International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 22, No. 2 (2008) 295–322

World Scientific
www.worldscientific.com

POLYNOMIAL NETWORKS VERSUS OTHER TECHNIQUES IN TEXT CATEGORIZATION

MAYY M. AL-TAHRAWI
Al-Ahliyya Amman University
P.O. Box 348, Amman 19374, Jordan
mtahrawi@ammanu.edu.jo

RAED ABU ZITAR
New York Institute of Technology
P.O. Box 8401, Amman 11121, Jordan
rzitar@philadelphia.edu.jo

Many techniques and algorithms for automatic text categorization had been devised and proposed in the literature. However, there is still much space for researchers in this area to improve existing algorithms or come up with new techniques for text categorization (TC). Polynomial Networks (PNs) were never used before in TC. This can be attributed to the huge datasets used in TC, as well as the technique itself which has high computational demands. In this paper, we investigate and propose using PNs in TC. The proposed PN classifier has achieved a competitive classification performance in our experiments. More importantly, this high performance is achieved in one shot training (noniteratively) and using just 0.25%–0.5% of the corpora features. Experiments are conducted on the two benchmark datasets in TC: Reuters-21578 and the 20 Newsgroups. Five well-known classifiers are experimented on the same data and feature subsets: the state-of-the-art Support Vector Machines (SVM), Logistic Regression (LR), the k-nearest-neighbor (kNN), Naive Bayes (NB), and the Radial Basis Function (RBF) networks.

Keywords: Polynomial networks; text categorization; document classification; document categorization.

1. Introduction

Text categorization (TC) can be defined as the task of assigning an unseen document to one or more predefined categories. Larger amounts of textual information are becoming available online everyday. Consequently, the need for automatic, fast, accurate and efficient classification of this information grows rapidly, in order to help people to get their needs from these huge amounts of information. In this paper, we investigate and propose using Polynomial Networks (PNs) in TC. Five other well-known classifiers are used in our experiments: the state-of-the-art Support Vector Machines (SVM), Logistic Regression (LR), the k-nearest-neighbor (kNN), Naive

Bayes (NB) and Radial Basis Function (RBF) Networks. All classifiers are tested using the same data and feature subsets, so as to provide a basis for direct comparisons. Classification experiments are conducted on the two benchmark datasets: Reuters-21578 and the 20 Newsgroups. Relative frequency (term frequency in a document normalized to document length) is used as a term-weighting function in all the classification algorithms used in this work, except the NB classifier. For NB, binary weights are used to represent documents. Chi square is used for feature selection. Several feature reduction techniques are experimented and evaluated using the six classifiers. PN classifiers have proved, in our experiments, to be a competitive algorithm in the field of TC. They have achieved this competitive performance noniteratively, and using just 0.25%–0.5% of the corpora features. Add to this, PN classifiers have recorded a high classification performance on rare classes and those closely related ones. The rest of the paper is organized as follows. We start, in Sec. 2, with an overview of the other five classification algorithms used in our experiments. Section 3 is devoted to explain, in detail, the proposed PN model we devise in this paper. A brief explanation of the datasets used and the processing steps performed on these datasets is presented in Sec. 4. In Sec. 5, we explain the feature selection methods and the performance evaluation measures used in our experiments. Section 6 presents a summary of the experiments we have conducted and the results reached in these experiments. Analysis of results takes place in Sec. 7 and some related work is presented in Sec. 8. Finally, we conclude and present our intended future work in Sec. 9.

2. Some Well-Known Text Categorization (TC) Methods

In this section, we review in brief the other five TC algorithms used in our experiments: kNN, NB, RBF networks, SVM, and LR.

2.1. *kNN*

kNN, stands for k-nearest neighbor classification, is a well-known statistical approach that has been applied to TC since the early stages of research. It is one of the top-performing methods on Reuters.[47] The algorithm is very simple. Given a test document to classify, the classifier finds the k nearest neighbors of the test document, and majority voting among the neighbors is used to decide the category of the test document. Similarity is measured by the cosine between the vectors representing the documents. If a category is shared by more than one of the k neighbors, then the sum of the similarity scores of these neighbors is the weight of that shared category.

2.2. *Naive bayes (NB)*

NB is a well-known and highly practical probabilistic classifier that has been widely used in TC. It uses the joint probabilities of words and categories to estimate the probabilities of categories, given a test document. The naive part in this algorithm

is the assumption of word independence: probability of a word given a category is assumed to be independent from the conditional probabilities of other words given that category; i.e. it does not use word combinations as predictors. This naive assumption results in saving computation time to a great extent. Several studies show that NB performs surprisingly well in TC, despite this wrong independence assumption.[14] Zheng *et al.*[50] and Lewis and Ringuette[32] found feature selection to be very useful for Reuters when classified with NB.

In TC, the probability of a class C, given a document d_j is calculated by Bayes' theorem as follows[26]:

$$P(C|dj) = \frac{P(dj|C)P(C)}{P(dj)} = \frac{P(dj|C)P(C)}{P(dj|C)P(C) + P(dj|\bar{C})P(\bar{C})}.$$

2.3. *Radial basis function (RBF) networks*

RBF network is an artificial neural network model motivated by the "locally tuned" response observed in biological neurons.[23] RBF networks were used early for interpolation,[34,40] probability density estimation,[15,37,42] and approximations of smooth multivariate functions.[38] They have also been applied with success to classification.[31,35,36,46] The RBF network has a feedforward structure consisting of a single hidden layer of a number of locally tuned units which are fully interconnected to an output layer of a number of linear units. All hidden units simultaneously receive the input vector. Hidden units outputs are calculated as the distance between the input vector and weight vector of the hidden unit multiplied by a bias b. The bias allows the sensitivity of the radial basis unit to be adjusted. It determines the width of the area in the input space to which each hidden unit responds. A distinguishing feature of a RBF network is its adaptive nature, which generally allows it to utilize a relatively smaller number of locally tuned units. Details of the transfer functions used in our experiments and other parameters settings are given in Sec. 6.

2.4. *Support vector machines (SVM)*

Support Vector Machines (SVM) are relatively new classifiers that were introduced by Vapnik.[6,11,44] Empirical studies in recent years have shown that SVM is the state-of-the-art technique among other well-known TC algorithms.[25] Furthermore, they are fully automatic, eliminating the need for manual parameter tuning.

SVMs are based on the *Structural Risk Minimization* principle[44] from computational learning theory. The idea of structural risk minimization is to find a hypothesis h for which the lowest true error can be guaranteed. The true error of h is the probability that h will make an error on a random unseen test example. An upper bound can be used to connect the true error of a hypothesis h with the error of h on the training set and the complexity of H (measured by VC-Dimension), the hypothesis space containing h.[44] SVMs find the hypothesis h which approximately

minimizes this bound on the true error by controlling the VC-Dimension of H efficiently. To learn nonlinear hypotheses, SVM makes use of convolution functions. Depending on the convolution function, SVMs learn *polynomial classifiers* (SVM Poly), *radial basis function* (SVM RBF) *classifiers*, or two-layer sigmoid neural nets. For details of SVMs computations, and how to apply them in TC, the reader can refer to Refs. 7 and 25.

2.5. *Logistic regression (LR)*

Logistic Regression (LR) has been a well-known and mature statistical model suitable for probabilistic classification. Recently, logistic regression has been studied in statistical machine learning community due to its close relation to SVMs and Adaboost.[24,45,49] It is a high performance classifier that can be efficiently trained with a large number of labeled examples. Previous studies have shown that the logistic regression model is able to achieve the similar performance of TC as SVMs.[30,49] These studies have also showed that the logistic regression model can be trained significantly more efficiently than SVMs, particularly when the number of labeled documents is large. Logistic regression can be applied to both real and binary data. It outputs the posterior probabilities for test examples that can be conveniently processed and engaged in other systems. In theory, given a test example x, logistic regression models the conditional probability of assigning a class label y to the example by[24]:

$$P(y|x) = \frac{1}{1 + \exp(-y\,\alpha^T x)}$$

where α is the model parameter. We used the Iteratively Re-weighted Least Squares (IRLS) nonlinear optimization algorithm as a fitting procedure.[30] Other efficient implementation algorithms of logistic regression have been developed in the recent literature (the reader can refer to related work in Sec. 8.2 for more details on this issue).

