Expert Systems with Applications 39 (2012) 10967–10976

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Automated text classification using a dynamic artificial neural network model

M. Ghiassi *, M. Olschimke, B. Moon, P. Arnaudo

Santa Clara University, 500 El Camino Real, Santa Clara, CA 95053-0388, USA

ARTICLE INFO

Keywords:
Classification
Textual document classification
Dynamic artificial neural networks
Pattern recognition
Machine learning
Artificial intelligence

ABSTRACT

Widespread digitization of information in today's internet age has intensified the need for effective textual document classification algorithms. Most real life classification problems, including text classification, genetic classification, medical classification, and others, are complex in nature and are characterized by high dimensionality. Current solution strategies include Naïve Bayes (NB), Neural Network (NN), Linear Least Squares Fit (LLSF), k-Nearest-Neighbor (kNN), and Support Vector Machines (SVM); with SVMs showing better results in most cases. In this paper we introduce a new approach called dynamic architecture for artificial neural networks (DAN2) as an alternative for solving textual document classification problems. DAN2 is a scalable algorithm that does not require parameter settings or network architecture configuration. To show DAN2 as an effective and scalable alternative for text classification, we present comparative results for the Reuters-21578 benchmark dataset. Our results show DAN2 to perform very well against the current leading solutions (kNN and SVM) using established classification metrics.

1. Introduction

Widespread adoption of the Internet, popularity of social networking, and the digitalization of information within organizations have led to large collections of digital documents. Automatic categorization and labeling are essential in processing digital information for all applications. For textual documents, classification is an important technique often utilized by researchers and practitioners. Text categorization research has its roots in the 1960s (Maron, 1961). Until the late 1980s, text categorization systems, based on knowledge-engineering, were built manually. In this approach, expert knowledge is encoded into a set of logical rules and is used to classify documents. An example of this approach is the categorization of news stories into a broad set of topical categories using pattern-matching techniques (Hayes, Knecht, & Cellio, 1988). This approach has been superseded by the machine learning approach in which the text classifier is automatically built by learning from an already classified text collection (the training set), as identified by domain experts. This approach is shown to be comparable with human performance while offering significant savings in costs and manpower (Sebastiani, 2002). However, despite advances in automated text categorization, manual categorization still remains the dominant approach in practice (Manning, Raghavan, & Schütze, 2008). The MEDLINE database, a bibliographic index of life sciences and biomedical documents, requires more than two million US dollars to categorize 350,000 new entries per year manually. To help reduce this cost, Hersh and Haynes (1991) developed an experimental information retrieval system with support for automatic indexing and natural language retrieval. Another application of machine learning approach is the work of Creecy, Masand, Smith, and Waltz (1992), who applied machine learning for the classification of responses to the 1990 Decennial Census. Their work is estimated to have saved 15 million US dollars. Automated text classification methods have also been used for document filtering, personalized news filters (Lang, 1995), Spam filters (Wang, Jones, & Pan, 2006), Web site categorization (Joachims, Freitag, & Mitchell, 1997), customer reviews, product evaluations and ratings with new applications sprouting out frequently.

In these machine learning methods, unprocessed texts are first preprocessed in order to create a representation of the text in vector space. This approach results in a very large vector of features; often 10,000s features. A common characteristic of this approach is that the matrix representing the text to be classified is often sparse. This is due to the fact that most words only appear in a few documents. Typical text classification experiments include a feature set ranging from 20,000 to more than 60,000 features. For instance, the two Spam filter datasets used by Wang et al. (2006), have initial dimensionalities of 24,749 and 65,694. Similarly, the Reuters RCV1 document collection (Lewis, Yang, Rose, & Li, 2004) includes 804,414 documents and consists of 47,152 features.

* Corresponding author. Address: 316 P Lucas Hall, Santa Clara University, 500 El Camino Real, Santa Clara, CA 95053-0388, USA. Tel.: +1 408 554 4687; fax: +1 408 554 2331.

E-mail addresses: mghiassi@scu.edu (M. Ghiassi), michael@olschimke.eu (M. Olschimke), bmoon@scu.edu (B. Moon), parnaudo@gmail.com (P. Arnaudo).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.03.027

In this research, we introduce a dynamic artificial neural network (DAN2) algorithm as an alternative approach for text classification. We use a standard text corpus, Reuters-21578, to demonstrate that DAN2 can be used as a machine learning tool with excellent results. We also demonstrate that the transformation and dimensionality reduction techniques used in DAN2 render its scalability and show that it can be used for classification problems of any dimension. Finally, the dynamic architecture of DAN2 does not require experimentation with parameter settings or network architecture configuration. These last two properties of DAN2 offer researchers an attractive alternative to existing approaches for automatic text classifications.

The benchmark used for this study is the Reuters dataset. This dataset has been studied extensively and is often used for benchmarking competing methods. Despite widespread usage, the prominent methods still struggle with achieving high levels of correct classification in all categories. The selection of this dataset allows us to compare and contrast our findings with results from other researchers. We discuss how we perform data preprocessing, the computational environment used, and the statistical measurements employed to study the effectiveness of our approach. We will use "performance" as a generic term for measures that evaluate the quality and accuracy of classification decisions, including precision, recall and F_1 measure. Additionally, we will include the time required for training and testing of the DAN2 model to record a benchmark of computational efficiency for future researchers.

The remainder of this paper is as follows: Section 2 presents a summary of existing strategies for text classification including text preprocessing steps, feature engineering methods, leading text classification methods, and the performance measurements metrics used to assess effectiveness of various strategies. The DAN2 algorithm and its properties are discussed in Section 3. Section 4 presents performance assessment of DAN2 and characterizes the Reuters dataset, followed by the document preprocessing steps applied, the feature engineering approach used and discusses DAN2 results. Finally, Section 5 presents our conclusion.

2. Background and literature review

Text classification problems are complex in nature and are always characterized by high dimensionality (Yang & Pedersen, 1997). To reduce this complexity researchers begin by applying preprocessing techniques to the original documents in order to produce a more simplified text. We group these preprocessing techniques into basic steps and standard steps. The basic steps include: (1) transforming the original, often XML-based files into plain text, (2) removing white spaces, (3) changing all words to lower case, and (4) removing punctuation and numbers. Once the basic preprocessing steps have been performed the document is ready for standard preprocessing activities. These are: (1) removing stop words, (2) stemming, (3) transforming the data into the vector space, (4) term weighting, (5) feature selection, and (6) splitting the data into training and test sets. These preprocessing steps transform the documents into a simpler form and a vector representation of the documents is produced (stages 1- 4 of standard steps). This vector is called the *bag-of-words*. Literature on information retrieval lists approaches commonly used for preprocessing activities and the steps necessary to produce such a vector (Frakes & Baeza-Yates, 1992; Korfhage, 1997; Kowalski, 1997; Lewis, 1992c; Lewis & Jones, 1996; van Rijsbergen, 1979). The transformed document's value for each term is defined using various *term weighting* methods. Common term weighting methods are binary weighting, the term frequency approach (TF), and the term-frequency/inverse document frequency (TF/IDF) method (Manning et al., 2008; Salton & Buckley, 1988). Automated software tools are available to perform some or all of these steps. In this research we use software tools from the R (Feinerer, 2009; Venables & Smith, 2009) and Weka (Hornik, Zeileis, Hothorn, & Buchta, 2006) systems to perform these activities.

