Distributional Clustering of Words for Text Classification

L. Douglas Baker[†‡]
www.cs.cmu.edu/~ldbapp

Andrew Kachites McCallum[‡†]
www.cs.cmu.edu/~mccallum

[†]School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[‡]Just Research
4616 Henry Street
Pittsburgh, PA 15213

Abstract This paper describes the application of Distributional Clustering [20] to document classification. This approach clusters words into groups based on the distribution of class labels associated with each word. Thus, unlike some other *unsupervised* dimensionality-reduction techniques, such as Latent Semantic Indexing, we are able to compress the feature space much more aggressively, while still maintaining high document classification accuracy.

Experimental results obtained on three real-world data sets show that we can reduce the feature dimensionality by three orders of magnitude and lose only 2% accuracy—significantly better than Latent Semantic Indexing [6], class-based clustering [1], feature selection by mutual information [23], or Markov-blanket-based feature selection [13]. We also show that less aggressive clustering sometimes results in improved classification accuracy over classification without clustering.

1 Introduction

The popularity of the Internet has caused an exponential increase in the amount of on-line text and in the number of people who create and use this text. As the amount of documents and number of users rise, automatic document categorization becomes an increasingly important tool for helping people organize this vast amount of data. Statistical document classification algorithms have been applied to categorizing newsfeeds [10], classifying Web pages [4], sorting electronic mail [17] and learning the interests of users [14].

In this paper we cluster words into groups specifically for the benefit of document classification. While much study has been devoted to word clustering for language modeling and word co-occurrence [1, 20], little work has been done on word clustering for document classification. The underlying clustering method we apply is Distributional Clustering [20]—an information-theoretic approach that has shown good performance in language modeling.

Word clustering methods create new, reduced-size event spaces by joining similar words into groups. Distributional Clustering does so by joining words that induce similar probability distributions among the target features that co-occur with the words in question. The reasoning can be understood intuitively as follows. If two different words "vote" similarly among the possible answers in the task at hand, then joining those two words, (and causing each of them to vote according to the average of their individual votes), will not hurt performance. In fact, performance may be increased by clustering when training data is sparse, because averaging statistics for similar words can result in more robust estimates. Similarity between distributions is measured by a variant of Kullback-Leibler divergence.

In document classification, the target concept is the class label. Thus, in this paper, we measure word similarity by the distributions of class labels associated with the words in question. For example, consider classifying documents about sports into categories by individual sport (*e.g.* baseball, hockey, tennis). In the training data, the words puck and goalie may occur only in the in the hockey class. Thus, for the purposes of this classification task, there is no need to distinguish between them. All words that are strongly indicative of the hockey class will be clustered together. Furthermore, Distributional Clustering will sensibly cluster words that are indicative of more than one class. The word team may occur with equal frequency in classes baseball and hockey; the word teammates may also occur equally in just those two classes. These words could be merged. The following section gives a more detailed example of this idea.

1.1 Benefits of Word Clustering

There are three key benefits of using word clustering: (1) useful semantic word clusterings, (2) higher classification accuracy and (3) smaller classification models. The second two reasons are shared with feature selection, and thus feature clustering can be seen as a complement or alternative to feature selection. Feature clustering is better at reducing the number of redundant features, whereas feature selection is better at removing detrimental, noisy features.

Word clustering can provide useful semantically-related groups of words—in effect, an automatically generated thesaurus. An interesting aspect of the semantic groups produced by our algorithm is that they depend on the class labels assigned to the documents. This reflects the fact that some words that are synonyms in one context are not in another. The clusters are based on a supervised machine learning paradigm, and are task-focussed. The usefulness of an automatically-generated thesaurus is difficult to evaluate, however, and is not the subject of this paper.

Second, word clustering can result in higher classification accuracy, as described above. This will be further discussed in later sections.

Third, the size of the classification model can be greatly reduced because separate sets of parameters for many words are replaced with a single set of parameters for a word cluster. Our results include successful size reductions by several orders of magnitude, from, for example, 50,000 to 50.

We argue that successful use of small-footprint text classification models becomes increasingly important with the wide-spread and popular use of text classification. For example, large population, high-volume routing tasks, as required by companies such as WiseWire [21], can involve text categorization with hundreds of thousands of class labels on a stream of documents arriving a rate of hundreds per second—the use of word clustering can avoid the need for machines with many gigabytes of memory. At the other end of the scale, consider hand-held computers that automatically organize their data by text classification—word clustering can allow classification models to fit in these restricted-memory machines. As text classification spreads beyond servers and research machines, and onto home computers, secretaries machines and network computers, reducing the number of features for which statistics must be maintained becomes more important.

Furthermore, we maintain that the dramatically-reduced dimensionality allows the use of more complex algorithms that would not have been feasible with the 50,000 original dimensions.