3. Polynomial Networks (PNs)

Polynomial Network (PN) classifiers have been known in the literature for many years,[20] and have been recently used in some areas like speaker verification and sign language recognition.[2,3,8–10] Traditionally, PNs have been difficult to use in TC because of their high computational demands, in terms of both time and storage. These requirements increase dramatically with the large datasets, usually used in TC. Nevertheless, PNs have many attractive properties for use in TC. Firstly, they do not require iterative training or a high degree of fine tuning of parameters. Secondly, the ability to partition the problem and execute its parts in parallel on more than one computer at a time, is a major advantage of this technique. One more important advantage of PN classifiers, which is realized in our experiments, is their ability to achieve high precision, even on rare classes and closely related ones,

using just a very small subset of the corpus features. Another advantage of using PNs in classification is that, unlike some lazy learners (kNN, for example), once the training phase is finished, neither the training data nor any special memory requirements are needed anymore. A small file that holds classes' models (weights) is all what we need to classify an unseen document.

3.1. *The architecture of PNs*

We adopt the PNs model proposed by Campbell *et al.*[9] for use in TC. The model consists of two layers. The first layer (the input layer) forms the monomial basis terms of the input vector $x(x_1, x_2, \ldots, x_N)$, such as $1, x_1, x_2, x_1^2, \ldots$ etc. where N is the number of features (dimensions) of x. A second layer then linearly combines the output of the first layer; i.e. the data is first expanded into a high-dimensional space in the first layer and then it is linearly separated using the second layer. The basic embodiment of a Kth order polynomial network consists of several parts. The N features of one observation $x(x_1, x_2, \ldots, x_N)$ are used to form a basis function $p(x)$; one $p(x)$ is formed for each observation. The elements of $p(x)$ for a polynomial of degree K are monomials of the form[2]:

$$\prod_{j=1}^{N} x_j^{k_j}, \quad \text{where } k_j \geq 0 \quad \text{and} \quad 0 \leq \sum_{j=1}^{N} k_j \leq K. \tag{1}$$

As an example, for an input vector (an observation) x containing two features x_1 and x_2, a second order polynomial network basis function $p(x)$ will appear as follows

$$p(x) = [1 \quad x_1 \quad x_2 \quad x_1^2 \quad x_1 x_2 \quad x_2^2]^t \tag{2}$$

and a third order polynomial network basis function $p(x)$ will appear as

$$p(x) = [1 \quad x_1 \quad x_2 \quad x_1^2 \quad x_1 x_2 \quad x_2^2 \quad x_1^3 \quad x_1^2 x_2 \quad x_1 x_2^2 \quad x_2^3]^t. \tag{3}$$

The second layer of the PN linearly combines all inputs to produce weights of classes (classes' models). The whole class is represented by one weight, which is computed during the training phase. Detailed training steps and examples are presented in the next section.

3.2. *The training phase*

A PN is trained to approximate an ideal output using mean squared error as the objective criterion. The polynomial expansion of the ith class feature vectors (observations) is denoted by Ref. 9:

$$M_i = [p(x_{i,1}) \quad p(x_{i,2}) \quad p(x_{i,3}) \quad \cdots \quad p(x_{i,Ni})]^t \tag{4}$$

where N_i is the number of training feature vectors for class i, and $p(x_{i,m})$ is the basis function of the mth feature vector for class i. After forming M_i for each class i of the nc training classes, a global matrix M is obtained for the nc classes, by concatenating the individual M_i's computed for each class[2]

$$M = [M_1 \quad M_2 \quad M_3 \quad \cdots \quad M_{nc}]^t. \tag{5}$$

The training problem then reduces to finding an optimum set of weights w (one weight for each class) that minimizes the distance between the ideal outputs (targets) and a linear combination of the polynomial expansion of the training data such that[2]:

$$w_i^{\text{opt}} = \arg\min_w \|Mw - o_i\|_2 \tag{6}$$

where o_i is the ideal output (a column vector which contains N_i ones in the rows where the ith class' data are located in M, and contains zeros otherwise). A class model w_i^{opt} can be obtained in one shot (noniteratively) by applying the normal equations method[2,21]:

$$M^t M w_i^{\text{opt}} = M^t \ o_i. \tag{7}$$

By defining MG as $M^t M$, Eq. (7) reduces to

$$w_i^{\text{opt}} = MG^{-1} \ M^t \ o_i. \tag{8}$$

3.3. *Recognition*

Recognition (classification of a new unseen document) consists of two parts: identification and verification. Identification involves finding the best matching class of an unseen input, given the feature vector of this input. In the verification phase, the claim made in the identification phase is either accepted or rejected. The identification phase proceeds as follows in the proposed PN technique. The feature vector x of the input (the unseen input to classify) is expanded into its polynomial terms $p(x)$ in a manner similar to what was done with the training inputs in the training phase (using the same polynomial degree, of course). Then, the new unseen input is assigned to the class c such that[2]

$$c = \arg\max_i \ w_i^{\text{opt}} \cdot p(x) \quad \text{for } i = 1, 2, \ldots, nc. \tag{9}$$

The method can be extended to handle multilabel categorization by recording all scores which result from $w_i \cdot p(x)$ for each class, then accepting all classifications with a score higher than a prespecified threshold in the verification phase.

In verification, a decision to accept or reject a certain classification can be based on using a certain threshold value. In our experiments, we accepted classifications with scores above 0.5, since the output score $w_i \cdot p(x)$ lies between 0 and 1. A threshold may also be decided on basis of using a validation set to decide the proper threshold value; either one threshold value for all classes or a threshold value for each class. If the score $w_i \cdot p(x)$ lies above the threshold, the classification is accepted; otherwise it is rejected.

3.4. *Text categorization (TC) using PNs*

The training phase of TC using PNs goes through the following steps. Each training document is represented by a vector of token features x using the vector space model. Tokens can be represented by their *binary* weights, *normalized frequencies*,

tf.idf weights, ... etc. We used relative frequencies in our experiments. Then, the kth order PN basis function $p(x)$ is formed for each training document, as in Eq. (1). Second order PNs are used in the experiments presented in this paper. So, if a training document is represented by the frequencies 1, 0.2, 0 in order; i.e. the feature vector $x = (1, 0.2, 0)$ for this document, then the second order PN basis function of this document will look as follows [Eq. (1)]:

$$p(x) = [1 \quad 1 \quad 0.2 \quad 0 \quad 1 \quad 0.2 \quad 0 \quad 0.4 \quad 0 \quad 0]. \tag{10}$$

The polynomial expansion of each class i training files, M_i, is then formed as in Eq. (4). Now, the global matrix for all the nc classes is obtained by concatenating all the individual M_i matrices into M as in Eq. (5). Once the global matrix M is formed, the PN is trained to approximate an ideal output using mean-squared error as the objective criterion [Eq. (6)]. Finally, the training phase ends with finding the optimum set of weights w as in Eqs. (7) and (8). To classify an unseen document, the feature vector x of the unseen document is expanded into its polynomial terms $p(x)$ as in Eq. (1). Then, the new unseen document is assigned to class c as explained in Eq. (9).

4. Data Sets

We used the two benchmark datasets: 20 Newsgroups and Reuters-21578 in our experiments. They are downloaded from Ref. 1, where versions of the 20 Newsgroups and subsets of Reuters-21578 suitable for use in single-label TC are available. We applied our own processing steps on the datasets downloaded from Ref. 1. The whole processing steps performed on the datasets can be summarized as follows:

(1) Tabs, new lines, and RETURN characters are converted to spaces.
(2) Only letters, hyphens '-' and underscores '_' are kept; any other character is eliminated. The hyphen and underscore characters are kept because of the commonality of compound words (noun groups); *object-oriented* is an example of such terms which is commonly used in *computer science* topics.
(3) All letters are converted to lowercase.
(4) We applied the Porter Stemmer,[39] with the following modifications: an ignore list of more than 1000 stop words is defined and used to reduce the number of tokens in the dataset. Then, any remaining word consisting of just one character is removed.
(5) Infrequent words which appear five times or less in the whole corpus are removed. This is a common preprocessing step in TC, which helps to reduce the size of the feature set to a great extent; accordingly, the computing demands for classification are significantly reduced. Eliminating these rare words has proved not to have a bad effect on classification accuracy.[17] Table 1 shows the 20 Newsgroups dataset after processing.

As for Reuters, **R10,** the set of the ten classes with the highest number of positive training examples of the ModApte version of Reuters-21578 is selected. To

Table 1. Distribution of documents and features among the 20 newsgroups dataset classes.

