Tackling the Poor Assumptions of Naive Bayes Text Classifiers

Jason D. M. Rennie JRENNIE@MIT.EDU
Lawrence Shih KAI@MIT.EDU
Jaime Teevan TEEVAN@MIT.EDU
David R. Karger KARGER@MIT.EDU
Artificial Intelligence Laboratory; Massachusetts Institute of Technology; Cambridge, MA 02139

Abstract

Naive Bayes is often used as a baseline in text classification because it is fast and easy to implement. Its severe assumptions make such efficiency possible but also adversely affect the quality of its results. In this paper we propose simple, heuristic solutions to some of the problems with Naive Bayes classifiers, addressing both systemic issues as well as problems that arise because text is not actually generated according to a multinomial model. We find that our simple corrections result in a fast algorithm that is competitive with state-of-the-art text classification algorithms such as the Support Vector Machine.

1. Introduction

Naive Bayes has been denigrated as "the punching bag of classifiers" (Lewis, 1998), and has earned the dubious distinction of placing last or near last in numerous head-to-head classification papers (Yang & Liu, 1999; Joachims, 1998; Zhang & Oles, 2001). Still, it is frequently used for text classification because it is fast and easy to implement. Less erroneous algorithms tend to be slower and more complex. In this paper, we investigate the reasons behind Naive Bayes' poor performance. For each problem, we propose a simple heuristic solution. For example, we look at Naive Bayes as a linear classifier and find ways to improve the learned decision boundary weights. We also better match the distribution of text with the distribution assumed by Naive Bayes. In doing so, we fix many of the classifier's problems without making it slower or significantly more difficult to implement.

In this paper, we first review the multinomial Naive Bayes model for classification and discuss several systemic problems with it. One systemic problem is that when one class has more training examples than another, Naive Bayes selects poor weights for the decision boundary. This is due to an under-studied bias effect that shrinks weights for classes with few training examples. To balance the amount of training examples used per estimate, we introduce a "complement class" formulation of Naive Bayes.

Another systemic problem with Naive Bayes is that features are assumed to be independent. As a result, even when words are dependent, each word contributes evidence individually. Thus the magnitude of the weights for classes with strong word dependencies is larger than for classes with weak word dependencies. To keep classes with more dependencies from dominating, we normalize the classification weights.

In addition to systemic problems, multinomial Naive Bayes does not model text well. We present a simple transform that enables Naive Bayes to instead emulate a power law distribution that matches real term frequency distributions more closely. We also discuss two other pre-processing steps, common for information retrieval but not for Naive Bayes classification, that incorporate real world knowledge of text documents. They significantly boost classification accuracy.

Our Naive Bayes modifications, summarized in Table 4, produces a classifier that no longer has a generative interpretation. Thus, common model-based techniques to uncover latent classes and incorporate unlabeled data, such as EM, are not applicable. However, we find the improved classification accuracy worthwhile. Our new classifier approaches the state-of-the-art accuracy of the Support Vector Machine (SVM) on several text corpora while being faster and easier to implement than the SVM and most modern-day classifiers.

2. Multinomial Naive Bayes

The Naive Bayes classifier is well studied. An early description can be found in Duda and Hart (1973). Some of the reasons the classifier is so common is that it is fast, easy to implement and relatively effective. Domingos and Pazzani (1996) discuss its feature independence assumption and explain why Naive Bayes performs well for classification even with such a gross

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

over-simplification. McCallum and Nigam (1998) posit a multinomial Naive Bayes model for text classification and show improved performance compared to the multi-variate Bernoulli model due to the incorporation of frequency information. It is this multinomial version, which we call "multinomial Naive Bayes" (MNB), that we discuss, analyze and improve on in this paper.

