On Using Partial Supervision for Text Categorization

Charu C. Aggarwal, Stephen C. Gates, and Philip S. Yu, *Fellow*, *IEEE*

Abstract—In this paper, we discuss the merits of building text categorization systems by using supervised clustering techniques. Traditional approaches for document classification on a predefined set of classes are often unable to provide sufficient accuracy because of the difficulty of fitting a manually categorized collection of documents in a given classification model. This is especially the case for heterogeneous collections of Web documents which have varying styles, vocabulary, and authorship. Hence, this paper investigates the use of clustering in order to create the set of categories and its use for classification of documents. Completely unsupervised clustering has the disadvantage that it has difficulty in isolating sufficiently fine-grained classes of documents relating to a coherent subject matter. In this paper, we use the information from a preexisting taxonomy in order to supervise the creation of a set of related clusters, though with some freedom in defining and creating the classes. We show that the advantage of using partially supervised clustering is that it is possible to have some control over the range of subjects that one would like the categorization system to address, but with a precise mathematical definition of how each category is defined. An extremely effective way then to categorize documents is to use this a priori knowledge of the definition of each category. We also discuss a new technique to help the classifier distinguish better among closely related clusters.

Index Terms—Clustering, categorization, supervision, taxonomy, text.

1 Introduction

The amount of online text data has grown greatly in recent years because of the increase in popularity of the World Wide Web. As a result, there is a need to provide effective content-based retrieval, search, and filtering for these huge and unstructured online repositories. In this paper, we consider the problem of automated text categorization, in which we desire to find the closest matching subjects for a given test document. Such a system has several applications, such as the construction of recommendation systems or providing the ability to categorize very large libraries of text collections on the Web in an automated way. We assume that a preexisting sample of documents with the associated classes is available in order to provide the supervision to the categorization system.

Several text classifiers have recently been proposed, such as those discussed in [4], [5], [6], [13], [14]. These classifiers have shown excellent results on document collections such as the *Reuters* data set or the US patent database [5] and to a somewhat lesser extent on the Web with the *Yahoo*! taxonomy. Categorization of Web documents has proven to be especially difficult because of the widely varying style, authorship, and vocabulary in different documents.

Most of the above-mentioned categorizations are created using manual categorizations by subject experts. The apparent inaccuracy of classification methods on large document collections is a result of the fact that a large heterogeneous collection of manually categorized documents is usually a poor fit for any given classification model. Thus, it is interesting to investigate the construction of categorization systems which relax the restriction imposed by predefined sets of classes. We study the use of clustering in order to create the categories. Once such a set of categories has been obtained, it is easy to perform the categorization by using the same distance measures as were used to perform the clustering. The initially available document taxonomy can provide sufficient supervision in creating a set of categories which can handle similar subjects as the original, but with some freedom in choosing exactly how to define and create the classes. As long as the final application of the categorization system does not restrict us to the use of a fixed set of class labels, this approach may provide considerable advantage because of the tight integration of the measures which are used for clustering and classification.

The fact that we actually know the model used to construct each partition in the clustering ensures that we can theoretically obtain a perfect accuracy on this categorization. Therefore, the quality of categorization depends completely on the quality and coherence of each cluster in the new taxonomy, rather than the accuracy of a training procedure on the original taxonomy. Thus, if the supervised clustering procedure can create a new set of classes which are qualitatively comparable to the original taxonomy (in terms of human perception and judgment), the accuracy of the overall categorization system is substantially improved.

The use of clustering for providing browsing capabilities has been espoused in earlier work by Cutting et al. [7], [8]. Other work on clustering algorithms for text data may be found in [3], [9], [11], [18], [19], [20]. These methods do not use any kind of supervision from a preexisting set of classes and are attractive for creation of a small number of clusters such as 50 or so, though the clustering rapidly degrades in quality when there is a need to find more fine-grained

• *The authors are with the IBM T.J. Watson Research Center, Yorktown Heights, NY 10598. E-mail: {charu, scgates, psyu}@us.ibm.com.*

Manuscript received 8 Sept. 1999; revised 8 Oct. 2001; accepted 22 Oct. 2002. For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 110549.

partitions. Typically, when categories are related sufficiently such that some documents can be considered to be related to both, unsupervised clustering methods are unable to create distinct sets of classes for such categories. The use of a preexisting manual categorization helps in the creation of a new set of clusters, so that we have some control over the range of subjects that we would like the categorization system to address. The resulting set of clusters may contain additional, new, or similar classes to the original taxonomy, and may be quite different in terms of the distribution of the documents among the different classes.

In this paper, we will use the $Yahoo!$ taxonomy in order to study our categorization system. This is one of many hierarchical organizations of documents which are built by manual categorizations of documents. This is also one of the larger categorizations of documents which are currently available and, hence, was a good choice for our study. Some of the studies discussed in this paper have also been investigated briefly in [2].

We mention an interesting technique discussed in [15], which shows how to augment a small number of labeled documents with a large pool of unlabeled documents in order to improve the effectiveness of text classifiers. However, the aim and scope of this work is somewhat different from our paper in that the former is mainly concerned with the practical problems of obtaining large sets of labeled training documents. Our paper is, however, concerned with the issue of using partially supervised clustering for categorization.

This paper is organized as follows: In Section 2, we will discuss the details of the cluster generation and categorization. In Section 3, we provide an intuitive discussion of the observed behavior of the engine. A conclusion and summary is provided in Section 4.

1.1 Contributions of this Paper

The contributions of this paper are as follows:

1. We discuss the merits of using supervised clustering for performing document categorization over the traditional approach of training on a predefined set of categories. We propose an algorithm for supervised text clustering.
2. The importance of creating a classification process which is able to distinguish between closely related subjects in the taxonomy has been pointed out in [5]. We discuss a process for using this set of clusters in order to create a classifier which is able to distinguish between very closely related subjects in the taxonomy. The work discussed in [5] uses a hierarchical taxonomy for effectively distinguishing between closely related categories. Such classifiers are quite fast, though the accuracy can be sensitive to the quality of the hierarchical organization. Our method is able to provide high-quality categorizations on a flat set of classes without compromising on speed.

2 A Description of the Categorization System

In this section, we will provide a description of our categorization system including feature selection, clustering, and classification. We will first begin with the definitions and notations which we will need for further development of our ideas.