Class #	Class	# Train Docs.	# Test Docs.	Total # Docs.	# Features
1	Alt.atheism	480	319	799	1610
2	Comp.graphics	584	389	973	1563
3	Comp.os.ms-windows.misc	572	394	966	1173
4	Comp.sys.ibm.pc.hardware	590	392	982	1173
5	Comp.sys.mac.hardware	578	385	963	1100
6	Comp.windows.x	593	392	985	1816
7	misc.forsale	585	390	975	1152
8	Rec.autos	594	395	989	1612
9	Rec.motorcycles	598	398	996	1634
10	Rec.sport.baseball	597	397	994	1531
11	Rec.sport.hockey	600	399	999	1991
12	Sci.crypt	595	396	991	2061
13	Sci.electronics	591	393	984	1525
14	Sci.med	594	396	990	2177
15	Sci.space	593	394	987	2208
16	soc.religion.christian	598	398	996	1945
17	talk.politics.guns	545	364	909	2073
18	talk.politics.mideast	564	376	940	2618
19	talk.politics.misc	465	310	775	2159
20	talk.religion.misc	377	251	628	1474
	Total	**11293**	**7528**	**18821**	**14900** (after removing duplicates)

use **R10** in single-label categorization, only documents with a single topic and the classes which still have at least one train and one test example were considered. As a result, the set of the ten most frequent classes, **R10** was reduced to eight classes **(R8)**. From R10 to R8, the classes *corn* and *wheat*, which are intimately related to the class *grain* disappeared and this last class lost many of its documents. We used **R8** in our experiments. The same processing steps are applied on R8, as in the 20 Newsgroups, except that infrequent words were not removed, since the size of the processed feature set was reasonable. We ended up with the distribution of documents and features, per class, shown in Table 2. The large variation between classes in the number of training and test documents, and in the number of features is clear from this table.

5. Feature Selection and Performance Evaluation Measures

This section explains the feature selection methods and the performance evaluation measures used in the research presented in this paper.

5.1. *Feature selection*

Applying a proper feature selection criterion in TC results in using a small, but proper, subset of the corpora features in classification. It was proved in several researches that, in most of the cases, only a small, but proper, subset of the corpus features is useful in classification.[17,48] In fact, using many features or the entire

Table 2. Distribution of documents and features among **R8** classes.

R8					
Class #	Class	# Train Docs	# Test Docs	Total # Docs	# Features
1	Acq	1596	696	2292	7323
2	crude	253	121	374	2751
3	Earn	2840	1083	3923	7188
4	grain	41	10	51	1038
5	interest	190	81	271	1448
6	money-fx	206	87	293	1992
7	ship	108	36	144	1676
8	trade	251	75	326	2652
	Total	5485	2189	7674	13891 (after removing duplicates among classes)

feature set may affect the classification performance negatively.[17] Chi square (χ^2) is used as a feature selection criterion in our experiments. It has shown to yield good results and has proved to maximize precision of classification, compared to other feature selection methods including Information Gain (IG), Document Frequency (DF), Odds Ratio (OR), Log Probability Ratio, Mutual Information (MI), and Term Strength (TS).[17,19,41,48] Chi square was originally used in the statistical analysis of independent events. Its application in feature selection for TC goes through the following steps:

(1) For each term in each class in the training set, compute the chi square score to measure the correlation or dependency between the term and its containing class. The higher this score is, the more discriminating the term is for that class. A zero score means that the term is independent of its containing class; i.e. it will be of no use in predicting the class of an unseen document. The chi square measure is computed for each term t in each class c_i as follows[50]:

$$\chi^2(t, c_i) = \frac{N \times (AD - CB)^2}{(A + C) \times (B + D) \times (A + B) \times (C + D)} \tag{11}$$

where N is the total number of training documents in the dataset, A is the number of documents belonging to class c_i and containing t, B is the number of documents belonging to class c_i but not containing t, C is the number of documents not belonging to class c_i but containing t and D is the number of documents neither belonging to class c_i nor containing t.

(2) Combine the class-term chi square measures for terms that appear in more than one class (usually with different chi square measures in different classes) in one score. Either the sum, the maximum or the average score can be used as a globalization mechanism. We used the maximum score in our work.

(3) Choose the reduced feature set from the topmost chi square measure features, and eliminate those features with zero or small measures which indicate their small or nondiscriminating power.

After computing the chi square measure for each term in each class in the training set, we experimented with several methods to select a subset of the corpus features to be used in classification. The aim of this experimentation is to try to find out the method which generates the best classification performance for each classifier. We took a specified number of the corpus topmost chi square measure features, regardless of the share of each class from these features. We also selected an equal number of features from each class, regardless of the original number of features in the class; selection from the class is also done from the topmost chi square measure terms in the class. Finally, we selected an equal percentage of features from each class (also taking the topmost chi square measure features in the class). As will be shown in Sec. 6, the latter method yielded the best performance using the PN classifier.

5.2. *Performance measures*

Classifiers are evaluated either using their training efficiency (the average time to build the classifier), classification efficiency (the average time to classify an unseen document) or effectiveness (the average correctness of classification). A classifier effectiveness can be measured by computing its *accuracy*, *precision*, *recall* and (or) F1 measure. F1 measure has been shown to be more reliable metric than *Accuracy*.[48]

Accuracy of a class c_i, Acc_i is computed as follows:

$$\mathrm{Acc}_i = \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FN}_i + \mathrm{FP}_i} \tag{12}$$

where

TP_i: True Positives with respect to a category c_i; the number of documents correctly claimed by the classifier as belonging to category c_i.

FP_i: False Positives with respect to c_i ; the number of documents incorrectly claimed by the classifier as belonging to c_i.

FN_i: False Negatives with respect to c_i ; the number of documents incorrectly claimed by the classifier as not belonging to c_i.

Precision refers to the proportion of test files classified into a class that really belong to that class, while recall is the proportion of test files belonging to a class and were claimed by the classifier as belonging to that class. Precision of a class c_i, P_i can be defined as follows[13]:

$$P_i = \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i} \tag{13}$$

and recall of a class c_i, (R_i) can be computed using the following formula[13]:

$$R_i = \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FN}_i}. \tag{14}$$

The F1 measure, introduced by Rijsbergen,[43] is the harmonic average of both *precision* and *recall*. High F1 means high performance of the system. F1 is computed

as follows[13]:

$$\mathrm{F1} = \frac{2 \times \mathrm{recall} \times \mathrm{precision}}{\mathrm{recall} + \mathrm{precision}} \tag{15}$$

$$= \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}. \tag{16}$$

Individual results of categories can be either microaveraged or macroaveraged to give an idea of the classification performance on the corpus as a whole. We evaluated the four classifiers using *accuracy*, *precision*, *recall*, and F1. Individual results per class, as well as micro- and macro-averaged results are presented, where appropriate, to give a clear idea of the classifiers performance. For detailed formulae for computing *microaveraged* and *macroaveraged* results, the reader can refer to Ref. 13.

6. Experiments and Results

TC experiments are conducted on the 20 Newsgroups and Reuters (R8) using PNs, SVM, LR, kNN, NB and RBF networks. Several feature subsets were used in classification (details of these subsets are explained in Sec. 6.1.1) and each classifier was tested on all these feature subsets.

6.1. *Experiments on R8 dataset*

Four different reduction strategies were used to select different feature subsets. Each categorization algorithm was experimented using the four feature sets, in order to find out the subset that results in the optimal classification performance for each algorithm. Various parameters settings were tested for SVM, kNN and RBF networks. For kNN, several values of K in the range $[1 \ldots 100]$ were tried for each feature subset. Regarding RBF networks, the Gaussian kernel was used, and several experiments were tried for each feature subset. Various settings for the spread value, the number of neurons in the hidden layer, the goal error, and the number of neurons added in each iteration were experimented. SVM was experimented with the quadratic polynomial kernel and the RBF kernel, with various parameter settings. The best results reached for each classifier are presented in Sec. 6.1.2. All experiments were conducted on a 2.66 GHZ, 1GB RAM PC, and they were coded using MATLAB. Details of the feature reduction strategies are explained next.

6.1.1. *Feature reduction methods*

We started by selecting the topmost 100 chi square measure features from the corpus as a whole (0.72% of the corpus features), and these 100 features are distributed among the classes as shown in Table 3. It is to be noted that the total number of features is greater than 100 since some topmost features have appeared in more than one class. To clarify the table contents, we explain the meaning of the *acq* entry in

Table 3. Distribution of the 100 topmost corpus features among the classes in R8.

Class	# of Class Features Out of the Topmost 100 Features Selected for Classification	% of Class Features Selected Out of the Topmost 100
Acq	14	0.19%
Crude	17	0.62%
Earn	13	0.18%
Grain	10	0.96%
Interest	4	0.28%
Money-fx	13	0.65%
Ship	10	0.6%
Trade	26	0.98%

Table 3. The table states that 14 features out of the 100 selected for classification were chosen from *acq* features. These 14 features represent only 0.19% of the original 7323 features of *acq* (the reader can refer to Table 2). This table shows that a very small subset of each class features is used to build the classifier.