2.1. Modeling and Classification

Multinomial Naive Bayes models the distribution of words in a document as a multinomial. A document is treated as a sequence of words and it is assumed that each word position is generated independently of every other. For classification, we assume that there are a fixed number of classes, $c \in \{1, 2, \ldots, m\}$, each with a fixed set of multinomial parameters. The parameter vector for a class c is $\vec{\theta}_c = \{\theta_{c1}, \theta_{c2}, \ldots, \theta_{cn}\}$, where n is the size of the vocabulary, $\sum_i \theta_{ci} = 1$ and θ_{ci} is the probability that word i occurs in that class. The likelihood of a document is a product of the parameters of the words that appear in the document,

$$p(d|\vec{\theta}_c) = \frac{(\sum_i f_i)!}{\prod_i f_i!} \prod_i (\theta_{ci})^{f_i}, \qquad (1)$$

where f_i is the frequency count of word i in document d. By assigning a prior distribution over the set of classes, $p(\vec{\theta}_c)$, we can arrive at the minimum-error classification rule (Duda & Hart, 1973) which selects the class with the largest posterior probability,

$$\begin{aligned} l(d) &= \operatorname{argmax}_c \left[\log p(\vec{\theta}_c) + \sum_i f_i \log \theta_{ci} \right], && (2) \\ &= \operatorname{argmax}_c \left[b_c + \sum_i f_i w_{ci} \right], && (3) \end{aligned}$$

where b_c is the threshold term and w_{ci} is the class c weight for word i. These values are natural parameters for the decision boundary. This is especially easy to see for binary classification, where the boundary is defined by setting the differences between the positive and negative class parameters equal to zero,

$$(b_+ - b_-) + \sum_i f_i \, (w_{+i} - w_{-i}) = 0.$$

The form of this equation is identical to the decision boundary learned by the (linear) Support Vector Machine, logistic regression, linear least squares and the perceptron. Naive Bayes' relatively poor performance results from how it chooses the b_c and w_{ci}.

2.2. Parameter Estimation

For the problem of classification, the number of classes and labeled training data for each class is given, but the parameters for each class are not. Parameters must be estimated from the training data. We do this by selecting a Dirichlet prior and taking the expectation of the parameter with respect to the posterior. For details, we refer the reader to Section 2 of Heckerman (1995). This gives us a simple form for the estimate of the multinomial parameter, which involves the number of times word i appears in the documents in class c (N_{ci}), divided by the total number of word occurrences in class c (N_c). For word i, a prior adds in α_i imagined occurrences so that the estimate is a smoothed version of the maximum likelihood estimate,

$$\hat{\theta}_{ci} = \frac{N_{ci} + \alpha_i}{N_c + \alpha}, \qquad (4)$$

where α denotes the sum of the α_i. While α_i can be set differently for each word, we follow common practice by setting $\alpha_i = 1$ for all words.

Substituting the true parameters in Equation 2 with our estimates, we get the MNB classifier,

$$l_{\mathrm{MNB}}(d) = \operatorname{argmax}_c \left[\log \hat{p}(\theta_c) + \sum_i f_i \log \frac{N_{ci} + \alpha_i}{N_c + \alpha} \right],$$

where $\hat{p}(\theta_c)$ is the class prior estimate. The prior class probabilities, $p(\theta_c)$, could be estimated like the word estimates. However, the class probabilities tend to be overpowered by the combination of word probabilities, so we use a uniform prior estimate for simplicity. The weights for the decision boundary defined by the MNB classifier are the log parameter estimates,

$$\hat{w}_{ci} = \log \hat{\theta}_{ci}. \qquad (5)$$

3. Correcting Systemic Errors

Naive Bayes has many systemic errors. Systemic errors are byproducts of the algorithm that cause an inappropriate favoring of one class over the other. In this section, we discuss two under-studied systemic errors that cause Naive Bayes to perform poorly. We highlight how they cause misclassifications and propose solutions to mitigate or eliminate them.

3.1. Skewed Data Bias

In this section, we show that *skewed data*—more training examples for one class than another—can cause the decision boundary weights to be biased. This causes the classifier to unwittingly prefer one class over the other. We show the reason for the bias and propose to alleviate the problem by learning the weights for a class using all training data *not* in that class.

Table 1. Shown is a simple classification example with two classes. Each class has a binomial distribution with probability of heads $\theta = 0.25$ and $\theta = 0.2$, respectively. We are given one training sample for Class 1 and two training samples for Class 2, and want to label a heads (H) occurrence. We find the maximum likelihood parameter settings ($\hat{\theta}_1$ and $\hat{\theta}_2$) for all possible sets of training data and use these estimates to label the test example with the class that predicts the higher rate of occurrence for heads. Even though Class 1 has the higher rate of heads, the test example is classified as Class 2 more often.