2.1 Some Definitions and Notations

In order to represent the documents, we used the vector space model [16]. In the vector space model, it is assumed that each document can be represented as *term vector* of the form $\overline{a} = (a_1, a_2, \dots a_n)$. Each of the terms a_i has a weight w_i associated with it, where w_i denotes the normalized frequency of the word in the vector space. A well-known normalization technique is the cosine normalization. In cosine normalization, the weight w_i of the term i is computed as follows:

$$w_i = \frac{tf_i \cdot idf_i}{\sqrt{\sum_{i=1}^{n} (tf_i \cdot idf_i)^2}}. \tag{1}$$

Here, the value of tf_i denotes the *term frequency* of a_i, whereas the value of idf_i denotes the *inverse document frequency*. The inverse document frequency is the inverse of the number of documents in which a word is present in the training data set. Thus, less weight is given to words which occur in larger number of documents, ensuring that the commonly-occuring words are not given undue importance.

The similarity between two documents may be measured by calculating the *cosine* similarity between the documents. The cosine similarity between two documents with weight vectors $U = (u_1 \dots u_n)$ and $V = (v_1 \dots v_n)$ is given by:

$$cosine(U, V) = \frac{\sum_{i=1}^{n} f(u_i) \cdot f(v_i)}{\sqrt{\sum_{i=1}^{n} f(u_i)^2} \cdot \sqrt{\sum_{i=1}^{n} f(v_i)^2}}. \tag{2}$$

Here $f(\cdot)$ is a *damping function* such as the square root or the logarithmic function. We note that the normalization used is derived from the cosine normalization technique which is often used in the text retrieval literature [16].

A *centroid* of a set of documents is defined by a concatenation of the documents in the set. Thus, a centroid of a set of documents is a metadocument which contains all the terms in that set with the appropriate term frequencies added. A *damped centroid* (or pseudocentroid) of a set of documents is defined in the same way as the centroid, except that in this case, a damping function is applied to the frequencies of the terms in each document before adding them together. The damping function ensures that the repeated presence of a word in a single document does not affect the pseudocentroid of the entire cluster excessively. Thus, the pseudocentroid is often a much more stable representation of a central point in a cluster of documents as compared to the centroid.

A projection of a document is defined by setting the term frequencies (or weights) of some of the terms in the vector representation of the document to zero. These are the terms which are said to be *projected out*. We will use the process of projection frequently in the course of the supervised clustering algorithm. Each cluster is represented by a seed vector containing only a certain maximum number of projected words. The aim in projection is to isolate a relatively small vocabulary which describes the subject matter of a cluster well, while filtering out the nonrelevant features for that class. We use an incremental process of gradually finding the best set of projected words, while simultaneously refining the clusters, so as to gradually converge to an optimum feature set for each cluster. The iterative approach of our method is somewhat similar to the K-means algorithm, although in our case, a successive merging process was also used.

2.2 Feature Selection

Our first phase was to perform the feature selection in such a way so that only the more differentiating words are used in order to perform the clustering. Note that in unsupervised clustering methods, where a preexisting taxonomy is not used, the feature selection is somewhat rudimentary in which only *stop words* (very commonly-occuring words in the English language) are removed. In this case, since more information is available, we use it in order to prune the feature set further and bias the clustering process to use words which are discriminatory with respect to the original class labels. We use a number, called the *normalized gini index* of a word, in order to calculate its importance in the clustering process.

Let there be K classes $C_1, C_2 \dots C_K$ at the lowest level in the original taxonomy. Let $f_1, f_2 \dots f_K$ be the number of occurences of that word in each of the K classes, and let $n_1 \dots n_K$ be the total word count for the documents in each of the K classes. Thus, the *fractional presence* of a word in a particular class is given by f_i/n_i. We define the *skew fraction* of a word for class i by

$$\frac{f_i/n_i}{\sum_{i=1}^{K} f_i/n_i}.$$

We shall denote this skew fraction by p_i. Note that if the word is very noisy and is very evenly distributed among the different classes, then the skew fraction for the word is likely to be approximately $1/K$ for many classes.

The normalized gini index of a word with skew fractions $p_1 \dots p_K$ is given by

$$1 - \sqrt{\sum_{i=1}^{K} p_i^2}.$$

If the word is distributed evenly across the different classes, then the gini index is $1 - 1/\sqrt{K}$. This is the maximum possible value of the gini index. On the other hand, when the word is highly correlated with particular categories and is very skewed in its distribution, then the normalized gini index is much lower.

For our feature selection phase, we calculated the normalized gini index of each word in the lexicon in order to calculate its significance to the lexicon. All those words whose gini index was higher than a predefined value were removed from contention. Thus, the removal of these words ensures the use of a much better set of features than the simple stopword removal of unsupervised clustering techniques. In subsequent phases of clustering and categorization, only the reduced feature set was used for all analysis.

2.3 Supervised Cluster Generation

The clustering algorithm uses a seed-based technique in order to create the clusters. Traditional clustering methods such as K-means have often used seed-based algorithms in order to serve as an anchor point for the creation of the clusters. In other words, seeds form an implicit representation of the cluster partitioning in which each item to be categorized is assigned to its closest seed based on some distance (or similarity) measure. In the context of information retrieval, a seed is a metadocument which can be considered as a pseudorepresentation of a central point in a given cluster. Most of the current clustering algorithms discussed in [3], [7], [8], [11] are based on finding a set of seeds in order to define the implicit partitions.

Since the focus of the algorithm is on *supervised clustering*, we started off with a set of seeds which are representative of the classes in the original taxonomy. These representative seeds are constructed by finding the damped centroids (or pseudocentroids) of the corresponding classes. This choice of starting point (and features picked) ensures the inclusion of supervision information from the old taxonomy, but the subsequent clustering process is independent of any further supervision. Providing this level of independence is critical in the construction of a much more refined set of classes, which are based purely upon content. One of the aspects of the algorithm is that it projects out some of the words in order to represent the seeds. Thus, each seed consists of a vector in which the number of words with a nonzero weight is restricted to a predefined maximum. This vector of words is indicative of the subject material which is most relevant to that cluster. The algorithm starts with a projected dimensionality of about 500 words and gradually reduces it in each iteration as the clusters get more refined, and a smaller number of words are required in order to isolate the subject of the documents in that cluster. This technique of representing clusters by using both the documents and the projected dimensions in order to represent a cluster is referred to as *projected clustering* [1], and is an effective technique for the creation of clusters for very high-dimensional data. The idea of using truncation for speeding up document clustering has been discussed in [18], though our focus for using projections is different and is designed in order to improve the quality of the clustering by iteratively refining the dimensions and clusters. Thus, the projected clustering technique merges the problem of finding the best set of documents and features for a cluster into one framework. More details on the advantages of using projected clustering for very high-dimensional data may be found in [1]. The basic framework of the clustering algorithm is illustrated in Fig. 1. The following four steps (detailed in Figs. 2, 3, 4, and 5) are applied iteratively in order to converge to the final set of clusters in the taxonomy. We assume that the set of seeds available to the algorithm at any stage is denoted by S and the documents which are being clustered by the algorithm are denoted by D.

