Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

Combining Machine Learning and Hierarchical Indexing Structures for Text Categorization

Miguel E. Ruiz & Padmini Srinivasan
School of Library and Information Science
The University of Iowa, Iowa City, Iowa

Abstract

This paper presents a method that exploits the hierarchical structure of an indexing vocabulary to guide the development and training of machine learning methods for automatic text categorization. We present the design of a hierarchical classifier based on the divide and conquer principle. The method is evaluated using backpropagation neural networks, as the machine learning algorithm, that learn to assign MeSH categories to a subset of MEDLINE records. Comparisons with traditional Rocchio's algorithm adapted for text categorization, as well as flat neural network classifiers are provided. The results indicate that the use of hierarchical structures improves performance significantly.

Introduction

Text categorization, also known as automatic indexing, is the process of algorithmically analyzing an electronic document to assign a set of categories (or index terms) that succinctly describe the content of the document. This assignment can be used for classification, filtering, or retrieval purposes. Manual indexing has been applied since the invention of writing to facilitate access to information. For years librarians have work on indexing using controlled vocabularies such as the Library of Congress Subject Headings, and Medical Subject Headings (MeSH). The increasing amount of information available in different areas of knowledge creates the need to automate part of this process. Automatic indexing algorithms based on statistical patterns in language appeared during the 1960's, and 1970's (Luhn, 1958; Salton, 1983). During the 1980's several systems were created for computer aided indexing. During the late 1980's several works applied expert systems to create knowledge-based indexing systems, e.g. MedIndeEx System at the National Library of Medicine (Humphrey, 1988). The 1990's could be characterized by the advent of the World Wide Web (WWW) which has made available a vast amount of information that is potentially useful. By the end of 1997 the estimated size of the WWW was 320 million pages and it is expected to double its size every year. The

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

information overload created by the WWW has stimulated the creation of reliable automatic indexing methods that could help users filter large amounts of documents.

Sections 2 and 3 will present an overview of machine learning, and its application to automatic text categorization. Section 4 presents the hierarchical model, and the details for designing and building such a model. Sections 5, 6 and 7 explain the experimental settings, evaluation measures and the experiments performed. Finally we present the results obtained and its analysis in section 8. Section 9 presents our conclusions and future work.

Theoretical Background in Machine Learning

Since computers were invented, we have wondered whether it is possible for them to learn how to perform specific tasks. Machine learning is an area of artificial intelligence that has been dedicated to this goal. Although it is not known yet how to make computers learn as well as people do, some algorithms have been invented for certain types of learning tasks. Machine learning algorithms have proven to be very successful in solving many problems, for example, the best results in speech recognition have been obtained with such algorithms. Machine learning algorithms learn by performing a search on the space of the problem to be solved. Two kinds of machine learning algorithms have been developed: supervised learning, and unsupervised learning. Supervised learning algorithms operate by learning the objective function from a set of training examples and then applying the learned function to the target set. Unsupervised learning operates by trying to find useful relations between the elements of the target set. A large number of machine learning algorithms have been invented and a detailed review of all of them is out of the scope of this paper but the reader is referred to (Mitchell, 1998) for such a presentation.

Text categorization can be characterized as a supervised learning problem. We have a set of example documents that have been correctly categorized (usually by human indexers). This set is then used to train a classifier based on a machine learning algorithm. The trained classifier is then used to categorize the target set.

Different machine learning algorithms such as decision trees (Moulinier, 1997), inductive learning (Apte and Damerau, 1994), neural networks (Ng et al, 1997; Weiner et al., 1995), linear classifiers (Lewis et al., 1996), K-nearest neighbor algorithms (Yang, 1999), support vector machines (Joachims, 1997), and naive Bayes classifiers (Lewis and Ringuette, 1994; McCallum et al., 1998) have been explored to build text categorization systems. Most of these studies build classifiers without regard of the hierarchical structure of the indexing vocabulary. Only recently some authors (Koller and Sahami, 1997; Ng et al., 1997; McCallum et al. 1998; Mladènic,

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

1998) have started to explore and use the hierarchical structure of the indexing vocabulary.

Our hypothesis is that the hierarchical structure of a vocabulary system could be used to guide the construction and training of machine learning methods that will significantly improve the performance of a text categorization system. In particular, we have developed a hierarchical neural network model inspired by the hierarchical mixture of experts model proposed by Jordan and Jacobs (1993). Section 3 will describe in general the Hierarchical Mixture of Experts model and in detail our model.

Artificial Neural Networks

Artificial Neural Networks (ANN) are machine learning methods that provide a robust approach to approximating real-valued, discrete-valued, and vector-valued functions. ANN are inspired by the observation of biological organisms and their large interconnected webs of neurons. ANN are a rough approximation of this because they are built out of small units (neurons) that are interconnected to form a web. Each neuron receives one or more real-valued inputs and produces a single real-valued output, which may become the input of another unit. Despite this similarity, there are many complexities in the biological systems that are not captured by the ANN models. The main goal of many researchers in the machine learning community, which is also ours, is to obtain effective algorithms rather than simulate the behavior of biological organisms.

A perceptron is a type of neural network that takes a vector of real-valued inputs, computes a linear combination of these inputs, and produces a single output. The perceptron is a device that decides whether an input pattern belongs to one of two classes. Given the input vector $(x_1,\ldots,x_n)$, the output of the perceptron is computed as a linear function of the form:

$$O(x_1,\ldots,x_n)=\begin{cases}1 & \text{if } \sum_{i=1}^{n} w_i x_i + \theta > 0\\ 0 & \text{otherwise}\end{cases}$$

where each w_i is a weight that determines the contribution of the corresponding input value x_i, and θ is a threshold that the weighted combination of inputs must surpass to set the output to 1. The learning process in a perceptron involves choosing the best values of w_i and θ based on the set of training examples.

