Using Density Estimation to Improve Text Categorization

Carl Sable, Kathy McKeown, and Vasileios Hatzivassiloglou
Department of Computer Science
450 Computer Science Building Columbia University
1214 Amsterdam Avenue
New York, NY 10027
{sable,kathy,vh}@cs.columbia.edu

ABSTRACT

This paper explores the use of a statistical technique known as density estimation to potentially improve the results of text categorization systems which label documents by computing similarities between documents and categories. In addition to potentially improving a system's overall accuracy, density estimation converts similarity scores to probabilities. These probabilities provide confidence measures for a system's predictions which are easily interpretable and could potentially help to combine results of various systems. We discuss the results of three complete experiments on three separate data sets applying density estimation to the results of a TF*IDF/Rocchio system, and we compare these results to those of many competing approaches.

Keywords

density estimation, text categorization

1. INTRODUCTION

Text categorization can help organize documents to allow for better browsing and search capabilities. In this paper, we present our research on the use of a statistical technique known as density estimation [16] to potentially improve the results of certain text categorization methods. The methods which are eligible are those that label documents by computing a similarity measure (or other score, such as a probability estimate) for every document/category pair. These methods include Rocchio/TF*IDF, Naive Bayes, K-Nearest Neighbor, certain implementations of support vector machines, and many other commonly used techniques. Density estimation converts vectors of such similarity measures to probabilities of category membership. These probabilities provide confidence measures of systems' predictions, and we will show, through experiments, that the technique also often improves the accuracy of a system.

In a previous paper [12], we have presented research on categorization algorithms applied to the categorization of images using associated text as well as categorization of text alone. Our system applies a Rocchio-based method involving TF*IDF text similarity measures [13, 14] in conjunction with novel features, such as consideration of parts of speech and different spans of text. Standard TF*IDF/Rocchio computes a similarity measure for every document/category pair and makes its decisions based on these measures (e.g. by choosing the category with the highest score). The Rocchio approach is often used as a baseline for text categorization [4, 8, 15, 19]. In recent years, advanced methods such as support vector machines (SVMs) [5] have achieved significantly better results.

We will explain how density estimation can be applied to the results of systems such as ours to convert numerical similarity scores for categories to probabilities of membership in each category by estimating the proportion of documents with similar scores in the training set that fall into the category. A Rocchio-based system is an ideal type of system to which density estimation can be applied because it computes similarity scores for categories that do not already have any intrinsic meaning. We will show that density estimation, in addition to providing confidence measures for predictions, improves the accuracy of our system in the majority of cases, boosting results to surpass those of several advanced methods. Density estimation is a technique which can be applied to the results of any system which computes, for every test document, a score for every category.

2. RELATED WORK

Our approach to categorization stems from a long line of work in measuring text similarity in both information retrieval and categorization contexts. Salton and Buckley [14] generalized Rocchio's method for relevance feedback and applied it to text categorization. Two other common methods used for text categorization are Naive Bayes [7] and K-Nearest Neighbor (kNN) [20]. Recently, many researchers have attempted to apply more advanced methods to various text categorization tasks. These include maximum entropy [10], Widrow-Hoff and exponentiated gradient [8], boosting [15], and support vector machines [5].

Joachims [4] implemented a probabilistic version of a Rocchio-based text categorization system that uses TF*IDF representations of documents and categories to compute probabilities of categories given a test document. The method which that system uses appears quite different from standard Rocchio, but Joachims shows that the two are similar and that they would be equivalent under certain assumptions. Our system applies standard Rocchio first, and then applies density estimation to convert similarity scores to probabilities. Joachims tests his system on two representative categories of a Reuters corpus, whereas we test our system on 90 Reuters categories representing a common benchmark used in recent years to compare advanced categorization systems.[1] Joachims compares his probabilistic TF*IDF system to a standard version and Naive Bayes, whereas we compare our system against all methods tested by Yang and Liu [20] for Reuters categorization, and against several systems which comprise the publicly available Rainbow package [9] for categorization of our own corpus.