Even after preprocessing, for large scale documents, the size of the resulting matrix is still very large and further activities are required to reduce its size. Feature engineering methods are introduced to reduce the size of such matrices. In text classification models, a feature represents a variable. Most real life text classification problems, including the Reuters benchmark used in this study, often involve 1000s or 10,000s of features. Feature engineering is applied in order to reduce the number of features. This is a common step for classification methods in order to reduce complexity (Yang & Pedersen, 1997). Additionally, feature engineering serves as a means to remove noise from the input documents (Manning et al., 2008). The most widely used feature engineering approaches are the *information gain* (IG) (Joachims, 1998; Lewis & Ringuette, 1994; Mitchell, 1996; Moulinier, Raskinis, & Ganascia, 1996; Quinlan, 1986), *document frequency thresholding* (DF) (Yang & Pedersen, 1997), *mutual information* (MI) (Manning et al., 2008), *term strength* (TS) (Yang & Pedersen, 1997), *latent semantic indexing* (LSI) (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990), and *the* χ^2 *statistic* (CHI) (Yang & Pedersen, 1997). We also use feature engineering to reduce complexity in our analysis.

Once the document is fully preprocessed and feature engineering techniques have been applied, the documents are divided into training and testing datasets and the process of classifying the documents can begin. The classification strategies used for analysis are binary and multi-class categorization. In binary classification only two categories are present: either the document belongs to the class or it does not. Most multi-class classification strategies are multiple iterations of binary classification. The most commonly used multi-class approaches are *one-against-all approach* (Manning et al., 2008; Sebastiani, 2002), *one-against-one approach* (Anthony, Gregg, & Tshilidzi, 2007; Friedman, 1996; Hsu & Lin, 2002; Kreßel, 1999), and *the hierarchical approach* (Ghiassi & Burnley, 2010; Lewis & Jones, 1996; Mahinovs & Tiwari, 2007). The only *"true"* multi-class approaches are decision tree methods (which are not efficient for large scale text classification problems) and the kNN method.

The methods and algorithms used for text classification using these strategies are few in number. The large number of possible features in text classification presents a problem for many classification algorithms. For instance the *Bayesian approach* will become computationally intractable for large applications (Yang & Pedersen, 1997). In order to address computational complexity issues, the number of features has to be reduced significantly. For example, Lewis has used Naïve Bayes approach with only 10 features on the Reuters-21450 dataset (Lewis, 1992a, 1992b). Using the same dataset, the best decision tree model used 90 features. Wiener, Pedersen, and Weigend (1995) report that their neural network based approach yielded the best average classification performance using about 20 features on the Reuters-22173 dataset, which is comparable to other research results (Apté, Damerau, & Weiss, 1994; Lewis, 1992a; Lewis & Ringuette, 1994). They also report that when using more than 20 features, the classification performance on the test set quickly decreases. The authors concluded that the neural network model used in their research was not able to extract features to generalize well on the test dataset. Alternatively, they suggest using much more than six hidden layers in their model; however, they found out that the classification performance increased only slightly. Schütze, Hull, and Pedersen (1995) evaluated linear discriminant analysis (LDA), logistic regression, and neural networks on the TREC-2 and TREC-3 (Harman, 1995a, 1995b) collections. They selected the 200 most relevant features using the χ^2 statistic. They also experimented with

Latent Semantic Indexing (LSI) (Deerwester et al., 1990) with 100 features. From their experiments, they conclude that LDA and neural networks show significantly better classification performance compared to a simple baseline classifier which ranks documents according to the proximity to the query vector in the vector space. Logistic regression performed better only when LSI was used for indexing. Other experiments with LSI have been performed by Yang (1994). In his experiments, he used medical data from the MEDLINE database to evaluate noise reduction strategies for Linear Least Squares Fit (LLSF) mapping. He evaluated strategies including removing non-informative words from the training dataset, using truncated singular value decomposition, and eliminating non-influential components in the LLSF matrices (Yang, 1994). He concludes that text classification of documents with a large number of features requires *"aggressive"* feature selection due to the noise in the training dataset and the computational requirements. Alternatively, some methods such as k-Nearest Neighbors (kNN) do not have problems with the large dimensionality of the feature space. This method has been used since the beginning of the machine learning approach for text categorization (Iwayama & Tokunaga, 1995; Masand, Linoff, & Waltz, 1992; Yang, 1994) and continues to be one of the better performing methods (Yang & Liu, 1999). Most recently, Support Vector Machines (SVM) have been shown to outperform kNN (Joachims, 1998, 1999, 2002). However, both methods are still limited by the size of the training corpus (Joachims, 2006). New attempts, such as parallel processing (Chang et al., 2007; Zhang, Li, & Yang, 2005) and the use of a Cutting-Plane Algorithm for solving the SVM optimization problem (Joachims, 2006) have been utilized to overcome the training size limitation problem.

Finally, in order to assess the effectiveness of various classification methods, performance measurement metrics are required. When evaluating the performance of a classifier, the classifier's judgment is evaluated against the judgment of a domain expert (Lewis, 1997). It is assumed that the expert's judgment is correct (Table 1). The relationship between the expert judgment and the classifier system judgment can be best expressed by a contingency table (Lewis, 1997).

The number of documents classified by both the classifier and the expert as being within the given category are called true positives (a). False positives (b) are documents assigned as being within the class but the experts disagree. If the system assigns a document as out of the class but the expert judges the document the opposite, the judgment is called a false negative (c). True negatives (d) are the judgments by both the experts and the system that put the document out of the class. (r) is the sum of all documents the experts classified as part of the class (thus n–r the opposite), and (k) is the sum of all documents the system classified as part of the class. (n) is the total number of judgments (i.e., the total number of documents in the test dataset).

The easiest approach to evaluate the performance of a classifier is to compare the number of right decisions (true positives and true negatives) with the number of wrong decisions (false positives and false negatives). This is defined as the *error rate* (a measure of accuracy). This approach works well with balanced datasets, i.e., where the number of documents per class is about the same as in the training set as it is in the test dataset (Ghiassi & Burnley, 2010).

Table 1
Contingency table.

	Expert says no	Expert says yes	
System says no	d (correct)	c (incorrect)	c + d = n − k
System says yes	b (incorrect)	a (correct)	a + b = k
	b + d = n − r	a + c = r	a + b + c + d = n

However, many text classification problems are outbalanced, including the Reuters benchmark dataset used in this study. To overcome this limitation of error rate and accuracy, researchers in text classification often use other metrics, for example *precision*, *recall*, *fallout,* and the F_1 measure. Precision is the percentage of documents correctly assigned to the class, i.e., these documents have also been assigned by the experts as part of the class (Lewis, 1997), or in other words: "How many positive decisions by the classifiers are true?" An alternative to precision is *fallout*. This "*is the proportion of non-class members that the system assigns to the class*," (Lewis, 1997). Precision has some limitations: it does not give guidance if all truly positive documents have been classified as such. *Recall,* on the other hand, is a measure that provides this information: it "*is the proportion of class members that the system assigns to the class*," (Lewis, 1997). In other words, of all positive expert judgments, how many have been "found" by the classifier? A perfect classifier would have an accuracy, precision and recall of 1.0 and error-rate and fallout of 0.0 (Lewis, 1997). However, both quantities trade off against each other: it is easy to achieve a recall of 1.0 but very low precision, just by classifying all documents into the class (Manning et al., 2008). The precision/recall break-even point is the point where both metrics are in balance (Manning et al., 2008). This metric is widely used in text categorization research (Joachims, 1998, 2006). An alternative solution to trade-off precision versus recall is the F measure, commonly written as the F_1 measure (Manning et al., 2008). For the Reuters benchmark dataset, researchers have used various performance measures. The variety of metrics used in the literature as well as the actual texts selected in their analysis do not allow for an easy and direct comparison of competing methods (Yang & Liu, 1999). To allow assessment of our results and to ensure compatibility with other researchers, we present DAN2's results using many metrics in order to compare with corresponding values offered in the literature. Another difficulty in comparing and evaluating performance of various algorithms is lack of consistency in feature selections. For most published results, there are insufficient details about the feature list and/or size of the training and testing datasets. We present our approach in detail and compare our results with the best published results we have found for each algorithm in order to avoid any biases.