Geometrically speaking, in two dimensions, the formula of the perceptron represents a line. Any point above the line makes the output of the perceptron 1. Perceptrons can represent many primitive boolean functions but they have the limitation that they can learn only linearly separable problems. In Figure 1 two classes (+ and -) are represented in a two dimensional space. Figure 1(a) shows an example of linearly separable categories, while Figure 1(b) shows an example of non-linearly separable

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

categories (observe that there is no way to separate the two categories using a single line).

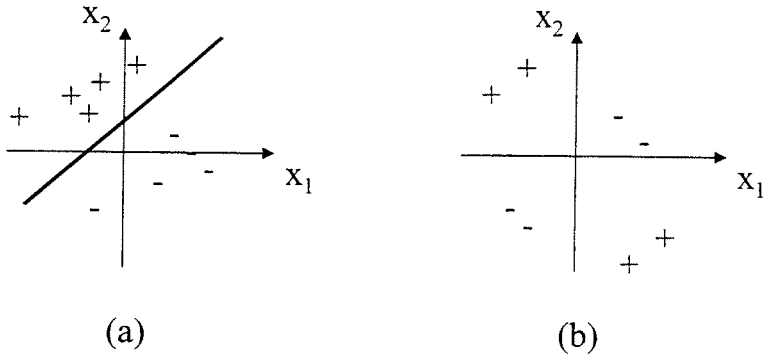

Figure 1. (a) Linearly separable categories (b) Non linearly separable categories

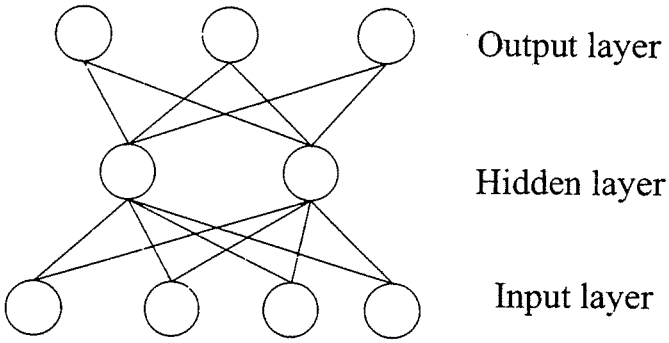

Figure 2. Multi-layered neural network

Backpropagation Networks

In contrast to the perceptron model presented previously, multi-layered networks have its neurons organized by layers and each neuron is fully connected to the neurons in the next layer (see Figure 2). This configuration allows multi-layered networks to be able to approximate almost any vector-valued function.

Backpropagation networks are multi-layered networks in which the weights are learned using a gradient descent algorithm. Given a network with a fixed set of units and interconnections, the backpropagation algorithm employs a gradient descent strategy to find the set of weights that minimizes the squared error between the network output value and the correct values. The reader is referred to chapter 4 of (Mitchell, 1998) for more details about ANN and the backpropagation algorithm.

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

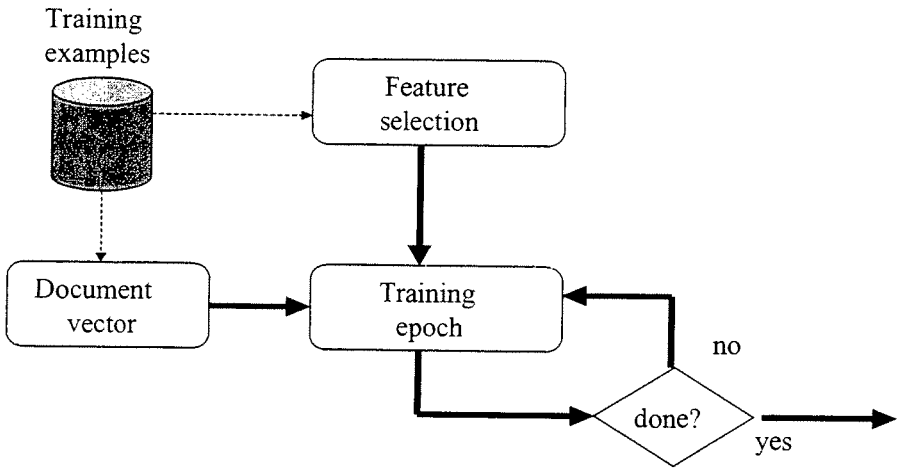

Figure 3. Training procedure for Text Categorization using a machine learning algorithms

Machine Learning and Text Categorization

So far we have discussed the generalities of machine learning and in particular some details of neural networks. This section will present the general procedure for training a machine learning algorithm for text categorization. As mentioned before, text categorization can be characterized as a supervised learning problem. We have a set of examples (abstracts or full text documents) that have been correctly categorized (usually by human indexers). Given this set of training examples, we can train a machine learning method. Figure 3 shows the general procedure for training a machine learning algorithm to perform text categorization.

Document Representation:
At this point we assume that the documents are electronically available (title and abstract, or full text). We use the vector space model (Salton, 1983) to represent a document. First, all the words in the document are tokenized, filtered using a stop list (words such as articles, prepositions, common verbs, etc are discarded), and stemmed using Porter's stemmer (Porter, 1980). Unique stems with its corresponding document frequency are kept. Each document is then represented by a vector:

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

$$D = (x_1, ..., x_m)$$
$$x_i = tf * idf$$
$$idf = \log \frac{N}{n_i}$$

where m is the total number of unique stems (terms) in the training collection, tf is the term frequency within the document, idf is the inverse document frequency, N is the total number documents in the training set, and n_i is the number of documents in the training set that contain the term i. The vector space model allows us to represent a document as a real-valued vector that can be presented as an input to a neural network, or to any other machine learning algorithm.