The concept of converting category similarity scores to probabilities (or recalibrating category probabilities) is not new. Some recent research exploring this potential is discussed in two recent papers by Bennett. In [1], Bennett provides evidence that Naive Bayes systems typically produce probabilities close to 0 or 1, a problem which we have noticed but have never formally analyzed, and begins to dicuss methods which could be used to recalibrate the probabilities. In a later paper [2], he introduces methods of fitting asymmetric Gaussian and Laplace distributions to the output of a Naive Bayes classifier and a linear SVM classifier. The methods described in these papers are designed to work for binary categorization tasks involving separate YES/NO decisions for all document/category pairs.

In a previous paper [12], we reported detailed results of classifying images based on associated text as either Indoor or Outdoor. Even then, our system, which relied on a combination of Rocchio and a much simpler version of density estimation, performed better for our corpus than a competing image-based system [17] which was optimized to perform the same task. In another paper [11], we reported the results of integrating our system at that time with an image-based system. We have since improved our system using the generalized method of density estimation described in this paper. The previous method could only be applied when there were exactly two mutually exclusive categories, but our current version can be applied when dealing with any number of mutually exclusive categories or when we are dealing with a multi-label, binary categorization task. In addition, our current method leads to significantly better results even for the task involving two mutually exclusive categories.

3. DENSITY ESTIMATION

Density estimation is a statistical technique for estimating a probability density[2] for the distribution assumed to generate a set of empirically obtained data points. Our approach to using density estimation for the purposes of text categorization can be applied in conjunction with any text categorization system that expresses a similarity score between each document and category. It works regardless of the number of categories, and it works with either mutually exclusive categories or with independent, binary categories.

Assume we are dealing with a set of N categories (C_1, C_2, ..., C_N). Let us also assume that we have a text categorization system using a method that can assign to any given document d a similarity score to category C_i, namely $S_{(d,C_i)}$. This can be done for every category, and so we can obtain for each document a vector of similarity scores, one for each category. The vector for some given document d can be represented as $V_d = [S_{(d,C_1)}, S_{(d,C_2)}, ..., S_{(d,C_N)}]$.

Rocchio/TF*IDF systems create just such a set of similarity scores for every document. These scores only have meaning in comparison with each other, and so this constitutes an ideal type of system to which density estimation can be applied. Systems that use methods such as Naive Bayes or K-Nearest Neighbor generally compute probabilities of category membership for each category. These probabilities can be used in place of similarity scores in the vector above, and density estimation can be used to re-scale the probabilities. SVMs, in their basic form, make binary (YES/NO) decisions for individual categories by deciding whether a transformation of a representation of a document falls on a particular side of a hyperplane in a mapped vector space. When used for mutually exclusive categories, the distances from the hyperplanes are sometimes used to determine positive or negative (depending on the side of the hyperplane) scores for categories, and these scores could be used for the vectors to which density estimation could apply. Density estimation potentially can be applied to most commonly used techniques for text categorization; exceptions include decision tree approaches and expert systems which follow a specific chain of rules that ends in a prediction of a specific category without generating scores for each category.

When dealing with mutually exclusive categories, a standard text categorization system, after generating a vector of category similarity scores for a document, would assign the document to the category with the highest score. While this seems intuitive, it is not always the best solution, because the similarity measures do not always have an intrinsic meaning, and the scale is not always the same for every category. When dealing with binary categories, there are several standard methods of converting similarity scores to YES/NO decisions for every possible document/category pair. One, known as Scut [19], involves determining the optimal threshold for each category based on training data. (Optimality can be determined, for example, by maximizing F_1 measures, which will be described in Section 6). Another method, known as Pcut [19], involves measuring the percentage of training documents that fall into each category in the training set and assuming a similar percentage will fall into the category in the test set. Therefore, for each category, we assign the category to the x test documents with the highest similarity where x is chosen based on the training set. A third method is to create, for each category, a separate category consisting of all documents not in the actual category. To obtain a YES/NO decision for the actual category, we compare the similarity score of a test document and the actual category to the similarity score of the test document and the created category, thus converting our multi-label, binary categorization task to a set of categorization tasks each with two mutually exclusive categories.