Another reduced feature set is formed by selecting the topmost 70 chi square measure features from the corpus as a whole (0.5% of the corpus features). We created this feature set to compare the classification performance using the same reduction method (the corpus topmost features) but with a smaller number of features. These 70 features are distributed among the classes as shown in Table 4. Again, it is to be noted that the total number of features is greater than 70 since some topmost features have appeared in more than one class.

An equal number of features is chosen from each class as a third feature reduction strategy. Here, we tried to find out the effect of the variation in the number of features chosen from each class (as a part of the corpus top most features selected for building the classifier), on the classifier performance. We selected the topmost 13 chi square measure features from each class, and these 104 features were reduced to 96 features (0.7% of the corpus features) after elimination of duplicates. This experiment did not come up with a clear enhancement of the performance of the PN classifier compared with using the corpus 70 topmost features. However, it was better than using the corpus 100 topmost features, which is very close to 96 as the number of features. The last feature reduction strategy we tried was to select

Table 4. Distribution of the topmost R8 — 70 features among classes.

Class	# of Class Features Out of the 70 Topmost Features Selected for Classification	% of Class Features Selected Out of the Topmost 70
Acq	9	0.12%
Crude	13	0.47%
Earn	8	0.11%
Grain	5	0.48%
Interest	4	0.28%
Money-fx	9	0.45%
Ship	8	0.48%
Trade	18	0.68%

an equal percentage of the topmost chi square measure features from each class (0.5% of each class). We got 131 features and these 131 features were reduced to 108 features after elimination of duplicates. This reduction strategy recorded a clear enhancement on the classifiers' performance, compared with the other three reduction methods. Detailed results of using these four reduced feature sets in classification, as well as an analysis of these results, will follow in the subsequent sections.

6.1.2. *Results*

TC experiments on R8 were conducted using the six categorization algorithms: PNs, SVM, LR, kNN, NB, and RBF Networks. Each algorithm was experimented using the four reduced feature sets explained in Sec. 6.1.1. Each feature subset was tried with various parameter settings for each TC algorithm. Tables 5 and 6 show detailed results of the PN classifier on R8, and Table 7 summarizes these results and shows training and testing times for the four feature sets. We include micro- and macro-averaged F1 as it summarizes the classifier *precision* and *recall*. Micro- and macro-averaged *accuracy* are included, as well, in this table. For the rest of the experiments (using the other classifiers), and for the sake of space, we just include summarized performance results in Tables 8 through 13. Detailed results will be included, where necessary, in the analysis part of the paper (Sec. 7).

Table 5. Results of the PN classifier on R8.

	R8 — 100 Features				R8 — 70 Features			
Class	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy
Acq	93.8623	90.0862	91.9355	85.0746	89.6739	94.8776	92.1788	85.4922
Crude	93.5185	83.4711	88.2096	78.9063	93.6364	85.124	89.1775	80.4688
Earn	92.7944	97.5069	95.0923	90.6438	96.7351	95.7525	96.2413	92.7549
Grain	88.8889	80	84.2105	72.7273	76.9231	100	86.9565	76.9231
Interest	89.0625	70.3704	78.6207	64.7727	89.8305	65.4321	75.7143	60.9195
Money-fx	75	75.8621	75.4286	60.5505	77.7778	72.4138	75	60
Ship	79.4118	75	77.1429	62.7907	87.5	77.7778	82.3529	70
Trade	82.5	88	85.1613	74.1573	80.2326	92	85.7143	75

Table 6. Results of the PN classifier on R8.

	R8 — 108 Features				R8 — 96 Features			
Class	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy
Acq	96.4	92.24	94.3	89.1667	93	90.52	92.1788	84.7914
Crude	95.37	85.12	90	81.746	93.46	82.64	89.1775	78.125
Earn	93.7	98.43	96	92.2944	93.77	97.32	96.2413	91.4137
Grain	90.9	100	95.24	90.9091	81.82	90	86.9565	75
Interest	87.14	75.31	80.8	67.7778	87	66.67	75.7143	60.6742
Money-fx	84.15	79.31	81.66	69	72.4	72.4	75	56.7568
Ship	87.88	80.56	84.01	72.5	87.1	75	82.3529	67.5
Trade	85.19	92	88.47	79.3103	75.6	90.67	85.7143	70.1031

Table 7. Summarized results of the PN classifier on R8.

	100 Features	70 Features	108 Features	96 Features
MicroAverage F1	91.7314	92.4166	**93.60**	91.60
MacroAverage F1	84.9752	85.4169	**88.80**	84
MicroAverage Accuracy	84.7257	85.9023	**87.9777**	84.4922
MacroAverage Accuracy	73.7029	75.1948	**80.338**	73.0455
Training Time for the corpus in secs	1.1572×10^4	5.697×10^3	1.3762×10^4	1.0442×10^4
Avg. Test Time per file in secs	2.41	1.28	3.35	2.07

*Boldface indicates best results.

Table 8. Results of the kNN classifier on R8.

	100 Features, $k=7$	70 Features, $k=5$	108 Features, $k=4$	96 Features, $k=15$
MicroAverage F1	92.65	**92.74**	**92.74**	92.28
MacroAverage F1	86.16	**87.14**	86.53	84.14
MicroAverage Accuracy	86.2979	**86.4566**	**86.4566**	85.6658
MacroAverage Accuracy	76.4297	**78.1272**	77.367	73.9627
Training + Testing Time for the whole corpus in secs	719.334	395.17	605.94	669.914

Table 9. Results of the NB classifier on R8.

	100 Features	70 Features	108 Features	96 Features
MicroAverage F1	91.73	92.33	**92.65**	92.28
MacroAverage F1	82.4	**84.06**	83.83	83.71
MicroAverage Accuracy	84.7257	85.7446	**86.2979**	85.6658
MacroAverage Accuracy	71.0836	**73.6019**	73.1588	73.0455
Training + Testing Time for the corpus in secs	146.265	135.406	181.672	173.063
Avg. Test Time per file in secs	0.0342	0.0253	0.0388	0.0366

Table 10. Results of the RBF classifier on R8.

	100 Features 500 Neurons	70 Features 500 Neurons	108 Features 750 Neurons	96 Features 500 Neurons
MicroAverage F1	76.63	76.43	**77.94**	77.38
MacroAverage F1	49.31	47.22	**51.65**	48.46
MicroAverage Accuracy	62.1155	61.8495	**63.8526**	63.1083
MacroAverage Accuracy	36.7562	35.537	**38.7656**	36.589
Training Time of the corpus in secs	15944.61	13851.03	33101.63	15306.42
Avg. Test Time per file in secs	0.01	0.0087	0.014	0.0099

6.2. *Experiments on the 20 newsgroups dataset*

For the 20 Newsgroups dataset, 0.25% of the topmost chi square measure features from each class are selected to form a reduced feature set for use in classification. This feature set includes 88 features in total, and ends with 86 features after

Table 11. Results of the SVM RBF classifier on R8.

	100 Features	70 Features	108 Features	96 Features
MicroAverage F1	94.11069	93.1932	**95.3403**	93.6958
MacroAverage F1	87.1019	86.4063	**89.801**	86.2126
MicroAverage Accuracy	88.8697	87.2541	**91.0956**	88.1392
MacroAverage Accuracy	78.3568	77.4128	**82.2067**	77.0883
Training + Testing Time for the corpus in secs	2.7702×10^3	2.5069×10^4	2.1336×10^3	2.7956×10^3

Table 12. Results of the SVM POLY classifier on R8.

	100 Features	70 Features	108 Features	96 Features
MicroAverage F1	88.1681	88.9904	**89.0818**	88.0767
MacroAverage F1	63.3149	**65.6089**	63.9956	63.4646
MicroAverage Accuracy	78.8399	**80.1646**	**80.313**	78.6939
MacroAverage Accuracy	52.8007	**54.8686**	53.9245	52.7945
Training + Testing Time for the corpus in secs	1.2311×10^4	3.0176×10^4	2.2804×10^4	6.345×10^4

Table 13. Results of the LR classifier on R8.

	100 Features	70 Features	108 Features	96 Features
MicroAverage F1	94.1983	93.2846	**95.0662**	94.4267
MacroAverage F1	87.3856	87.2397	**89.9377**	**89.8624**
MicroAverage Accuracy	89.0328	87.4144	**90.596**	89.4418
MacroAverage Accuracy	78.7487	78.1421	**82.212**	**82.3384**
Training + Testing Time for the corpus in secs	359.7030	282.9690	347.4070	345.9060

eliminating duplicates among classes. The 88 features are distributed among the classes as shown in Table 14.