1.2 Contributions

This paper introduces the application of Distributional Clustering to document classification with a naive Bayes classifier. We derive naive Bayes, explain its assumptions, and discuss its close ties to cross-entropy. We describe Distributional Clustering and show how Distributional Clustering clusters features so as to minimize errors in cross entropy.

We present experimental results on three real-world text corpora, including newswire stories, UseNet articles and Web pages. Results show that Distributional Clustering can reduce the feature dimensionality by three orders of magnitude, and lose only 2% accuracy. This performance is significantly better than class-based clustering using mutual information [1], clustering by Latent Semantic Indexing [6], feature selection by information gain [23] and feature selection by Markov-blanket [13]. On one of the data sets we show that clustering increases classification accuracy. We hypothesize why this did not happen in more cases, and discuss possible future improvements.

2 Clustering Words by Class Distributions

This section introduces our probabilistic framework, derives the naive Bayes classifier and explains Distributional Clustering.

Like previous work in Distributional Clustering [20] we use a form of "Kullback-Leibler divergence to the mean." Unlike their work, we use a weighted average instead of a simple average, we use hard clustering instead of soft, and we use a greedy agglomerative method instead of a divisive entropy-based method.

2.1 Probabilistic Framework and Naive Bayes

We approach text classification in a Bayesian learning framework. We assume that the text data was generated by a parametric model, and use training data to calculate Bayes optimal estimates of the model parameters. Then, equipped with these estimates, we classify new test documents by using Bayes rule to turn the generative model around and calculate the probability that a class would have generated the test document in question. Classification then becomes a simple matter of selecting the most probable class.

The training data consists of a set documents, $\mathcal{D} = \{d_1, d_2, ..., d_n\}$, where each document is labeled with a class from a set of classes $\mathcal{C} = \{c_1, c_2, ..., c_m\}$.

We assume that the data is generated by a mixture model, (parameterized by θ), with a one-to-one correspondence between mixture model components and classes. Thus, the data generation procedure for a document, d_i, can be understood as (1) selecting a class according to the class priors, $\mathrm{P}(c_j|\theta)$, then (2) having the corresponding mixture generate a document according to its own parameters, with distribution $\mathrm{P}(d_i|c_j;\theta)$. The probability of generating document d_i independent of its class is thus a sum of total probability over all mixture components:

$$\mathrm{P}(d_i|\theta) = \sum_{j=1}^{|\mathcal{C}|} \mathrm{P}(c_j|\theta)\mathrm{P}(d_i|c_j;\theta) \qquad (1)$$

Now we expand our notion of how a document is generated by an individual mixture component. In this paper we approach document generation as language modeling. Thus, unlike some notions of naive Bayes in which documents are 'events' and the words in the document are 'attributes' of that event (a multi-variate Bernoulli model), we instead consider words to be 'events' (a multinomial model) [19]. Multinomial naive Bayes has been shown to out-perform the multi-variate Bernoulli on many real-world corpora [19]. We say a document is comprised of an ordered sequence of word events, and write d_{ik} for the word in position k of document d_i. Given this, we can expand the expression for the probability of a document given class c_j, $\mathrm{P}(d_i|c_j;\theta)$, saying that the probability of the sequence is equal the product of the probabilities of the events in the sequence, also remembering that each event may depend on the events that preceded it:

$$\mathrm{P}(d_i|c_j;\theta) = \mathrm{P}(|d_i|) \prod_{k=1}^{|d_i|} \mathrm{P}(d_{ik}|c_j;\theta;d_{iq}, q < k), \qquad (2)$$

where we assume that document length, $|d_i|$, is distributed independently of class.

Next we make the naive Bayes assumption: we assume that the probability of each word event in a document is independent of the word's context, and furthermore, independent of its position in the document. Note that this is the same as saying we use a uni-gram language model. Each word event is drawn from a multinomial distribution over the set of all words in the vocabulary, V. We write w_t for the t-th word in V, and given that $d_{ik} = w_t$, we can express the naive Bayes assumption by writing

$$\mathrm{P}(d_{ik}|c_j;\theta;d_{iq}, q < k) = \mathrm{P}(w_t|c_j;\theta). \qquad (3)$$

Given the assumption about one-to-one correspondence between mixture model components and classes, and the naive Bayes assumption, the mixture model is composed of disjoint sets of parameters for each class c_j, and the parameter set for each class is composed of probabilities for each word, $\theta_{w_t|c_j} \equiv \mathrm{P}(w_t|c_j;\theta)$. The only other parameters in the model are the class prior probabilities, written $\theta_{c_j} \equiv \mathrm{P}(c_j|\theta)$.