Class 1 $\theta = 0.25$	Class 2 $\theta = 0.2$	p(data)	$\hat{\theta}_1$	$\hat{\theta}_2$	Label for H
T	TT	0.48	0	0	*none*
T	{HT,TH}	0.24	0	$\frac{1}{2}$	Class 2
T	HH	0.03	0	1	Class 2
H	TT	0.16	1	0	Class 1
H	{HT,TH}	0.08	1	$\frac{1}{2}$	Class 1
H	HH	0.01	1	1	*none*

$$p(\hat{\theta}_1 > \hat{\theta}_2) = 0.24$$
$$p(\hat{\theta}_2 > \hat{\theta}_1) = 0.27$$

Table 1 gives a simple example of the bias. In the example, Class 1 has a higher rate of heads than Class 2. However, our classifier labels a heads occurrence as Class 2 more often than Class 1. This is not because Class 2 is more likely by default. Indeed, the classifier also labels a tails example as Class 1 more often, despite Class 1's lower rate of tails. Instead, the effect, which we call "skewed data bias," directly results from imbalanced training data. If we were to use the same number of training examples for each class, we would get the expected result—a heads example would be more often labeled by the class with the higher rate of heads.

Let us consider the more complex example of how the weights for the decision boundary in text classification, shown in Equation 5, are learned. Since log is a concave function, the expected value of the weight estimate is less than the log of the expected value of the parameter estimate, $E[\hat{w}_{ci}] < \log E[\hat{\theta}_{ci}]$. When training data is not skewed, this difference will be approximately the same between classes. But, when the training data is skewed, the weights will be lower for the class with less training data. Hence, classification will be erroneously biased toward one class over the other, as is the case with our example in Table 1.

To deal with skewed training data, we introduce a "complement class" version of Naive Bayes, called Complement Naive Bayes (CNB). In estimating weights for regular MNB (Equation 4) we use training data from a single class, c. In contrast, CNB estimates parameters using data from all classes *except* c. We think CNB's estimates will be more effective because each uses a more even amount of training data per class, which will lessen the bias in the weight estimates. We find we get more stable weight estimates and improved classification accuracy. These improvements might be due to more data per estimate, but overall we are using the same amount of data, just in a way that is less susceptible to the skewed data bias.

CNB's estimate is

$$\hat{\theta}_{\tilde{c}i} = \frac{N_{\tilde{c}i} + \alpha_i}{N_{\tilde{c}} + \alpha}, \tag{6}$$

where $N_{\tilde{c}i}$ is the number of times word i occurred in documents in classes other than c and $N_{\tilde{c}}$ is the total number of word occurrences in classes other than c, and α_i and α are smoothing parameters, as in Equation 4. As before, the weight estimate is $\hat{w}_{\tilde{c}i} = \log \hat{\theta}_{\tilde{c}i}$ and the classification rule is

$$l_{\text{CNB}}(d) = \text{argmax}_c \left[\log p(\vec{\theta}_c) - \sum_i f_i \log \frac{N_{\tilde{c}i} + \alpha_i}{N_{\tilde{c}} + \alpha} \right].$$

The negative sign represents the fact that we want to assign to class c documents that *poorly* match the complement parameter estimates.

Figure 1 shows how different amounts of training data affect (a) the regular weight estimates and (b) the complement weight estimates. The regular weight estimates shift up and change their ordering between 10 examples of training data and 1000 examples. In particular, the word that has the smallest weight for 10 through 100 examples moves up to the 11th largest weight (out of 18) when estimated with 1000 examples. The complement weights show the effects of smoothing, but do not show such a severe upward bias and retain their relative ordering. The complement estimates mitigate the problem of the skewed data bias.