Six different TC algorithms were tried using the same 86 features: PNs, kNN, NB, SVM RBF, SVM POLY and LR. Various parameter settings were tried for kNN and SVM, and the best results are recorded here. Detailed F1 performance per class, using each classification algorithm, is presented in Table 15, and summarized results are shown in Table 16.

7. Analysis of Results

This section discusses the results of the experiments conducted in this research on Reuters-21578 and the 20 newsgroups datasets, using different reduced feature sets with the six classifiers.

7.1. *Analysis of the results of the experiments on Reuters*

As an overall performance of the six classifiers on the four reduced feature sets, the top performers on the 108 feature set are SVM-RBF, LR and PNs, while the

Table 14. Distribution of the 88 features among classes in the 20 newsgroups dataset.

Class#	Class	# of Class Features Out of the 88 Features Selected
1	Alt.atheism	4
2	comp.graphics	4
3	comp.os.ms-windows.misc	3
4	comp.sys.ibm.pc.hardwar	3
5	comp.sys.mac.hardware	3
6	comp.windows.x	5
7	misc.forsale	3
8	Rec.autos	4
9	Rec.motorcycles	4
10	Rec.sport.baseball	4
11	Rec.sport.hockey	5
12	Sci.crypt	5
13	Sci.electronics	4
14	Sci.med	5
15	Sci.space	6
16	soc.religion.christian	5
17	talk.politics.guns	5
18	talk.politics.mideast	7
19	talk.politics.misc	5
20	talk.religion.misc	4
	Total	**88**

Table 15. F1 results of classifying 20 newsgroups using 86 features (0.25% of each class).

Class #	Class	PNs	kNN	NB	SVM-RBF	SVM-Poly2	LR
1	Alt.atheism	53.211	14.998	**56.2893**	49.242	38.889	52.852
2	comp.graphics	**57.1027**	**57.1798**	56.0197	52.59	55.162	55.788
3	comp.os.ms-windows.misc	56.6893	57.377	**57.8534**	56.713	**57.46**	56.595
4	comp.sys.ibm.pc.hardware	35.337	32.3024	**38.3057**	34.686	28.077	36.887
5	comp.sys.mac.hardware	60.2476	60.485	**64.2356**	60.766	55.539	63.025
6	comp.windows.x	53.8721	48.7223	**58.8235**	53.427	49.815	**58.569**
7	misc.forsale	70.9677	70.0855	**74.2706**	70.839	67.755	72.677
8	rec.autos	**72.0798**	70.6215	71.0674	69.892	66.385	**72.214**
9	rec.motorcycles	**86.0963**	83.6763	85.1064	84.615	77.515	**86.207**
10	rec.sport.baseball	71.3725	66.0274	72.2013	70.444	66.562	**74.356**
11	rec.sport.hockey	**79.2503**	72.4005	**79.8883**	**79.245**	75.072	**79.767**
12	sci.crypt	72.6241	69.6133	**74.2297**	70.297	66.203	73.82
13	sci.electronics	**34.5059**	32.1555	23.7131	30.739	28.099	**34.936**
14	sci.med	**28.6599**	4.6908	5.2506	5.2632	0.46729	**28.683**
15	sci.space	**70.8725**	68.6441	**70.9424**	68.586	63.781	**71.056**
16	soc.religion.christian	63.5851	59.7701	**65.5367**	61.105	55.946	**65.227**
17	talk.politics.guns	**58.4329**	56.8214	**59.5399**	57.333	38.889	**58.39**
18	talk.politics.mideast	**78.0255**	73.8411	**80.2508**	26.517	75.205	**78.855**
19	talk.politics.misc	42.2535	43.2071	**48.2916**	43.88	40.404	45.089
20	talk.religion.misc	22.9412	14.5773	**40.0881**	2.3622	8.3283	23.03

top performers using the other feature sets are LR, SVM-RBF and kNN, then follows PNs.

Using equal percentage of class features (108) results in the best classification *precision* in all classifiers, compared with using equal number of features from each

Table 16. Summarized classifier results on the 20 newsgroups.

20 Newsgroups — 86 features						
Measure	PNs	NB	kNN, $k = 15$	SVM-RBF	SVM-POLY	LR
MacroAverage Precision	**67.5232**	**68.8322**	61.8359	**67.174**	**68.311**	**68.627**
MacroAverage Recall	**55.9918**	**56.5105**	49.976	52.13	44.993	**56.586**
MacroAverage F1	**58.41**	**59.09**	52.86	52.4272	50.7778	**59.401**
MicroAverage F1	**57.19**	**57.12**	50.85	53.3874	45.2444	**57.798**
Accuracy	**43.317**	**44.395**	38.658	38.346	36.522	**44.367**
MicroAverage Accuracy	**40.043**	**39.978**	34.093	36.414	29.236	**40.645**

Table 17. MicroAverage precision on Reuters.

Algorithm	108 Features	100 Features	96 Features	70 Features
PN	**93.60**	91.73	91.6	92.42
kNN	**92.74**	92.65	92.28	**92.74**
NB	**92.65**	91.73	92.28	92.33
RBF	**79.96**	77.64	78.36	77.41
SVM Poly	**89.08**	88.17	88.08	88.99
SVM RBF	**95.34**	94.11	93.7	93.19
LR	**95.07**	94.2	94.43	93.28

class, or just choosing a specified number of the corpus topmost features (refer to Table 17). This indicates that we have to make sure that all classes are covered evenly in the feature set selected for building classifiers, if we need high precision classification results.

For all classifiers, except LR and SVM-RBF, better F1 results are recorded using the corpus topmost 70 features compared with using the 100 or 96 reduced feature sets. This indicates that it is not always the case that using more features in training results in better classification performance. This remark was realized in some other studies.[32,50] This can be attributed to the low quality of the additional features, or the noise added with these additional features, especially in the case of rare classes (those with a small number of training documents and features) or closely related ones (those with many features in common). For NB, it could be that more features mean more violation of its naive assumption of words independence.

Trying to study the effect of the variation of the number of training documents among the R8 dataset classes, it is clear that common classes (those with a large number of training documents), e.g. *earn* (has 2840 documents) and *acq* (has 1596 documents) have much better classification performance compared with rare classes (those classes with a small number of training documents); e.g. *grain* (has just 41 training documents). Common classes recorded high classification performance using all classifiers tested in this research. But, the very important enhancement we could get using our PN classifier was clear when we selected equal percentage of features from each class (and actually only 0.5% of each class features). We could get a microaverage F1 measure of 95.24% for the rare class *grain* using just five features from this class in building the classifier. This result is very close to the

results recorded for the two common classes *acq* and *earn* (*acq* had a microaverage F1 of 94.3 and *earn* had a 96% microaverage F1 in the same experiment). Other classifiers recorded lower performance on this rare class using the same feature set, though LR recorded 100% F1 using 96 and 100 features on *grain*, and kNN and SVM-RBF recorded 94.74% F1 using 70 features on this rare class. This implies that the best performance for each classifier differs as the way to select the features and the number of features vary. The optimal number of features and the optimal selection method differs from one classifier to another, even when working on the same dataset. For a summarized performance of the six classifiers on common and rare classes, the reader can refer to Tables 18 through 21.

In most of the experiments, the lowest performance was recorded for the three classes: *money-fx*, *interest* and *ship*. After investigating test files misclassifications,

Table 18. Results of common and rare classes in R8 using 108 features.

	F1 Using 108 Features (Equal Percentage)						
Class	Poly	kNN	RBF	NB	SVM Poly	SVM RBF	LR
Earn	**96**	**96**	93.95	**95.88**	**97.2235**	**98.1986**	**97.494**
Acq	**94.3**	91.73	77.1	**94.5**	85.7858	**95.0774**	**95.722**
Grain	**95.24**	**94.74**	21.54	84.21	33.3333	**94.7368**	**95.238**

Table 19. Results of common and rare classes in R8 using 70 features.

	F1 Using 70 Topmost Corpus Features						
Class	Poly	kNN	RBF	NB	SVM Poly	SVM RBF	LR
Earn	**96.24**	**96.67**	93.34	**96.73**	**96.9612**	**97.0711**	**96.72**
Acq	**92.18**	91.5	78.75	**92.54**	86.2069	91.6953	**92.47**
Grain	86.96	**94.74**	17.54	86.96	46.1538	**94.7368**	90.91

Table 20. Results of common and rare classes in R8 using 100 features.