Feature Selection:

Despite using stop list and stemming, which are techniques that reduce the number of terms, the size of the document vector is usually too large to be useful to train a machine learning algorithm. Performance and training time of many machine learning algorithms are closely related to the quality of the features used to represent the problem. In previous work (Ruiz and Srinivasan, 1998), we used a frequency-based method to reduce the number of terms. The number of terms or features, is an important factor that affects the convergence and training time of most machine learning algorithms. For this reason it is important to reduce the set of terms to an optimal subset that achieve the best performance.

Two approaches for feature selection have been presented in the literature: the filter approach, and the wrapper approach (Liu & Motoda, 1998). The wrapper approach attempts to identify the best feature subset to use with a particular algorithm. For example, for a neural network the wrapper approach selects an initial subset and measures the performance of the network; then it generates an "improved set of features" and measures the performance of the network using this set. This process is repeated until it reaches a termination condition (either the improvement is below a predetermined value or the process has been repeated for a predefined number of iterations). The final set of features is then selected as the "best set". The filter approach, which is more commonly used, attempts to assess the merits of the feature set from the data alone independent of the particular learning algorithm. The filtering approach selects a set of features using a ranking criterion, based on the training data.

Once the feature set for the training set has been identified, the training process takes place by presenting each example (represented by its set of features) and letting the algorithm adjust its internal representation of the knowledge contained in the training set. After a pass of the whole training set, which is called an epoch, the algorithm

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

checks whether it has reached its learning goal. Some algorithms such as Bayesian learning algorithms need only a single epoch; others such as neural networks need multiple epochs to convert.

The trained classifier is now ready to be used for categorizing a new document. The classifier is typically tested on a set of documents that are distinct from the training set. Since we know the correct classification for this test set, we can determine the effectiveness of the classifier.

Hierarchical Model

Up to this point we have shown a general procedure to train and test a machine learning algorithm for text categorization. This procedure does not take into account the hierarchical structure of the indexing vocabulary. Vocabularies such as MeSH have associated relations that organize them in a hierarchical structure, i.e. using a parent child relation or a narrower term relation. These relations are built in the vocabulary to facilitate its organization and to help indexers. They usually reflect conceptual relationships in the domain. Except for few works most researchers in automatic text categorization have ignored these relations. We believe that since the arrangement of terms in a hierarchical tree reflects the conceptual structure of the domain, machine learning algorithms could take advantage of it and improve their performance.

Indexing a document is a task wherein multiple categories are assigned to a single document. Although human indexers are effective in this, it is quite challenging for a machine learning algorithm. Some algorithms even make simplifying assumptions that the categorization task is binary and that a document cannot belong to more than one category. For example, the naive Bayesian learning approach assumes that a document belongs to a single category. This problem can be solved by building a single classifier for each category, in such a way that the learning algorithm learns to recognize whether or not a particular term (category) should be assigned to a document. This transforms a multiple category assignment problem into a multiple binary decision problem.

We extend machine learning to consider the hierarchical structure of the indexing vocabulary by using a method inspired by the Hierarchical Mixture of Experts (HME) model (Jordan & Jacobs, 1993). HME is based on the divide-and-conquer principle. The main idea is to solve the problem by dividing it into smaller problems that are easier to solve, and then combine the solutions of the small problems to obtain the general solution. This principle has been successfully used to create algorithms that are elegant, and efficient. The HME model approaches the solution of a classification problem by dividing the input space into a nested sequence of regions and then

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

training smaller classifiers that are specialized in classifying a reduced domain. The HME model has two basic components: gating networks, and expert networks. These components are organized as a tree structure where the internal nodes are the gates, and the leaf nodes are the experts. The gating networks represent high level concepts. As their name implies, each gating network works as a gate to access nodes in the lower levels of the hierarchy. The expert networks, sited in the terminal nodes of the hierarchy, are specialized in recognizing documents corresponding to specific categories. The original HME model proposed by Jordan and Jacobs (1993) uses expectation maximization to train the gates, and linear perceptrons as experts. It works in a bottom up fashion. Initially all experts are activated and their output is combined by the parent gate. The output of the gate is then passed to the next upper level until it reaches the top level that will report the global answer.

Figure 4, shows a schematic view of our model for a two level hierarchy. We use the same division of the space, as the HME model, using experts and gates but our training procedure is different since we don't use expectation maximization. Another difference is that in our model each gate receives an input x, which is the document vector, and if the document contains the concept represented by that node the output of the network is set to 1 (true), else 0 (false). All the networks connected to an "open gate" (a gate whose output value is 1) are activated and thus the classification proceeds in a top down fashion. If a document reaches an expert node via the high level gates, then the expert decides whether or not the category that the expert "knows" should be assigned to the document.

