

# Modeling Document Taxonomies

Scott Spangler  
IBM Almaden Research  
Center  
650 Harry Road, San Jose,  
CA 95120  
408-927-2887  
spangles@us.ibm.com

Jeffrey Kreulen  
IBM Almaden Research  
Center  
650 Harry Road  
San Jose, CA 95120  
408-927-2431  
kreulen@us.ibm.com

Justin Lessler  
IBM Almaden Research  
Center  
650 Harry Road, San  
Jose, CA 95120  
408-927-2449  
lessler@us.ibm.com

David E. Johnson  
IBM Watson Research  
Center  
1101 Kitchawan Road,  
Yorktown Hts, NY 10598  
914-945-1036  
dejohns@us.ibm.com

## ABSTRACT

Taxonomies are meaningful hierarchical categorizations of documents into topics reflecting the natural relationships between the documents and their business objectives. Creating effective taxonomies and reducing the overall cost required to create them is an important area of research. Unsupervised text clustering is one part of the solution, but automated clustering alone is insufficient to consistently create effective taxonomies in a business environment. To address this problem, we have developed tools that allow for a “mixed initiative” approach to taxonomy development, where human expertise can be employed to edit and refine a text clustering to make it more meaningful for a given application. Document taxonomies developed using mixed initiative methods pose the following challenge: how do we model the taxonomy so that future documents will be classified correctly. We have developed a comprehensive approach to solving this problem, and implemented this approach in a software tool called eClassifier. The crux of our solution is to apply a suite of classifiers, including both statistical and rule based varieties at each level of the taxonomic hierarchy, and then to choose for each category the best classifier or set of classifiers, that produce the most accurate results on unseen test documents. We tested various methods of combining these multiple classifiers against several different mixed initiative taxonomies and against the standard Reuters data set. We show that in nearly all cases, one method in particular performed better than the others and that this method significantly improves upon any single classifier approach.

## Categories and Subject Descriptors

H.2.8 [Information Systems]: Data Mining;  
H.3.3 [Information Search & Retrieval]: Clustering;  
I.5.3 [Clustering]: Similarity Measures

## General Terms

Algorithms

## Keywords

Text Mining, Clustering, Mixed Initiative, Classification

## 1 INTRODUCTION

The ability to organize information around common themes, also known as taxonomy generation, is an increasingly important area of research, particularly in the area of search and information retrieval. We have developed a comprehensive text analysis application, eClassifier, that incorporates the ability to generate, edit, and validate taxonomies [3][18]. This tool was developed to fill a void where automated clustering algorithms fell short and manual categorization was too expensive and unscalable.

During the development and utilization of this tool, we have increasingly found that many useful taxonomies can be developed, even on the same set of information and that you need many different capabilities and approaches to generate these taxonomies. Additionally, as we applied this technology across many domains and integrated with larger applications, it became clear that these human edited taxonomies would need to be accurately modeled in order to categorize large numbers of new documents automatically. In short we needed to find a way to accurately model document taxonomies developed using multiple ad hoc techniques. For this purpose we implemented a suite of classifiers utilizing a broad range of techniques well established in the literature as being the most effective known for supervised categorization. In practice, we found that different classification algorithms performed better in different circumstances. What is more, we saw that even within the same taxonomy, some classifiers would perform better for one set of categories, whereas others would perform better on different categories. It seemed we needed an approach that would apply the most accurate categorizer on each category. This sounds simple enough, but in practice when two (or more) different classifiers differ in the way they would categorize a document, some accurate way of deciding which categorizer to trust must be found.

Past methods for combining different classifier fall into two groups. The first group consists of classifiers that attempt to compensate for classifier instability or improve accuracy by intelligent use of the training data. These techniques are usually used to combine several instantiations of a single classification

algorithm to produce a single classification. Bagging [1], Arcing [2] and Boosting [6] are all examples of this type of classifier combination.

The second camp, to which our multi-classifier approaches belongs, attempts to combine the results of several different classification algorithms into a single classifier [6][9][20]. This “mixture of experts” approach is partially based on the intuition that multiple generative processes may be involved in the creation of a taxonomy. As different classifiers are better suited for modeling different divisions in data, it stands to reason that a set of classifiers would better classify documents than a single classifier. In our approach we take this intuition one step further. We have observed that often when users of eClassifier produce a taxonomy, each class is created in a different manner. Some classes are created using a particularly distinguishing set of keywords, others via a quick scan of included documents, and some by making slight modifications to the initial clustering. Because of this our techniques are focused around predicting each combined classifier’s performance on a particular class, and favoring the classifiers that appear to be best able to model that class.

This approach is essentially a heuristic for combining classifiers. While it may be more limited than approaches that involve learning a combining function, such as those described in [6][9][20], it works well in practice. Benefits of our approach include the fact that category based distinctions in classifier performance will be captured with even small training sets. Additionally, by focusing in on the subset of the factors that might be affecting classifier performance that we feel is most likely important, we avoid the noise of the other factors that might prevent a learned approach from finding these class based differences and adjusting overall behavior accordingly.