CNB is related to the one-versus-all-but-one (commonly misnamed "one-versus-all") technique that is frequently used in multi-label classification, where each example may have more than one label. Berger (1999) and Zhang and Oles (2001) have found that one-vs-all-but-one MNB works better than regular MNB. The classification rule is

$$l_{\text{OVA}}(d) = \text{argmax}_c \left[\log p(\vec{\theta}_c) + \left(\sum_i f_i \log \frac{N_{ci} + \alpha_i}{N_c + \alpha} - \sum_i f_i \log \frac{N_{\tilde{c}i} + \alpha_i}{N_{\tilde{c}} + \alpha} \right) \right]. \tag{7}$$

This is a combination of the regular and complement classification rules. We attribute the improvement

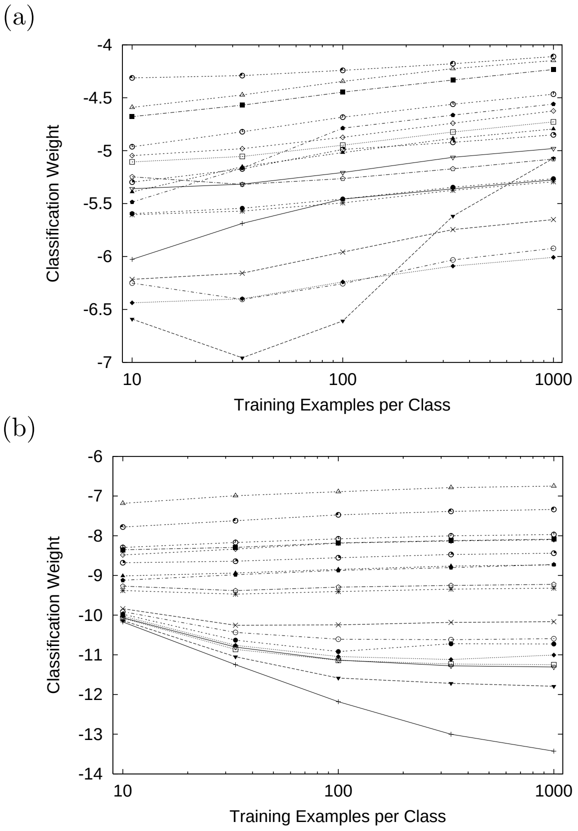


Figure 1. Average classification weights for 18 highly discriminative features from 20 Newsgroups. The amount of training data is varied along the x-axis. Plot (a) shows weights for MNB, and Plot (b) shows the weights for CNB. CNB is more stable across a varying amount of training data.

with one-vs-all-but-one to the use of the complement weights. We find that CNB performs better than one-vs-all-but-one and regular MNB since it eliminates the biased regular MNB weights.

3.2. Weight Magnitude Errors

In the last section, we discussed how uneven training sizes could cause Naive Bayes to bias its weight vectors. In this section, we discuss how the independence assumption can erroneously cause Naive Bayes to produce different magnitude classification weights. When the magnitude of Naive Bayes' weight vector $\vec{w}_c$ is larger in one class than the others, the larger-magnitude class may be preferred. For Naive Bayes, differences in weight magnitudes are not a deliberate attempt to create greater influence for one class. Instead, the weight differences are partially an artifact of applying the independence assumption to dependent data. Naive Bayes gives more influence to classes that most violate the independence assumption. The following example illustrates this effect.

Consider the problem of distinguishing between documents that discuss Boston and ones that discuss San Francisco. Let's assume that "Boston" appears in Boston documents about as often as "San Francisco" appears in San Francisco documents (as one might expect). Let's also assume that it's rare to see the words "San" and "Francisco" apart. Then, each time a test document has an occurrence of "San Francisco," Multinomial Naive Bayes will double count—it will add in the weight for "San" and the weight for "Francisco." Since "San Francisco" and "Boston" occur equally in their respective classes, a single occurrence of "San Francisco" will contribute twice the weight as an occurrence of "Boston." Hence, the summed contributions of the classification weights may be larger for one class than another—this will cause MNB to prefer one class incorrectly. For example, if a document has five occurrences of "Boston" and three of "San Francisco," MNB will label the document as "San Francisco" rather than "Boston."