We can now calculate estimates of θ, (written $\hat{\theta}$), of these parameters from the training data. The $\theta_{w_t|c_j}$ estimates consist of straightforward counting of events, supplemented by 'smoothing' with a Laplacean prior that primes each estimate with a count of one. Define $N(w_t, d_i)$ to be the count of the number of times word w_t occurs in document d_i, and define $\mathrm{P}(c_j|d_i) = \{0,1\}$ as given by the document's class label, then the estimate of the probability of word w_t in class c_j is

$$\hat{\theta}_{w_t|c_j} \equiv \frac{1 + \sum_{i=1}^{|\mathcal{D}|} N(w_t, d_i)\mathrm{P}(c_j|d_i)}{|V| + \sum_{s=1}^{|V|}\sum_{i=1}^{|\mathcal{D}|} N(w_s, d_i)\mathrm{P}(c_j|d_i)}. \quad (4)$$

The class prior parameters, θ_{c_j}, are estimated by maximum-likelihood estimate—the fraction of documents in each class in the corpus:

$$\hat{\theta}_{c_j} \equiv \frac{\sum_{i=1}^{|\mathcal{D}|} \mathrm{P}(c_j|d_i)}{|\mathcal{D}|}. \quad (5)$$

Given estimates of these parameters calculated from the training documents, classification can be performed on test documents by calculating the probability of each class given the evidence of the test document, and selecting the class with the highest probability. We formulate this by first applying Bayes rule, and then substituting for $\mathrm{P}(d_i|c_j;\theta)$ and $\mathrm{P}(d_i|\theta)$ using equations 1, 2 and 3.

$$\begin{aligned} \mathrm{P}(c_j|d_i;\hat{\theta}) &= \frac{\mathrm{P}(c_j|\hat{\theta})\mathrm{P}(d_i|c_j;\hat{\theta})}{\mathrm{P}(d_i|\hat{\theta})} \quad (6) \\ &= \frac{\mathrm{P}(c_j|\hat{\theta})\prod_{k=1}^{|d_i|}\mathrm{P}(w_{d_{ik}}|c_j;\hat{\theta})}{\sum_{r=1}^{|C|}\mathrm{P}(c_r|\hat{\theta})\prod_{k=1}^{|d_i|}\mathrm{P}(w_{d_{ik}}|c_r;\hat{\theta})} \end{aligned}$$

Both the mixture model and word independence assumptions are violated in practice with real-world data; however, there is empirical evidence that naive Bayes often performs well in spite of these violations [16, 23, 10, 4]. Friedman and Domingos and Pazzani discuss why the violation of the word independence assumption sometimes does little damage to classification accuracy [9, 7].

2.2 Measuring Word Similarity for Distributional Clustering

Now we address the question of how to cluster words in the context of our generative model and naive Bayes.

Word clustering algorithms define a similarity measure between words, and collapse similar word into single events that no longer distinguish among their constituent words. Typically, the parameters of the cluster become the weighted average of the parameters of its constituent words.

Consider, for example, the random variable over classes, C, and its distribution given a particular word, w_t. We write this distribution $\mathrm{P}(C|w_t)$. When words w_t and w_s are clustered together, the new distribution is the weighted average of the individual distributions

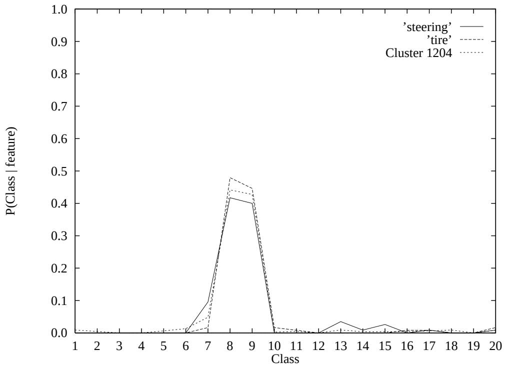

Figure 1: In the 20 Newsgroups data set, class probability distributions for words tire and steering and for their combination

$$\begin{aligned} \mathrm{P}(C|w_t \vee w_s) = & \frac{\mathrm{P}(w_t)}{\mathrm{P}(w_t)+\mathrm{P}(w_s)}\mathrm{P}(C|w_t) \\ & + \frac{\mathrm{P}(w_s)}{\mathrm{P}(w_t)+\mathrm{P}(w_s)}\mathrm{P}(C|w_s) \quad (7) \end{aligned}$$

Distributional Clustering differs from some other machine learning approaches to similarity metrics (*e.g.* k-nearest neighbor [2]) in that it measures similarity based on the target variable that it is trying to estimate for the task at hand, not the other "input" attributes. More specifically, it examines the the probability distribution over the target variable induced by the different events to be clustered, and measures similarity between the events as similarity between the induced target variable distributions.

In the context of document classification, the target variable for the task at hand is the class label. Distributional clustering thus measures the similarity between two words w_t and w_s as the similarity between the class variable distributions they induce: $\mathrm{P}(C|w_t)$ and $\mathrm{P}(C|w_s)$.