1. **Document Assignment**: (Fig. 2). In each iteration, we assign the documents to their closest seed in S. The similarity of each document to its closest seed is calculated using the cosine measure. Thus, a new partition of the documents in D is created by the set of seeds S. After the assignment process, the old set

```
Algorithm TClus(D)
begin
  S = Initial set of seed meta-documents;
  $iteration := 0$;
  $words = Initial\_Value1$;
  $threshold = Initial\_Value2$;
  $minimum = Initial\_Value3$;
  while not(termination-criterion)
  do begin
    $(S, D) = Assign(S, D)$;
    $S = Project(S, words)$;
    $S = Merge(S, threshold)$;
    $S = Kill(S, minimum)$;
    $iteration := iteration + 1$;
    $words = words * \theta$;
    { $\theta$ is a number which is less than 1, and
      indicates the rate at which the number of
      projected dimensions in each seed reduces }
  end
end
```

Fig. 1. The clustering algorithm.

of seeds S are discarded, and the new pseudocentroid of each partition is added to S as a seed. The procedure returns the new set of seeds S after the assignment of documents to seeds. Those documents which are not close enough to any of the seeds may be permanently discarded as outliers. Thus, the document set D is pruned in conjunction with the formation of clusters, so that documents which do not fit well in any category are quickly removed.

2. **Project**: (Fig. 3). In each iteration, we project out the words with the least weight from the pseudo-centroids of the previous iteration. This ensures that only the terms which are frequently occuring within a cluster of documents are used for the assignment process. The number of terms which are projected out in each iteration is such that the number of nonzero weight terms reduces by a geometric factor in each iteration. We denote this geometric factor by θ. The use of an iterative projection technique is useful in finding the words which are most representative of the subject material of a cluster. This is because in the first few iterations, when the clusters are not too refined, a larger number of dimensions need to be retained in the projection in order to avoid premature loss of information. In later iterations, the clusters become more refined and it is possible to project down to a fewer number of words.
3. **Merge**: (Fig. 4). In each iteration, we merge all the clusters where the similarity of the seeds in the corresponding partitions is higher than a predefined value (denoted by *threshold*). The merging process is implemented using a simple single linkage method [17]. Each cluster is represented by a node in an undirected graph and an edge is added between the two nodes if the similarity of the seeds of the corresponding clusters is larger than the predefined threshold value. Each connected component in this graph is then treated as a supercluster. In other words, the documents in each connected component are assigned to a single cluster. The set of pseudo-centroids of this reduced set of clusters is returned by the procedure. Although the simple linkage process is somewhat naive, it is very fast and effective for high values of the threshold similarity.
4. **Kill**: (Fig. 5). In each iteration, we discard all those seeds from S such that the number of documents in the corresponding clusters is fewer than a predefined number. This predefined parameter is denoted by *minimum* in Fig. 1. These documents either get redistributed to other clusters or get classified as outliers in later iterations.

These procedures are applied iteratively in order to create the clusters. We note that another interesting operation which could be used in such a technique could be a split method which creates clusters in two distinct categories from one large cluster. However, the problem with implementing such a split operation is that it is likely to be unsupervised, as a result it may sometimes create incoherence with respect to the original class labels. Consequently, we restricted our operations.

In the initialization process, we started off with a projected dimensionality of 500 words and reduced the number of words by a factor of 70 percent ($\theta = 0.7$) in each iteration. When the number of projected dimensions in the seed of each cluster was fewer than 200, we terminated the clustering process. The choice of final number of words was made in such a way that the average intercluster similarity was less than 30 percent of the average similarity of documents within a cluster. This ensures that the clusters are well separated and coherent collections of documents.

2.4 Categorization Algorithm

The definition of each cluster ensures that it is possible to categorize any test document very easily by assigning it to the class for which the corresponding seed is the closest. As in the case of the clustering, the cosine measure is used in order to perform the classification.

An important feature which we added to our categorization process was a method for distinguishing between very closely related subjects. This issue has been discussed earlier by Chakrabarti et al. [5] for building hierarchical categorization models. Here, we discuss this issue in the context of a flat set of clusters which are defined in terms of their seed vectors. If desired, a hierarchy can also be created by repeatedly applying agglomerative hierarchical clustering on the centroid metadocuments thus created. However, this is orthogonal to the primary aim of this paper, which is to illustrate the effectiveness of partially supervised clustering.

This is required because even a supervised clustering technique may not provide perfect subject isolation. Sometimes, a small percentage[1] of the documents do get clustered with documents from a closely related (though slightly inaccurate) category. Even though a theoretical accuracy of 100 percent can be obtained by reporting the cluster label for the most similar seed, it may sometimes be desirable to correct for the errors in the clustering process by using a context-sensitive comparison method.

1. See empirical section for details.

```
Algorithm Assign(Pseudo-centroids: S, Documents: D)
begin
  Initialize each of the clusters $C_1 \dots C_K$ to null;
  { Note that the clusters $C_1 \dots C_K$ correspond to the pseudo-centroids $s_1 \dots s_K$ in $S$ }
  for each document $d \in D$ do
  begin
    Calculate the cosine of $d$ to each pseudo-centroid in $S$;
    Find the pseudocentroid $s_i$, which is most similar to $d$;
    if $cosine(s_i, d) < OutlierThreshold$ then
      begin
      { $d$ is removed from $D$ as an outlier }
      $D = D - \{d\}$;
      end
    else
      Add $d$ to cluster $C_i$;
  end
  for each $s_i \in S$ do
    redefine $s_i \in S$ to the pseudo-centroid of cluster $C_i$;
  return$(S, D)$;
end
```

Fig. 2. Assigning documents to pseudocentroids.