In practice, it is often the case that weights tend to lean toward one class or the other. For the problem of identifying "barley" documents in the Reuters-21578 corpus, it is advantageous to choose a threshold term, $b = b_+ - b_-$, that is much more negative than one chosen by counting documents. In testing different smoothing values, we found that $\alpha_i = 10^{-4}$ gave the most extreme example of this. With a threshold term of $b = -94.6$, the classifier achieved as low an error rate as any other smoothing value. However, the threshold term calculated via the prior estimate by counting training documents was $\log \frac{p(\vec{\theta}_+)}{p(\vec{\theta}_-)} = -5.43$. This threshold yielded a somewhat higher rate of error. It is likely Naive Bayes' independence assumption lead to a strong preference for the "barley" documents.

We correct for the fact that some classes have greater dependencies by normalizing the weight vectors. Instead of assigning $\hat{w}_{ci} = \log \hat{\theta}_{ci}$, we assign

$$\hat{w}_{ci} = \frac{\log \hat{\theta}_{ci}}{\sum_k |\log \hat{\theta}_{ck}|}. \tag{8}$$

We call this, combined with CNB, Weight-normalized Complement Naive Bayes (WCNB). Experiments indicate that WCNB is effective. Alternately, one could address this problem by optimizing the threshold terms, b_c. Webb and Pazzani give a method for doing this by calculating per-class weights based on identified violations of the Naive Bayes classifier (Webb & Pazzani, 1998).

Since we are manipulating the weight vector directly,

Table 2. Experiments comparing multinomial Naive Bayes (MNB) with Weight-normalized Complement Naive Bayes (WCNB) over several data sets. Industry Sector and 20 News are reported in terms of accuracy; Reuters in terms of precision-recall breakeven. WCNB outperforms MNB.

	MNB	WCNB
Industry Sector	0.582	0.889
20 Newsgroups	0.848	0.857
Reuters (micro)	0.739	0.782
Reuters (macro)	0.270	0.548

we can no longer make use of the model-based aspects of Naive Bayes. Thus, common model-based techniques to incorporate unlabeled data and uncover latent classes, such as EM, are not applicable. This is a trade-off for improved classification performance.

3.3. Bias Correction Experiments

We ran classification experiments to validate the techniques suggested here. Table 2 gives classification performance on three text data sets, reporting accuracy for 20 Newsgroups and Industry Sector and precision-recall breakeven for Reuters. See the Appendix for a description of the data sets and experimental setup. We compared Weight-normalized Complement Naive Bayes (WCNB) with standard multinomial Naive Bayes (MNB), and found that WCNB resulted in marked improvement on all data sets. The improvement was greatest for data sets where training data quantity varied between classes (Reuters and Industry Sector). The greatly improved Reuters macro P-R breakeven score suggests that much of the improvement can be attributed to better performance on classes with few training examples. WCNB also shows an improvement (small, but significant) on 20 Newsgroups even though the distribution of training examples is even across classes.

In comparing, we note that our baseline, MNB, is similar to the MNB results found by others. Our 20 Newsgroups result closely matches that reported by McCallum and Nigam (1998) (85% vs. our 84.8%). The difference in Ghani (2000)'s Industry Sector result (64.5% vs. our 58.2%) is likely due to his use of feature selection. Zhang and Oles (2001)'s result on Industry Sector (84.8%) is significantly higher because they optimize the smoothing parameter. When we optimized the smoothing parameter for MNB via cross-validation, in experiments not reported here, our MNB results were similar. Smoothing parameter optimization also further improved WCNB. Our micro and macro scores on Reuters are reasonably similar to Yang and Liu (1999) (79.6% vs. our 73.9%, 38.9% vs. our 27.0%), with the differences likely due to their use of feature selection, a different scoring metric (F1), and a different pre-processing system (SMART).