3. A dynamic architecture for artificial neural networks

We have developed a neural network model, DAN2 (A Dynamic Architecture for Artificial Neural Networks), which employs a different architecture than the traditional neural network (FFBP) models. The general philosophy of the DAN2 model is based upon the principle of learning and accumulating knowledge at each layer, propagating and adjusting this knowledge forward to the next layer, and repeating these steps until the desired network performance criteria are reached. Fig. 1 presents the overall DAN2 architecture.

As in classical neural networks, the DAN2 architecture is composed of an input layer, hidden layers and an output layer. The input layer accepts external data to the model. In DAN2, unlike classical neural nets, the number of hidden layers is not fixed a priori. They are sequentially and dynamically generated until a level of performance accuracy is reached. Additionally, the proposed approach uses a fixed number of hidden nodes (four) in each hidden layer. This structure is not arbitrary, but justified by the estimation approach. At each hidden layer, the network is trained using all observations in the training set simultaneously, so as to minimize a stated training accuracy measure such as mean squared error (MSE) value or other accuracy measures such as F_1. As shown in Fig. 1, each hidden layer is composed of four nodes. The first node

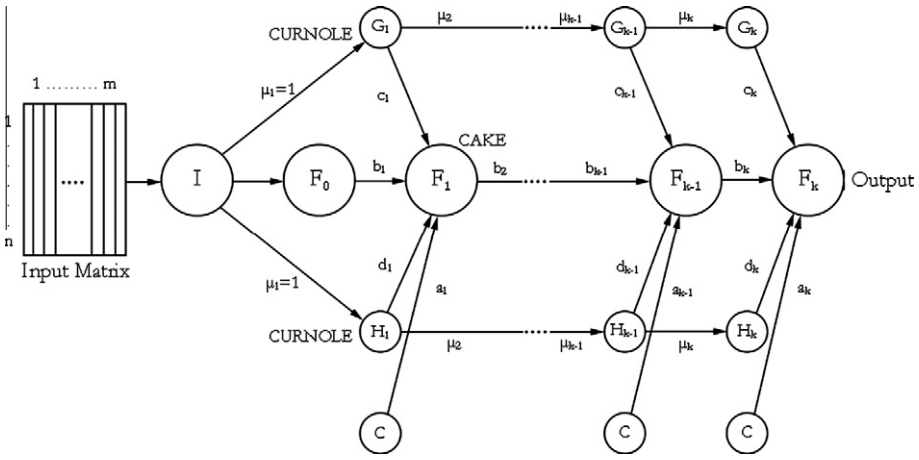

Fig. 1. The DAN2 network architecture.

is the bias or constant (e.g. 1) input node, referred to as the C node. The second node is a function that encapsulates the "Current Accumulated Knowledge Element" (CAKE node) during the previous training step. The third and fourth nodes represent the current residual (remaining) nonlinear component of the process via a transfer function of a weighted and normalized sum of the input variables. Such nodes represent the "Current Residual Nonlinear Element" (CURNOLE nodes). In Fig. 1, the "*I*" node represents the input, the "*C*" nodes are the constant nodes, the "G_k" and "H_k" nodes represent CURNOLE nodes, and the "F_k" nodes are the CAKE nodes. The final CAKE node represents the dependent variable or the output.

At each layer, the previous four nodes (C, G_k, H_k, and F_{k-1}) are used as the input to produce the next output value (F_k). The parameters on the arcs leading to the output nodes, (a_k, b_k, c_k, d_k), represent the weights of each input in the computation of the output for the next layer. The parameter connecting the CURNOLE nodes, μ_k, is used as part of the argument for the CURNOLE nodes and reflects the relative contribution of each input vector to the final output values at each layer. A detailed description of the architecture and its properties are fully presented in Ghiassi and Saidane (2005).

The training process begins with a special layer where the CAKE node captures the linear component of the input data. Thus, its input (content) is a linear combination (weighted sum) of the input variables and a constant input node. These weights are easily obtainable through classical linear regression. If the desired level of accuracy is reached, we can conclude that the relationship is linear and the training process stops. This step is used as the starting point. For classification problems this step can be replaced with alternative methods described in Section 3.2. For nonlinear relations additional hidden layers are required. At each subsequent layer the input to the CAKE node is a weighted sum (linear combination) of the previous layer's CAKE, CURNOLE, and C nodes. Throughout training, the CAKE nodes carry an adequate portion of learning achieved in previous layers forward. This process ensures that the performance or knowledge gained so far is adjusted and improved but not lost. This property of DAN2 introduces knowledge memorization to the model. Ghiassi and Saidane (2005) show that the DAN2 algorithm ensures that during network training, the residual error is reduced in every iteration and the accumulated knowledge is monotonically increased.

The training process defines creation of partitions among classes that could include linear and nonlinear components. The linear component of the input data is captured in the first CAKE node using ordinary least squares (OLS) or other simple and easy to compute approaches. The algorithm next transforms the input dataset to model the nonlinearity of the process in subsequent iterations. DAN2 uses a vector projection approach to perform data transformation. The transformation process defines a reference vector $R = \{r_j; j = 1, 2, \ldots, m\}$, where m represents the number of attributes of the observation records, and projects each observation record onto this vector to normalize the data as discussed in Ghiassi and Saidane (2005). This normalization defines an angle, α_i, between record i and the reference vector R. DAN2 uses the set of α_i's to train the network, and updates their values in every iteration. In Ghiassi and Saidane (2005), the authors show that this normalization can be represented by the trigonometric function Cosine $(\mu_k\alpha_i + \theta_k)$. In every hidden layer k of the architecture we vary $(\mu_k\alpha_i + \theta_k)$ and measure the impact of this change on the output value. The modification of the angle $(\mu_k\alpha_i + \theta_k)$ is equivalent to rotating μ_k and shifting θ_k, the reference vector, thus changing the impact of the projected input vectors and their contribution to the output for that iteration. The Cosine $(\mu_k\alpha_i + \theta_k)$ uses two (nonlinear) parameters, μ_k and θ_k. The use of the latter can be avoided through the expansion of the cosine function in the form: A Cosine $(\mu_k\alpha_i) + B$ Sine $(\mu_k\alpha_i)$. We use this functional form as the transfer function in our model. The two CURNOLE nodes in Fig. 1 represent this formulation. At any given hidden layer k, if the Cosine $(\mu_k\alpha_i + \theta_k)$ terms captured in previous layers do not adequately express the nonlinear behavior of the process, a new layer with an additional set of nodes is automatically generated, including a new Cosine $(\mu_k\alpha_i + \theta_k)$ term. This process is analogous to how the Fourier series adds new terms to improve function approximation. Therefore, the number of layers in the DAN2 architecture is dynamically defined and depends upon the complexity of the underlying process and the desired level of accuracy. Thus, the output of this model is represented by the linear combination of the constant, CAKE, and CURNOLE nodes. Eq. (1) represents the functional form of this relationship at iteration (layer) k