In this paper we will describe an approach for accurately modeling document taxonomies that are created using a mixed initiative approach in eClassifier. In section 2, we describe in detail how such taxonomies are created. Once created these taxonomies need to be modeled so that future documents can be accurately classified. We discuss several approaches for combining multiple classifiers in order to accurately model mixed initiative document taxonomies in section 3. In section 4, we discuss the results of testing the different modeling approaches on several document collections and taxonomies. Finally, in section 5 we summarize and outline areas for future research.

## 2 MIXED INITIATIVE TAXONOMIES

Mixed Initiative taxonomy generation is the process of creating categories of documents from a large collection using both text mining and human expertise. We have found this approach to be compelling in business application of taxonomies, because people and organizations have many different views of the world and how it should be organized. Taxonomies are used to organize information for many different purposes, necessitating organizing around things as disparate as technology, process, geography, business model and others. In this section we describe in detail how such taxonomies are created.

Our process for developing meaningful document taxonomies from a large collection of documents usually begins with text clustering. After eliminating common stop words and (high- and low-frequency) non-content-bearing words, we represent the text data set as a vector space model, that is, we represent each text example as a vector of certain weighted frequencies of the remaining words [17]. We used the **txn** weighting scheme [16]. This scheme emphasizes words with high frequency in a document, and normalizes each document vector to have unit Euclidean norm. For example, if a document were the sentence, “We have no bananas, we have no bananas today,” and the dictionary consisted of only two terms, “bananas” and “today”, then the unnormalized document vector would be {2 1} (to indicate two bananas and one today), and the normalized version would be:  $\left[ \frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}} \right]$ .

As our primary tool for automated classification, we used the k-means algorithm [6], [7] using a cosine similarity metric [14] to automatically partition the problem tickets into k disjoint clusters. The algorithm is very fast and easy-to-implement. See [14] for a detailed discussion of various other text clustering algorithms. The k-means algorithm produces a set of disjoint clusters and a *centroid* for each cluster that represents the cluster mean. Typically k is initially set to 30, for the highest level of the taxonomy, though the user may adjust this if desired. The initial taxonomy assigns each document to only one category (cluster). Any of the initial high level categories may be subcategorized if desired, either automatically and recursively, or manually one at a time.

An alternative to k-means clustering is to create an initial categorization via Boolean keyword queries. This approach is most useful when the domain expert already has a strong idea of how the taxonomy should be structured. Each category is described via a set of keywords connected by “and”, “or”, or “not”. The resulting query defines those document examples that belong to the category. The queries are then ordered and the document initially falls into the category of the query that matches its content. Any document that does not match any query is placed in a special “Miscellaneous” category. The user may reorder the queries based on the results to insure that every category starts with a significant number of matching documents. Once the initial taxonomy is created, further refinement can take place via k-means clustering (using centroids of the existing categories as seeds) or by manual editing of the categories (see next section).

### 2.1 Taxonomy Refinement

Once an initial taxonomy has been generated, the next step is to provide tools for rapidly changing the taxonomy to reflect the needs of the application. Keep in mind that our goal here is not to produce a “perfect” taxonomy for every possible purpose. Such a taxonomy may not even exist, or at least may require too much effort to obtain. Instead we want to focus the user’s efforts on creating a “natural” taxonomy that is practical for a given application. For such applications, there is no right or wrong change to make. It is important only that the change accurately reflect the expert user’s point of view about the desired structure. In this situation, the user is always right. The tool’s job is to allow the user to make whatever changes may be deemed desirable. In some cases such changes can be made at

the category level, in other cases a more detailed modification of category membership may be required. Our tool provides capabilities at every level of a taxonomy to allow the user to make the desired modifications with a simple point and click.

### 2.1.1 Category Level

Category level changes involve modifying the taxonomy at a macro-level, without direct reference to individual documents within each category. One such modification is merging. Merging two classes means creating a new category that is the union of two or more previously existing category memberships. A new centroid is created that is the average of the combined examples. The user supplies the new category with an appropriate name.

Deleting a category (or categories) means removing the category and its children from the taxonomy. The user needs to recognize this may have unintended consequences, since all the examples that formerly belonged to the deleted category must now be placed in a different category at the current level of the taxonomy. To make this decision more explicit, we introduce the graphic called “View Similar Categories” chart:

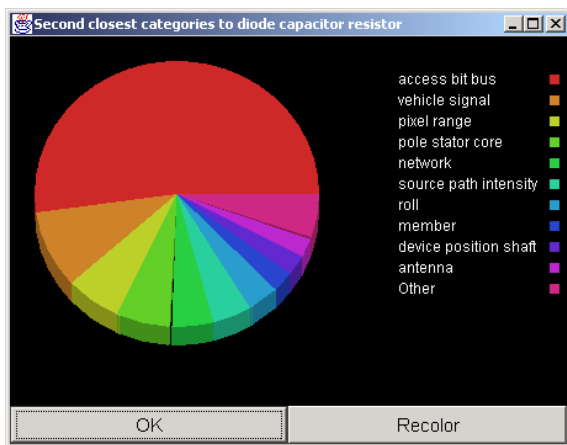


Figure 1: Similar Categories Display

This chart displays what percentage of a categories documents would go to which other categories if the selected category were to be deleted. Each slice of the displayed pie chart can be selected to view the individual documents represented by the slice. Making such information explicit allows the user to make an informed decision when deleting a category, avoiding unintended consequences.

