lsquare is more effective when there is a small number of training examples. Thus, Table 7 suggests that our method can be a good choice when we have limited amount of labeled training data.

In another set of tests on *20NG*, to evaluate our method and to find the effect of the changing number of categories on the performance, we performed experiments using only six arbitrarily chosen categories of 20NG. These categories do not correspond to each other and are not representative of remaining categories. This experiment is to illustrate the performance of our method when using fewer categories of the same data set and to see the effect of the changing number of categories on both learners. The results are shown in Table 8 and illustrated in Fig. 9. These results again demonstrate that Lsquare performed slightly better than SVM on a subset of the data set. Also, Table 8 shows the stability of the method across the categories. Furthermore, this is another evaluation that shows that Lsquare is more effective when we have a small number of categories. All these experiments and results present an experimental

TABLE 8
Accuracy, Break-Even Point, and F1 for Lsquare and SVM on 100 Examples of Six Categories of *20NG*

Training Size = 100 doc	Lsquare			SVM		
Category	**Accuracy**	**BEP**	**F1**	**Accuracy**	**BEP**	**F1**
alt.atheism	98.74	95.68	95.66	98.46	94.72	94.69
comp.windows.x	99.61	98.93	98.93	98.76	96.60	96.52
rec.autos	99.47	98.57	98.57	99.33	98.18	98.16
rec.sport.hockey	99.72	99.27	99.26	99.21	97.90	97.85
sci.electronics	97.60	94.14	93.83	99.74	99.30	99.30
talk.politics.guns	98.64	94.29	94.21	97.79	90.06	89.69
Average	**98.96**	**96.81**	**96.74**	**98.88**	**96.13**	**96.04**

justification of the effectiveness of our proposed method for text categorization.

The uniqueness and key points of strength of Lsquare are as follows: Lsquare deduces multiple classifiers from the training data. Recall that Lsquare derives from the training data (the sets A and B) d subsets $A^1, A^2, \ldots, A^d$, and $B^1, B^2, \ldots, B^d$, as explained in Section 3.3. Then, it creates two classifiers for each combination $\{A^k, B^k\}$ $(k = 1, 2, \ldots, d)$. Let us assume that $d = 10$, then 20 classifiers are created from A and B. A new testing vector is classified based on the collective decision of these 20 classifiers. Moreover, by computing the maximum number of $a \in A$ (respectively, $b \in B$) vectors that can be separated from B (respectively, from A), Lsquare captures most of the peculiarities of each $\{A, B\}$ training case. That is, Lsquare calibrates the derived classifiers (logic formulas) to the most discriminating attributes between A and B in each particular $\{A, B\}$ training combination. For an $\{A, B\}$ training combination, each one of the 20 derived classifiers captures certain discriminating features between A and B. Thus, the resulting collective classification decision is based on a rich set of discriminating features captured by 20 different classifiers.

The computational complexity of the approach depends mainly on the training size and on the number of word clusters used in document representation (dimensionality of document representation). For example, in the WebKB data set (4,199 documents) and with 200 training documents, a complete training for one experiment could take $\sim$ 1-2 hours to cluster words and generate/train the classifiers (four classifiers), using a modest machine (Pentium 4, 2.0GHz, 512MB RAM running Windows 2000 or sun Ultra SPAR 10 workstation). However, the testing phase is extremely fast, on the order of seconds (*less than half a minute in many cases*) to categorize all testing document (the highest testing times are for 20NG data set because we need to classify $\sim 20,000$ documents using 20 classifiers, see Table 9). These timings are very competitive if compared with previously published time complexities; see, for example, [4], [7]. The computational times for 20NG (20,000 documents) experiments are showed in Table 9 (On average, 7.2 sec are needed to classify 1,000 documents into 20 categories; see Table 9). These values indicate that the method is practical and computationally efficient.