$$F_k(X_i) = a_k + b_k F_{k-1}(X_i) + c_k G_k(X_i) + d_k H_k(X_i) \quad (1)$$

where X_i represents the n independent input records, $F_k(X_i)$ represents the output value at layer k, $G_k(X_i)$ = Cosine $(\mu_k\alpha_i)$, and $H_k(X_i)$ = -Sine $(\mu_k\alpha_i)$ represent the transferred nonlinear components, and a_k, b_k, c_k, d_k, and μ_k are parameter values at iteration k. The training process initially captures the linear component by using OLS or other simple and easy to compute approaches. If the desired level of accuracy is reached, the training terminates. Otherwise, the model generates additional layers to capture the nonlinear component of the process by minimizing a measure of total error as represented

by $SSE_k = \Sigma_i[F_k(X_i) - \hat{F}(X_i)]^2$. Substituting $F_k(X_i)$ from Eq. (1) results in:

$$SSE_k = \Sigma_i[a_k + b_k F_{k-1}(X_i) + c_k \mathrm{Cos}(\mu_k \alpha_i) + d_k \mathrm{Sin}(\mu_k \alpha_i) - \hat{F}(X_i)]^2 \quad (2)$$

where $\hat{F}(X_i)$ are the observed output values. Minimizing Eq. (2) requires the estimation of five parameters. This formulation is linear in the parameter set A_k, where $A_k = \{a_k, b_k, c_k, d_k\}$ and nonlinear in parameter μ_k. In Ghiassi and Saidane (2005), they present several nonlinear optimization strategies to estimate the nonlinear parameter μ_k. They also show that following this approach, at each layer the knowledge gained is monotonically increased, total error is reduced, and the network training improves.

In Ghiassi, Saidane, and Zimbra (2005), the authors compare DAN2 with traditional FFBP and recurrent neural network (RNN) models. The comparison spans both theoretical and computational perspectives using several benchmark datasets from the literature. Performance of DAN2 against these models as well as non neural network alternatives is also presented. Their study shows that DAN2 outperforms all other alternatives and produces more accurate training and testing results in every case.

3.1. DAN2 for text classification

DAN2 has been shown to effectively solve a variety of continuous problems (including forecasting and functional approximation) (Ghiassi & Saidane, 2005; Ghiassi et al., 2005). Classification differs in that its output is discrete. However, classification can also be viewed as the process of drawing a partition between classes. DAN2 can be used to approximate a function that identifies this partition. Unlike linear and quadratic discriminant analysis, DAN2 does not assume the shape of the partition. In contrast to the kNN method, DAN2 does not require storage of training data. Once the model has been trained, DAN2 performs much faster than kNN because it does not need to iterate through individual training samples. Finally, DAN2 does not require experimentation and final selection of a kernel function and a penalty parameter as is required by SVM. DAN2 solely relies on a training process in order to identify the final classifier model. In Ghiassi and Burnley (2010), the authors demonstrate the effectiveness of DAN2 for a large number of classification benchmark datasets. This research extends DAN2's application to the text classification domain.

Ideally, the DAN2 model should be trained on some measure of classification error rates. However, this is not always practical since the misclassification error may remain constant for several training cycles and then change by a relatively small amount. This lack of sensitivity to small changes reduces the efficiency of the gradient descent algorithm. Instead of inching along a hill looking for the direction of descent, the algorithm would need to follow a flat plain for a long distance before finding out if the direction results in a step up or a step down. For this reason, DAN2 was trained using the mean squared error (MSE) of its output before the output was transformed into the discrete class. The training algorithm could take advantage of the small changes in MSE, which would not have been measurable in the misclassification rate. The assumption is that improvements in MSE will eventually result in improvements in the misclassification error. A lower MSE indicates that more of the output points are closer to the target class values. With more training samples near their target class values, less samples will be closer to the incorrect class value. As the classification of each sample is determined by the class value to which it is closest, fewer misclassifications should occur, resulting in a lower classification error rate. Throughout this process, classification error rates (F_1 and others) are also generated and used to assist with error rate search optimization.

3.2. Starting point

DAN2's algorithm, at every iteration, solves a nonlinear minimization problem represented by Eq. (2). Specifically, the nonlinear optimization strategy used in DAN2 estimates the nonlinear parameter μ_k. Like all nonlinear optimization methods for non-convex/non-concave functions obtaining global optimum is never guaranteed. Similar to other optimization applications choice of a good starting point can improve convergence to local optimum and beginning the search at various starting points can facilitate reaching multiple local optima. For large scale problems, like the Reuters dataset, selection of a good starting point can be very helpful.

In Ghiassi and Saidane (2005), they identify the starting point as $F_0(X)$ and use the training data and the standard linear regression to obtain its value. In classification problems, $F(X)$ only takes binary values. For the classification problem examined in this research, in addition to the standard MLR, we have experimented with alternative methods to compute starting point values. These include simple methods such as random assignment of 0 or 1 for $F_0(X)$, assigning all records to one class (either 0 or 1), as well as a rudimentary kNN solution. The advantage of the random assignment is the simplicity and speed of computation. However, this approach often does not offer good starting solutions. The kNN approach used to capture a starting point is a simplified method that only considers one or two values for k and uses only a subset of the features from the actual problem in order to quickly obtain a starting point value. For example, for the Reuters dataset we only used two values for k ($k = 1$ or 5) and limited the number of features to only 500. We then use values with the best error statistics as the starting point for DAN2.

3.3. Scalability

DAN2's algorithm is highly scalable. The algorithm is composed of a series of layers that are automatically and dynamically generated without requiring any input from the analysts. At each layer the observation vectors are projected into the reference vector to create the angle α_i as discussed in Section 3. The training and optimization at each layer only uses the transformed data to obtain the five parameters of Eq. (2). Therefore, regardless of the original size of the dataset, DAN2 only needs to compute the set of four linear parameters, $A_k = \{a_k, b_k, c_k, d_k\}$, and one nonlinear parameter μ_k at each iteration. For instance, for text classification problems with large feature sets, once the starting point is computed, each observation record, as represented by its many features, is projected onto the reference vector and its angle, α_i, is computed. The training process only uses the α_i values and only needs to compute the five parameters. Therefore, the original problem with a large feature set is converted to a 5-parameter model at each layer; thus scaling the problem size and demonstrating the scalability of the model. The scalability of DAN2 is a distinguishing strength of the approach from traditional artificial neural networks.

3.4. Hierarchical DAN2 for multiple class classification

The additional complexity inherent in multiple class classification problems presents a challenge to many classification methods. An approach that has been used to improve multiple class performance of classifiers is hierarchical models. Common hierarchical classifiers include hierarchical neural networks and decision trees. Decision trees often identify one or more attributes used to split data at each node, until each sample reaches a leaf node at which it is assigned to a class (Amasyali & Ersoy, 2008). Hierarchical neural networks may come in a variety of architectures. Both Porter and Liu (1996) and Lee and Ersoy (2007) describe a hierarchical

neural network in which difficult training samples are rejected from the first layer and used for training in subsequent layers (Kim, Kehtarnavaz, Yeary, & Thornton, 2003). The hierarchical neural network model first branches into classes that separate easily and then uses feature selection to build additional branches for the remaining classes, determining which features to use at each node of the tree. Finally, neural networks were used at each node to improve the performance of the tree.