An example of these class distributions in data from the 20 Newsgroups corpus is shown in figure 1. Consider the line for the word tire. The horizontal axis has ticks for the (order-irrelevant) list of class labels. The vertical axis indicates the probability of each class given the word tire, and the shape of the line shows the probability distribution over classes given tire, $\mathrm{P}(C|\text{tire})$. The graph indicates that the word occurs mostly in classes 8 (rec.autos) and 9 (rec.motorcycles), and only mildly in other classes.

Remembering the classification task, the graph can also be interpreted as a picture of how much the word tire "votes" for each of the classes whenever it occurs in a test document. The line thus shows the essence of how tire contributes to the classification algorithm.

In the same figure, notice the line for the word steering. The shape of its distribution is quite similar to that of tire. The third line, labeled Cluster 1204, shows the class distribution from a cluster containing both words, and since the words have similar distributions, the distribution of the cluster is similar to each. Thus, if the word tire voted according to the distribution of Cluster 1204 instead of according to the tire class distribution, it would not be voting much differently, and the final classification scores would not be very far off. (Table 1 shows

Cluster 1204 (Motorcycle and Automobile)	Cluster 1287 (Motorcycle)	Cluster 1473 (Baseball and Hockey)
honda	bike	season
rear	biker	players
wheel	yamaha	scored
steering	harley	rookie
tire	riders	philly
suspension	bikers	roster
throttle	harleys	announcers
mechanic	countersteering	coaches
rust	wheelie	leagues

Table 1: Lists of highest probability words from three clusters (out of 1200) created by Distributional Clustering on the 20 Newsgroups dataset.

other words that also fall in cluster 1204, and two other clusters.)

This example expresses the core intuition behind Distributional Clustering for document classification: the class distributions, $\mathrm{P}(C|w)$, express how individual words contribute to classification, and we cluster words so as to preserve the shape of these distributions.

Now we turn to the question of how exactly to measure the difference between two probability distributions. Kullback-Leibler divergence is an information-theoretic measure that does just this.

The KL divergence between the class distributions induced by w_t and w_s is written $D(\mathrm{P}(C|w_t)||\mathrm{P}(C|w_s))$, and is defined

$$\sum_{j=1}^{|\mathcal{C}|} \mathrm{P}(c_j|w_t) \log \left(\frac{\mathrm{P}(c_j|w_t)}{\mathrm{P}(c_j|w_s)} \right) \tag{8}$$

In the context of information theory, KL divergence can be intuitively understood as a measure of inefficiency that occurs when messages are sent according to one distribution, $(\mathrm{P}(C|w_t))$, but encoded with a code that is optimal for a different distribution, $(\mathrm{P}(C|w_s))$.

KL divergence has some odd properties. It is not symmetric, and it is infinite when an event with non-zero probability in the first distribution has zero probability in the second distribution.

Thus, in Distributional Clustering we use a related measure that does not have these problems. It is the *average* of the KL divergence of each distribution to their mean distribution, called "KL divergence to the mean." Unlike earlier work [20] we use a weighted average instead of a simple average.

$$\begin{aligned} &\mathrm{P}(w_t) \cdot D(\mathrm{P}(C|w_t)||\mathrm{P}(C|w_t \vee w_s)) \\ +\ &\mathrm{P}(w_s) \cdot D(\mathrm{P}(C|w_s)||\mathrm{P}(C|w_t \vee w_s)) \end{aligned} \tag{9}$$

This metric can be understood as the expected amount of inefficiency incurred if, instead of compressing two distributions optimally with their own code, we use the code that would be optimal for their mean. This explanation makes it clear why this metric is such a good fit for a clustering distance metric. It describes perfectly the effect of clustering—events that formerly generated their own individual statistics, now, once clustered, generate combined statistics.

- Sort the vocabulary by mutual information with the class variable.
- Initialize the M clusters as singletons with the top M words.
- Loop until all words have been put into one of the M clusters:
    - Merge the two clusters which are most similar (Equation 9), resulting in $M-1$ clusters.
    - Create a new cluster consisting of the next word from the sorted list, restoring the number of clusters to M.

Table 2: The Algorithm

2.3 Distributional Clustering Minimizes Error in Naive Bayes Score

Classification by naive Bayes is intimately related to information theory. It can easily be shown that, assuming a uniform class prior, choosing the most probable class by naive Bayes is identical to choosing the class that has the minimal cross entropy with the test document.