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

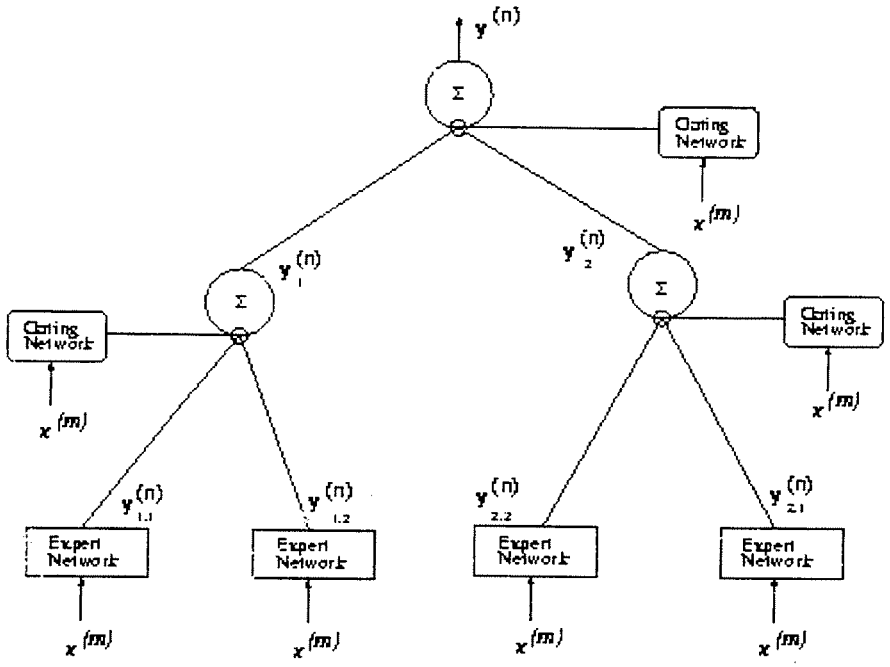

Figure 4. Hierarchical model

The categorization task starts at the root node and the gate decides whether the most general concept is present in the document. If this is true, then all the second level nodes are activated and the process repeats until a leaf node is reached. Observe that only the experts connected to gates that have value 1 are activated.

Implementation of the Hierarchical Model

For this study we use the Medical Subject Headings (MeSH) subset of the Unified Medical Language System (UMLS) Metathesaurus as our indexing vocabulary, and the parent child relationship as defining the hierarchy of concepts. The UMLS Metathesaurus is made up of concepts. Each concept has one or more strings associated, of which one is the preferred string (usually the MeSH term). The UMLS Metathesaurus offers hierarchical relationships in which more general concepts are parents of narrower or more specific concepts (See Figure 5). In our model the gates represent the general concepts, while the experts represent specific categories (or strings). For the purpose of comparing results with other studies we will show the results obtained using only the subtree of Heart Diseases, but the method is general and can be applied to the whole set or any subset of UMLS. We use backpropagation neural networks for both experts and gates. The experts are trained using a selected

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

subset of representative examples, which will be discussed in more detail later, to recognize the presence or absence of a particular category in a document. The gates are trained to recognize whether or not any of the categories of the descendants is present in the document. Observe that this definition implies that for each category that corresponds to a high level node there are a gate network and an expert network. For example, for the category "Heart Diseases", which is the root of our tree, there is a gate network that learns to recognize the general concept (all the documents that are about any of its descendants), and an expert network (that learns how to assign this specific category "Heart Diseases").

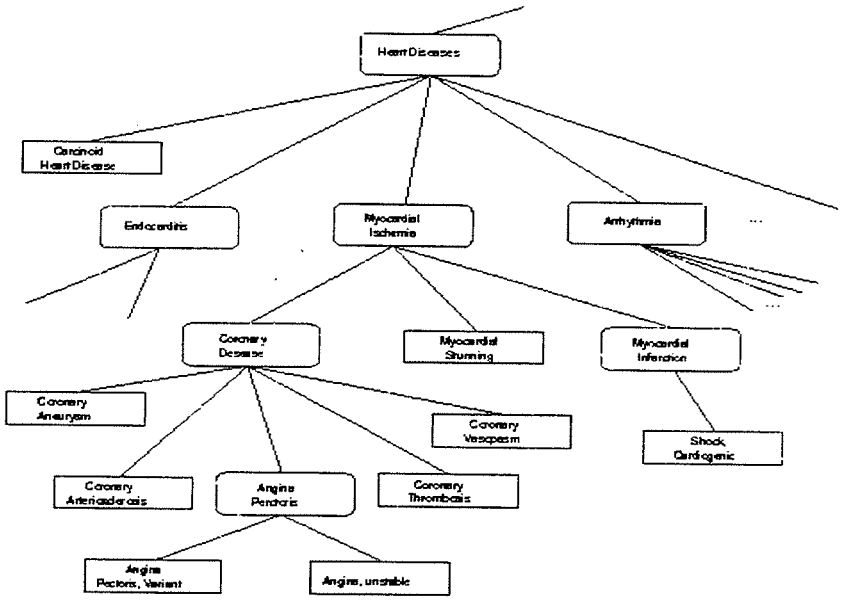

Figure 5. UMLS hierarchical structure for the "Heart Diseases" subtree

The backpropagation networks that we use have three layers. The input layer consists of a set of "n" features selected for each expert (or gate), the middle layer has "2n" hidden nodes, and the output layer is a single node. Given the best set of features and a training set, the backpropagation network learns to assign the category (or concept in the case of gates).

Feature selection:

As discussed before, feature selection is a critical step for automatic text categorization. Neural networks in particular require reduction of the feature set because the performance of the network and the cost of classification are sensitive to the size and quality of the input features used to train the network (Yang & Honovar,

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

1998). For this paper we will report our results using a filter approach based on a measure called Correlation Coefficient (Ng, Goh, & Low, 1997). The measure relates the occurrence of a term (in our case a stemmed word) in the relevant and the non-relevant documents for each category. The formula is as follow:

$$C(w) = \frac{(N_{r+}N_{n-} - N_{r-}N_{n+})\sqrt{N}}{\sqrt{(N_{r+} + N_{r-})(N_{n+} + N_{n-})(N_{r+} + N_{n+})(N_{r-} + N_{n-})}}$$

where N_{r+}(N_{n+}) is the number of relevant (non-relevant) texts in which the term *w* occurs, and N_{r-}(N_{n-}) is the number of relevant(non-relevant) texts in which the term *w* does not occur. This measure is derived from the χ^2 measure presented by Schütze *et al.* (1995), where $C^2 = \chi^2$. The correlation coefficient can be interpreted as a "one-side" χ^2 measure. This method promotes features that have high frequency in the relevant documents but are rare in the non-relevant documents. When features are ranked by this method, the positive values correspond to features that indicate presence of the category while the negative values indicate absence of the category. According to empirical results from Ng, Goh & Low (1997) using the features from relevant documents that are indicative of membership to the category gives better results than using features that are indicative of non-membership. We tested this hypothesis for a few categories and our results also support it.