The use of density estimation replaces those methods described in the previous paragraph. First, the system is used not only to obtain similarity score vectors for each test document, but also for each training document. (Of course, the vectors for the training documents only need to be calculated once ahead of time.) So, a vector of category similarity scores, as described above, is computed for every training

[1]Joachims also tests his system on a collection of articles from 20 newsgroups.

[2]Or a probability mass function, if the distribution is assumed to be discrete.

document as well as every test document. Let d_1 and d_2 be two documents in the corpus. We can then measure the Euclidean distance between the similarity score vectors for d_1 and d_2 as follows:

$$D_{(d_1,d_2)} = \sqrt{\sum_{i=1}^{N}[S_{(d_1,C_i)} - S_{(d_2,C_i)}]^2}$$

To use density estimation to label a test document d, distances are computed between the document's similarity score vector and those of every training document. The k training documents with the closest similarity score vectors to that of d, and thus the smallest distances according to the above formula, are selected. In other words, we are choosing training documents whose similarity score vectors fall within an N-dimensional hypersphere centered at the point specified by the similarity score vector for d. The labels of these document will be used to determine the predicted label of the test document as described below. We select the same number of training documents for each test document (as opposed to examining a hypersphere of fixed radius) due to the potential variability in sparseness of data for different values of similarity scores. The parameter k (the number of training documents used) can be chosen arbitrarily, or it can be determined based on cross-validation experiments within the training set.

Once training documents with similarity score vectors close to that of d have been identified, density estimation provides a probability of membership of d in each category C, estimated as the proportion of those neighbors from the training set which were assigned the label C. In calculating this proportion, individual training documents are weighted inversely proportional to the distance of their similarity score vectors from the center of the hypersphere (corresponding to the similarity score vector of the test document d). A separate decision can be made for each category based on the (weighted) percentage of close training documents that have been assigned to the category, and probability estimates for all possible categories can be assigned to the test document. More formally, let t_i be the i^{th} training document out of the k training documents selected as described above, and let $I_{(t_i,C)}$ be 1 if t_i belongs to category C and 0 otherwise. The estimated probability that the current document d belongs to some specific category C is:

$$P(C|d) = \frac{\sum_{i=1}^{k} I_{(t_i,C)} \frac{1}{D_{(d,t_i)}+\epsilon}}{\sum_{i=1}^{k} \frac{1}{D_{(d,t_i)}+\epsilon}}$$

Note that the numerator and denominator in the above formula are the same except for the $I_{(t_i,C)}$ term, so that documents which belong to category C contribute to both the numerator and denominator, and documents that do not belong to category C contribute only to the denominator. The epsilon in the formula is just an arbitrary, very small constant to avoid infinities in the case that there is some training document with a similarity score vector that exactly matches that of d.

Density estimation as just described can be applied with any number of categories and it can be used as described regardless of whether we are dealing with mutually exclusive categories or binary categories. If we are dealing with mutually exclusive categories, every selected training document will have exactly one label, and the probabilities assigned to categories for a given test document will add up to 1. If we are dealing with binary categorization, each individual category will be assigned a probability ranging from 0 to 1, but there is no further restriction on the sum of these probabilities (since the categories are independent and a document may belong to more than one category or none at all). When dealing with binary categorization, density estimation is applied to the raw similarity scores between a test document and categories; this method replaces the use of Scut, Pcut, or conversion to a set of mutually exclusive categorization tasks described earlier in the section.

Those who are familiar with K-Nearest Neighbor approaches to text categorization may notice a strong resemblance between these and density estimation. In each case, we are choosing certain training examples and using their categories to predict the category or categories of a test document. The important distinction is that density estimation is *not* comparing the actual test document to any training documents. It is only comparing the category similarity score vector of the test document to category similarity score vectors of training documents. It is very possible that a document that might not share any words in common with a test document will have a very similar (maybe even an identical) category similarity score vector. The purpose of density estimation is not to find training documents which are similar to the test document, but rather to interpret the category similarity scores of the test document which can be the result of some entirely different system. For each experiment discussed in this paper, we do compare the results of our system using density estimation to those of one or more kNN systems.