In addition to merging and deleting, the user can select any category and drag and drop the category into any existing folder (a category with children). An example of when such an operation might be performed is when a very specific category is created at the root node of the tree, which would more naturally belong within an already existing, more general, category. The operation of dragging and dropping a category to a folder has consequences to all other folders in a direct line from the root of the tree to the destination node (which gain the contents of the source node) and to all other folders in a direct line from the root to the source node (which lose the contents of the source node).

All such consequences are automatically handled by eClassifier.

### 2.1.2 Document Level

While some changes to a taxonomy may be made at the class level, others require a finer degree of control. These are called document level changes, and consist of moving or copying selected documents from a source category to a destination category. The most difficult part of this operation from the users point of view is selecting exactly the right set of documents to move so that the source and destination categories are changed in the manner desired. To facilitate this eClassifier provides a number of mechanisms for selecting documents.

One of the most natural and common ways to select a set of documents is with a keyword query. EClassifier allows the user to enter a query for the whole document collection or for just a specific category. The query can contain keywords separated by “and” and/or “or” and also negated words. Words that co-occur with the query string are displayed for the user to help refine the query. Documents that are found using the keyword query tool can be immediately viewed and selected one at a time or as a group to move or establish a new category.

Another way to select documents to move or copy is via the “Most/Least Typical” sorting technique whereby the example documents are displayed in sorted order by cosine distance from the centroid of their category [18]. For example, the documents that are least typical of a given category can be located, selected, and moved out of the category they are in. They may then be placed elsewhere or in a new category.

The scatter plot visualization display [5] can also be a powerful tool for selecting individual or groups of documents. Using a “floating box”, groups of contiguous points (documents) can be selected and moved to the new desired class.

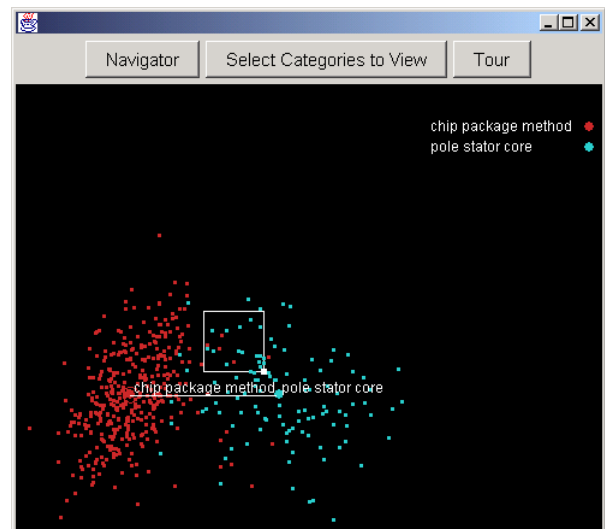


Figure 2: Floating box for moving documents

Independent of the document selection method, the user is allowed to choose between moving, copying, or deleting the selected documents. Moving is generally preferable because

single class membership generally leads to more distinct categories which are better for the classification of future documents. Still, in cases where a more ambiguous category membership better reflects the user's natural understanding of the taxonomy, eClassifier allows the user to create a copy of the documents to be moved and to place this copy in the destination category. In such cases the individual document will actually exist in two (or more) categories at once, until or unless the user deletes the example. Deletion is the third option. It allows the document to be removed entirely from the taxonomy, if it is judged to be not applicable.

### 3 MODELING APPROACHES

The mixed initiative method of taxonomy development described in the previous section introduces a unique set of problems when trying to model the taxonomy created with an automated classifier. First of all, each level of the taxonomic hierarchy may be created using a different categorization approach (e.g. text clustering vs. keyword queries). But even within a single taxonomy level some categories may have been created by k-means and left alone, while others will be either created or edited by a human expert, so that a centroid model will no longer apply uniformly across all categories.

But if a centroid-based classifier is not sufficient, then what is the modeling approach that should be used? The necessity for using some kind of multi-classifier approach seems clear, since we cannot predict ahead of time which style of categorization the user will employ to create the taxonomy. A simple solution would therefore train and test a suite of different classifiers at each level of the taxonomy, and choose the most accurate classifier (the one having the optimum combination of precision and recall) at each level. This would be an improvement over the single classifier approach, but it still would not handle the problem of different categorization approaches being used within a single level of the taxonomy. To handle this configuration optimally a multi-classifier approach was developed. In this section we describe the single classifier components of the multi-classifier, and then describe several possible methods of combining these individual classifiers into a coherent multi-classifier.