Training set selection

When working with large collections we have a considerable number of categorization examples. Theoretically the availability of a large number of examples seems to be beneficial for training a machine learning algorithm. However, when the number of possible categories is also considerably large (such as the UMLS Metathesaurus with 350,000 concepts), we face a situation in which each category is assigned to a small number of documents. This creates a situation in which each category has a relatively small number of positive examples and an overwhelming number of negative examples (every thing that has not been labeled with this category). When a machine learning algorithm is trained with such an unbalanced training set it will learn that the safest decision is not to assign the category because this is correct most of the time. The overwhelming number of negative examples hides the assignment function. To overcome this problem, an appropriate training set must be selected. We call this training subset the "category zone". This concept is inspired by the query zone, which was proposed by Singhal, Mitra & Buckley (1997). They describe a method called "query zoning" which is based on the observation that in a large collection a query will have a set of documents that constitute its domain. Non-relevant documents that are outside the domain are easy to identify, but it is more difficult to differentiate between relevant and non-relevant documents in the query domain. We observe that in text categorization, each category also has its own domain and that training a machine learning algorithm with examples in this domain could improve the effectiveness of a specialized classifier. The problem is that the

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

category zone is not explicitly known. We use the following procedure to obtain an approximation of the category zone:

1. Take all positive examples and compute their centroid vector. Positive examples are those documents that have been assigned the category.
2. Using this centroid as a query, perform retrieval and obtain the top 10,000 documents. This subset will contain most of the positive examples and many negative examples that are "closely related" to the domain of the category.
3. Obtain the category zone by adding any unretrieved positive examples to the set obtained in the previous step.

The use of this category zone as the training set for a specific category will speed up the training process and will also improve the effectiveness of the classifier.

Experimental Setting

Our experiments use the OHSUMED collection (Hersh *et al.* 1994) which is a subset of the MEDLINE collection that contains 348,543 records from 1987 to 1991. Each record from this collection has several fields. We use the title, and abstract fields. In the training set we use the MeSH field which represents the manual indexing performed by professional indexers at the National Library of Medicine (NLM). From this collection we select the subset of 233,455 records that have title, abstract, and MeSH terms. We select the first four years for training (183,229 records) and the records from year 1991 (50,216 records) for testing. The subtree of Heart Diseases, which is the one we will explore in this study, contains 119 categories. These experimental settings are the same used by Lewis *et al.* (1996) and will allow us to compare our results with their method. Lewis *et al.* (1996) focussed their study on those categories that have at least 75 examples in the training collection obtaining a subset of 49 categories. In our previous work (Ruiz & Srinivasan, 1998) where we studied the minimum number of positive examples needed to properly train a neural network we found that we should have at least 20 positive examples in the training set. There are 71 categories in the "Heart Diseases" subtree that have at least 20 examples. For purpose of comparison we will report only the subset of 49 categories that have been used by Lewis et al. (1996).

The 119 "Heart Diseases" categories form a 5 level hierarchy where the first level corresponds to the root node and the fifth level has only leaf nodes. The number of gates in each level starting from the root is 1, 11, 9, and 3. Not all branches have experts with the required number of examples, as a consequence the hierarchy for the

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

49 categories that have at least 75 examples is reduced to three levels with 1, 6, and 4 gates respectively (see Figure 6).

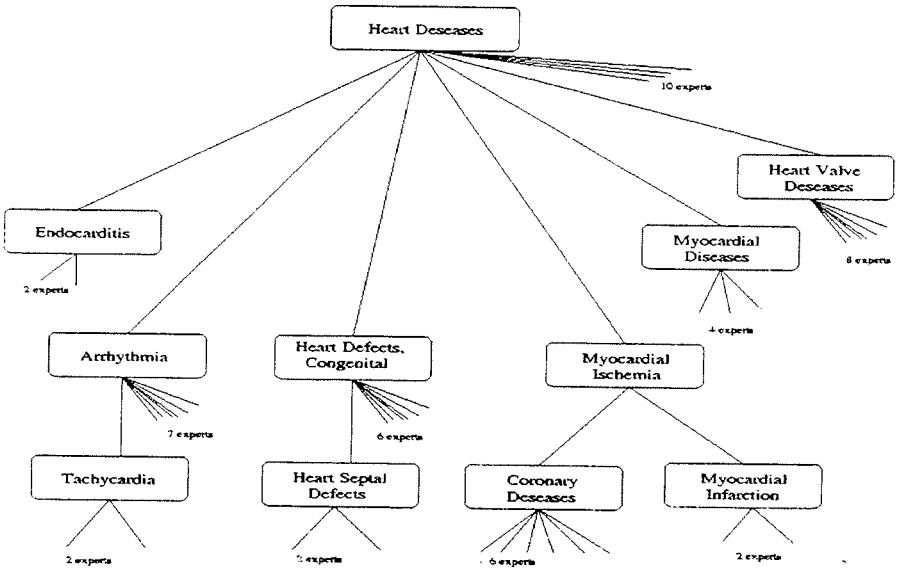

Figure 6. Hierarchy for the 49 categories that have at least 75 examples.

Evaluation Measurements

Several measurements have been used in previous studies to evaluate performance of text categorization systems. The contingency table that relates the system assignments and the human indexer's assignment (see table 1). Several measures have been defined in the artificial intelligence and information retrieval communities based on this contingency table (see table 2).

F_β and Break-Even Point (BEP) are two common measures in text categorization that combine recall and precision. BEP was proposed by Lewis (1992) and is defined as the point at which recall equals precision. Van Rijsbergen's F_β measure (van Rijsbergen, 1979) combines recall and precision into a single score according to the following formula:

$$F_\beta = \frac{(\beta^2 + 1)P \times R}{\beta^2 P + R}$$

The value $\beta \in [0,\infty)$ defines the relative importance of recall and precision; F_0 is the same as precision, F_∞ is the same as recall. Intermediate values between 0 and ∞ are different weights assigned to recall and precision. The most common values assigned

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

to β are: 0.5 (recall half as important as precision), 1.0 (recall and precision equally important), 2.0 (recall twice as important as precision).