Reasons for using hierarchical classifiers focus on reducing complexity. Porter and Liu (1996) describe hierarchical classifiers as a subset of modular classifiers. They suggest that modular classifiers often arise when a combination of factors include a large number of classes, classes have difficult shapes (are not compact, convex, or connected), classes do not have distinct boundaries, boundaries are highly nonlinear, and misclassification of some points carries a high penalty. Lee and Ersoy (2007) describe hierarchical classification as a way in which to detect data which are more difficult to classify in order to classify these data differently.

In Ghiassi and Burnley (2010), they follow this same reasoning and have developed a hierarchical DAN2 model. They postulate that if a class can be removed from the dataset, the remaining classes may become easier to classify without the influence of the removed class. Clearly, this approach assumes samples only belong to one class. Following research can examine DAN2's accuracy improvements using alternative approaches, such as those described in Kim et al. (2003), Lee and Ersoy (2007) and Porter and Liu (1996).

Since in the Reuters-21578 dataset, a sample can belong to multiple classes, therefore, most hierarchical models, including hierarchical DAN2, are not suitable. In order to simplify the training of DAN2 and improve its performance, the multiple class classification problem is broken down into several two-class datasets. Linear and quadratic discriminant analysis methods follow a similar approach, as both of these classification methods rely on functions that only apply in the case of two class datasets. Similar to the approach used by these two methods, DAN2 can also be trained on a collection of one-versus-all versions of the dataset. Similar to the case of discriminant analysis, these methods are then reconciled by classifying a sample as belonging to the version in which it had the highest output. As with the linear and quadratic discriminant analysis, a one-versus-all model for each class is constructed and run.

4. Performance assessment of DAN2 on a text classification benchmark

This section presents the results of using DAN2 on a benchmark dataset. We first present information about the choice of the benchmark used and follow that with a discussion of the computational environment for our study and the choice of metrics used to assess DAN2's effectiveness. The actual results of DAN2 on these benchmarks are then presented.

4.1. The Reuters benchmark dataset

To assess the effectiveness of DAN2 in solving text classification problems, we will evaluate its performance using the widely used Reuters benchmark dataset. This corpus has been used extensively in the literature and has become a de-facto benchmark (Joachims, 2006; Lewis, 1995; Sebastiani, 2002). Since we seek to demonstrate and evaluate the effectiveness of DAN2 for text classification and compare it with leading alternative models, we needed a benchmark which had been thoroughly researched with well documented test results. Reuters-21578 possesses these properties. This dataset, consisting of 21,578 documents, is based on newswire data from 1987 (Lewis, 1997). Similar to other researchers, we use the widely adopted ApteMod split of Reuters-21578. This version is obtained by eliminating unlabelled documents and selecting the categories with at least one document in both the training and test datasets (Yang & Liu, 1999). It consists of 118 topic categories. The ten largest classes in training and test datasets are presented in (Joachims, 1998; Manning et al., 2008). As stated by other researchers (Sebastiani, 2002; Yang & Liu, 1999) although this corpus is the most widely used benchmark dataset, researchers have used many configurations of this dataset in their analysis. Some have included "unlabeled" articles while others have only considered articles with more than 2000 words (Umar & Khiyal, 2007) resulting in the size of the dataset to range from high of 10,667 articles to low of 479 articles. The choice of categories have also varied from 93 to only five categories (Sebastiani, 2002; Umar & Khiyal, 2007). Unfortunately, this wide variation of test data and categories does not allow for a reliable comparison. Yang and Liu (1999) and Sebastiani (2002) have outlined three conditions for experimentation using this corpus. They are: (1) experimentation with the same dataset and same categories; (2) same split of the training and testing datasets; and (3) same measurement metrics. Sebastiani (2002) and others suggest the Reuters-21578 as the "standard" variant of Reuters, with 10 categories and use of BEP or F_1 as the measurement metrics. Table 2 lists the most widely used number of training and test articles for each category. We developed DAN2 classification models for these ten classes and compared our results with other methods using the datasets described in Table 2.

As discussed earlier, the raw text from this dataset first needs to be preprocessed. The basic and standard preprocessing steps discussed earlier are applied first. We follow the index representation approach used by other researchers (Lewis, 1992c; Lewis & Jones, 1996). In this approach, for each unique word in the document, a dimension is added to the document vector. The document's value for each dimension depends on the feature generation method employed. Once again, we follow earlier researchers' choice of feature generation and have adopted the TF/IDF weighting method (Joachims, 1998; Lewis et al., 2004) for our research. Finally, following studies of feature selection (Joachims, 1998; Lewis & Ringuette, 1994; Mitchell, 1996; Moulinier et al., 1996; Quinlan, 1986), we have selected the information gain (IG) approach as the method for this process. The splitting of the datasets into training and testing sets also followed earlier studies.

4.2. Computational strategies, requirements, and environment

Complex text classification problems require significant computational capabilities. A fast computer with sufficient memory and computing power and appropriate software systems is always

Table 2
Training and testing documents for Reuters-21578.

Category	Training	Testing	Total
Earn	2877	1087	3964
Acq	1650	719	2369
Money fx	538	179	717
Grain	433	149	582
Crude	389	189	578
Trade	369	118	487
Interest	347	131	478
Ship	197	89	286
Wheat	212	71	283
Corn	182	56	238
Total	7194	2788	9982

needed for such computation. Below we briefly describe the hardware and software configuration used in this research.

The computer system used for this experiment consists of a multi-core server with the following configuration: (a) 2 × Intel Quad Core Xeon @ 3.2 GHz, (b) 16 GB of RAM, (c) Adaptec Raid Controller with 4 SAS hard drives in RAID 1/0 configurations.

The operating system used on this computer was the 64-bit version of the SuSE Linux Enterprise Server (SLES, 11). We added the VMWare Server (VMWare, 2009) software system to provide support for our own tools. The primary modeling environment for preprocessing was the R environment (Feinerer, 2009; Venables & Smith, 2009) with, rWeka, *tm* (Hornik et al., 2006), and the latest version of the Java Runtime Environment (Java, 2009). In order to support the multiprocessor environment the Message Passing Interface (MPI) libraries were added to the system. This interface is a library that is used on clusters and super computers to distribute workloads for computation among multiple nodes. One such open-source MPI implementation is OpenMPI (Open MPI, 2009) which was installed on our system. We found these tools to offer the best available software environment for our research.

4.3. Performance measurement metrics

In order to compare our results with existing research we have selected the performance measurement metrics that were employed in earlier studies. The primary statistics used for performance measurement were break-even-point (Joachims, 1998) and F_1 (Manning et al., 2008) measures. We report our results using the exact same performance statistics in order to be able to offer a comparative analysis.

As noted by Sebastiani (2002) and Yang and Liu (1999), although many researchers have used the Reuters-21578 dataset to assess effectiveness of various methods, it has been difficult to compare and contrast reported results for these methods. The reasons for the difficulties are due to researchers' choices of "non-comparable performance measures" and/or the choice of selecting only a subset of the dataset (Yang & Liu, 1999). The authors, however, have presented results of a controlled study in which kNN and SVM were identified as the two best performing methods. These results were consistent with findings of Joachims (1998) and Manning et al. (2008). A comparative assessment based on these results can offer researchers guidelines on the effectiveness of available algorithms, even though as noted by other researchers, a controlled and statistically proven comparison is not feasible.