	F1 Using Topmost 100 Features						
Class	Poly	kNN	RBF	NB	SVM Poly	SVM RBF	LR
Earn	95.09	96.29	92.89	95.69	96.8927	**97.821**	**97.80**
Acq	91.94	91.59	79.42	92.73	84.769	93.3426	**95.03**
Grain	84.21	88.89	26.87	80	46.1538	94.7368	**100**

Table 21. Results of common and rare classes in R8 using 96 features.

	F1 Using 96 Features (Equal Number)						
Class	Poly	kNN	RBF	NB	SVM Poly	SVM RBF	LR
Earn	96.24	96.51	93.66	96.48	96.8956	**97.7716**	**97.57**
Acq	92.18	90.77	80.87	92.75	84.6779	92.8276	**94.39**
Grain	86.96	88.89	20	85.71	46.1538	94.7368	**100**

we found that the poor performance recorded on these classes is mainly attributed to the fact that (*interest*, *money-fx*), and (*ship*, *trade*) have many features in common. These common features cause misclassifications between these classes. In fact, (*acq, earn*) also have many features in common, but they have many training documents; a factor that reduces misclassifications of documents belonging to these classes. Nevertheless, using equal percentage of features from each class together with PN, SVM, and LR classifiers has recorded results above 80% microaverage F1 for these poor classes. Detailed classification results of these classes, using all classifiers, are given in Tables 22 through 25.

Regarding the distinguishable performance recorded by LR in all the experiments conducted in this research, this is totally consistent with recent studies which compared this classifier performance in TC with the top performers in this area, including the state-of-the-art SVM classifiers.[28–30,49]

Table 22. Results of poor performance classes in R8 using 108 features.

	F1 Using 108 Features (Equal Percentage)						
Class	Poly	kNN	RBF	NB	SVM Poly	SVM RBF	LR
Interest	80.8	**87.58**	57.86	73.1	83.1169	82.1192	83.44
Money-fx	**81.6**	78.48	48.37	71.68	41.0714	80.7453	**82.56**
Ship	**84.01**	65.5	18.18	76.06	5.4054	80.6452	**83.08**

Table 23. Results of poor performance classes in R8 using 70 features.

	F1 Using 70 Topmost Corpus Features						
Class	Poly	kNN	RBF	NB	SVM Poly	SVM RBF	LR
Interest	75.71	78.57	56.47	71.9	77.3006	**83.3333**	78.621
Money-fx	75	70.5	41.06	**86.64**	27.451	75.1678	79.07
Ship	82.35	**83.87**	6.06	78.79	20	65.4545	75.758

Table 24. Results of poor performance classes in R8 using 96 features.

	F1 Using 96 Features (Equal Number)						
Class	Poly	kNN	RBF	NB	SVM Poly	SVM RBF	LR
Interest	75.71	78.67	51.22	74.17	**80**	77.2414	78.6667
Money-fx	75	72.26	36.36	67.88	32.7273	76.6467	**80.4598**
Ship	82.35	64.15	11.76	76.92	5.4054	67.8571	**87.8788**

Table 25. Results of poor performance classes in R8 using 100 features.

	F1 Using 100 Topmost Corpus Features						
Class	Poly	kNN	RBF	NB	SVM Poly	SVM RBF	LR
Interest	**78.62**	**78.95**	56.63	73.1	**78.7097**	**78.3784**	76.3158
Money-fx	75.43	75.16	42.95	67.84	30.4762	77.7778	**78.5714**
Ship	**77.14**	76.67	14.63	76.47	5.4054	70.1754	76.9231

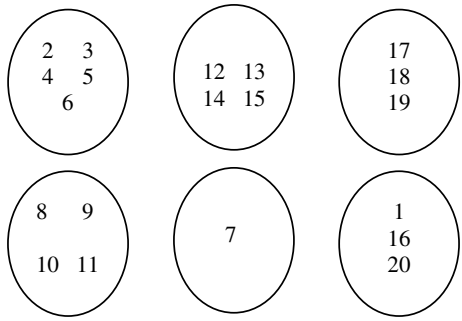

Fig. 1. 20NG classes: close classes (in topic) are encircled.

7.2. *Analysis of the results of the experiments on the 20 newsgroups dataset*

The 20 newsgroups are known to be a tougher domain for TC than Reuters. This is mainly due to two reasons. Firstly, many classes in the 20 newsgroups are very closely related to each other (in topic — refer to Fig. 1), which causes high misclassifications among these close classes.

Secondly, the way the two datasets articles were indexed is totally different: Reuters was indexed manually into categories on the basis of a restricted set of keywords, while the 20 newsgroups articles were labeled by their own creators based on a full understanding of the articles and their context. Consequently, automatic TC results of the 20 newsgroup are usually much lower than those of Reuters.

We conducted classification experiments on the 20 newsgroups using a reduced feature set, which contained 0.25% of each class features, using PNs, kNN, NB, SVM with RBF kernel, SVM with Polynomial kernel of degree 2, and LR. The top performers on the 20 newsgroups are PNs, NB, and LR. For NB, this is not surprising. Some recent papers have found NB to perform surprisingly well.[4,27,33] Despite the small feature set used in classification, we have recorded a PN classification performance close to many published results on this dataset (the reader can refer to related work in Sec. 8). Tables 15 and 16 in the previous section summarize the F1 measure per class in the 20 newsgroups dataset using the six classification algorithms, as well as other performance results. We believe that more experiments should be conducted on this dataset, using various settings, so as to come up with better conclusions regarding how to improve the classification results of this tough dataset.

8. Related Work

This section previews some earlier work in TC. It starts with a brief presentation of earlier work on Logistic Regression and Support Vector Machines. Then, some results of earlier research in TC are presented in brief.

8.1. *Support vector machines (SVM)*

Several studies in the literature of TC have investigated and proposed the use of SVMs as a machine learning technique for TC. Joachims[25] explored the use of Support Vector Machines (SVMs) in TC. He analyzed the properties of learning with text data and identified why SVMs are appropriate for TC. SVM (with RBF, Poly kernels) achieved great improvements over good traditional performers on TC (Naive Bayes, Rocchio, kNN, and Decision tree) in his experiments. He tested all algorithms on Reuters-21578 and Ohsumed. On Reuters, kNN performed best among the conventional methods and SVM was the best performer among all.

Dumais *et al.*[16] compared the effectiveness of five TC algorithms (SVM, Rocchio, Decision Trees, Naïve Bayes, and Bayes Nets) on Reuters-21578 in terms of learning speed, classification speed and classification accuracy. SVMs were the best among those algorithms tested in their work.

Basu *et al.*[5] examined the effect of feature reduction on the performance of text classifiers. They conducted their experiments on Reuters-21578 using two classifiers: SVM and Artificial Neural Networks (ANN). SVM outperformed ANN significantly in their reported results, and they found that reducing the feature set helps to improve performance in both classifiers.

8.2. *Logistic regression (LR)*

Komarek and Moore[28] investigated Logistic Regression's performance as a statistical method for use with large sparse datasets. Their results on a large life sciences dataset indicate that LR can perform surprisingly well, both statistically and computationally, when compared with a group of more recent classification algorithms (Linear SVM, RBF SVM, kNN, Decision tree, Bayes Classifier, and others). They conclude that there are circumstances in which LR equals or exceeds the performance of more recent algorithms.

Zhang *et al.*[49] used a modified version of LR to approximate the optimization of SVM by a sequence of unconstrained optimization problems. They proved that their approximation converges to SVM and proposed an iterative algorithm "MLR-CG" which uses Conjugate Gradient as its inner loop. They compared the MLR-CG with SVM over different TC collections (Reuters-21578 and Reuters Corpus volume I - RCV1) and showed that their algorithm is much more efficient than SVM when the number of training examples is very large.

Komarek and Moore[29] have conducted an empirical comparison of LR (IRLS+CG) to several classical and modern learners (Linear SVM, RBF SVM, kNN, Decision trees) on a variety of learning tasks on a group of datasets (Life sciences, Reuters-21578, citeseer, imb). LR recorded better performance on Reuters compared with SVM Linear and SVM RBF. The same authors demonstrated, in Ref. 30, that a very simple parameter-free implementation of LR (TR-IRLS) is sufficiently accurate and fast to compete with the state-of-the-art classifiers (Linear SVM, RBF SVM, kNN, Bayes) on large datasets (citeseer, imdb, life sciences, and modapte.sub).