None of the measures is perfect or appropriate for every problem. For example, recall (sensitivity), and precision (predictive value (+)) if used alone might show deceiving results, i.e. a system that assigns the category to every document will show perfect recall (1.0). Accuracy, which is a popular measure in machine learning, is appropriate when the number of positive examples and the number of negative examples are balanced, but in extreme conditions it might also be deceiving. If the number of negative examples is much larger than the number of positive examples, a system that assigns no documents to the category will obtain an almost perfect accuracy (close to 1).

BEP also shows some problems. Usually the value of BEP has to be interpolated. If the values of recall and precision are too far apart then BEP will show values that are not achievable by the system. Also the point where recall equals precision is not necessarily desirable from the user's perspective.

Van Rijsbergen's F measure is the best suited measure, but still has the drawback that it might be difficult for the user to define the relative importance of recall and precision. We will report F_1 values because it will allows us to compare results with other researchers that use the same data set (Lewis *et al.*, 1996). When working with several categories we report a macro-averaged value.

	Indexer assigns the category (class positive C+)	Indexer doesn't assign the category (Class negative C-)
System assigns the category (Assigned positive R+)	a (True Positive)	b (False Positive)
System doesn't assign the category (Assigned negative R-)	c (False Negative)	d (True Negative)

Table 1. Contingency table for binary decision classifications.

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

Table 2. Efficiency Measures for Binary Classification defines in Information Retrieval (IR) and Artificial Intelligence (AI).

IR	AI	Formula
Recall (R)	Sensitivity	$\frac{a}{a+c}$
Precision (P)	Predictive_value(+)	$\frac{a}{a+b}$
Fallout	Predictive_value (-)	$\frac{b}{b+d}$
	Accuracy	$\frac{a+c}{a+b+c+d}$
Error_rate		$\frac{b+c}{a+b+c+d}$

Experiments

We want to address two questions with our experiments: (1) Does the hierarchical classifier based on our hierarchical model improve performance when compared to a non-hierarchical classifier? (2) How does our hierarchical method compare with other text categorization approaches? With these questions in mind we present a series of experiments using the OHSUMED collection.

Baselines

We will define two baselines to answer our research questions. The first base line is represented by a classical Rocchio classifier, which is described next. This will allow us to address the issue of comparing our hierarchical model against another approach. The second base line is a "flat" neural network to address our first research question. We also will cite works published with the same collection to answer the second research question.

Rocchio Classifier

Rocchio's algorithm was developed in the mid 60's to improve queries using relevance feedback. It has been proven to be one of the most successful feed forward algorithms. Rocchio (1971) showed that the optimal query vector is the difference vector of the centroid vectors for the relevant and the non-relevant documents. Salton & Buckley (1990) included the original query to preserve the focus of the query, and added coefficients to control the contribution of each component. The mathematical expression of this version is:

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

$$Q_{new} = \alpha Q_{orig} + \beta \frac{1}{R} \sum_{d \in rel} d - \gamma \frac{1}{N-R} \sum_{d \notin rel} d$$

where d is the weighted document vector, R=|*Rel*| is the number of relevant documents, and N is the total number of documents. Any negative component of the final vector Q_{new} is set to zero. Several techniques have been proposed to improve the effectiveness of Rocchio's method: better weighting schemes (Singhal, Buckley & Mitra, 1996), query zoning (Singhal, Mitra & Buckley, 1997), and dynamic feedback optimization (Buckley & Salton, 1995).

We use the first two of these techniques to get an "optimal" Rocchio classifier for text categorization. We use the same approach presented by Schapire, Singer & Singhal (1998) to build their baseline Rocchio text filter. Since there is no initial query in text categorization, the first term of Rocchio's formula is set to zero. We use the centroid of the positive examples of the category to build the category zone. The following steps summarize our algorithm for a given category:

1. Create the centroid vector using Rocchio's formula: as a preprocessing step discard stop words, stem the rest and weight them using the *atn*[1] weighting scheme. Select the top ranked 100 terms.
2. Build the category zone: Use the centroid as initial query and retrieve the top 10,000 documents from the training set. Create the category zone by adding any unretrieved relevant documents.
3. Using the category zone, build a classifier by using Rocchio's formula with α=0, β=8, γ=8. Observe that this will create a vector that is the difference between the centroid of the positive examples and the centroid of the negative examples in the query zone. Select the top 100 terms with their respective weights. This is the classifier vector.
4. Using the classifier vector obtained in the previous step, rank the full training collection according to the similarity with the classifier vector. A threshold (τ) on the similarity value that maximizes the F1 measure (described in the evaluation section) is selected. This is our optimal Rocchio classifier.
5. Evaluate the classifier on the test set: use the Rocchio's classifier to rank the test collection and assign the category to all those documents above the threshold τ.

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

Hierarchical Classifier

The hierarchical classifier is represented in Figure 6. In order to train each expert network we first build its ``category zone" using steps 1 and 2 of the Rocchio classifier[2] as explained before. Feature selection is then applied on each category zone to extract the ``best" set of features for each category. As described in section 4.2 we use the Correlation Coefficient for this. For each expert network a backpropagation neural network is trained using the corresponding category zone, and the selected set of features.

Similarly, each gating network is also a backpropagation network, however the training subset is the combined category zones of its descendants in the hierarchy. Feature selection is performed on the previously mentioned subset to obtain the best set of features for the corresponding gate.