Our goal is to present DAN2 as an effective alternative approach for text classification rather than claiming superior performance. We present a detailed discussion of our experiment to clarify how our results are obtained. As such we have selected to first present our results using a host of performance metrics to show its effectiveness. Presenting DAN2's results using multiple measurement metrics avoids biases that might favor the algorithm as stated by Yang and Liu (1999). We then compare our results with kNN and SVM values using the specific metrics used by these approaches. Since there are multiple reported results for some of the algorithms, we use the best (most favorable) values from the literature and compare DAN2's results against them.

4.4. DAN2 results for the Reuters-21578

Even once the Reuters-21578 dataset has been preprocessed, the results remain very large. In any text processing problem, feature engineering techniques are used to further reduce the size of the resulting vector space to lessen the problem of complexity. Although DAN2 is a scalable approach, the convergence of the internal nonlinear optimization algorithms employed in DAN2 improves with reduced dimensionality. Reducing the feature size, therefore, can greatly enhance performance. Similar to other researchers we have used the "information gain" measure to select an appropriate set of features. We also show, confirming earlier studies (Yang & Wilbur, 1996), that reducing features beyond a certain level will result in less accurate models. Accordingly, choosing the number of features selected for each model is a challenging decision. We experimented with various feature sizes by starting at 1000 features and incrementing by 250. We noted that beyond 3000 features the model does not gain additional benefit. Specifically, we noticed a drop of 5% in forecasting precision when using more than 2500 features. We also noticed the same drop off of 5% for the precision/recall break-even point metric for feature size set beyond the 2000–2250 range. We further explored this region by incrementing the feature set by only 50 features and found the best number of features at around 2200. Fig. 2 shows this analysis which confirms findings of Yang and Wilbur (1996).

The selection and choice of feature set has been a challenging topic for all researchers using the Reuters dataset. Yang and Liu (1999) report using either χ^2 statistics or information gain criterion and experimenting with 1000 features for the NN model, 2000 features for the NB model, 2415 features for the kNN model (with $k = 45$), 2415 features for the LLSF model, and 10,000 features for the SVM model to optimize performance of each classifier using global F_1 score. The authors do not clarify which approach was used for each case. Similarly, Joachims (1998) reports experimenting with 500, 1000, 2000, 5000, and 10000 features for all classifiers. He uses the information gain criterion for feature selection and the "tfc" method for feature vector scaling. For the kNN model values of $k = 1$, 15, 30, 45, and 60 were used. The best results were reported in (Joachims, 1998).

Clearly, there is no established approach for selecting the best feature set for each model. However, researchers acknowledge that a small feature set masks effectiveness of the classifier and including a very large feature set beyond a "critical" level degrades classifiers' performance. In this study, we perform a systematic experimentation to identify a range for the feature set that offers the best performance. Our experimentation supports the general guidelines suggested by other researchers. Furthermore, we show that a feature set beyond a critical point indeed decreases classification effectiveness (Fig. 2). Although our experimentation is limited to the Reuters-21578 dataset, the approach used can be effective for all text classification models.

Similar to other studies, once the feature set was selected we developed ten models, one for each class of the Reuters-21578 dataset. For each model we used a rudimentary kNN with only 500 features and $k = 1$ or $k = 5$ to generate a starting point for DAN2. For each model we used the training dataset to train the

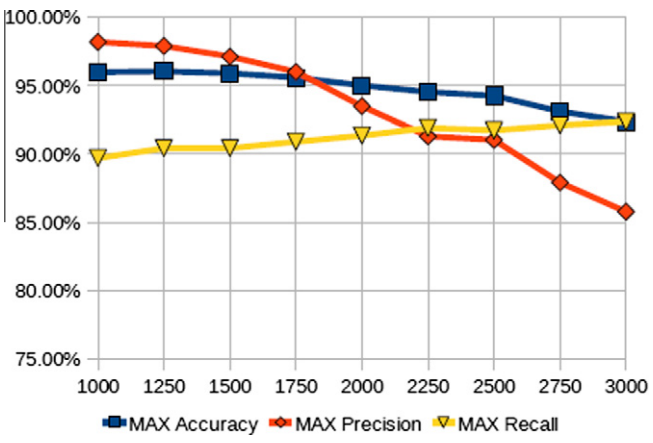

Fig. 2. Number of features for training.

model and the testing dataset to measure its generalization accuracy. To avoid under/over fitting of the model we made sure that the accuracy measures for both the training and testing datasets were reasonably close. We report both these values for all models. Table 3 presents our results using precision, recall, break-even-point, and F_1 measures for both the training and testing datasets. Similar to other studies, the best results were obtained for categories that were more balanced (earn, acq, and grain), and the worst results were for less balanced categories (interest and trade). When comparing the effect of the metrics used, we observed that for compound metrics such as F_1 and break-even-point measures, (a) the absolute break-even-point values are slightly better (higher values) than the corresponding F_1 values, (b) the relative performance measures within the categories for both metrics follow the same direction with both metrics identifying "earn" as the most accurate and "interest" as the least accurate models, and (c) both metrics offer results that are very close to one another. To assess the effectiveness of DAN2 against other approaches, therefore, we could have used any one of these metrics without the loss of generality. Overall, the performance of DAN2 for text classification, using the Retuers-21578 dataset, has shown remarkable results and is consistent with its performance using other classification benchmarks as reported in Ghiassi and Burnley (2010).

Table 4 presents results for kNN, SVM, and DAN2 for the top ten most common categories of the Reuters-21578 dataset. We note that although the literature presents multiple results for kNN and SVM methods, Joachims (1998, 2002), Manning et al. (2008) and Yang and Liu (1999) published results for kNN and SVM lack detailed information about the individual training and testing datasets for each category. For SVM, specifically, results are reported for linear, polynomial, and RBF kernels. The choices of measures used for feature selection and accuracy reporting also vary (Joachims, 2002; Manning et al., 2008). However, the most widely reported results are the Joachims (1998) results for both kNN and SVM. We use these values for our comparisons and they are presented in Table 4. We note that (a) the purpose of this comparison is to demonstrate effectiveness of DAN2 as measured in comparison with other methods as opposed to claiming or demonstrating superiority of the approach, and (b) various reported results for kNN and SVMs, even though the training and testing dataset sizes are not clearly stated, do not significantly differ from values reported in Table 4, and (c) values reported for kNN and SVM in Table 4 use the break-even-point metrics which allow us to directly compare with DAN2 values. Therefore, we report kNN values and the best overall SVM values (as measured by the combined micro average value) for the RBF-based SVM with the parameter value of $\gamma = 0.8$ (Joachims, 1998). Table 4 shows break-even-point values for DAN2 (testing dataset) to be uniformly better than the corresponding values for kNN, with the corn category showing the most improvement (up to 26%). When comparing DAN2 break-even-point values with the corresponding ones from the best SVM values reported (testing dataset), DAN2 produces better results in all cases except the "Acq" category.

Table 4
DAN2 break-even-point value comparison with kNN and SVM (in percent).