Beginning with the naive Bayes classification formula in equation 6, assume uniform class prior by dropping $\mathrm{P}(c_j|\theta)$, then make a series of transformations that do not change which class gets the highest score: (1) drop the denominator (which is a constant over all classes), (2) transform the product over word position in the document into an equivalent expression with a product over words in the vocabulary, (3) take the log of the entire expression, and finally (4) divide by document length, $|d_i|$. This results in

$$-\sum_{t=1}^{|V|} \mathrm{P}(w_t|d_i) \cdot \log \left(\mathrm{P}(w_t|c_j;\hat{\theta}) \right) \tag{10}$$

which is precisely the expression of the cross entropy between the distribution of words in the document, $\mathrm{P}(W|d_i)$ and the distribution of words in the class $\mathrm{P}(W|c_j)$ [3], where W is a random variable over words.

Using this cross entropy as a representative of the naive Bayes score for each class, we can express the "error in naive Bayes score incurred by clustering two words." It is the difference between (1) the cross entropies before two words are joined and (2) cross entropy after two words are joined.

Simple algebraic manipulation of this error expression results exactly in equation 9, the weighted sum of two KL divergences to the mean. Thus, we conclude, when words are clustered according to this similarity metric, increase in the "error in naive Bayes scores" is minimized.

2.4 Clustering Algorithm

Now we address the question of how to use the similarity metric to form clusters. We create clusters with deterministic word membership using a simple, greedy agglomerative approach that works well in practice, while scaling extremely efficiently to large vocabulary sizes. Instead of comparing the similarity of all possible pairs of words, (a daunting $O(|V|^2)$ operation), we consider all pairs of a much smaller subset, of size M, where M is the final number of clusters desired. At all stages, the algorithm has not more than M clusters. The clusters are initialized with the M words that have highest mutual information with the class variable. The most similar two clusters are joined, then the next word is added as a singleton cluster to bring the total number of clusters back up to M. Table 2 contains an outline of our algorithm.

In contrast, probabilistic "soft" clustering, as used in previous Distributional Clustering work [20, 15], is more formally rigorous, and allows the clustering to be less greedy. However, we avoid the costly EM-style update procedure that must be used to find a stable configuration of the cluster centroids and the cluster membership probabilities.

3 Related Work

Distributional Clustering has been used [20, 5, 15] to address the problem of sparse data in building statistical language models for natural language processing, but it has not previously been applied to document classification. We have used larger data sets with more prevalent sparseness and fewer class labels.

ChiMerge [12] uses a form of Distributional Clustering to discretize numeric attributes for subsequent classification. It is an agglomerative, hard clustering algorithm that uses the χ^2 statistic as the the similarity metric. We have also tried χ^2 in our experiments and found that the KL divergence average yields better performance.

Chi2 [18] is an extension of ChiMerge for use as a feature selector of numeric attributes. Liu and Setiono observe that if all the values of any attribute are clustered together, then that value is irrelevant to the classification task and can be removed.

Class-based clustering [1] uses an agglomerative, hard clustering algorithm where the clustering criterion is designed to maximize the overall average mutual information between clusters and the class variable. This criterion implicitly measures the similarity between the distributions $\mathrm{P}(C|w_t)$ and $\mathrm{P}(C|w_s)$ as well as the similarity between the distributions $\mathrm{P}(C|\neg w_t)$ and $\mathrm{P}(C|\neg w_s)$ for two features. We find that average mutual information is not a good clustering criterion for text classification with a multinomial naive Bayes model because it considers the information about the class label that is indicated by both the presence and the absence of a word in a document, whereas the classifier only considers those words that are present in a document. A clustering criterion based only on class distributions given words that do appear is better suited to a multinomial naive Bayes classifier. We argue that KL-divergence is a good choice among such criteria.

In that clustering reduces the dimensionality of feature space, our work can be seen as a form of feature selection, although we do not actually remove any features. A previous study found feature selection by mutual information with the class label to be the best for text, among several common, time/space-efficient methods [23].

However, mutual information between words and classes does not capture dependencies between words. Koller and Sahami present a Markov-blanket-based feature selection algorithm that aims to address exactly this [13]. Their technique is based on the same principles as Distributional Clustering—it examines $\mathrm{P}(C|w_t)$, and tries to preserve the proper C distribution.

Latent Semantic Indexing [6] is an unsupervised dimensionality reduction technique for information retrieval that explicitly accounts for the dependencies between words. In brief, it applies Principle Component Analysis (PCA) to documents represented as word vectors. Dumais applies it to text classification [8] by representing each class as a centroid, which is the vector sum of all the feature vectors of all the documents in that class. A new document is labelled with the class of the centroid to which its feature vector is closest, as measured by the cosine-similarity between the two vectors.

The Linear Least Squares Fit (LLSF) method [22] is another classification algorithm based on PCA, which is equivalent to Dumais' use of LSI for classification except that LLSF uses the dot-product to compute similarity instead of the cosine and is thus sensitive to the length of the two vectors being compared.