4. Modeling Text Better

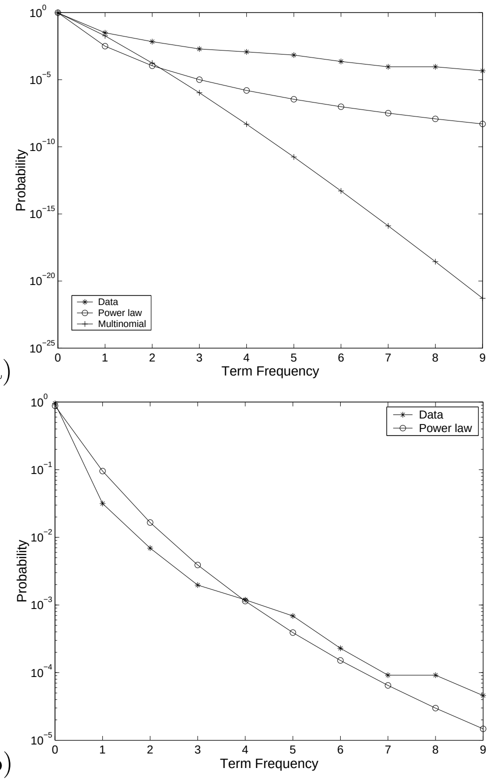
So far we have discussed systemic issues that arise when using any Naive Bayes classifier. MNB uses a multinomial to model text, which is not very accurate. In this section we look at three transforms to better align the model and the data. One transform affects frequencies—term frequency distributions have a much heavier tail than the multinomial model expects. We also transform based on document frequency, to keep common terms from dominating in classification, and based on length, to keep long documents from dominating during training. By transforming the data to be better suited for use with a multinomial model, we find significant improvement in performance over using MNB without the transforms.

4.1. Transforming Term Frequency

In order to understand if MNB would do a good job classifying text, we looked at empirical term distributions of text. We found that term distributions had heavier tails than predicted by the multinomial model, instead appearing like a power-law distribution. Using a simple transform, we can make these power-law-like term distributions look more multinomial.

To measure how well the multinomial model fits the term distribution of text, we compared the empirical distribution to the maximum likelihood multinomial. For visualization purposes, we took a set of words with approximately the same occurrence rate and created a histogram of their term frequencies in a set of documents with similar length. These term frequency rates and those predicted by the best fit multinomial model are plotted in Figure 2 on a log scale. The figure shows the empirical term distribution is very different from what a multinomial model would predict. The empirical distribution has a much heavier tail, meaning multiple occurrences of a term is much more likely than expected for the best fit multinomial. For example, the multinomial model predicts the chance of seeing an average word occur nine times in a document is $p(f_i = 9) = 10^{-21.28}$, so low that such an event in unexpected even in a collection of all news stories ever written. In reality the chance is $p(f_i = 9) = 10^{-4.34}$, very rare for a single document, but not unexpected in a collection of 10,000 documents.

This behavior, also called "burstiness", has been ob-



(a)

(b)

Figure 2. Shown is an example term frequency probability distribution, compared with several best fit analytic distributions. The data has a much heavier tail than the multinomial model predicts. A power law distribution ($p(f_i) \propto (d + f_i)^{\log \theta}$) matches more closely, particularly when an optimal d is chosen (Figure b), but also when $d = 1$ (Figure a).

served by Church and Gale (1995) and Katz (1996). While they developed sophisticated models to deal with term burstiness, we found that even a simple heavy tailed distribution, the power law distribution, could better model text and motivate a simple transform to the features of our MNB model. Figure 2(b) shows an example empirical distribution, alongside a power law distribution, $p(f_i) \propto (d + f_i)^{\log \theta}$, where d has been chosen to closely match the text distribution. The probability is also proportional to $\theta^{\log(d+f_i)}$. Because this is similar to the multinomial model, where the probability is proportional to θ^{f_i}, we can use the multinomial model to generate probabilities proportional to a class of power law distributions via a simple transform, $f_i' = \log(d + f_i)$. One such transform, $f_i' = \log(1 + f_i)$, has the advantages of being an identity transform for zero and one counts, while pushing down larger counts as we would like. The transform allows us to more realistically handle text while not giving up the advantages of MNB. Although setting $d = 1$ does not match the data as well as an optimized