4. DATA SETS

We have evaluated the benefits of density estimation through experimentation on three data sets taken from two corpora. For each data set, we consider a different text categorization task. The first task involves non-topical categories (categorization of images as either Indoor or Outdoor), the second task involves broad, high-level topical categories (categorization of news documents into general categories), and the third task deals with Reuters topic categories. The first two tasks use data sets that we have created using our own corpus (as described below), and we have defined the categories for these tasks to be mutually exclusive. The third task uses a commonly used, publicly available data set in which the categories are binary (i.e. each document can have any number of labels up to the number of categories).

4.1 Data Set 1

The raw data from our first corpus consists of news postings from November 1995 through January 1999 from a variety of Usenet newsgroups. All of the postings contain a text article and some contain one or two images with associated captions. For our experiments on this corpus, we define a document as an image along with its caption and corresponding article; there are 2,312 documents in total. Most captions are two or three sentences long. Usually, the first sentence describes the associated image and the rest give background information about the related story.

In order to create training and test sets for our first two experiments, we set up two user-friendly, web-based interfaces that allow users to label documents manually. Detailed instructions including definitions and guidelines for the categories were provided. The first such interface asked users to categorize images according to the Indoor versus Outdoor distinction, and the choices given were *Indoor*, *Likely Indoor*, *Ambiguous*, *Likely Outdoor*, and *Outdoor*. The instructions, category definitions, guidelines, and interface for this set of categories can be viewed at http://www.cs.columbia.edu/~sable/research/readme.html.

Using the first interface, three volunteers manually labeled a total of 1,675 images, and the same 1,675 images were also labeled by the first author of this paper. These four evaluators were given access to the images along with their captions. A total of 1,339 images (79.9% of the 1675) were assigned a definite label in the same direction by both evaluators, and these 1,339 images comprised the data set for the first experiment discussed in this paper. 401 (29.9%) of these images were classified as Indoor and 938 (70.1%) were classified as Outdoor.

4.2 Data Set 2

The second interface asked users to categorize entire documents into one of the mutually exclusive categories *Struggle*, *Politics*, *Disaster*, *Crime*, or *Other*. We will refer to these as our Events categories in the remainder of this paper. The information provided for this set of categories can be viewed at http://www.cs.columbia.edu/~sable/research/instr.html.

Using this interface, 28 volunteers manually labeled a total of 1,750 documents, and the same 1,750 documents were also labeled by the first author of this paper. This time, we asked evaluators to categorize entire documents, each consisting of an article, an image, and a corresponding caption. A total of 1,328 (75.9%) of the 1,750 documents were assigned identical labels by both evaluators, and these 1,328 documents comprised the data set for the second experiment discussed in this paper. 417 (31.4%) of these documents were classified as Struggle, 387 (29.1%) were classified as Politics, 296 (22.3%) were classified as Disaster, 150 (11.3%) were classified as Crime, and 78 (5.9%) were classified as Other.

4.3 Data Set 3

For the third experiment, we used the ModApte split of the Reuters-21578 [6] corpus, a common benchmark for comparing different methods of text categorization [5, 15, 20]. This split includes 9,603 training documents and 3,299 test documents. To allow for direct comparison with Yang and Liu [20], we eliminated all categories which did not contain at least one training document and one test document and then discarded unlabeled documents. This left 90 topic categories, 7,770 training documents, and 3,019 test documents. Most of the documents (9160 of the 10,789, or 84.9%) were assigned to exactly one category, and the rest were assigned to more.[3] The most categories assigned to any one document was 15, and the average number of categories assigned to a document was approximately 1.24. Category distribution was quite skewed. The most common category had 3,964 instances (2,877 training instances and 1,087 test instances), while, on average, each category had approximately only 148 instances, and some categories had only two instances.

[3]This is because we have eliminated documents with no assigned categories to be consistent with Yang and Liu. It is a bit unusual for binary categorization, and our system does not assume that a document has at least one category.