We build a *domination matrix* on a subset of the universe of categories, such that we know that all of these categories are good candidates for being the best match. As we will see, the simplicity of this process ensures that speed is not compromised by the use of the flat organization of clusters.

The first step in the algorithm is to find the k closest cluster seeds to the test document. The similarity of each cluster to the test document is calculated by using the cosine measure of the test document to the seed corresponding to each cluster. The value of k is a user-chosen parameter and is typically a small number compared to the total number of nodes in the taxonomy. These k categories are the candidates for the best match and may often contain a set of closely related subjects. This ranking process is designed to rerank these categories more appropriately.

In order to understand the importance of distinguishing among closely related subjects, let us consider the seeds for two nodes in the taxonomy: **Business Schools** and **Law Schools**. Recall that our process of projection limits the number of words in each seed to only words which are relevant to the corresponding categories. Some examples of words (with nonzero weights) which could be represented in the seed vector of each of these categories are as follows:

1. **Business Schools:** business (35) , management (31), school (22), university (11), campus (15), presentation (12), student (17), market (11), operations (10)....
2. **Law Schools:** law (22), university (11), school (13), examination (15), justice (17), campus (10), courts (15), prosecutor (22), student (15) ...

```
Algorithm Project(Pseudo-centroids: S, Words: l)
begin
  for each $s_i \in S$ do
  begin
    Retain the $l$ terms in the pseudo-centroid $s_i$ with
    maximum weight, setting the weight of all other terms to 0;
  end
end
```

Fig. 3. Projecting out the less important terms.

A document in the generic category of schools is likely to contain all of the words such as university and school. Thus, both these categories may be among the k closest seeds for the document. In order to establish the relative closeness of two categories to a given document more accurately, we need to ignore the contributions of the words common to both categories to the cosine measure. In other words, we need to compare the closeness based on the words which are *not* common in the seed vector of both categories. This is done by performing a *relative seed subtraction* operation on the seed vectors of each of the categories. The seed subtraction operation is defined as follows: Let S_1 and S_2 be two seed vectors. Then, the seed $S_1 - S_2$ is obtained by taking the seed S_1 and setting the weight of all those words which are common to S_1 and S_2 to 0.

We say that the seed S_1 dominates the seed S_2 under the following conditions:

- The (cosine) similarity of S_1 to the test document T is larger than the similarity of S_2 to T by at least a predefined threshold referred to as the *domthresh*.
- The (cosine) similarity of S_1 to T is not larger than the similarity of S_2 to T by the predefined threshold, but the similarity of $(S_1 - S_2)$ to T is larger than the similarity of $(S_2 - S_1)$ to T.

The use of a domination threshold ensures that it is only possible to reorder seeds whose similarity to the test document are very close together. This is because it is primarily in these cases that the differences in the contributions of the common words tends to be a result of noise, rather than any actual pattern of difference in the frequencies of the (common) words in the seeds for the two categories. For each pair of the closest k seeds to the test document, we compute the domination matrix, which is the pairwise domination of each seed over the other. In order to rank order the k candidate seeds, we compute the *domination*

```
Algorithm Merge(PseudoCentroids: S, MergingThreshold: t)
begin
  for each pair of pseudo-centroids $s_i, s_j \in S$ do
    calculate $cos[ij] = cosine(s_i, s_j)$;
  Construct a graph with one node for each pseudo-centroid;
  for each pair $s_i, s_j$ of pseudo-centroids do
    if $cos[ij] > t$ then add an edge between the nodes for $s_i$ and $s_j$;
  for each connected component of the graph, concatenate the
      metadocuments for the corresponding pseudo-centroids in order
      to create a new pseudo-centroid. Let $S$ be the new set of
      pseudo-centroids formed out of these connected components;
  return($S$)
end
```

Fig. 4. Merging very closely related clusters.

number of each seed. The *domination number* of a seed is equal to the number of seeds (among the remaining $(k-1)$ seeds) that it dominates. The k seeds are ranked in closeness based on their domination number; ties are broken in favor of the original ordering based on cosine measure. The algorithm for returning the ranked set of k categorizations is illustrated in Fig. 6.

It is obvious that the best matching category is more likely to be contained among the top k categories based on cosine measure, than only the closest category based on this measure. (If the clustering is perfect, then it suffices to use $k = 1$.) The reranking process is then expected to rank this category highly among the k choices. If there are a total of K classes created by the clustering algorithm, then the categorization algorithm needs to perform $O(K + k^2)$ cosine similarity calculations. Further, since the projected dimensionality of each seed is restricted to a few 100 words, each similarity calculation can be implemented efficiently. Thus, the categorization system is extremely fast because of its simplicity and scales almost linearly with the number of classes. This feature is critical for its use in performing automated categorization of large libraries of documents.

3 Perfomance of the Categorization System

The assessment of the performance of a categorization system based on supervised clustering presents new challenges. Existing benchmarks, such as the well-known Reuters set, are designed principally to test the performance of a new classifier against an predefined set of classes, i.e., the class label for each document is defined externally, and the goal is to measure the accuracy in terms of this "expert" classification. However, when a new set of classes are created, such as by our clustering, these categories may not have a precise correspondence to the original set of classes and an accuracy measurement with respect to this new set is meaningless.

We know that since the classifier uses the same similarity model as the clustering system does, the key issues are clustering quality and classifier speed. Another point to understand is that the use of supervised clustering to create the new set of categories makes it difficult to apply the standard synthetic data models and techniques [14], [15] which are used for evaluating unsupervised clustering; therefore, our discussion of the performance of the system is primarily an intuitive one based on real data.

As indicated earlier, we used a scan of the *Yahoo*! taxonomy from November 1996. This taxonomy contained a total of 167,193 Web documents, over a lexicon of approximately 700,000 words. The unusually large size of this lexicon is a result of the fact that many of the words in Web document collections tend to be nonstandard words which could be misspellings or creative variations on standard words. Such words are so sparse that they do not have much of a role to play in the clustering process. Only 87,000 words occured in seven or more documents in the entire collection. We truncated the *Yahoo*! tree taxonomy to obtain a set of 1,463 classes corresponding to higher level nodes. The purpose was to use the lowest level nodes in the taxonomy which contained at least 50 or more documents. (Otherwise, it is difficult to use such sparsely populated nodes for any kind of reasonable categorization or clustering.) The slight shortening and variation of names illustrated in Table 1 from the actual *Yahoo*! names is because of this truncation. The total number of categories at the leaf level of this truncated *Yahoo*! taxonomy was about 1,463.