8.3. *Results of some earlier experiments in TC*

Direct comparisons between TC algorithms had always been difficult due to the large variation between researchers in the datasets they use, even from benchmark data sets like Reuters and the 20 newsgroups. Variation in the feature selection methods, feature reduction methods, the number of features selected for use in classification, or the preprocessing steps applied on the documents, makes direct comparisons not possible. We have chosen to present here some studies which had worked on data sets, categorization algorithms, and evaluation measures that are close to those used in our experiments, in order to ease the indirect comparisons to some extent, and give a clearer idea about the performance of PN classifiers.

In a study conducted by Guo *et al.*,[22] the authors investigated two well known similarity-based learning approaches to TC: kNN and Rocchio classifiers. They propose a new classifier called the kNN model-based classifier by unifying the strengths of kNN and Rocchio classifiers. They conducted their experiments using subsets of the 20 newsgroup collection and Reuters-21578 news stories. In their experiments, they used a subset of the 20 newsgroup that includes 20 categories, each having 200 documents. For Reuters, they used the seven most frequent categories, each containing only 200 documents. These seven categories are: *acq*, *corn*, *crude*, *earn*, *interest*, *ship*, and *trade*. *IG* (Information Gain) was used as a feature selection criterion and the normalized *tf.idf* as the term weighting function. They set $K = 35$ for Reuters and $K = 45$ for the 20 newsgroup. Tables 26 and 27 show their F1 results on Reuters, and the 20 newsgroups respectively.

Another important study we present here was conducted by Debole and Sebastiani.[12] They proposed a number of supervised variations of *tf.idf* for term weighting. In their experiments, they used Reuters-21578 with SVM classifier, and three methods for feature selection (IG, Chi square, and gain ratio). They reported results for the Reuters subset of 115 categories with at least one training example, the subset of 90 categories with at least one training example and one test example and the set of ten categories with the highest number of training examples (R10). Their best results on R10 are presented in Table 28.

Table 26. F1 measure on Reuters.

Category	kNN, $K = 35$	Rocchio	kNN Model
Acq	78.81	84.03	86.08
Corn	87.08	87.74	91.85
Crude	84.12	84.51	84.72
Earn	88.02	90.39	89.45
Interest	79.67	80.84	83.26
Ship	84.02	86.22	86.73
Trade	80.69	81.64	80.00
Macroaveraged F1	83.2	85.05	86.01

Table 27. F1 measure on the 20 newsgroups.

Class #	Class	kNN, $K = 45$	Rocchio	kNN Model
1	alt.atheism	88.67	83.85	91.38
2	Comp.graphics	65.47	70.00	68.06
3	Comp.os.ms-windows.misc	65.27	66.67	67.40
4	Comp.sys.ibm.pc.hardware	55.78	55.89	58.70
5	Comp.sys.mac.hardware	56.92	57.91	61.04
6	Comp.windows.x	80.00	80.21	80.10
7	Misc.forsale	67.92	72.68	73.13
8	rec.autos	76.17	75.81	77.75
9	rec.motorcycles	88.50	88.42	89.41
10	rec.sport.baseball	85.86	88.61	90.00
11	rec.sport.hockey	90.12	93.27	92.73
12	sci.crypt	89.16	89.43	88.61
13	sci.electronics	70.17	69.09	73.60
14	sci.med	88.4	88.22	90.54
15	sci.space	83.45	86.76	86.83
16	soc.religion.christian	83.17	81.00	83.33
17	Talk.politics.guns	89.05	85.51	88.94
18	Talk.politics.mideast	91.39	93.09	91.54
19	Talk.politics.misc	85.11	75.37	83.29
20	Talk.religion.misc	80.83	74.18	79.47
MacroAveraged F1		79.07	78.80	80.79

Table 28. Results on R10.

Classifier	MicroAverage F1	MacroAverage F1
SVM	92	86

Table 29. Microaverage F1 — R10.

NB, MI	NB, FIS	SVM, MI	SVM, FIS
83.02	89.21	88.88	89.72

In a study by Fragoudis *et al.*[18] Feature and Instance Selection (FIS) was experimented on R10 and the 20 newsgroups using NB, SVM and other classifiers. We present some of their results in Tables 29 through 31.

Finally, we present the results of an experimental study conducted by Debole and Sebastiani[13] on the three subsets of Reuters-21578 that have been most popular among TC researchers, in order to find the relative hardness of each subset. They used two different weighting policies: cosine normalized form of *tf.idf* and a supervised term-weighting policy,[12] three different reduction factors of features: 0.90, 0.50, 0.0, and three feature selection criteria: Chi square, IG, and GR. They used three different classifiers: Rocchio, kNN and SVM. They tried all combinations of classifiers, reduction factors, term-weighting methods and feature selection criteria. Table 32 shows their best F1 scores achieved by any of the text classifiers (indeed it was SVM) using different parameters (term weighting policy, reduction factor, and feature selection criterion) on R10.

Table 30. A detailed F1 performance (per class) on R10.

	Classifier & Feature Selection Method	
Class	NB, MI	SVM, MI
Earn	96.9	**97.66**
Acq	87.33	91.47
Money-fx	57.45	65.06
Grain	75.08	91.75
Crude	80.11	80.87
Trade	54.98	70.20
Interest	50.90	62.50
Wheat	69.66	84.29
Ship	81.08	77.22
Corn	52.48	87.72

Table 31. Microaverage F1 — 20 newsgroups.

NB, MI	NB, FIS	SVM, MI	SVM, FIS
57.5	**69.52**	64.99	66.23

Table 32. Best results of Ref. 13 on **R10**.

MicroAverage F1	92
MacroAverage F1	86

9. Conclusion

In this paper, we have proposed and investigated using polynomial networks (PNs) as a machine learning approach in automated text categorization (TC). The two benchmark datasets: Reuters-21578 and the 20 newsgroups are used to test the proposed PN classifier. To be able to evaluate the performance of PNs in TC, we conducted our experiments, on the same data and feature subsets, using other well-known (including the state-of-the-art) algorithms in the literature of TC: Support Vector Machines (SVM), Logistic Regression (LR), the k-nearest-neighbor (kNN) Naive Bayes (NB), and the Radial Basis Function networks (RBF).

Direct comparisons show that our PN classifier is competitive to the top performers in TC. More importantly, PN classifiers are built in one shot (noniteratively), without the need for fine parameter tuning. Our proposed PN classifiers have recorded high classification performance on rare classes and closely related ones. Usually, rare classes (those with very few features or training documents) record lower classification results using any classifier. The same situation holds for classes that are closely related in topic. We could achieve high PN classification performance on these two groups of classes by applying feature reduction methods which cover all classes evenly in the reduced feature set selected for classification. All classifiers achieved their optimal precision on Reuters when an equal percentage of features is selected from each class to build the classifier.

Regarding the 20 newsgroups dataset, the PN classifier was among the top performers in our experiments, recording results close to those recorded in the literature, using just 0.25% of each class features.

Some of our intended, as well as other possible, extensions of this research are presented next.

We are working on another technique of applying PNs in TC that dramatically saves training and testing times as well as memory requirements. Such technique will enable using a larger set of features and higher polynomial degrees in classification. As a result, we would be able to compare the performance we will get using more features or higher polynomial degrees with the one resulted from using 0.25%–0.5% of the benchmark datasets' features with quadratic polynomials. This will answer very important questions: Do we really need to use the entire feature set or a larger number of features to get better performance when applying PNs in TC? Do we need to use polynomials of a degree higher than two to enhance PN classifiers performance? If not, this means that a major point that prohibited the use of PNs in TC for decades is not really an illness point or an obstacle.

Our proposed PN classifiers can be easily modified to handle multilabel TC; one of our intended near future work is to extend them to handle multilabel TC on Reuters. A possible trial to improve classifiers' performance is related to feature generation and weighting. New mechanisms can be experimented to enhance classifiers' performance. In addition, other feature reduction and selection methods can be tried and compared to those used in this research.

It is worth noting that our work can be further extended to handle Arabic texts instead of English texts; all what needs to be changed is the preprocessing phase.