5. EXPERIMENTS

For each of the data sets described in section Section 4, we first applied our own TF*IDF/Rocchio system which we have previously described in [12] and then we applied density estimation to the results of the system to measure its effect on performance. TF*IDF/Rocchio systems base categorizations on similarities between documents and categories. The term *document* here is used in a general sense to refer to whatever type of entity is being categorized. Each document and each category is represented by a weighted word vector. Each word is weighted according to the product of the word's TF, or *term frequency*, and the word's IDF, or *inverse document frequency* [13, 14]. Once every document and category is represented by a vector of TF*IDF values, a document can be compared to a category by taking the dot product of the document's vector and the category's vector. Our system also allows many parameters which are not typical of other systems using this methodology (e.g. the restriction to words of specific grammatical categories as determined by a statistical part-of-speech tagger [3]).

Our system automatically performs cross-validation experiments within the training set to choose the settings for the optional parameters that are likely to maximize performance. For the experiments discussed in this paper, we performed these cross validation experiments with and without density estimation, as it is possible that the use of density estimation will change the optimal settings for certain other parameters. For example, one of our parameters deals with normalization of category word vectors, which becomes more important when density estimation is not used since both normalization and density estimation can account for skewed category sizes. For the cross-validation experiments using density estimation, we also tried out multiple possible values of k which, as described in Section 3, is the number of training documents used to determine the predicted label or labels of each test document.

In addition to measuring the effect of density estimation on the performance of our system for each data set, we also wanted to compare our results against those of other recent, advanced methods. For our first two experiments, we therefore also tested six competing systems which comprise the publicly available Rainbow package (cited in Section 2). All of these systems use text categorization techniques which have been previously described in literature. Since we were particularly interested in comparing our results with those of a K-Nearest Neighbor system, and that which is part of the Rainbow package seemed to perform poorly, we also implemented our own version of a kNN system which uses the same word vectors for documents as does our TF*IDF Rocchio system. For our third experiment, which uses binary Reuters categories, we could not test the Rainbow systems or our own version of a kNN system because these were implemented to handle only data sets with mutually exclusive categories. Instead, we compare results against all systems test by Yang and Liu [20] (including a kNN system) in a controlled study conducted using the same data set. These systems were chosen because they use well-known methods and achieved strong performance scores in previously reported studies discussing similar experiments [20].

6. EVALUATION MEASURES

For our first two experiments, the main measure we consider is overall accuracy. Each test document is assigned one category, and the overall accuracy of the system is the percentage of such assignments that are correct. We also report the F_1 measures for each category. The F_1 measure [18] combines precision and recall into a single measurement:

$$F_1 = \frac{2 \times Precison \times Recall}{Precision + Recall}$$

This measure computes a score which is closer to the lower of precision and recall, and thus requires good results for both measurements to achieve a high F_1 score.

When dealing with binary categorization, such as is the case with our third experiment, values of precision, recall, and F_1 can be computed for each category and then averaged together (macro-averaging), or they can be computed once based on all binary decisions being made (micro-averaging). The former method treats all categories equally, while the later method treats all documents equally. Yang and Liu [20] report micro-averaged precision, micro-averaged recall, micro-averaged F_1, macro-averaged F_1, and overall error (which is 1 minus overall accuracy), so we do the same for our third experiment to allow for direct comparison. We focus on the micro-averaged F_1 score which has been more widely used for comparing methods and systems than the macro-averaged alternative [20].

7. RESULTS AND EVALUATION

7.1 First Experiment

For our first experiment, we randomly selected 894 (approximately two thirds) of the 1,339 images in our first data set (those that had definite agreement between two evaluators on the Indoor versus Outdoor classification question) for training, and we used the remaining 445 images for testing. The training set contained 276 (30.9%) indoor images and 618 (69.1%) outdoor images, while the test set contained 125 (28.1%) indoor images and 320 (71.9%) outdoor images. Therefore, a baseline classifier which labels every image as Outdoor achieves an overall accuracy of 71.9% (although the F_1 measure of this classifier would be 0 for the Indoor category).