We first performed unsupervised clustering of the data by using an improved variation of the algorithm discussed in [18].[2] We found about 1,000 clusters from the original set of documents. Although unsupervised clustering was able to group together similar documents in each cluster, it was unable to perform the fine-grained level of subject isolation that one would expect from such a large number of clusters. For example, a cluster was formed such that it contained constituent documents that were drawn from *Yahoo*! categories related to computer generated art, hand-crafted arts, artists, painting, sculpture, museums, and architecture. Although these documents shared considerable similarity in the subject material and vocabulary, the overall subject

```
Algorithm Kill(Pseudo-centroids: S, MinimumLimit: l)
begin
  for each pseudo-centroid $s_i \in S$ such that the
  associated cluster $C_i$ has fewer than $l$ documents, discard the
  corresponding pseudo-centroids from $S$ ( $S = S - \{s_i\}$);
  return($S$);
end
```

Fig. 5. Removal of poorly defined clusters.

2. We omit the exact details of the implementation of this algorithm since it is beyond the scope of this paper.

```
Algorithm Classify(TestDocument: T)
begin
  Use cosine measure to find the $k$ closest seeds $\{S_1 \ldots S_k\}$
            to the test document $T$;
  for $i = 1$ to $k$ $domination[i] = 0$;
  for $i = 1$ to $k$ do
    for $j = (i + 1)$ to $k$ do
    begin
      if $(cosine(S_i, T) > cosine(S_j, T) + domthresh)$ then
      $domination[i] = domination[i] + 1$;
      else if $(cosine(S_i - S_j, T) > cosine(S_j - S_i, T))$ then
      $domination[i] = domination[i] + 1$;
      else
      $domination[j] = domination[j] + 1$;
      end
    end
  Rank order the $k$ categorizations in decreasing order of $domination[i]$;
end
```

Fig. 6. The classification algorithm.

TABLE 1
Some Examples of Constituent Documents in Each Cluster

Category	*Yahoo*! Categories of constituent documents
1. Wine	@Entertainment@Drinks_and_Drinking@Alcoholic_Drinks@Wine (28)
	@Business_and_Economy@Companies@Drinks@Alcoholic@Wine@Wineries (9)
	@Business_And_Economy@Companies@Drinks@Alcoholic@Wine@Other (2)
	@Entertainment@Magazines@Other (2)
2. Fitness	@Health@Fitness (29)
	@Business_and_Economy@Companies@Health@Fitness (7)
	@Recreation@Sports@Other (3)
	@Business_and_Economy@Companies@Sports@Other (3)
	@Business_and_Economy@Products_and_Services@Magazines@Sports (3)
3. Health Organizations	@Health@Medicine@Organizations (39)
	@Health@Medicine@Other (16)
	@Health@Other (4)
	@Education@Other (3)
	@Business_And_Economy@Organizations@Public_Interest_Groups@Other (3)
	@Business_And_Economy@Companies@Health@Other (3)
	@Business_And_Economy@Companies@Books@Titles@Other (2)
4. National Parks	@Recreation@Outdoors@Parks@National_Parks... (37)
	@Recreation@Outdoors@Other (8)
	@Recreation@Travel@Regional@US_States@Other (7)
	@Business_And_Economy@Companies@Travel@Tour_Operators@Adventure (7)
	@Recreation@Travel@Regional@Countries@Other (2)
	@Business_...@Companies@Travel@Lodging@Regional@...@Canada (2)
5. Star Trek	@News_And_Media@Television@Genres@Science_FF&H@Star_Trek@Other (120)
	@Entertainment@Movies_And_Films@Actors_And_Actresses (13)
	@Entertainment@Science_FF&H (4)
	@Entertainment@Movies_And_Films@Genres@Science_FF&H@Other (4)
	@Entertainment@Humor_Jokes_And_Fun@Other (3)
	@Recreation@Hobbies_And_Crafts@Collecting (3)

material in the documents was relatively generic (art), and it was difficult to find the level of fine-grained subject isolation that is available in the *Yahoo*! taxonomy. Almost all the clusters found using the unsupervised clustering technique provided categories in which the overall subject was as generic as a top-level category of the hierarchical *Yahoo*! organization. This is consistent with our earlier observation that unsupervised methods are often unable to create a sufficiently fine-grained subject isolation. In such cases, it is not even possible to provide meaningful subject labels for the different clusters. We also tested the supervised version of the clustering algorithm without the use of feature selection and discovered similar results; wherein documents from different categories got combined in an incoherent way.

In our implementation of the supervised clustering algorithm, we first calculated the gini index of the different words in the clusters and removed about 10,000 words with the highest gini index. We also removed the very infrequently occuring words in order to remove misspellings and creative variations on ordinary words. Specifically, we removed all those words which occured in fewer than seven documents out of the original training data set of 167,193 Web documents. At this stage, we were left with a lexicon of about 77,000 words. We found that the use of the *idf* normalization actually decreased the quality of the clustering and, therefore, we used only the term frequencies in order to represent the weights of the terms in the vector-space representation of the documents.[3] We restricted the number of words in the pseudocentroid of each cluster to about 200. The algorithm started with about 500 projected words in each seed, and successively removed words from the seed, until a maximum of about 200 words was obtained in each seed. The algorithm was terminated as soon as fewer than 200 words were remaining in the seeds. The value of the seed reduction factor θ was 0.7. The value of the parameter *minimum* used to decide when to kill a cluster was 8. The value of the merging threshold (the parameter *threshold* in Fig. 1) was 0.95. The choice of 200 words as a threshold was made by running experiments on the coverage of the significant vocabulary by the clusters. In general, we wanted each word to be covered by at least one cluster, but at the same time not be covered by five to 10 words on the average. In order to achieve this goal, a choice of 200 words turned out to be quite effective. The algorithm required a total of three hours to complete on a 233 MHz AIX machine with 100 MB of memory. We obtained a total of 1,167 categories in a flat taxonomy.