#### 3.1 Classification Approaches

We incorporated the following classifiers into our suite of available classifiers in the eClassifier toolkit.

##### 3.1.1 Centroid

This is the simplest classifier. It classifies each document to the nearest centroid (mean of the category) using a cosine distance metric.

##### 3.1.2 Decision Tree

We use two types of decision tree classifiers in eClassifier. The core decision tree algorithm both share is an implementation of the well-known ID3 algorithm [15]. In addition to eClassifier's core feature selection process (which selects only the N most frequently occurring words in the text corpus to include in the feature dictionary) some classification algorithms benefit from additional reduction in the feature space [12][6]. In these

algorithms we use a method to select terms based on their mutual information with the categories [12][4], and selecting all terms where the mutual information is above some threshold.

##### 3.1.2.1 Single Decision Tree

In the first method we use ID3 to learn a single decision-tree that classifies each document.

##### 3.1.2.2 Set of Binary Decision Trees

In this method we learn a binary or "in/out" decision tree for each class using the ID3 algorithm. Each decision tree is run on each document and returns whether or not the document belongs in the class corresponding to that tree. In the case that more than one tree classifies a document as in, the document is placed into the class with the highest prior probability among those selected. In the case that no decision tree classifies the document as in it is placed in the largest overall class.

##### 3.1.3 Naïve Bayes

We have incorporated two variations of Naïve Bayes classifier into our suite. The first is based upon numeric features, the second on binary features. Both use the well known Bayes decision rule and make the Naïve Bayes assumption [11][12][13] and differ only in how the probability of the document given the class,  $P(d | C_k)$ , is calculated.

##### 3.1.3.1 Numeric Features:

This method, also known as the multinomial model [12], classification is based upon the number of occurrences of each word in the document:

$$P(d | C_k) = \prod_{w_i \in d} P(w_i | C_k)$$

Where the individual word probabilities are calculated from the training data using Laplace smoothing [12]:

$$P(w_i | C_k) = \frac{n_{k,i} + 1}{n_k + |V|}$$

Where  $n_k$  is the total number of word positions in documents assigned to class  $C_k$  in the training data,  $n_{k,i}$  is the number of positions in these documents where  $w_i$  occurs, and  $V$  is the set of all unique words.

##### 3.1.3.2 Binary Features:

This method, also known as the multivariate model [12], calculates probabilities based on the presence or absence of words in documents, ignoring their frequency of occurrence:

$$P(d | C_k) = \prod_{w_i \in V} [B_i P(w_i | C_k) + (1 - B_i)(1 - P(w_i | C_k))]$$

Where  $B_i$  is 1 if  $w_i$  occurs in  $d$  and 0 otherwise, and the individual word probabilities are calculated as:

$$P(w_i | C_k) = \frac{1 + \sum_{d \in D} B_i P(C_k | d)}{2 + \sum_{d \in D} P(C_k | d)}$$

where  $P(C_k | d)$  is 1 if  $d$  is in class  $C_k$  and 0 otherwise.

Since this method works best in the case of smaller feature spaces [12] we use the same method as in the Decision Tree algorithm to do additional feature selection.

### 3.1.4 Rule Based

The rule induction classifier [8] is based on a fast decision tree system that takes advantage of the sparsity of text data, and a rule simplification method that converts a decision tree into a logically equivalent rule set. The system also uses a modified entropy function that both favors splits enhancing the purity of partitions and, in contrast to the gini or standard entropy metrics, is close to the classification error curve, which has been found to improve text classification accuracy.

### 3.1.5 Statistical

The statistical classifier is a version of regularized linear classifier that has similar behavior as a support vector machine, but also provides a probability estimate for each class. It also employs the sparse regularization condition described in [21] to produce a sparse weight vector.

The numerical algorithm is described in [21].

## 3.2 Multi-Classifer Approaches

Each of the multi-classifier approaches we created utilizes all of the classifiers described in section 3.1. To prevent overspecialization on the training data a two-stage training process was employed. Our general approach was to train each of the individual classifiers listed in the previous section on a random sample containing 67% of the original training data set. The remaining 33% of the training set is used for the purpose of classifier weighting/evaluation. We designed three possible approaches to combine classifier results at the category level. In each case once the mathematical formula for determining the classifier weighting used for each category has been determined, the individual classifiers are then retrained on 100% of the training set, in order to achieve optimal performance. The remainder of this section describes the three approaches we employed for combining the individual classifier results.

### 3.2.1 Winner take all

1. For a particular classification algorithm,  $a$ , and category,  $c$ , let

$$F_1(c, a) = \frac{2 * P(c, a) * R(c, a)}{P(c, a) + R(c, a)}$$

where  $P(c, a)$  is the precision of the algorithm on the category and  $R(c, a)$  is the recall.