4 Experimental Results

This section provides empirical evidence that Distributional Clustering is able to aggressively reduce the number of features while maintaining high classification accuracy. At equal feature dimensionality, it achieves significantly higher accuracy than four other feature clustering and feature selection algorithms: supervised Latent Semantic Indexing [8], class-based clustering [1], feature selection by mutual information with the class variable [23] and feature selection by a Markov-blanket method [13].

The 20 Newsgroups data set, collected by Ken Lang, contains about 20,000 articles evenly divided among 20 UseNet discussion groups [10]. Several of the topic classes are quite confusable: four of them are about computers; three discuss religion. In tokenizing the data we skipped all UseNet headers, used a stoplist, but did not stem because we found it to hurt accuracy. The resulting vocabulary, after removing words that occur only once, has 62258 words.

The 'ModApte' test/train split of the Reuters-21578 data set (http://www.research.att.com/~lewis) contains 9603 training documents and 3299 testing documents, gathered from the Reuters newswire. There are 135 overlapping topic categories, but we used only those 90 for which there exists at least one training and one testing document. The number of training documents per class varies from 1 to nearly 4000. The largest 10 classes contain 77.5% of the documents; 28 classes have fewer than 10 training documents. We removed all words that had less than three occurrences. The resulting vocabulary has 16177 words.

We gathered the entirety of the *Yahoo!* 'Science' hierarchy in July 1997. The 6294 web pages are divided into 41 disjoint classes by chopping the hierarchy two levels deep. After removing stopwords and words that occur only once, the vocabulary contains 44383 words.

Figure 2 (top) shows classification accuracy results on the 20 Newsgroups data set for four of the five methods considered. The horizontal axis indicates the number of features that were used in the classification model, and the vertical axis the percentage of the test documents that were correctly classified. Results are averages of 5-20 trials of randomized 1/3–2/3 test-train splits, except in the case of supervised LSI, which uses only 1/3 for training instead, because training was too slow on 13000 articles. Thus, for comparison, we show two performance curves for Distributional Clustering, one with 2/3 training and one with 1/3 training. Notice that with only 50 features (a reduction of more than three orders of magni-

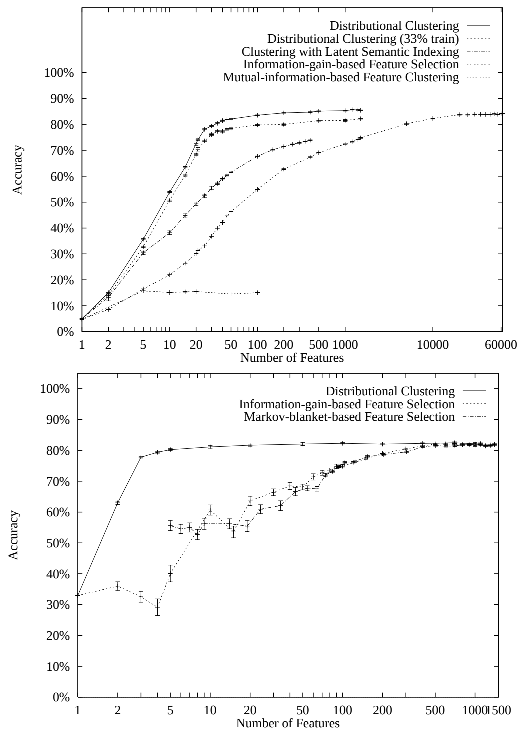

Figure 2: **Top:** Classification accuracy on the 20 Newsgroups data set, for varying numbers of features. The two highest curves are both for Distributional Clustering. The (extremely tight) bars on each data point show standard error. **Bottom:** Classification accuracy on a subset of the 20 Newsgroups data set. Temporary technical problems prevent the curve for the Markov-blanket feature selector from continuing under five features.

tude) Distributional Clustering achieves 82.1% accuracy, only 2% lower than at the full vocabulary. In comparison, supervised LSI reaches only 60% accuracy. Feature selection by mutual information and class-based clustering are lower still with 46.3% and 14.5% respectively.

Furthermore, note that Distributional Clustering actually provides a small, but statistically-significant increase over the best performance possible without clustering. The highest accuracy without clustering is 84.2%, with the full vocabulary. All of our clustering results with more than 400 features are higher than this; the best, with 1200 clusters, is 85.7%. The increased performance indicates that Distributional Clustering is providing slightly more accurate estimates of $\theta_{w_t|c_j}$. Supervised LSI never resulted in higher accuracy than the raw feature set on this data.

The Markov blanket feature selector is missing from this graph because, due to the memory and CPU requirements of the algorithm, we were not able to run it on the full data set. The bottom graph in Figure 2 shows results on a corpus consisting of only the three `talk.politics.*` classes of the 20 Newsgroups data set. This reduced data set has 3000 documents, and after removing words occurring in fewer than 50 documents, a 1407 word vocabulary.