Using the chosen parameters based on the cross-validation experiments discussed in Section 5 (with and without density estimation), we trained our system on the entire training set and tested on the previously unseen test set. The overall accuracy of our system when density estimation was *not* used was 80.7%. Table 1 shows the precision and recall values achieved on our test set for each category. The F_1 measures for the Indoor and Outdoor categories were 69.9% and 85.7% respectively. The overall accuracy of our system after density estimation was applied rose to 86.1%.[4] Table 2 shows the precision and recall values achieved on our test set for each category. The F_1 measures for the Indoor and Outdoor categories were 73.7% and 90.5% respectively. So density estimation not only had a significant effect on the overall accuracy of our system (a chi-square test shows a P-value of 3.8%), but it also improved performance (based on the F_1 measure) for both categories.

[4]Whichever category has the highest probability according to density estimation is considered to be the system's prediction.

	Actual Indoor	Actual Outdoor	Precision
System Indoor	100	61	62.1%
System Outdoor	25	259	91.2%
Recall	80.0%	80.9%	

Table 1: Results for the first data set without density estimation.

	Actual Indoor	Actual Outdoor	Precision
System Indoor	87	24	78.4%
System Outdoor	38	296	88.6%
Recall	69.6%	92.5%	

Table 2: Results for the first data set using density estimation.

Table 3 shows the results of all systems tested on our first data set. As mentioned in the previous paragraph, density estimation led to a significant increase in accuracy for this experiment. In addition, our system with density estimation performed better than all but one competing system, that being a Probabilistic Indexing system which beat our system with density estimation by only one test document. All systems beat a baseline of 71.9% accuracy which could be achieved by a system which chooses the largest category every time. For this data set, we also measured the performance of humans who were asked to predict whether the images in our test set were Indoor or Outdoor by looking only at the textual captions. The overall accuracy of humans was 87.6%, and we consider this to be a reasonable upper bound for how well an automatic system using only text might be expected to do. Note that the best systems, including our system with density estimation, were within 2% of this result.

7.2 Second Experiment

For our second experiment, we randomly selected 885 (approximately two thirds) of the 1,328 documents in our second data set (those that had agreement between two evaluators for our Events categories) for training, and we used the remaining 443 documents for testing. The largest category in the training set was Struggle, which accounted for 282 (31.9%) of the training documents and 135 (30.5%) of the test documents. Incidentally, the largest category in the test set was Politics, which accounted for 243 (27.5%) of the training documents and 144 (32.5%) of the test documents. A baseline classifier which predicted the largest category ev-

System	Overall Accuracy %	Indoor F_1 %	Outdoor F_1 %
Rocchio/TF*IDF	80.8	69.9	85.7
Density Estimation	86.1	73.7	**90.5**
K-Nearest Neighbor	82.7	65.8	88.4
Naive Bayes (Rainbow)	85.4	73.5	89.9
Rocchio/TF*IDF (Rainbow)	84.5	73.2	89.1
K-Nearest Neighbor (Rainbow)	77.8	65.3	83.6
Probabilistic Indexing (Rainbow)	**86.3**	**78.1**	90.0
SVMs (Rainbow)	82.0	66.9	87.7
Maximum Entropy (Rainbow)	84.5	70.9	89.4

Table 3: Density estimation led to a significant increase in accuracy for the Indoor versus Outdoor data set.

System	Overall Accuracy %	Struggle F_1 %	Politics F_1 %	Disaster F_1 %	Crime F_1 %	Other F_1 %
Rocchio/TF*IDF	87.1	85.0	88.4	**98.8**	79.2	60.0
Density Estimation	84.9	83.7	86.0	97.3	80.0	34.3
K-Nearest Neighbor	84.0	81.1	82.1	93.9	81.3	65.0
Naive Bayes (Rainbow)	87.6	86.2	86.3	96.7	89.1	61.5
Rocchio/TF*IDF (Rainbow)	87.4	81.1	85.3	97.7	88.4	**68.3**
K-Nearest Neighbor (Rainbow)	81.9	80.0	79.7	95.6	75.6	63.2
Probabilistic Indexing (Rainbow)	86.5	83.6	84.8	97.2	89.4	65.0
SVMs (Rainbow)	**88.7**	**88.1**	**89.2**	96.2	87.0	57.9
Maximum Entropy (Rainbow)	88.3	**88.1**	87.9	95.7	87.9	55.6