Experts and gates are trained independently using the following parameters: learning rate = 0.5, error tolerance = 0.01, maximum number of cycles = 100. The training of each network takes about 30 minutes for an expert, and 90 minutes for a gating network using a HP-700 workstation. Using 12 workstations and a dynamic scheduling program specifically designed for this task we train the 71 experts (with at least 20 positive examples) and the 15 gating networks in about 6 to 7 hours. We tried several values for the number of input features of the networks (5, 10, 25, 50, 100, and 150 features). The best result was obtained using 25 input features. As mentioned before, the architecture of the backpropagation network for 25 inputs has 50 hidden units and a single output unit. The inputs are the *tf*×*idf* weights of each selected feature (term) in the document, where *tf* is the frequency of the term in the document, and *idf* is the inverse document frequency.

Once experts and gates have been trained individually, we assemble them according to the hierarchical structure. Since the output of our networks is a real value between 0 and 1, we need to transform it to a binary decision. This step is called defuzzification. We do this by selecting a threshold that optimizes the F_1 value for each category. We use the complete training set to select the optimal threshold. Since we are working with a modular hierarchical structure we have several choices to perform defuzzification. Our approach is to make a binary decision in each of the gates and then optimize the threshold on the experts using only those examples that reach the leaf nodes.

Observe that computing the optimal threshold for binary decision in the gates and then the thresholds for the binary decision in the experts implies a multidimensional optimization problem. We decided to optimize the gates by grouping them into levels and finding the value of the threshold at each level that maximizes the average F_1 value for all the experts. Each expert's threshold is optimized to maximize the F_1 value of the examples in the training set that reach the expert.

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

Once the optimal thresholds are set in the training set, the test set is processed. The values reported in Section 8 are the F_1 values obtained in the test set.

Flat Neural Network Classifier

To assess the advantage gained by exploiting the hierarchical structure of the classification scheme, we build a flat neural network classifier. We tried building a single network that learns all the 49 categories that have at least 75 examples but after a long training time only a few iterations on the training set were completed and the network's performance was extremely low. We then decided to build a flat modular classifier that will combine the outputs of the 49 individual expert networks using a simple mixture model. In this mixture model the experts are trained independently and their outputs are combined to get a single output vector $y^{(n)}$. The defuzzification step is performed by optimizing the F_1 value of each expert for the entire training set. The performance of this model is considerably higher than the performance of the single network. These are the values that we report in the next section for the flat neural network classifier.

Results and Analysis

Table 3 shows the performance of the hierarchical model, against the Rocchio classifier, and the flat neural networks for the 49 categories with at least 75 examples. The flat classifier performs significantly better than Rocchio's classifier (47.9% of improvement in macro-averaged F_1). In terms of class by class comparison the flat neural network performs better than Rocchio's classifier in 42 of the 49 categories. The addition of the hierarchical structure improves the results significantly (65.7% of improvement with respect to Rocchio classifier, 12% of improvement when compared to flat classifiers). The hierarchical classifier performs better than the flat classifier in 41 of the 49 categories.

When compared to other reported results, the HME model achieved the same macro-average F_1 (0.50) as the *exponentiated gradient* (EG) algorithm reported by Lewis *et al.* (1996), but lower than that of the Widrow-Hoff Algorithm (0.55) reported in the same work. We would like to compare our results with the *generalized instance set* method proposed by Lam and Ho (1998). However they used a different subset for training and testing (They use year 1991 and take the first 33,478 documents for training and the last 16,738 documents for testing). Hence we need to repeat all the experiments using the same split in order to do a fair comparison.

A detailed analysis of the behavior of the HME with respect to the flat neural network shows two facts. The defuzzification threshold for the hierarchical classifier is less than or equal to the threshold for the flat classifier in 45 of the 49 categories. This is an expected result because the intermediate layers perform a prefiltering of "bad

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

candidate texts"' hence the experts receive a smaller number of examples. Since the optimization process sets these thresholds to maximize the F1 values in the training set, when the "bad matches"' have been filtered the algorithm will be able to set a lower threshold that increases the number of true positive without incrementing significantly the number of false positives. Even if the threshold is not changed we can expect a higher performance if the hierarchy is good at filtering false positives. Table 4 shows the number of documents that pass each gate in the test set. The number of documents in the test collection is 50,216. The root node filters most of the documents and only 8,792 reach the nodes connected to the root.

	Rocchio (baseline)	Flat neural network (% of improvement)	Hierarchical neural network (% of improvement)
Macro-Averaged F_1	0.3007	0.4449 (47.9%)	0.4984 (65.7%)
Variance	0.0194	0.0320	0.0254
Median	0.2791	0.4642	0.5167

Table 3. Comparison of average performance of the 49 categories for the three classifiers.

Level 1 (threshold=0.01)	Level 2 (threshold=0.005)	Level 3 (threshold=0.01)	# of doc >= threshold
(root) Heart Diseases			8792
	1.1 Arrhythmia		1978
		1.1.1 Tachycardia	562
	1.2 Endocarditis		289
	1.3 Heart Defects Congenial		1238
		1.3.2 Heart Septal Defects	188
	1.4 Heart Valve Diseases		1130
	1.5 Myocardial Diseases		3401
	1.6 Myocardial Ischemia		7052
		1.6.3 Coronary Diseases	5357
		1.6.4 Myocardial Infarction	5893

Table 4. Number of documents that pass each gate in the test set.

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

Conclusions

This paper presents a machine learning method for text categorization that takes advantage of the hierarchical structure of the indexing vocabulary. The results indicate that the use of the hierarchical structure of the indexing vocabulary improves performance significantly.

Our method is scalable to large test collections and vocabularies because it divides the problem into smaller tasks allowing a modular approach.

The results obtained by our hierarchical model are comparable with those reported previously in the literature, and significantly better than the classical Rocchio classifier and our flat neural network classifier.

Future work

We plan to study this approach using other machine learning methods to verify whether the improvement obtained by including the hierarchical model will also be obtained using other categorization methods.

We also will explore alternative methods for category zone approximation. It would also be interesting to explore alternative methods, such as expectation maximization, for training the hierarchical classifier.