Category	DAN2 BEP training	DAN2 BEP testing	kNN BEP	SVM BEP
Earn	99.32	99.23	97.30	98.50
Acq	96.56	93.85	92.00	95.30
Money-fx	79.86	83.17	78.20	75.40
Grain	98.68	96.21	82.20	91.90
Crude	89.62	94.10	85.70	89.00
Trade	79.60	84.07	77.40	78.00
Interest	94.80	83.59	74.00	75.00
Ship	92.42	92.03	79.20	86.50
Wheat	97.33	88.28	76.60	85.90
Corn	95.77	98.53	77.90	85.70

To statistically assess DAN2's results against kNN and SVM values, we use the "Macro T-test" significance test suggested in Yang and Liu (1999). This is a *T*-test for comparing the means of two systems. We present the difference between any two values for each category by X_i for $i = 1, 2, \ldots, 10$. Let $\overline{X}$ to be the simple average for X_i. The null hypothesis is $\overline{X} = 0$ with the alternative hypothesis as $\overline{X} > 0$. The computed *T*-value is defined as:

$$T = (\overline{X}/S \cdot E(\overline{X}))$$

We use the testing dataset BEP values for DAN2, kNN and SVM from Table 4 to compute the T-values. For DAN2 vs. kNN the computed T-value is 5.29 and for DAN2 vs. SVM the computed T-value is 2.11. The computed *T*-values are compared against the *t*-student values with $(n - 1)$ degrees of freedom at $P = 0.05$ (1.833). The statistical test of significance shows that DAN2 performs better than kNN and SVM at $P = 0.05$. Clearly, the results show DAN2 to be a very effective alternative to existing approaches for text categorization, as measured in this experiment.

4.5. DAN2 computational times for Reuters-21578

All machine learning algorithms, including DAN2, NN, SVM and kNN, require experimentation to find the best training model. For example, for SVM, the value of the penalty parameter and the choice of the kernel function needs to be selected experimentally (Joachims, 1998), while with kNN, the best *k* value can only be determined by trial and error. For NN models, the architecture (number of layers and number of hidden nodes) is experimentally defined, and then the parameters of the model are computed. Similarly, training of a DAN2 model requires experimentation. DAN2 initially defines a set of stopping conditions. These conditions include the maximum number of iterations and the target accuracy (the algorithm will stop at whichever condition is met first). At each iteration, DAN2 solves a nonlinear optimization problem. In Ghiassi and Saidane (2005), the authors offer a number of alternatives to solve the intermediate nonlinear optimization problem.

Table 3
DAN2 training and testing results for Reuters-21578 (in percent).

Category	Precision training	Precision testing	Recall training	Recall testing	BEP training	BEP testing	F_1 training	F_1 testing
Earn	98.87	98.80	99.77	99.66	99.32	99.23	99.32	99.23
Acq	99.51	97.56	93.62	90.14	96.56	93.85	96.47	93.70
Money-fx	96.14	97.69	63.58	68.65	79.86	83.17	76.54	80.63
Grain	98.43	94.74	98.94	97.67	98.68	96.21	98.68	96.18
Crude	98.05	99.13	81.18	89.06	89.62	94.10	88.82	93.83
Trade	96.33	99.06	62.87	69.08	79.60	84.07	76.09	81.40
Interest	97.98	98.67	91.62	68.52	94.80	83.59	94.69	80.87
Ship	95.65	95.00	89.19	89.06	92.42	92.03	92.31	91.94
Wheat	98.04	96.55	96.62	80.00	97.33	88.28	97.30	87.50
Corn	97.94	100.00	93.60	97.06	95.77	98.53	95.72	98.51

Table 5
Training and testing times for reuters categories with DAN2.

Category	DAN2 training time (s)	DAN2 testing time (s)
Earn	410.2969	9.5859
Acq	203.1875	8.2891
Money-fx	9.3828	0.1563
Grain	71.7500	0.3359
Crude	8.9390	0.3460
Trade	4.0781	0.0586
Interest	11.3672	0.3281
Ship	27.4453	0.7109
Wheat	30.0469	0.0081
Corn	23.9453	0.2500

The implementation used in this research utilizes the grid search approach. In DAN2 a 2-dimensional search (interval and step sizes) is employed to find the best μ_k at each iteration. The algorithm begins with larger intervals and step sizes to determine promising search regions. When promising sectors are identified, subsequent more granular 2-D searches are performed on each sector to find the (local) optima. When a step produces inferior results, it is discarded and the process starts over with a different interval and step size. Once the desired level of training accuracy is reached, the search terminates. The algorithm uses internally set metrics to avoid over fitting as described in Ghiassi et al. (2005).

We timed the computational effort required to train each category to assess DAN2's efficacy. Table 5 reports training and testing times required for each of the ten models. We note that although the size of the training set is an important variable in model training time, it is not the only significant factor. Clearly, the complexity of the problem, the underlying nonlinear model and the starting points, are also influential. As presented in Table 5, in our computational environment, the largest category (earn) takes the longest (410 s) to train. The shortest training times were for trade and crude models, at 4 and 9 s, respectively. Once the models are trained, however, the actual runtime of the model for testing sets is very short and is mostly proportional to the size of the test sets. For this data set, the longest testing time for the trained DAN2 models is 9.6 seconds for earn and the shortest time is for wheat 0.008 s. Similar information for SVM and kNN was not provided in the literature and thus, we cannot directly compare DAN2's computational times with others.

5. Conclusions

This paper presented DAN2 as an alternative approach for automated text categorization. DAN2 is scalable and can be used for document text categorization without requiring experimentation with parameter settings or network architectural configurations. We first presented a summary of DAN2's theoretical foundations and then showed DAN2's effectiveness by comparing its performance against the leading text categorization methods (kNN and SVM) using the established Reuters-21578 benchmark dataset. Results show DAN2 as outperforming kNN for all cases and SVM for nine out of the ten cases. Overall, results show that DAN2 performs better than both kNN and SVM, at $P = 0.05$ level of significance, for this dataset. The excellent uniform performance of DAN2 indicates the robustness of the approach and its generality. Finally, the transformation and dimensionality reduction property of DAN2 makes its application to text classification especially important.

References

Amasyali, M. F., & Ersoy, O. (2008). Cline: A new decision-tree family. *IEEE Transactions on Neural Networks, 19*(2), 356–363.

Anthony, G., Gregg, H., & Tshilidzi, M. (2007). Image Classification Using SVMs: One against-One Vs One-against-All. In *Proc. 28th Asian Conference on Remote Sensing ARCS, Learning; Artificial Intelligence. Computer Vision and Pattern Recognition, 2007*, 12–16.

Apté, C., Damerau, F., & Weiss, S. M. (1994). Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems, 12*(3), 233–251.

Chang, E. Y., Zhu, K., Wang, H., Bai, H., Li, J., Qiu, Z., et al. (2007). PSVM: parallelizing support vector machines on distributed computers. *In Advances in Neural Information Processing Systems, 20*, 1–8.

Creecy, R. H., Masand, B. M., Smith, S. J., & Waltz, D. L. (1992). Trading MIPS and memory for knowledge engineering. *Communications of the ACM, 35*(8), 48–63.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science, 41*(6), 391–407.

Feinerer, I. (2009). A framework for text mining applications within R. http://cran.r-project.org/web/packages/tm/index.html. Accessed 7 June 2009.

Frakes, W. B., & Baeza-Yates, R. (1992). *Information Retrieval: Data Structures and Algorithms*. Prentice Hall.

Friedman, J. H. (1996). Another approach to polychotomous classification. Stanford University, Technical Report.

Ghiassi, M., & Burnley, C. (2010). Measuring Effectiveness of a Dynamic Artificial Neural Network Algorithm for Classification Problems. *Expert Systems with Applications Journal, 37*(4), 3118–3128l.

Ghiassi, M., & Saidane, H. (2005). A dynamic architecture for artificial neural network. *Neurocomputing, 63*, 397–413.

Ghiassi, M., Saidane, H., & Zimbra, D. K. (2005). A dynamic artificial neural network model for forecasting time series events. *International Journal of Forecasting, 21*, 241–362.