2. Select for each category the classification algorithm with the highest  $F_1$  score. So each category is owned by exactly one classifier (the "winner")

3. Run each classifier on each new example. If exactly one categorizer classifies a new example into a category that it "owns" then that categorizer chooses the membership of the example. If no categorizer does this, then select the largest

category in the training set. If multiple categorizers classify the example into a category they "own", then select the categorizer with the best  $F_1(c, a)$  score for the example's predicted category.

### 3.2.2 Weighted Voting

In the voting approach, we calculate  $F_1$  as in 3.3.1. Then at runtime:

1. Run each classifier on each new example.
2. Each classifier gets one "vote" for a possible category for each new example.
3. The value of this vote is the  $F_1(c, a)$  score for that category and algorithm.
4. The category that gets the highest vote sum of squares total is the predicted category.

### 3.2.3 Hybrid

1. Same as Step 1 in Winner take all (section 3.2.1).
2. Select for each category the classifier with the best  $F_1$  score on that category.
3. Run each classifier on each new example. If exactly one categorizer classifies a new example into a category that it "owns" then that categorizer chooses the membership of the example. If no categorizer does this then use a voting classifier approach (section 3.2.2), selecting the category with the highest  $F_1(c, a)$  sum of squares total. If multiple categorizers classify the example into a category they "own", then give each of these categorizers a vote equal to its  $F_1(c, a)$  score (sum of squares).

Note that other ways of combining precision and recall besides  $F_1$  may be used (e.g. weighting precision more than recall) in this formulation. In our experience, the variants with a more balanced weighting of precision vs. recall appear to do better than those which are unbalanced (at least for datasets where each example is categorized in one and only one class).

The next section compares these three approaches alongside each of the single classifier approaches to see which do better on typical "mixed-initiative" taxonomies.

## 4 RESULTS OF TESTING

To test our single and multi-classifier approaches we selected four data sets. The first data set was categorized into an artificial taxonomy to illustrate the usefulness of a multi-classifier approach. The next two data sets were categorized into multiple taxonomies by a domain expert using eClassifier. The fourth data set is the standard Reuters data set [10] used to evaluate text classification approaches.

## 4.1 Almaden Helpdesk Data

In our first test data set consisting of 6684 Almaden helpdesk problem tickets, we purposefully designed a taxonomy using eClassifier that clearly had categories designed with a mixture of approaches. In this example we used eClassifier to create 10 mutually exclusive problem categories, roughly equal in size. The first 5 categories were based each on a simple keyword query and the next 5 categories were based on a k-means clustering approach.

The overall results are shown in Table 1.

**Table 1: Almaden Helpdesk Results**

Algorithm	Accuracy	Std. Dev.	Confidence
Hybrid-Algorithm	96.72%	0.40%	0.25%
Winner-take-all	95.54%	0.58%	0.36%
Voting Classifier	94.44%	0.42%	0.26%
Statistical Classifier	93.64%	0.33%	0.21%
Set of Binary Decision Trees	91.25%	0.52%	0.32%
Decision Tree	90.89%	0.59%	0.36%
Naive Bayes (binary features)	90.40%	0.59%	0.37%
Centroid	89.76%	0.58%	0.36%
Rule Based Classifier	87.93%	1.28%	0.79%
Naive Bayes (numeric features)	82.28%	0.94%	0.58%

Each of the single classifier approaches failed to accurately model one or more of the categories, leading to poorer performance overall. The multi-classifier algorithms were able to choose the correct classification approach to use on each individual category, and thus achieve better results overall. Furthermore when we look at the classifier performance at the individual category level we see a much more even level of performance across all categories for the multi-classifiers.

**Table 2: Precision & Recall for each Category on Almaden Helpdesk Data (sample single classifiers)**

Class Name	Centroid	Rule Based Classifier
print	99.72%/89.58%	70.86%/100.00%
password	93.90%/84.03%	100.00%/100.00%
network	80.46%/77.51%	100.00%/100.00%
email	96.04%/76.23%	100.00%/100.00%
vm	73.95%/84.54%	100.00%/100.00%
file system	88.55%/94.41%	90.64%/53.27%
install problems	85.03%/99.38%	93.01%/95.65%
server and address problems	86.25%/92.54%	91.07%/64.15%
afs	95.77%/93.74%	86.69%/78.45%
lotus notes	79.51%/97.18%	92.09%/87.69%

**Table 3: Precision & Recall for each Category on Almaden Helpdesk Data (sample multi-classifiers)**

Class Name	Hybrid	Winner Algorithm
print	99.82%/100.00%	88.97%/100.00%
password	100.00%/100.00%	100.00%/99.85%
network	100.00%/100.00%	100.00%/99.62%
email	100.00%/100.00%	100.00%/100.00%
vm	100.00%/100.00%	98.92%/100.00%
file system	96.58%/89.49%	96.86%/89.21%
install problems	93.64%/96.38%	93.64%/95.64%
server and address problems	95.73%/91.93%	98.10%/88.28%
afs	90.85%/97.52%	98.11%/93.37%
lotus notes	94.07%/92.46%	94.19%/90.58%

We wish to stress here the advantage of having a modeling approach that works consistently well across all categories. A model with high precision and recall overall that has relatively low precision and recall for one of the smaller categories in the taxonomy may still deliver unacceptable results, depending on the importance of the poorly modeled category. In general, we found the Hybrid and Winner algorithms performed much more uniformly well on all categories in the taxonomy when compared to the other approaches. The Voting Classifier was not as effective as the other two multi-classifiers in this regard.