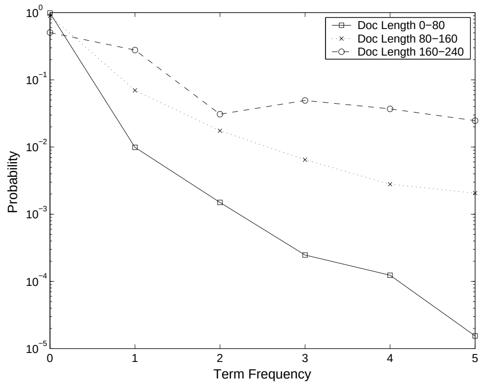


Figure 3. Plotted are average term frequencies for words in three classes of Reuters-21578 documents—short documents, medium length documents and long documents. Terms in longer documents have heavier tails.

d, it does produce a distribution that is much closer to the empirical distribution than the best fit multinomial, as shown by the "Power law" line in Figure 2(a).

4.2. Transforming by Document Frequency

Another useful transform discounts terms that occur in many documents. Common words are unlikely to be related to the class of a document, but random variations can create apparent fictitious correlations. This adds noise to the parameter estimates and hence the classification weights. Since common words appear often, they can hold sway over a classification decision even if their weight differences between classes is small. For this reason, it is advantageous to down-weight these words.

A heuristic transform in the Information Retrieval (IR) community, known as "inverse document frequency", is to discount terms by their document frequency (Jones, 1972). A common way to do this is

$$f_i' = f_i \log \frac{\sum_j 1}{\sum_j \delta_{ij}},$$

where δ_{ij} is 1 if word i occurs in document j, 0 otherwise, and the sum is over all document indices (Salton & Buckley, 1988). Rare words are given increased term frequencies; common words are given less weight. We found it to improve performance.

4.3. Transforming Based on Length

Documents have strong word inter-dependencies. After a word first appears in a document, it is more likely to appear again. Since MNB assumes occurrence independence, long documents can negatively effect parameter estimates. We normalize word counts to avoid

Table 3. Experiments comparing multinomial Naive Bayes (MNB) to Transformed Weignt-normalized Complement Naive Bayes (TWCNB) and the Support Vector Machine (SVM) over several data sets. TWCNB's performance is substantially better than MNB, and approaches the SVM's performance. Industry Sector and 20 News are reported in terms of accuracy; Reuters results are precision-recall breakeven.

	MNB	TWCNB	SVM
Industry Sector	0.582	0.923	0.934
20 Newsgroups	0.848	0.861	0.862
Reuters (micro)	0.739	0.844	0.887
Reuters (macro)	0.270	0.647	0.694

this problem. Figure 3 shows empirical term frequency distributions for documents of different lengths. It is not surprising that longer documents have larger probabilities for larger term frequencies, but the jump for larger term frequencies is disproportionally large. Documents in the 80-160 group are, on average, twice as long as those in the 0-80 group, yet the chance of a word occurring five times in the 80-160 group is larger than a word occurring twice in the 0-80 group. This would not be the case if text were multinomial.

To deal with this, we again use a common IR transform that is not seen with Naive Bayes. We discount the influence of long documents by transforming the term frequencies according to

$$f'_i = \frac{f_i}{\sqrt{\sum_k (f_k)^2}}.$$

yielding a length 1 term frequency vector for each document. This transform is common within the IR community because the probability of generating a document within a model is compared across documents; in such a case one does not want short documents dominating merely because they have fewer words. For classification, however, because comparisons are made across classes, and not across documents, the benefit of such normalization is more subtle, especially as the multinomial model accounts for length very naturally (Lewis, 1998). The transform keeps any single document from dominating the parameter estimates.

4.4. Experiments

We have described a set of transforms for term frequencies. Each of these tries to resolve a different problem with the modeling assumptions of Naive Bayes. The set of modifications and the procedure for applying them is shown in Table 4. When we apply those modifications, we find a significant improvement in text classification performance over MNB. Table 3 shows classification accuracy for Industry Sector and 20 Newsgroups and precision-recall breakeven for Reuters. In tests, we found the length normalization transform to be the most useful, followed by the log transform. The document frequency transform seemed to be of less import. We show results on the Support Vector Machine (SVM) for comparison. We used the transforms described in Section 4 for the SVM since they improved classification performance.

Table 4. Our new Naive Bayes procedure. Assignments are over all possible index values. Steps 1 through 3 distinguish TWCNB from WCNB.