Harman, D. (1995a). Overview of the Second Text Retrieval Conference (TREC-2). *Information Processing Management, 31*(3), 271–289.

Harman, D. (1995b). *Overview of the Third Text Retrieval Conference (TREC-3)*. Gaithersburg, MD: National Institute of Standards and Technology. 20899.

Hayes, P., Knecht, L., & Cellio, M. (1988). A news story categorization system. In *Proceedings of ANLP-88, 2nd Conference on Applied Natural Language Processing*.

Hersh, W. R., & Haynes, R. B. (1991). Evaluation of SAPHIRE: an automated approach to indexing and retrieving medical literature. In *Proceedings of the annual symposium computer application to, medical care* (pp. 808–812).

Hornik, K., Zeileis, A., Hothorn, T., & Buchta, C. (2006). RWeka: An R Interface to Weka. http://www.cran.r-project.org/web/packages/RWeka/index.html. Accessed 7 June 2009.

Hsu, C.-W., & Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks, 13*(2), 415–425.

Iwayama, M., & Tokunaga, T. (1995). Cluster-based text categorization: a comparison of category search strategies. In *Proceedings of SIGIR'95, Seattle WA* (pp. 273–280).

Java Runtime. What is Java? (2009). http://www.java.com/en/download/whatis_java.jsp. Accessed 7 June 2009.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning, Chemnitz, Germany, April 21–23* (pp. 137–142).

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the sixteenth international conference on machine learning, Bled, Slovenia* (pp. 200–209).

Joachims, T. (2002). SVM Light: Support Vector Machine. http://svmlight.joachims.org. Accessed 31 July 2009.

Joachims, T. (2006). Training linear SVMs in linear time. In *Proceedings of KDD'06, Philadelophia, PA* (pp. 217–226).

Joachims, T., Freitag, D., & Mitchell, T. (1997). WebWatcher: A Tour Guide for the World Wide Web. In *Proceedings of the 1997 international joint conference on artificial intelligence*.

Kim, N., Kehtarnavaz, N., Yeary, M. B., & Thornton, S. (2003). DSP-based hierarchical neural network modulation signal classification. *IEEE Transactions on Neural Networks, 14*(5), 1065–1071.

Korfhage, R. R. (1997). *Information storage and retrieval*. Hoboken, NJ: John Wiley & Sons, Inc..

Kowalski, G. (1997). *Information retrieval systems: Theory and implementation*. New York, NY: Springer-Verlag.

Kreßel, U. (1999). Pairwise classification and support vector machines. In *Advances in Kernel Methods: Support Vector Learning*. Cambridge, MA: MIT Press, pp. 255–268.

Lang, K. (1995). NewsWeeder: Learning to Filter Netnews. In *Proceedings of the 12th international machine learning conference (ICML 95)*, (pp. 331–339).

Lee, J., & Ersoy, O. K. (2007). Consensual and hierarchical classification of remotely sensed multispectral images. *IEEE Transactions on Geoscience and Remote Sensing, 45*(9), 2953–2963.

Lewis, D. D. (1992a). Representation and learning in information retrieval. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, MA.

Lewis, D. D. (1992b). An evaluation of phrasal and clustered representations on a text categorization task. In E. A. Fox, P. Ingwersen, & R. Fidel (Eds.), *Proceedings of SIGIR-95, 18th ACM internationa1 conference on research and development in 1nformation retrieva1* (pp. 246–254), Seatt1e, WA, ACM Pres, New York.

Lewis, D. D. (1992c). Text representation for intelligent text retrieval: A classification-oriented view. In *Proceedings of the 15th annual international SIGIR '92* (pp. 37–50), Denmark.

Lewis, D. D. (1995). Evaluating and Optimizing Autonomous Text Classification System. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 246–254). WA: Seattle.

Lewis, D. D. (1997). Reuters-21578 text categorization test collection. http://kdd.ics.uci.edu/databases/reuters21578/README.txt. Accessed 7 June 2009.

Lewis, D. D., & Ringuette, M. (1994). A comparison of two learning algorithms for text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval* (Las Vegas, NV) (pp. 81–93).

Lewis, D. D., & Jones, K. S. (1996). Natural language processing for information retrieval. *Communications of the ACM, 39*(1), 92–101.

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research, 5*, 361–397.

Mahinovs, A., & Tiwari, A. (2007). Text classification method review. Decision Engineering Report Series, http://dspace.lib.cranfield.ac.uk/bitstream/1826/1860/1/mahinovs.pdf. Accessed 7 June 2009.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Maron, M. (1961). Automatic indexing: an experimental inquiry. *Journal of the ACM, 8*(3), 404–417.

Masand, B., Linoff, G., & Waltz, D. (1992). Classifying news stories using memory based reasoning. In *Proceedings of the 15th annual international SIGIR '92* (pp. 59–65), Denmark.

Mitchell, T. (1996). *Machine learning*. Columbus, OH: McGraw Hill.

Moulinier, I., Raskinis, G., & Ganascia, J. (1996). Text categorization: a symbolic approach. In *Proceedings of SDAIR-96, 5th annual symposium on document analysis and information retrieval* (Las Vegas, NV).

Open MPI, Open Source High Performance Computing, (2009). http://www.open-mpi.org/. Accessed 7 June 2009.

Porter, W. A., & Liu, W. (1996). Fuzzy HMC classifiers. *Information Sciences, 94*, 151–178.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*, 81–106.

Salton, G., & Buckley, C. (1988). Term weighting approaches in automatic text retrieval. *Information Processing & Management, 24*(5), 513–523.

Schütze, H., Hull, D. A., & Pedersen, J. O. (1995). A comparison of classifiers and document representations for the routing problem. In *Proceedings of SIGIR-95,18th ACM International Conference on Research and Development in Information Retrieval* (pp. 229–237). WA: Seattle.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys, 34*(1), 1–47.

Umar, M. F., & Khiyal, M. S. H. (2007). Classification of textual documents using learning vector quantization. *Information Technology Journal, 6*(1), 154–159.

van Rijsbergen, C. J. (1979). *Information Retrieval* (2nd Ed.). Jordan Hill, Oxford: Butterworth-Heinemann.

Venables, W. N., & Smith, D. M. (2009). *An introduction to R*. Network Theory Ltd.

VMWare (2009). http://www.vmware.com/products/server. Accessed 7 June 2009.

Wang, B., Jones, G. J. F., & Pan, W. (2006). Using online linear classifiers to filter spam emails. *Pattern Analysis & Applications, 9*, 339–351.

Wiener, E., Pedersen, J. O., & Weigend, A. S. (1995). A neural network approach to topic spotting. In *Proceedings of the fourth annual symposium on document analysis and, information retrieval (SDAIR'95)*.

Yang, Y. (1994). Expert network: effective and efficient learning from human decisions in text categorization and retrieval. In *17th Annual international ACM SIGIR conference on research and development in, information retrieval (SIGIR'94)* (pp. 13–22).

Yang, Y. & Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of SIGIR-99, 22nd ACM international conference on research and development in information retrieval* (pp. 42–49). Berkeley, CA.

Yang, Y., & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th international conference on machine learning* (pp. 412–420). TN: Nashville.

Yang, Y., & Wilbur, J. (1996). Using corpus statistics to remove redundant words in text categorization. *Journal of the American Society for Information Science, 57*(5), 357–369.

Zhang, J.-P., Li, Z.-W., & Yang, J. (2005). A parallel SVM training algorithm on large scale classification problems. *Machine Learning and Cybernetics, 3*, 1637–1641.