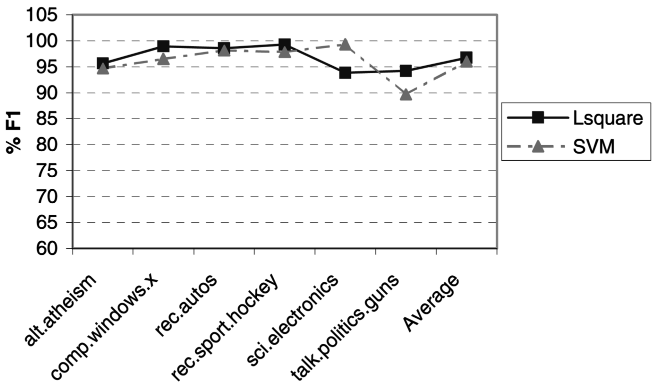


Fig. 9. Illustration of F1 results for *Lsquare* and SVM on 100 examples of six categories of *20NG*.

6 Conclusion

As most of the recent text categorization research focuses on addressing specific issues in TC (e.g., feature selection, clustering, and dimensionality reduction), very few new approaches are being devised. This paper proposes a new TC approach benefiting from the recent advances in feature clustering and dimensionality reduction coupled with a fairly effective logic-based learning technique. Logic-learning techniques have not been investigated in TC and this work serves this purpose and shows that logic-learning can be very competitive and effective in TC. Our method employs an effective representation of documents using distributional clustering of words. The method was extensively tested with numerous experiments using well-known benchmark data sets and compared with exact experimental settings against SVM. The proposed method outperformed the SVM-based method on all training-testing settings using WebKB data set and on most experiments conducted with the 20NG data set. On the Reuters-21578 data set, the method showed equally good and very close performance results to SVM. Finally, the paper showed that the proposed method represents a new and effective TC method that is particularly useful when there is limited training data.

TABLE 9
Training and Testing Times for 20 Categories of *20NG*

Number of training Examples	Training Time (*hr.min.sec*)	Number of testing examples	Testing time (*min.sec*)
10	0.18.04	19,990	2.24
20	0.23.39	19,980	2.18
50	0.58.34	19,950	2.25
100	1.39.29	19,900	2.27
200	3.15.03	19,800	2.27

REFERENCES

[1] H. Al-Mubaid and S. Nagula, "Machine Learning Approach for Context-Sensitive Error Detection," *Proc. Int'l Conf. Intelligent Computing and Information Systems (ICICIS '05),* 2005.

[2] H. Al-Mubaid and K. Truemper, "Learning to Find Context-Based Spelling Errors," *Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques,* 2006.

[3] L.D. Baker and A.K. McCallum, "Distributional Clustering of Words for Text Classification," *Proc. Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* 1998.

[4] R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter, "Distributional Word Clusters vs Words for Text Categorization," *J. Machine Learning Research,* vol. 3, 2003.

[5] B.E. Boser, I. Guyon, and V. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," *Proc. Ann. Workshop Computational Learning Theory (COLT '92),* pp. 144-152, 1992.

[6] M. Craven, D. DiPasquo, D. Freitag, A.K. McCallum, T.M. Mitchell, K. Nigam, and S. Slattery, "Learning to Extract Symbolic Knowledge from the World Wide Web," *Proc. Nat'l Conf. Artificial Intelligence (AAAI '98),* 1998.

[7] I. Dhillon, S. Mallela, and R. Kumar, "A Divisive Information-Theoretic Feature Clustering Algorithm for Text Classification," *J. Machine Learning Research,* vol. 3, 2003.

[8] S.T. Dumais, J. Platt, D. Heckerman, and M. Sahami, "Inductive Learning Algorithms and Representations for Text Categorization," *Proc. Seventh Int'l Conf. Information and Knowledge Management,* 1998.

[9] G. Felici, F. Sun, and K. Truemper, "A Method for Controlling Errors in Two-Class Classification," *Proc. 23rd Ann. Int'l Computer Software and Applications Conf. (COMPSAC-99),* 1999.