Finally an aspect that still remains to be tested is the effectiveness of our method in classifying full text documents. Given the effectiveness and speed of our classifiers this could be an important tool for real-time indexing of web pages in the medical domain.

Notes

1. SMART notation for weighting scheme consists of three letters that indicate the values for the term frequency, collection frequency, and normalization respectively. The first letter *a* is the augmented normalized term frequency (0.5+0.5*tf/maxtf) where tf is the raw term frequency, the second component *t* indicates that the inverse document frequency is used as a collection frequency factor, the third component indicates that no normalization is performed.

2. We use the SMART system for this task.

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

References

Apte, C. and Damerau, F. (1994) Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 3(12), 233-251.

Buckley, C. and Salton, G. (1995). "Optimization of relevance feedback weights". In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 351—357). New York, NY: ACM press.

Hersh, W., Buckley, C., Leone, T.J. and Hickam, D. (1994). "OHSUMED: An interactive retrieval evaluation and new large test collection for research". In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 192-201). New York, NY: ACM press.

Humphrey, S. and Kapoor, A. (1988). *The MedIndEx system: Research on interactive knowledge-based indexing of medical literature.* Washington, D.C.: National Library of Medicine.

Joachims, T. (1997). *Text categorization with support vector machines: Learning with many relevant features.* Technical Report LS-8 Report 23, University of Dortmund.

Jordan, M. I. and Jacobs, R.A. (1993). *Hierarchical Mixture of experts and the EM algorithm.* Technical Report, Massachusetts Institute of Technology.

Koller, D. and Sahami, M. (1998). "Hierarchically classifying documents using very few words". In *ICML-1997: Proceedings of the 14th International Conference on Machine Learning*, (pp 170-178). New York, NY: AAAI press.

Lam, W. and Ho, C. Y. (1998). "Using a generalized instance set for automatic text categorization". In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 81-89). New York, NY: ACM press.

Lewis, D. D. (1992). "An evaluation of phrasal and clustered representations on a text categorization task". In *Proceedings of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 37—50). New York, NY: ACM press.

Lewis, D. and Ringuette, M. (1994). "Comparison of two learning algorithms for text categorization". In *Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94).*

Lewis, D. D., Schapire, R. E., Callan, J.P., and Papka, R. (1996). "Training algorithms for linear classifiers". In *Proceedings of he 19th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 298-303). New York, NY: ACM press.

Liu, H. and Motoda, H. (1998). "Less is more". In H. Liu and H. Motoda (Eds.) *Feature extraction, construction and selection: A data mining perspective* (chapter 1, pp. 3-12). New York: Kluwer Academic Publisher.

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of research and Development* 2 (April): 159-65.

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

McCallum, A, Rosenfeld, R., Mitchell, T. and Ng, Y. (1998). "Improving text classification by shrinkage in a hierarchy of classes". In *Proceedings of the 15th International Conference on Machine Learning*. New York, NY: AAAI.

Mitcell, T. (1998). *Machine Learning*. Boston, MA, McGraw-Hill.

Mladènic, D. (1998). *Machine learning on non-homogeneous distributed text data*. PhD. Thesis, University of Ljibljana, Faculty of Computer and Information Science, Ljubljana, Slovenia.

Moulinier, I. (1997). *Is learning bias an issue on text categorization?* Technical Report, LaFORIA-LIP6, Universite Paris VI.

Ng, H. T., Goh, W. B., and Low, K. L. (1997). "Feature selection, perceptron learning, and a usability case study for text categorization". In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (pp. 67-73). New York, NY: ACM press.

Porter, M.F. (1980) An Algorithm for Suffix Stripping. *Program*, 14(3), 130-137.

Rocchio, J. J. (1971). "Relevance feedback in information retrieval". In G. Salton (Ed.) *The SMART Retrieval System: Experiments in Automatic Document Processing* (pp.313—323). Englewood Cliff, NJ, Prentice Hall.

Ruiz, M. E. and Srinivasan, P. (1998). "Automatic text categorization using neural networks". In E. Efthimiadis (Ed.) *Advances in Classification Research Volume 8* (pp. 59-72). Melford, NJ:Information Today, Inc.

Salton, G and Buckley, C (1990) Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*. 41, 288-297.

Salton, G and McGill M.J. (1983). *Introduction to modern information retrieval*. New York, NY, McGraw-Hill.

Schapire, R. E., Singer, Y. and Singhal, A. (1998). "Boosting and rocchio applied to text filtering". In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 215-223). New York, NY: ACM press.

Singhal, A., Buckley, C. and Mitra, M. (1996). Pivoted document length normalization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp.21-29). New York,NY: ACM press.

Singhal, A., Mitra, M. and Buckley, C. (1997). "Learning routing queries in a query zone". *In Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 25—32). New York, NY: ACM press.

van Rijsbergen, C. J. (1979). *Information Retrieval*. Second edition, London, Butterworths.

Weiner, E., Pedersen, J. O. and Weigend A. S. (1995). "A neural network approach to topic spotting". In *Proceedings of SDAIR'95* (pp. 317-—332).

Yang, J. and Honovar, V. (1998). "Feature subset selection using a genetic algorithm". In H. Liu and H. Motoda (Eds.) *Feature Extraction, Construction*

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

and Selection: A Data Mining Perspective (pp. 117-136). New York, NY: Kluwer Academic Publishers.

Yang, Y. (1999) An evaluation of statistical approaches to text categorization. *Information Retrieval* , 1(1), 69-90.

ISSN: 2324-9773

Ruiz, M. E., & Srinivasan, P. (1999). Combining machine learning and hierarchical indexing structures for text categorization. Proceedings of the 10th ASIS SIG/CR Classification Research Workshop, 125-148. doi: 10.7152/acro.v10i1.12485

ISSN: 2324-9773