The performance of Distributional Clustering on this data set is striking. Not only is it consistently better than the other techniques, but with only three features it maintains accuracy near 80% while the other techniques fall into the 50's and 40's. Examination of the three features show clusters indicative of each of the three classes. The Markov-blanket feature selector sometimes performed slightly better than feature selection by mutual information, but mostly performed about the same or worse. We believe Distributional Clustering performs better than feature selection because merging preserves information instead of discarding it. Some features that are infrequent, but useful when they do occur, get removed by the feature selector; feature merging keeps them.

Clustering with LSI also has the advantage that it combines information rather than discarding it. And, indeed the top graph shows LSI out-performing the mutual-information-based feature selector. However, the initial dimensionality reduction in LSI is unsupervised, whereas Distributional Clustering is supervised. Supervised techniques can take advantage of the class labels in order to concentrate their efforts on the specific task at hand. We believe this difference explains the accuracy increase of Distributional Clustering over LSI in the top graph. Linear Discriminant Analysis [11] is a supervised technique similar to LSI which we feel may work well for text classification, although we have not yet experimented with this technique.

Since LSI takes advantage of word co-occurrences, we thought that perhaps the traditional LSI classification method may not put LSI in its best light. (The traditional method classifies test documents by measuring cosine-similarity to a class centroid. The class centroid is an average of all the training documents in the class, and thus, like naive Bayes, loses document boundaries and word co-occurrence statistics.) We tested this hypothesis by replacing the centroid distance component with a nearest neighbor classifier that measures distances to all the individual training documents. The change did indeed increase LSI's performance from 73.7% to 74.9% at 400 features, but still did not beat Distributional Clustering's naive Bayes performance of 80.0%. Note also that nearest neighbor is more computationally expensive than centroid methods.

Of all the techniques in this comparison, class-based clustering performed worst. As discussed in the previous section, this technique is not a good match for classification with a multinomial naive Bayes model.

On the 20 Newsgroups data set, wall clock training times for the algorithms are: our Distributional Clustering 7.5 minutes, LSI 23 minutes, Markov-blanket feature selection 10 hours, mutual information feature selection 30 seconds.

Figure 3 shows classification accuracy results on Reuters-21578. Again, when the number of features is small, Distributional Clustering outperforms the other methods. In this data set a document can be labelled with multiple classes. A prediction for a test document is considered correct if it is any one of the given classes.

Figure 4 shows classification accuracy results on the *Yahoo!* data set. Unlike the other two data sets, here feature selection improves performance significantly. With all features, naive Bayes gets 52.9% accuracy; naive Bayes obtains 66.4% by using just the 500 features that have highest mutual information with the class label. This is a case in which "losing information" is beneficial, because the data are so noisy that the information hurts more than it helps.

Distributional Clustering, when used as a substitute for feature selection (*i.e.* clustering with all the words), does provide some benefit over the raw feature space (see the "all words" line in figure 4), however, it gets even better performance if it begins clustering with only the

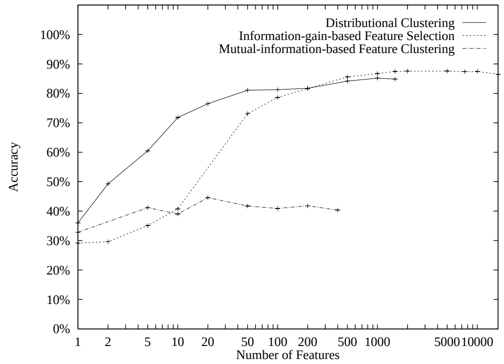

Figure 3: Classification accuracy on the Reuters-21578 data set. Computational constraints prevented us from getting results with LSI and Markov-blanket feature selection in time for the submission.

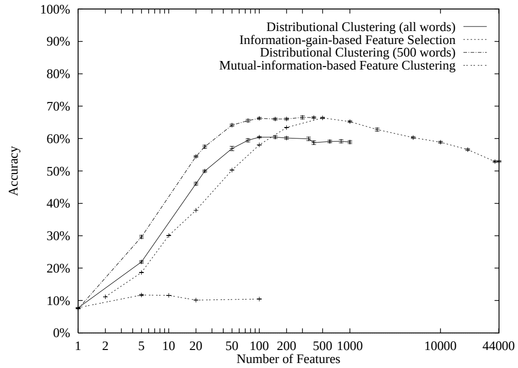

Figure 4: Classification accuracy on the Yahoo! data set, averaged over 10 runs. The error bars on each data point show standard error.

500 selected features, (the "500 words" line). This result indicates that Distributional Clustering was somewhat able to overcome noise by clustering, but further suggests that there is a place for feature-selection–feature-clustering combinations. Principled approaches to combinations of feature selection and feature merging will be a topic of future work.