References

1. Ana site for data sets suitable for single-label text categorization, http://www.gia.ist.utl.pt/∼acardoso/datasets/
2. K. T. Assaleh and M. Al Rousan, A new method for Arabic sign language recognition, Personal communications (2004).
3. K. T. Assaleh and W. M. Campbell, Speaker identification using a polynomial-based classifier, *Int. Symp. Signal Processing and Its Applications* (1999), pp. 115–118.
4. L. D. Baker and A. K. Mccallum, Distributional clustering of words for text categorization, *Proc. 21st Ann. Int. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR'98)* (1998), pp. 96–103.
5. A. Basu, C. Watters and M. Shepherd, Support vector machines for text categorization, *Proc. 36th Hawaii Int. Conf. System Sciences (HICSS'03), IEEE* (2002).
6. B. Boser, M. Guyon and V. Vapnik, A training algorithm for optimal margin classifiers, *Conf. Computational Learning Theory (COLT)* (1992), pp. 144–152.
7. C. J. C. Burges, A tutorial on support vector machines for pattern recognition, *Data Min. Knowl. Discov.* **2**(2) (1998) 121–167.
8. W. M. Campbell and K. T. Assaleh, Polynomial classifier techniques for speaker verification, *Proc. Int. Conf. Acoustics, Speech and Signal Processing* (1999), pp. 321–324.

9. W. M. Campbell, K. T. Assaleh and C. C. Broun, A novel algorithm for training polynomial networks, *Int. NAISO Symp. Information Science Innovations ISI'2001*, Dubai, UAE (March 2001).
10. W. M. Campbell and C. C. Boun, Using polynomial networks for speech recognition, Personal communications (2004).
11. C. Cortes and V. Vapnik, Support vector networks, *Mach. Learn.* **20** (1995) 273–297.
12. F. Debole and F. Sebastiani, Supervised term weighting for automated text categorization, *Proc. SAC-03, 18th ACM Symp. Applied Computing*, Melbourne, US (2003), pp. 784–788.
13. F. Debole and F. Sebastiani, An analysis of the relative hardness of Reuters-21578 subsets, *JASIS* **56**(6) (2005) 584–596.
14. P. Domingos and M. J. Pazzari, On the optimality of the simplest Bayesian classifier under zero-one loss, *Mach. Learn.* **29** (2/3) (1997) 103–130.
15. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis* (Wiley, New York, 1973).
16. S. Dumais, J. Platt, D. Heckerman and M. Sahami, Inductive learning algorithms and representations for text categorization, *Proc. ACM-CIKM'98* (1998).
17. G. Forman, An extensive empirical study of feature selection metrics for text classification, *J. Mach. Learn. Res.* **3** (2003) 1289–1305.
18. D. Fragoudis, D. Meretakis and S. Likothanassis, Integrating feature and instance selection for text classification, *SIGKDD'02* (2002), pp. 501–506.
19. K. Fuka and R. Hanka, Feature set reduction for document classification problems, *IJCAI-01 Workshop: Text Learning: Beyond Supervision*, Seattle, USA (August 2001) (2001).
20. K. Fukunaga, *Introduction to Statistical Pattern Recognition* (Academic Press, 1990).
21. G. H. Golub and C. F. Van Loan, *Matrix Computations* (John Hopkins, 1989).
22. G. Guo, H. Wang, D. Bell, Y. Bi and K. Greer. An kNN model-based approach and its application in text categorization, CICLing (2004), pp. 559–570.
23. M. H. Hassoun, *Fundamentals of Artificial Neural Networks* (The MIT Press, 1995).
24. S. C. H. Hoi, R. Jin and M. R. Lyu, Large-scale text categorization by batch mode learning, *Proc. 15th Int. Conf. World* (2006).
25. T. Joachims, Text categorization with support vector machines: learning with many relevant features, *Proc. 10th Euro. Conf. Machine Learning (ECML)* **1398** (1998) 137–142.
26. S.-B. Kim, H.-C. Seo and H.-C. Rim, Poisson Naïve Bayes for text classification with feature weighting, *Proc. 6th Int. Workshop on Asian Language*, Sapporo, Japan, ACL (July 2003), pp. 33–40.
27. D. Koller and M. Sahami, Hierarchically classifying documents using very few words, *Fourteenth Int. Conf. Machine Learning (ICML'97)* (1997), pp. 170–178.
28. P. Komarek and A. Moore, Fast robust logistic regression for large sparse datasets with binary output, *Artificial Intelligence and Statistics (AISTAT)* (2003).
29. P. Komarek and A. Moore, Fast logistic regression for data mining, text classification and link detection, paper submitted to *Neural Information Processing Systems Conf.* La Jolla, CA (July 2003).
30. P. Komarek and A. Moore, Making logistic regression a core data mining tool: a practical investigation of accuracy, speed, and simplicity, Technical Report TR-05-27 at the Robotics Institute, Carnegie Mellon University (May 2005).
31. Y. Lee, Handwritten digit recognition using k-nearest neighbor, radial basis functions, and backpropagation neural networks, *Neural Comput.* **3**(3) (1991) 440–449.

32. D. D. Lewis and M. Ringuette, A comparison of two learning algorithms for text categorization, *Proc. Third Ann. Symp. Document Analysis and Information Retrieval, (SDAIR'94)* (1994), pp. 81–93.
33. A. McCallum and K. Nigam, A comparison of event models for Naive Bayes text classification, *AAAI-98 Workshop on Learning for Text Categorization* (1998).
34. C. A. Micchelli, Interpolation of scattered data: distance and conditionally positive definite functions, *Constr. Approx.* **2** (1986) 11–22.
35. M. Niranjan and F. Fallsise, Neural networks and radial basis functions in classifying static speech patterns, Technical Report CUEDIF-INFENG17R22, Engineering Department, Cambridge University (1988).
36. S. J. Nowlan, Maximum likelihood competitive learning, in *Advances in Neural Information Processing Systems* **2**(Denver, 1989). ed. D. Touretzky, 1990, pp. 574–582.
37. E. Parzen, On estimation of a probability density function and mode, *Ann. Math. Stat.* **33** (1962) 1065–1076.
38. T. Poggio and F. Girosi, Networks for approximation and learning, *Proc. IEEE* 78(9) (1990) 1481–1497.
39. M. F. Porter, An algorithm for suffix stripping, *Program* **14**(3) (1980) 130–137.
40. M. J. D. Powell, Radial basis functions for multivariate interpolation: a review, *Algorithms for the Approximation of Functions and Data*, eds. J. C. Mason and M. G. Cox (Clarendon Press, Oxford, England, 1987), pp. 143–167.
41. M. Rogati and Y. Yang, High-performing feature selection for text classification, *CIKM'02* (November 2002), pp. 4–9.
42. D. F. Specht, Probabilistic neural networks, *Neural Networks* **3**(1) (1990) 109–118.
43. C. J. Van Rijsbergen, *Information Retrieval*, 2nd edn. (Butterworths, London, 1979).
44. V. Vapnik, *The Nature of Statistical Learning Theory* (Springer, New York, 1995).
45. V. Vapnik, *Statistical Learning Theory* (John Wiley & Sons, 1998).
46. D. Wettschereck and T. Dietterich, Improving the performance of radial basis function networks by learning center locations, *Advances in Neural Information Processing Systems* **4**(Denver, 1991), eds. J. E. Moody, S. J. Hanson and R. P. Lippmann (1992), pp. 1133–1140.
47. Y. Yang and X. Liu, A re-examination of text categorization methods, *SIGIR'99*, *ACM* (1999), pp. 42–49.
48. Y. Yang and J. Pederson, A comparative study on feature selection in text categorization, *Proc. Fourteenth Int. Conf. Machine Learning* (Morgan Kaufmann, San Francisco, 1997) pp. 412–420.
49. J. Zhang, R. Jin, Y. Yang and A. Hauptmann, Modified logistic regression: an approximation to SVM and its applications in large-scale text categorization, *Proc. 20th Int. Conf. Machine Learning (ICML)*, Washington, DC, USA (2003).
50. Z. Zheng, X. Wu and R. Srihari, Feature selection for text categorization on imbalanced data, *SIGKDD Explorations* **6**(1) (2004) 80–89.

Mayy M. Al-Tahrawi obtained her B.Sc. degree in computer science from Kuwait University in 1986, her M.Sc. also in computer science (compiler generators) from Kuwait University in 1988, and her Ph.D. in computer information systems from The Arab Academy for Banking and Financial Sciences, Jordan in 2006. She has worked as a lecturer in several universities in Jordan and Kuwait since 1988.

Currently, she is an Assistant Professor of computer information systems at Al-Ahliyya Amman University, Amman, Jordan.

Her research interests are in machine learning, pattern recognition, text categorization and information retrieval.

Raed Abu Zitar obtained his B.S. degree in electrical engineering from the University of Jordan in 1988, his M.S. in computer engineering from North Carolina A&T State University, U.S.A., in 1989, and his Ph.D. in computer engineering from Wayne State University, Detroit, U.S.A in 1993.

Currently he is a Professor of computer science and engineering at New York Institute of Technology, Amman, Jordan.

He has more than 50 publications in international journals and conferences, his research interests are in machine learning, pattern recognition and modeling.