We labeled the nodes by examining the constituent *Yahoo*! categories in this set of newly created clusters. Typically, the clustering did a very excellent job in grouping together documents from very closely related categories in the *Yahoo*! taxonomy in a creative way, so as to result in a coherent subject for each cluster. This kind of grouping differs from the case of unsupervised clustering in terms of providing much cleaner subject isolation. We have illustrated some examples of the clusters obtained in Table 1. In the right-hand column of the table, we have illustrated the constituent documents in each *Yahoo*! category in the cluster. In order to give a flavor of how well the clustering performed, we provide some interesting observations for these examples:

1. **Wine.** The cluster for wine was constructed by picking out documents from the wine segment of the *@Entertainment* and *@Business_And_Economy* subcategories of *Yahoo*!. Two of the documents in the category were also drawn from a *Yahoo*! category on magazines, but in both the cases, we found that the documents were magazine articles related to wine. Although the original *Yahoo*! categorization was accurate, the actual content of the page cannot be directly related to the topic of magazines, without prior knowledge of the metainformation that the document is a magazine article. Thus, the training phase of a classifier on the *Yahoo*! taxonomy would not be helped by the presence of such a document in the category on magazines.
2. **Fitness.** Most of the documents in the **Fitness** category were drawn from either fitness, health, or sports related categories from the *Yahoo*! taxonomy. Again, these categories occured in widely separated branches of the *Yahoo*! hierarchy. This is an example of a case where the clustering was able to pick out specific documents from interrelated categories which were more general than the topic of fitness. The presence of very closely related documents in many widely separated branches of a hierarchy is somewhat detrimental to the performance of a hierarchical classifier because it becomes more difficult to distinguish nodes at the higher levels of the taxonomy.
3. **Health Organizations.** This cluster consisted of documents from commercial health related categories, noncommercial health related categories, and education related categories, which were related to the general topic of health organizations.
4. **National Parks.** In the original *Yahoo*! categorization, many of the documents on parks fall under the travel, tourism, and outdoors category. Correspondingly, the clustering grouped together Web pages which were closely related to the subject material of parks but drawn from these very distinct *Yahoo*! categories. Again, the presence of documents on parks in travel related categories of *Yahoo*! may often confuse classifiers since this subject is only peripherally related to the topic of travel and tourism in general.
5. **Star Trek.** The clustering was able to pick out documents from various *Yahoo*! categories, which dealt with different aspects of Star Trek including the categories on television shows, motion pictures, actors and actresses, and collectibles. Two documents were also drawn from the Humor category of *Yahoo*! which had Star Trek related material in them.

In many cases, we found a correspondence between some original *Yahoo*! class and the final class which was created by the supervised clustering technique. However, the documents in these categories were often quite different and more directly related to the actual content of the page. Manual categorizations take into account factors which are not reflected in the content of a given document. Although this is often desirable to effectively index documents, it does not help provide the ability to train classifiers accurately. The redistribution of such documents to clusters which are more related on content provides a cleaner set of classes from the perspective of a content-based classification engine. (See, for example, the case for the category of wine above.) In all cases, we found that the projected set of words for a cluster corresponded very closely with its subject.

3.1 Categorization

We ran the classifier and reported the three best categories as the results. The domination threshold used for 0.025. We found that the use of the domination matrix approach caused a different ranking (from the original ranking of the categories based on cosine measure) in about 8 to 9 percent of the cases. In most cases, the use of the domination number lead to an improvement in the quality of the categorization. In order to provide a flavor of the performance of categorization, we provide some examples of the classifications which were reported in Table 2. One interesting observation was that when we tested documents which were related to multiple subjects in the taxonomy, the classifier was able to get the different related subjects as the first,

3. The fact that *idf* normalizations reduce cluster quality has also been observed in earlier work [18].

TABLE 2
Some Examples of Classifier Performance

Example	**Comments**
1. http://www.wpi.edu/t̃anis/lawyer.html: **Title:** Canonical List of Lawyer Jokes **First Category:** Jokes **Second Category:** Individual Attorneys **Third Category:** Misc. Humor, Jokes and Fun	Since there is no category on lawyer jokes, the classifier finds the categories most closely related to both attorneys and jokes.
2. http://www.doubleclick.com/advertisers: **Title:** Solutions for Advertisers **First Category:** Misc. Advertising (Commercial) **Second Category:** Misc. Marketing (Commercial) **Third Category:** Direct Marketing (Commercial)	Can distinguish pages about advertising from pages containing advertising
3. http://www.pliant.org/personal/TomErickson/hawaii.html: **Title:** The Key **First Category:** Web Science, Fiction, Fantasy, Horror **Second Category:** Misc. Humor Jokes and Fun **Third Category:** Jokes	This is a humorous narrative on the Web by a scientist.
4. http://www.crstat.com/default.htm: **Title:** Charles River Strategies Inc. **First Category:** Management Consulting (Commercial) **Second Category:** Market Research (Commercial) **Third Category:** Information Technology Consulting (Commercial)	Charles River Strategies (CRS) is "the computer industry's prominent research firm focussed on integrated end-user and channel marketing strategy"
5. http://research.microsoft.com/datamine/kdd99: **Title:** Conference on Knowledge Discovery and Data Mining **First Category:** Computer Conventions and Conferences **Second Category:** Miscellaneous Computer Science **Third Category:** Artificial Intelligence (Computer Science)	Finds all the topics- conferences, computer science (general) and artificial intelligence (more specific) which are related to data mining and knowledge discovery
6. http://users.neca.com/vmis/wsockexp.htm: **Title:** Getting back to the basics **First Category:** Misc. Internet Software Protocols **Second Category:** Microsoft Windows 95 **Thord Category:** Internet Information and Documentation	A page of documentation on the Winsock Protocol
7. http://www.software.ibm.com/ad/vajava: **Title:** IBM Visual Age for Java **First Category:** Object-Oriented Programming Tools (Commercial) **Second Category:** Programming Languages (Commercial) **Third Category:** Software Consulting (Commercial)	The word object-oriented is not mentioned on the Web-page

second, and third categories. As an example (see Table 2), when we tested a Web page containing a compilation of lawyer jokes, the classifier was able to pick out both the closely related subjects (lawyer and jokes) among its different choices. Another similar example was (3) in Table 2, where a humorous narrative on the Web was categorized to belong to both fiction and humor related categories.

When we tested the classifier on the homepage of the 1999 Conference on Knowledge Discovery and Data Mining, we found that the top three categories provided some interesting information:

1. The document was related to a computer conference.
2. The document was related to computer science.
3. The document was related to artificial intelligence.