## 4.2 Music Industry Data Set

The music industry data set is a set of 8182 news articles found on the World Wide Web each of which describes some aspect of the music industry. From this data set, 4 taxonomies were developed using eClassifier. In each case the documents are categorized in only one category of a flat taxonomy. Each taxonomy contains a Miscellaneous category which represents those documents which were deemed not to belong in any of the predefined categories.

1. Infrastructure Management - Each category contains a relevant technology. There are 6 categories. Miscellaneous contains 23% of the data.
2. Geography - Each category represents the geographic location of a particular musical event. There are 17 categories. Miscellaneous contains 53.14% of the examples.
3. Company Organization - Each category represents the related department of a music/record company. There are 12 categories, Miscellaneous contains 2.9% of the data.
4. Company - Each category represents a company in the music industry. There are 24 categories. Miscellaneous, containing 51.06% of the data.

Each data set was split randomly into 2/3 training and 1/3 test set, 100 different times. The average accuracy, standard deviation, and confidence of these 100 trials for each taxonomy is described in the following four tables, listing the most accurate algorithms first. Note that accuracy = precision = recall in all cases since each document has one and only one correct category.

**Table 4: Music Data, Company Taxonomy Results**

Algorithm	Accuracy	Std. Dev.	Confidence
Hybrid-Algorithm	85.77%	1.03%	0.20%
Winner-take-all	84.49%	1.13%	0.22%
Voting Classifier	83.66%	0.98%	0.19%
Statistical Classifier	82.74%	0.87%	0.17%
Naive Bayes (binary features)	82.46%	0.94%	0.18%
Decision Tree	82.14%	0.91%	0.18%
Rule Based Classifier	81.98%	1.05%	0.21%
Set of Binary Decision Trees	79.01%	1.23%	0.24%
Naive Bayes (numeric features)	41.88%	1.12%	0.22%
Centroid	37.57%	1.15%	0.23%

**Table 5: Music Data, Infrastructure Mgmt. Taxonomy Results**

Algorithm	Accuracy	Std. Dev.	Confidence
Hybrid-Algorithm	77.35%	1.10%	0.22%
Winner-take-all	77.01%	1.04%	0.20%
Centroid	76.87%	0.90%	0.18%
Voting Classifier	76.17%	0.93%	0.18%
Naive Bayes (numeric features)	75.42%	1.05%	0.20%
Statistical Classifier	71.76%	0.96%	0.19%
Naive Bayes (binary features)	61.78%	1.04%	0.20%
Rule Based Classifier	60.59%	1.18%	0.23%
Decision Tree	56.28%	1.30%	0.26%
Set of Binary Decision Trees	55.99%	1.17%	0.23%

**Table 6: Music Data, Geography Taxonomy Results**

Algorithm	Accuracy	Std. Dev.	Confidence
Hybrid-Algorithm	88.38%	0.79%	0.15%
Set of Binary Decision Trees	87.61%	0.72%	0.14%
Voting Classifier	87.56%	0.75%	0.15%
Decision Tree	87.26%	0.77%	0.15%
Winner-take-all	86.88%	0.79%	0.15%
Naive Bayes (binary features)	85.88%	0.78%	0.15%
Rule Based Classifier	85.08%	0.86%	0.17%
Statistical Classifier	84.68%	0.91%	0.18%
Naive Bayes (numeric features)	41.14%	1.12%	0.22%
Centroid	33.94%	1.03%	0.20%

**Table 7: Music Data, Company organization taxonomy Results**

Algorithm	Accuracy	Std. Dev.	Confidence
Hybrid-Algorithm	80.66%	1.26%	0.25%
Winner-take-all	79.10%	1.15%	0.23%
Centroid	78.69%	0.85%	0.17%
Voting Classifier	78.09%	1.02%	0.20%
Naive Bayes (numeric features)	74.64%	1.04%	0.20%
Statistical Classifier	68.22%	1.12%	0.22%
Set of Binary Decision Trees	51.85%	1.30%	0.25%
Decision Tree	50.47%	1.16%	0.23%
Rule Based Classifier	49.29%	1.30%	0.26%
Naive Bayes (binary features)	42.83%	1.28%	0.25%

### 4.3 Automotive Industry Data Set

The automotive industry data set is a set of 7793 news articles found on the World Wide Web each of which describes some aspect of the auto industry. From this data set, 5 taxonomies were developed using eClassifier. In each case the documents are categorized in only one category of a flat taxonomy. Each taxonomy contains a Miscellaneous category which represents those documents which were deemed not to belong in any of the predefined categories.