[10] G. Felici and K. Truemper, "A Minsat Approach for Learning in Logic Domains," *Informs J. Computing,* vol. 14, no. 1, Winter 2002.

[11] P.A. Flach, "On the Logic of Hypothesis Generation," *Applied Logic Series,* vol. 18, chapter 6, pp. 89-106, 2000.

[12] G. Forman, "An Extensive Empirical Study of Feature Selection Metrics for Text Classification," *J. Machine Learning Research,* vol. 3, 2003.

[13] M Hearst et al., Xerox TREC4 site report, TREC 4, 1996.

[14] T. Hofmann, "Probabilistic Latent Semantic Indexing," *Proc. Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* Aug. 1999.

[15] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," *Proc. 10th European Conf. Machine Learning (ECML '98),* 1998.

[16] T. Joachims, "A Statistical Learning Model of Text Classification with Support Vector Machines," *Proc. Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* 2001.

[17] D. Koller and M. Sahami, "Hierarchically Classifying Documents Using Very Few Words," *Proc. Int'l Conf. Machine Learning (ICML '97),* 1997.

[18] T.M. Mitchell, *Machine Learning.* McGraw-Hill, 1997.

[19] K. Nigam, A.K. McCallum, S. Thrun, and T.M. Mitchell, "Learning to Classify Text from Labeled and Unlabeled Documents," *Proc. Nat'l Conf. Artificial Intelligence (AAAI '98),* 1998.

[20] F. Pereira, N. Tishby, and L. Lee, "Distributional Clustering of English Words," *Proc. 31st Ann. Meeting of the ACL,* pp. 183-190, 1993.

[21] Reuters-21578: http://www.daviddlewis.com/resources/testcollections/reuters21578/, 2004.

[22] M.E. Ruiz and P. Srinivisan, "Hierarchical Neural Networks for Text Categorization," *Proc. Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* 1999.

[23] F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM Computing Surveys,* vol. 34, no. 1, pp. 1-47, 2002.

[24] N. Slonim and N. Tishby, "The Power of Word Clusters for Text Classification," *Proc. 23rd European Colloquium on Information Retrieval Research (ECIR-01),* 2001.

[25] N. Tishby, F.C. Pereira, and W. Bialek, "The Information Bottleneck Method," *Proc. 37th Ann. Allerton Conf. Comm., Control, and Computing,* 1999.

[26] V. Vapnik, *The Nature of Statistical Learning Theory.* Springer, 1995.

[27] Y. Yang and X. Liu, "A Re-Examination of Text Categorization Methods," *Proc. ACM Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* pp. 42-49, 1999.

[28] Y. Yang and J.O. Pedersen, "A Comparative Study on Feature Selection in Text Categorization," *Proc. 14th Int'l Conf. Machine Learning (ICML '97),* pp. 412-420, 1997.

[29] Z. Zheng and R. Srihari, "Optimally Combining Positive and Negative Features for Text Categorization," *Proc. Workshop Learning from Imbalanced Data Sets,* 2003.

[30] 20 News Group: http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.htm, 2004.

Hisham Al-Mubaid received the BS degree in computer science from the University of Jordan in 1989 and the MS and PhD degrees in computer science from the University of Texas at Dallas in 1997 and 2000, respectively. He worked one year as an assistant professor at the State University of New York (SUNY) at Geneso. Then, he joined the University of Houston-Clear Lake in 2001 as an assistant professor of compute science. His research interests and publications are mainly centered around natural language processing and include text categorization, machine learning, text mining, semantic, and ontology. Also, he has interests and publications in bioinformatics and teaching-learning research.

Syed A. Umair received the BE degree in computer science from the Muffakham Jah College of Engineering and Technology, Hyderabad, India, in 2003 and the MS degree in computer science from the University of Houston-Clear Lake, in 2005. From 2004 to 2005, he was with the Department of Computer Science at the University of Houston-Clear Lake working as a research assistant. His research interests include text categorization, feature selection, machine learning, and data mining.