5 Discussion

We have shown that Distributional Clustering is an effective technique for reducing the number of features needed for text classification. We are able to reduce the feature space by one to three orders of magnitude while losing only a few percent in classification accuracy. This result is important because, as the use of text classification becomes more widespread, and its application more diverse, the size of classification models is of increasing concern. Furthermore, the reduced dimensionality will allow the application of more complex methods.

We found that Distributional Clustering is better than feature selection at preserving the information contained in redundant features. It allows the size of the model to be reduced much more aggressively while maintaining good performance. However, it is still susceptible to detrimental features.

Earlier work with Distributional Clustering [20, 15] shows that Distributional Clustering addresses the sparse data problem (improving what were previously detrimental features). We also observed a small increase in classification accuracy, but this happened only on the one data set with the most, and most evenly distributed, data. We are not surprised that Distributional Clustering does not address the sparse data problem in more of our experiments because it clusters words based on the same estimate that affects performance. If we have a bad estimate of $\mathrm{P}(C|w_i)$ to begin with, our clustering criterion is strongly biased toward preserving that distribution, so that we will not overcome our lack of data. We hypothesize that previous work in Distributional Clustering saw sparse data improvements because (1) their data was not as sparse as ours (2) they had more target variable values, thus a larger number of "correct" $\mathrm{P}(c_j|w_i)$ values on which to pattern-match in order to fill in a "bad" $\mathrm{P}(c_j|w_i)$ estimate. We are currently investigating ways to augment Distributional Clustering to address this deficiency.

We also plan to look at techniques for sensibly combining feature clustering and feature selection to take advantage of the strengths of both, and to overcome the need for specifying in advance the number of clusters to create or features to remove.

As previously mentioned, LSI is an unsupervised dimensionality reduction technique based on the Singular Value Decomposition of a term-document matrix. The underlying technique in LSI is to find an orthonormal basis for the term-document space for which the axes lie along the dimensions of maximum variance. Linear Discriminant Analysis [11] is a related technique which instead attempts to find a basis such that the distance between the means of the members of each class is maximized while the variance within each class is minimized. We plan to investigate its use for text classification.

Acknowledgments

We are grateful to Mehran Sahami for his Markov-blanket feature selection code, as well as to Susan Dumais for the LSI code and her friendly and responsive help making it run on our system. Thorsten Joachims provided useful advice on the formatting of the Reuters data. Keiko Hasegawa graciously answered questions regarding Linear Discriminant Analysis. We thank Yahoo for permission to use their data.

References

[1] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[2] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[3] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* John Wiley, 1991.

[4] Mark Craven, Daniel DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Sean Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[5] Ido Dagan, Fernando Pereira, and Lillian Lee. Similarity-based estimation of word cooccurrence

probabilities. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, 1994.

[6] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[7] P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of the simple bayesian classifier. *Machine Learning*, 29:103–130, 1997.

[8] Susan T. Dumais. Using LSI for information filtering: TREC-3 experiments. Technical Report 500-225, National Institute of Standards and Technology, 1995.

[9] Jerome H. Friedman. On bias, variance, 0/1 - loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1:55–77, 1997.

[10] Thorsten Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *International Conference on Machine Learning (ICML)*, 1997.

[11] J. D. Jobson. *Applied Multivariate Data Analysis – Volume II: Categorical and Multivariate Methods*. Springer Verlag, 1992.

[12] R. Kerber. Chimerge: Discretization of numeric attributes. In *Proceedings of Tenth National Conference on Artificial Intelligence (AAAI-92)*, 1992.

[13] D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of Thirteenth International Conference on Machine Learning (ICML-96)*, 1996.

[14] Ken Lang. Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning (ICML)*, pages 331–339, 1995.

[15] Lillian Lee. *Similarity-Based Approaches to Natural Language Processing*. PhD thesis, Harvard University, 1997. (also Technical Report TR-11-97).

[16] David Lewis and Marc Ringuette. A comparison of two learning algorithms for text categorization. In *Third Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, 1994.

[17] David D. Lewis and Kimberly A. Knowles. Threading electronic mail: A preliminary study. *Information Processing and Management*, 33(2):209–217, 1997.

[18] H. Liu and R. Setiono. Chi2: Feature selection and discretization of numeric attributes. In *Proceedings of 7th IEEE Int'l Conference on Tools with Artificial Intelligence*, 1995.

[19] Andrew McCallum and Kamal Nigam. A comparison of event models for naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998. http://www.cs.cmu.edu/~mccallum.

[20] Fernando Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of english words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 183–90, 1993.

[21] WiseWire. http://www.wisewire.com.

[22] Yiming Yang. Noise reduction in a statistical approach to text categorization. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95)*, pages 256–263, 1995.

[23] Yiming Yang and Jan Pederson. Feature selection in statistical learning of text categorization. In *ICML-97*, pages 412–420, 1997.