Table 4: Density estimation degraded performance for our Events categories.

method	miR %	miP %	miF1 %	maF1 %	error %
Rocchio/TF*IDF (Pcut)	71.21	70.72	70.96	50.14	0.803
Density Estimation	78.93	87.48	82.98	40.52	0.446
Combo	80.48	83.90	82.15	51.18	0.482
Support Vector Machines	81.20	91.37	85.99	52.51	0.365
K-Nearest Neighbor	83.39	88.07	85.67	52.42	0.385
Least Squares Fit	85.07	84.89	84.98	50.08	0.414
Neural Net	78.42	87.85	82.87	37.65	0.447
Naive Bayes	76.88	82.45	79.56	38.86	0.544

Table 5: Density estimation leads to a huge improvement for the Reuters data set.

ery time (based on the training set) would therefore achieve only a 30.5% overall accuracy.

On this particular data set, our TF*IDF/Rocchio system achieved a better performance without density estimation. Without density estimation, the overall accuracy of our system was 87.1%. The F_1 measures for the categories Struggle, Politics, Disaster, Crime, and Other were 85.0%, 88.4%, 98.8%, 79.2%, and 60.0% respectively. With density estimation, the overall accuracy was 84.9%. The F_1 measures for the categories Struggle, Politics, Disaster, Crime, and Other were 83.7%, 86.0%, 97.3%, 80.0%, and 34.3% respectively. Therefore, density estimation degraded performance on this data set by 2.2%, and performance was worse on four of the five categories according to F_1 measures. This does not necessarily mean that a user would not want to use density estimation. The difference in overall accuracy is not statistically significant (a chi-square test shows a P-value of 38.4%), and using density estimation assigns confidence measures in terms of probability to all predictions, whereas with a standard approach, the best one can get is a ranked list of categories.

For further comparison, we tested all of the same systems used in our first experiment. Table 4 shows the results. In this experiment, several of the competing systems had a higher overall accuracy than our own. All systems far outperform the baseline of 30.5%, and our system with density estimation still beats both versions of kNN systems tested.

7.3 Third Experiment

Table 5 show the results of all systems tested on the Reuters data set. The five columns of results represent micro-averaged recall, micro-averaged precision, micro-averaged F_1, macro-averaged F_1, and overall error (equal to 1 minus overall accuracy). The bottom five rows of the table are a reproduction of Table 1 from [20], summarizing the results of all categorizers Yang and Liu tested. The five methods used were Support Vector Machines, K-Nearest Neighbor, Least

Squares Fit, Neural Net, and Naive Bayes. They are ranked from top to bottom in the table based on micro-averaged F_1, generally considered to be the most important of these performance measures [20].

The top row of the table shows the results of our system without density estimation. Without density estimation, some other technique is needed to convert category similarity scores to YES/NO decisions for each category as described in Section 3, since Reuters is a binary categorization corpus. We tried all three of the standard methods described in Section 3, and Pcut worked by far the best, so that was used for the result shown in the first line of the table. The micro-averaged F_1 measure, the one that we tried to optimize, was only 70.96%, far lower than the five methods tested by Yang and Liu. This is not surprising, as standard Rocchio does not generally fare well against other methods for binary categorization. For example, see Yang [19] in which she includes Rocchio among the compared systems tested on similar Reuters corpora. Interestingly, the macro-averaged F_1 measure was higher than three of the five systems tested by Yang and Liu.

The second row of the table shows the results of our system with density estimation. As can be seen, the micro-averaged F_1 went up from 70.96% to 82.98%, a major improvement. In addition, the overall error went down from 0.803% to 0.446%. These scores are better than those for the Naive Bayes approach and marginally better than those for the Neural Net approach examined by Yang and Liu. Density estimation took a system which was far under-performing top competing systems and improved its performance to such a degree that it was in the pack.