Since there is no category on data mining conferences, the classifier finds the closest set of general categories which are related to data mining conferences. Thus, each of the first, second, and third categories provide different pieces of relevant information about this document.

We observed this kind of behavior by the categorization system on a very regular basis. Another example was a page of documentation on the Winsock Protocol, in which case, it provided the categories of Internet Software Protocols, Windows 95, and Internet Information and Documentation as relevant categories. Again, we see that although there is no category in the taxonomy which relates to documentation on the Winsock Protocol, the classifier is able to find categories, all of which are closely related to some aspect of the document. The other property which we noted about the classifier was that it was often able to infer peripherally related subjects as its second and third choices. For example, the first choice for the page on IBM Visual Age for Java was on object-oriented programming tools, the second choice was on programming languages, and the third choice was on software consulting. Clearly, the first choice was a very exact match, whereas the second and third choices were peripherally related. It was very rare that the classifier reported totally unrelated choices for any subject.

The simplicity of the classifier ensured that it was extremely fast in spite of the very large number of classes used. The classifier required about two hours to categorize about 160,000 documents. This averaged at about 45 milliseconds per categorization. This does not include the time for parsing and tokenizing the document. In fact, the parsing procedure dominated the overall time for categorization (0.1 seconds for parsing). Since most text categorization techniques would need to parse the document anyway, this indicates that our categorization system is within a

TABLE 3
Survey Results

Case	Percentage of Instances
Better than *Yahoo*!	8%
Not as good as *Yahoo*!	8%
Both were equally correct	78%
Neither is correct	6%
Unknown	1%

reasonable factor of the best possible overall speed of parsing and subsequent classification.

3.2 An Empirical Survey of Categorization Effectiveness

It is hard to provide a direct comparison of our technique to any other classification method by using a measure such as classification accuracy. This is because our algorithm does not use the original set of classes, but it actually defines the categories on its own. Therefore, the accuracy of such a classifier is high, whereas the actual quality of categorization is defined by clustering quality. Therefore, we need to provide some way to measure the clustering effectiveness.

Since the entire thrust of our supervised clustering algorithm was to facilitate the generation of a set of good clusters (in terms of human judgment) from a manually-created taxonomy of real Web pages, it is also impossible to use the traditional synthetic data techniques (used for unsupervised clustering algorithms) in order to test the effectiveness of our technique. Thus, we need to find some way of quantifying the results of our clustering technique on a real data set such as *Yahoo*!.

We used a survey technique by using input from external survey respondents in order to measure the quality of the clustering. We sampled 141 documents from the clusters obtained by our algorithm, and asked about 10 respondents to indicate how well the corresponding subject labels defined it with respect to the original *Yahoo*! categorization. Specifically, for each document, we asked respondents to indicate one of the following five choices:

1. *Yahoo*! categorization was better.
2. Our categorization was better.
3. Both were similar.
4. Neither were correct.
5. Do not know.

The results of our survey are indicated in Table 3.

One of the interesting aspects of the results in Table 3 is that the quality of our categorization was as good as *Yahoo*! for 78 percent of the documents. Out of this 78 percent, the two categorizations reported the same label in 88 percent of the cases, while the remaining were judged to be qualitatively similar. Among the remaining documents, the opinions were evenly split (8 percent: 8 percent) as to whether *Yahoo*! or our scheme provided a better categorization. For the 8 percent of the cases in which our clustering algorithm provided a categorization which was not as good as *Yahoo*!, we found that most of these instances belonged to one of two kinds:

1. Neither categorization was particularly well suited, though the page was better categorized in *Yahoo*!. Typically, the content of the page did not reflect either category well, though some more metaunderstanding of the page was required in order to accurately classify it. An example of such a page was http://nii.nist.gov, which discusses the United States Information Infrastructure Virtual Library. The Web page discusses the National Information Infrastructure ("information superhighway"), which is an interconnection of computers and telecommunication networks, services, and applications. The document was present in the *Yahoo*! category on Government, though we clustered it along with telecommunication documents. The fact that the document is government related is metainformation which cannot be automatically derived from its content. We assert that the effective categorization of such documents may be difficult for any system which is based purely on content.
2. Our algorithm inserted the document in a closely related cluster, though the original *Yahoo*! categorization was slightly more accurate. This was a more common event than case (1). For example, a URL (http://www.i-channel.com) from the *Yahoo*! category on Cable Networks was grouped with miscellaneous documents on television by our clustering algorithm. Another URL (http://fluxnet.com) from the *Yahoo*! category on Rock Music CDS and Tapes was categorized by our algorithm in the general Rock Music topic. Most of these categorizations (though less accurate) were good enough to be not considered unreasonable.

Another way of interpreting the survey results is that that our automated (supervised) scheme provided clusters which were as good as or better than the *Yahoo*! partitions in $78 + 8 = 86$ percent of the cases, whereas the vice-versa was true for $78 + 8 = 86$ percent of the cases. Among the 10 percent of the documents, in which *Yahoo*! was judged better, a substantial fraction belonged to Case (2) above, in which the documents were inserted in a reasonably good cluster although not quite as accurately as its classification in *Yahoo*!. To summarize, the respondents found little or no quality difference in the (manual) categorization of *Yahoo*! versus our supervised cluster creation. The use of domination numbers improved the performance of our categorizer by about 5 percent.

It is important to understand that while the two categorizations are qualitatively similar, one of them (Yahoo!) is a high-cost manually built system, while the other provides an automated technique for effectively categorzing large libraries of text collections.

This is because an ability to express each category in a structured way is a key advantage from the perspective of a classifier. Thus, the perceived accuracy of the overall categorization system is expected to be much higher than one built from training on a predefined set of classes and the benefits of the use of partially supervised clustering are apparent.

4 Conclusions and Summary

In this paper, we proposed methods for building categorization systems by using supervised clustering. We also discussed techniques for distinguishing closely related classes in the taxonomy. We built such a categorization system using a set of classes from the *Yahoo*! taxonomy and using them as a base in order to create the supervised clusters. We showed that the supervised clustering created a new set of classes which were surveyed to be as good as the original set of classes in the *Yahoo*! taxonomy, but which are naturally suited to automated categorization. The result is a system which has much higher overall quality of categorization. Combined with the low cost of an automated categorization scheme compared to a manual scheme, such a system is likely to have wide applicability to large document repositories.