**Table 8: Auto Taxonomy Characteristics**

Taxonomy Name	Num. Categories	Miscellaneous Size
Companies	20	33.2%
Design	7	82.3%
Geography	18	24.9%
Manufacturing	10	89.3%
Media	9	85.1%

Each data set was split randomly into 2/3 training and 1/3 test set, 100 different times. The average accuracy, standard deviation, and confidence of these 100 trials for each taxonomy is described in the following five tables, listing the most accurate algorithms first. Note that accuracy = precision = recall in all cases since each document has one and only one correct category.

**Table 9: Auto Data, Company Taxonomy Results**

Algorithm	Accuracy	Std. Dev.	Confidence
Hybrid-Algorithm	88.09%	0.76%	0.15%
Winner-take-all	86.41%	0.84%	0.16%
Voting Classifier	86.03%	0.81%	0.16%
Rule Based Classifier	85.06%	0.83%	0.16%
Decision Tree	84.79%	0.87%	0.17%
Statistical Classifier	84.60%	0.84%	0.16%
Naive Bayes (binary features)	81.58%	0.72%	0.14%
Set of Binary Decision Trees	81.56%	0.94%	0.19%
Centroid	49.04%	1.14%	0.22%
Naive Bayes (numeric features)	48.78%	1.16%	0.23%

**Table 10: Auto Data; Design Taxonomy Results**

Algorithm	Accuracy	Std. Dev.	Confidence
Rule Based Classifier	98.78%	0.30%	0.06%
Hybrid-Algorithm	98.70%	0.31%	0.06%
Winner-take-all	98.69%	0.31%	0.06%
Set of Binary Decision Trees	98.62%	0.30%	0.06%
Voting Classifier	98.57%	0.29%	0.06%
Decision Tree	98.47%	0.33%	0.06%
Statistical Classifier	97.76%	0.37%	0.07%
Naive Bayes (binary features)	87.64%	0.80%	0.16%
Naive Bayes (numeric features)	57.01%	1.33%	0.26%
Centroid	52.75%	1.44%	0.28%

**Table 11: Auto Data; Geography Taxonomy Results**

Algorithm	Accuracy	Std. Dev.	Confidence
Hybrid-Algorithm	99.23%	0.28%	0.05%
Winner-take-all	98.94%	0.45%	0.09%
Voting Classifier	97.56%	0.46%	0.09%
Decision Tree	96.92%	0.74%	0.15%
Rule Based Classifier	96.36%	0.56%	0.11%
Naive Bayes (binary features)	95.90%	0.51%	0.10%
Statistical Classifier	92.02%	0.71%	0.14%
Set of Binary Decision Trees	91.04%	1.15%	0.22%
Naive Bayes (numeric features)	34.77%	1.23%	0.24%
Centroid	29.49%	1.02%	0.20%

**Table 12: Auto Data, Manufacturing Taxonomy Results**

Algorithm	Accuracy	Std. Dev.	Confidence
Hybrid-Algorithm	96.13%	0.39%	0.08%
Winner-take-all	96.10%	0.40%	0.08%
Rule Based Classifier	96.09%	0.39%	0.08%
Statistical Classifier	96.04%	0.42%	0.08%
Voting Classifier	95.80%	0.45%	0.09%
Set of Binary Decision Trees	93.50%	0.74%	0.14%
Decision Tree	91.99%	0.73%	0.14%
Naive Bayes (binary features)	89.74%	0.77%	0.15%
Naive Bayes (numeric features)	51.61%	1.68%	0.33%
Centroid	42.45%	1.44%	0.28%

**Table 13: Auto Data, Media Taxonomy Results**

Algorithm	Accuracy	Std. Dev.	Confidence
Rule Based Classifier	94.63%	0.57%	0.11%
Hybrid-Algorithm	94.49%	0.61%	0.12%
Winner-take-all	94.46%	0.61%	0.12%
Voting Classifier	94.32%	0.62%	0.12%
Set of Binary Decision Trees	94.19%	0.59%	0.12%
Decision Tree	93.71%	0.58%	0.11%
Statistical Classifier	93.55%	0.60%	0.12%
Naive Bayes (binary features)	83.11%	0.87%	0.17%
Naive Bayes (numeric features)	47.17%	1.35%	0.26%
Centroid	42.69%	1.53%	0.30%

## 4.4 Reuters

The Reuters ModApte data set [10] used for this test contained 9603 news articles in 79 categories, with each news article classified in only one category. The average accuracy, standard deviation, and confidence of these 100 trials for each taxonomy is described in the following table, listing the most accurate algorithms first. Note that accuracy = precision = recall in all cases since each document has one and only one correct category.