Interestingly, density estimation actually lowered the macro-averaged F_1 of our system from 50.14% to 40.52%. This led us to believe that density estimation may perform better for large categories while the standard approach may perform better for small categories (since macro-averaged F_1 treats all categories equally). We therefore performed a final experiment combining the two approaches. For all categories for which density estimation performed better according to F_1 measures based on cross-validation within the training set, we also used density estimation on the test set, but for other categories we used Pcut on the test set. It turns out that Pcut was used for 52 of the 90 categories, but these categories accounted for only 1,257 assignments in the training set, whereas density estimation was used for only 38 categories, but these categories accounted for 8,626 assignments in the training set. The results of our final experiment are summarized in row 3 of Table 5. As can be seen, the micro-averaged F_1 and overall error were slightly worse than when density estimation was used for all categories, but still much better than Pcut and better than the Naive Bayes approach examined by Yang and Liu. The macro-averaged F_1 of 51.18% was better than when either Pcut or density estimation was used alone, beating three of the five methods tested by Yang and Liu.

8. CONCLUSION

This paper has presented a novel application of an established mathematical technique, density estimation, that significantly improved the performance of our Rocchio-based text categorization system in two out of three experiments. This technique could be applied to the results of any system which categorizes documents by computing a similarity score for every category. In addition to improving performance, density estimation provides probabilistic confidence measures for a system's predictions, whereas a standard Rocchio-based system (and many other alternatives to which density estimation could be applied) can only provide category rankings or YES/NO decisions. We suggest that the use of density estimation should be considered for categorization tasks; experiments within the training set can be used to determine whether it is likely to improve the performance of a system.

9. REFERENCES

[1] P. Bennett. Assessing the calibration of Naive Bayes' posterior estimates. Technical report, Carnegie Mellon University, 2000. CMU-CS-00-155.

[2] P. Bennett. Using asymmetric distributions to improve classifier probabilities: A comparison of new and standard parametric methods. Technical report, Carnegie Mellon University, 2002. CMU-CS-02-126.

[3] K. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing (ANLP-88)*, pages 136–143, Austin, Texas, February 1988. Association for Computational Linguistics.

[4] T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of the International Conference on Machine Learning (ICML-97)*, 1997.

[5] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, 1998.

[6] D. Lewis. Reuters-21578 text categorization test collection, readme file (version 1.2), 1997.

[7] D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the European Conference on Machine Learning*, 1998.

[8] D. Lewis, R. Schapire, J. Callan, and R. Papka. Training algorithms for linear text classifiers. In *Proceedings of the 19th International ACM SIGIR Conference on Researce and Development in Information Retrieval (SIGIR-96)*, 1996.

[9] A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification, and clustering. http://www.cs.cmu.edu/~mccallum/bow, 1996.

[10] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering (IJCAI-99)*, 1999.

[11] S. Paek, C. Sable, V. Hatzivassiloglou, A. Jaimes, B. Schiffman, S. Chang, and K. McKeown. Integration of visual and text-based approaches for the content

labeling and classification of photographs. In *ACM SIGIR Workshop on Multimedia Indexing and Retrieval (SIGIR-99)*, 1999.

[12] C. Sable and V. Hatzivassiloglou. Text-based approaches for non-topical image categorization. *International Journal of Digital Libraries*, **3**(3):261–275, 2000.

[13] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, Massachusetts, 1989.

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

[15] R. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, **39**(2):135–168, 2000.

[16] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman Hall, London, 1986.

[17] M. Szummer and R. W. Picard. Indoor-outdoor image classification. In *IEEE Workshop on Content Based Access of Image and Video Databases (CAIVD-98)*, pages 42–51, Bombay, India, January 1998.

[18] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 2nd edition, 1979.

[19] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval Journal*, **1**(1/2):69–90, 1999.

[20] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-99)*, 1999.