References

[1] C.C. Aggarwal, C. Procopiuc, J.L. Wolf, P.S. Yu, and J.-S. Park, "Fast Algorithms for Projected Clustering," *Proc. ACM SIGMOD Conf. Management of Data,* 1999.

[2] C.C. Aggarwal, S.C. Gates, and P.S. Yu, "On the Merits of Using Supervised Clustering for Building Categorization Systems," *Proc. ACM SIGKDD Conf. Knowledge Discovery and Data Mining,* 1999.

[3] P. Anick and S. Vaithyanathan, "Exploiting Clustering and Phrases for Context-Based Information Retrieval," *Proc. SIGIR,* pp. 314-322, 1997.

[4] C. Apte, F. Damerau, and S.M. Weiss, "Automated Learning of Decision Rules for Text Categorization," *ACM Trans. Information Systems,* 1994.

[5] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan, "Using Taxonomy, Discriminants, and Signatures for Navigating in Text Databases," *Proc. VLDB Conf.,* Aug. 1997. Extended Version: "Scalable Feature Selection, Classification, and Signature Generation for Organizing Text Databases into Hierarchical Topic Taxonomies," *VLDB J.,* vol. 7, pp. 163-178, 1998.

[6] S. Chakrabarti, B. Dom, and P. Indyk, "Enhanced Hypertext Categorization Using Hyperlinks," *Proc. 1998 ACM SIGMOD Conf. Management of Data,* 1998.

[7] D.R. Cutting, D.R. Karger, J.O. Pedersen, and J.W. Tukey, "Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections," *Proc. SIGIR,* pp. 318-329, 1992.

[8] D.R. Cutting, D.R. Karger, and J.O. Pedersen, "Constant Interaction-Time Scatter/Gather Browsing of Very Large Document Collections," *Proc. 16th Ann. ACM SIGIR,* 1993.

[9] B.L. Douglas and A.K. McCallum, "Distributional Clustering of Words for Text Classification," *Proc. ACM SIGIR,* pp. 96-103, 1998.

[10] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data.* Englewood Cliffs, N.J.: Prentice Hall, 1988.

[11] M.A. Hearst and J.O. Pedersen, "Re-Eexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results," *Proc. ACM SIGIR,* pp. 76-84, 1996.

[12] D. Koller and M. Sahami, "Hierarchically Classifying Documents Using Very Few Words," *Proc. Int'l Conf. Machine Learning,* July 1997.

[13] W. Lam and C.Y. Ho, "Using a Generalized Instance Set for Automatic Text Categorization," *Proc. ACM SIGIR,* pp. 81-88, 1998.

[14] D.D. Lewis, "Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval," *Proc. ECML,* 1998.

[15] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, "Learning to Classify Text from Labeled and Unlabeled Documents," *Proc. AAAI,* 1998.

[16] G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval.* New York: Mc Graw Hill, 1983.

[17] R. Sibson, "SLINK: An Optimally Efficient Algorithm for the Single Link Cluster Method," *Computer J.,* vol. 16, pp. 30-34, 1973.

[18] H. Schutze and C. Silverstein, "Projections for Efficient Document Clustering," *Proc. ACM SIGIR,* pp. 74-81, 1997.

[19] C. Silverstein and J.O. Pedersen, "Almost-Constant Time Clustering of Arbitrary Corpus Sets," *Proc. ACM SIGIR,* pp. 60-66, 1997.

[20] O. Zamir and O. Etzioni, "Web Document Clustering: A Feasibility Demonstration," *Proc. ACM SIGIR.* pp. 46-53, 1998.

Charu C. Aggarwal received the BTech degree in computer science from the Indian Institute of Technology (1993) and the PhD degree in operations research from the Massachusetts Institute of Technology (1996). He has been a research staff member at the IBM T.J. Watson Research Center since June 1996. He has applied for or been granted 40 US patents and has published in numerous international conferences and journals. He has been designated master inventor at IBM Research. His current research interests include algorithms, data mining, and information retrieval. He is interested in the use of data mining techniques for Web and ecommerce applications.

Stephen C. Gates received the BS degree in physics (1969) and the PhD degree in biochemistry (1977), both from Michigan State University. He is manager of Internet Categorization Systems at the IBM T.J. Watson Research Center, where he has been since 1987. He was a tenured college associate professor of biochemistry prior to joining IBM. His research interests are in high-performance taxonomies and categorization systems suitable for Web-scale and larger problems.

Philip S. Yu received the BS degree in electrical engineering from the National Taiwan University, the MS, and Ph.D. degrees in electrical engineering from Stanford University, and the MBA degree from New York University. He is with the IBM T.J. Watson Research Center and is currently the manager of the Software Tools and Techniques group. His research interests include data mining, bioinformatics, Internet applications and technologies, database systems, multimedia systems, parallel and distributed processing, performance modeling, and workload analysis. Dr. Yu has published more than 340 papers in refereed journals and conferences. He holds or has applied for 254 US patents. Dr. Yu is a fellow of the ACM and a fellow of the IEEE. He is the editor-in-chief of *IEEE Transactions on Knowledge and Data Engineering*. He is also an associate editor of *ACM Transactions on the Internet Technology* and that of *Knowledge and Information Systems*. He is a member of the IEEE Data Engineering steering committee and is also on the steering committee of the IEEE Conference on Data Mining. In addition to serving as program committee member on various conferences, he was the program co-chairs of the 11th International Conference on Data Engineering and the Sixth Pacific Area Conference on Knowledge Discovery and Data Mining, and the program chairs of the Second International Workshop on Research Issues on Data Engineering: Transaction and Query Processing, the PAKDD Workshop on Knowledge Discovery from Advanced Databases, and the Second International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems. He served as the general chair of the 14th International Conference on Data Engineering and the Second IEEE International Conference on Data Mining. He has received several IBM and external honors including the Best Paper Award, two IBM Outstanding Innovation Awards, an Outstanding Technical Achievement Award, two Research Division Awards, and the 71st plateau of Invention Achievement Awards. He also received an IEEE Region 1 Award for "promoting and perpetuating numerous new electrical engineering concepts" in 1999. Dr. Yu is an IBM Master Inventor and was recognized as one of the IBM's 10 top leading inventors in 1999.