**Table 14: Reuters Data, Reuters Taxonomy Results**

Algorithm	Accuracy	Std. Dev.	Confidence
Hybrid-Algorithm	85.47%	0.95%	0.19%
Statistical Classifier	84.64%	0.83%	0.16%
Winner-take-all	84.29%	1.02%	0.20%
Naive Bayes (numeric features)	83.82%	0.89%	0.17%
Voting Classifier	83.79%	0.90%	0.18%
Rule Based Classifier	71.69%	1.03%	0.20%
Naive Bayes (binary features)	70.62%	1.00%	0.20%
Centroid	70.11%	0.98%	0.19%
Set of Binary Decision Trees	68.91%	1.11%	0.22%
Decision Tree	60.48%	1.17%	0.23%

In summary, the Hybrid multi-classifier algorithm was consistently the most accurate classifier, or no worse than the most accurate classifier. Moreover, the Hybrid algorithm exhibited the greatest degree of consistency in accuracy both within the categories of individual taxonomies and among multiple taxonomies. This consistency of performance achieves our design goal of having a classifier that can accurately model mixed initiative taxonomies.

## 5 CONCLUSIONS

The Hybrid multi-classifier approach appeared to do as well or better than all other multi-classifiers and all single algorithm classifiers on all taxonomies we tested. Performance improvement is most marked where multiple strategies are employed in generating the taxonomy categories. Performance on taxonomies that are uniformly generated is no better than the best single classifier.

We believe we have shown an effective method of combining multiple classifiers to accurately model a taxonomy developed using a mixed-initiative methodology. Future work still needs to be done to study “fuzzy clustering” taxonomies, or those taxonomies that allow classifications of documents into more than one category.

## 6 ACKNOWLEDGEMENTS

The authors gratefully acknowledge Dharmendra Modha and Ray Strong for their contributions to the original design of eClassifier; Lucian V. Lita, Vikas Krishna, and Tong Zhang for providing classifier implementations; and Norm Pass and Bill Cody for initiating the eClassifier and eClassifier for Lotus Discovery Server projects.

## 7 REFERENCES

[1] Breiman, L. (1996), Bagging Predictors, *Machine Learning*, Vol. 24, No. 2, pp. 123-140.  
 [2] Breiman, L. (1998). Arcing classifiers. *The Annals of Statistics*, 26(3), 801--849.  
 [3] Cody, W., Kreulen, J., Spangler, S., Krishna, V. (2002). *The Integration of Business Intelligence and Knowledge*

*Management*. IBM Systems Journal, Vol. 41, No. 4, pp 697-713.  
 [4] Cover, Thomas M., Thomas, Joy A. (1991). *Elements of Information Theory*. Wiley-Interscience.  
 [5] Dhillon, I., Modha, D., and Spangler, S. (2002). Visualizing class structure of multidimensional data with applications. *Journal of Computational Statistics & Data Analysis* (special issue on Matrix Computations & Statistics) Vol 4:1. November 2002. pp 59-90.  
 [6] Duda, Richard O., Hart, Peter E., Stork, David E. (2001). *Pattern Classification*, 2<sup>nd</sup> Ed. Wiley-Interscience.  
 [7] Hartigan, J. A. (1975) *Clustering Algorithms*. Wiley.  
 [8] Johnson, D. E., Oles, F. J., Zhang, T., and Goetz, T., 2002. A decision-tree-based symbolic rule induction system for text categorization. *IBM Systems Journal* 41:3, pp. 428-437.  
 [9] Kittler, J., Hatef, M., Duin, R., and Matas, J., (1998) "On Combining Classifiers", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20, pp. 226-239.  
 [10] Lewis, D. Reuters-21578 text categorization test collection. <http://www.research.att.com/~lewis>. 1999.  
 [11] Manning, Christopher D., Schütze, Hinrich (2000). *Foundations of Statistical Natural Language Processing*. The MIT Press.  
 [12] McCallum, Andrew, Nigam, Kamal. *A Comparison of Event Models for Naïve Bayes Text Classification*, AAAI-98.  
 [13] Mitchell, Tom M. (1997). *Machine Learning*. McGraw-Hill.  
 [14] Rasmussen, E. (1992). Clustering algorithms. In Frakes, W. B. and Baeza-Yates, R., editors, *Information Retrieval: Data Structures and Algorithms*, pages 419-442. Prentice Hall, Englewood Cliffs, New Jersey.  
 [15] Quinlan, J.R. (1986) Induction of Decision Trees. *Machine Learning* 1 (1):81-106.  
 [16] Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 4(5):512:523.  
 [17] Salton, G. and McGill, M. J. (1983). *Introduction to Modern Retrieval*. McGraw-Hill Book Company.  
 [18] Spangler, S. and Kreulen, J. (2002). Interactive Methods for Taxonomy Editing and Validation. *Proceedings of the Conference on Information and Knowledge Mining (CIKM 2002)*.  
 [19] Ting, W. K. and Witten, I., (1997). Stacking bagged and dagged models. 367-375. *Proc. of ICML'97*. Morgan Kaufmann.  
 [20] Wolpert, D.H. (1992), Stacked Generalization, *Neural Networks*, Vol. 5, pp. 241-259, Pergamon Press.  
 [21] Zhang, T. (2002), On the Dual Formulation of Regularized Linear Systems, *Machine Learning*, Vol. 46, pp 91-129.