- Let $\vec{d} = (\vec{d}_1, \ldots, \vec{d}_n)$ be a set of documents; d_{ij} is the count of word i in document j.
- Let $\vec{y} = (y_1, \ldots, y_n)$ be the labels.
- TWCNB$(\vec{d}, \vec{y})$
  1. $d_{ij} = \log(d_{ij} + 1)$ (TF transform § 4.1)
  2. $d_{ij} = d_{ij} \log \frac{\sum_k 1}{\sum_k \delta_{ik}}$ (IDF transform § 4.2)
  3. $d_{ij} = \frac{d_{ij}}{\sqrt{\sum_k (d_{kj})^2}}$ (length norm. § 4.3)
  4. $\hat{\theta}_{ci} = \frac{\sum_{j:y_j \neq c} d_{ij} + \alpha_i}{\sum_{j:y_j \neq c} \sum_k d_{kj} + \alpha}$ (complement § 3.1)
  5. $w_{ci} = \log \hat{\theta}_{ci}$
  6. $w_{ci} = \frac{w_{ci}}{\sum_i w_{ci}}$ (weight normalization § 3.2)
  7. Let $t = (t_1, \ldots, t_n)$ be a test document; let t_i be the count of word i.
  8. Label the document according to

$$l(t) = \arg\min_c \sum_i t_i w_{ci}$$

We discussed similarities in our multinomial Naive Bayes results in Section 3.3. Our Support Vector Machine results are similar to others. Our Industry Sector result matches that reported by Zhang and Oles (2001) (93.6% vs. our 93.4%). The difference in Godbole et al. (2002)'s result (89.7% vs. our 86.2%) on 20 Newsgroups is due to their use of a different multiclass schema. Our micro and macro scores on Reuters differ from Yang and Liu (1999) (86.0% vs. our 88.7%, 52.5% vs. our 69.4%), likely due to their use of feature selection, a different scoring metric (F1), and a different pre-processing system (SMART). The larger difference in macro results is due to the sensitivity of macro calculations, which heavily weighs small classes.

5. Conclusion

We have described several techniques, shown in Table 4, that correct deficiencies in the application of the

Naive Bayes classifier to text data. A series of transforms from the information retrieval community, Steps 1-3 in Table 4, improves the performance of Naive Bayes text classification. For example, the transform described in Step 1 converts text, which can be closely modeled by a power law, to look more multinomial. Training with the complement class, Step 4, solves the problem of uneven training data. Normalizing the classification weights, Step 6, improves upon the Naive Bayes handling of word occurrence dependencies. These modifications better align Naive Bayes with the realities of bag-of-words textual data and, as we have shown empirically, significantly improve its performance on a number of data sets. The modified Naive Bayes is a fast, easy-to-implement, near state-of-the-art text classification algorithm.

Acknowledgements We are grateful to Yu-Han Chang and Tommi Jaakkola for their input. We also thank the anonymous reviewers for valuable comments. This work was supported by the MIT Oxygen Partnership, the National Science Foundation (ITR) and Graduate Research Fellowships from the NSF.

References

Berger, A. (1999). Error-correcting output coding for text classification. *Proceedings of IJCAI '99.*

Church, K., & Gale, W. (1995). Poisson mixtures. *Natural Language Engineering*, *1*, 163–190.

Domingos, P., & Pazzani, M. (1996). Beyond independence: conditions for the optimality of the simple Bayesian classifier. *Proceedings of ICML '96.*

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis.* Wiley and Sons, Inc.

Ghani, R. (2000). Using error-correcting codes for text classification. *Proceedings of ICML '00.*

Godbole, S., Sarawagi, S., & Chakrabarti, S. (2002). Scaling multi-class Support Vector Machines using inter-class confusion. *Proceedings of SIGKDD.*

Heckerman, D. (1995). *A tutorial on learning with Bayesian networks* (Technical Report MSR-TR-95-06). Microsoft Research.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *Proceedings of ECML '98.*

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, *28*, 11–21.

Katz, S. (1996). Distribution of content words and phrases in text and language modelling. *Natural Language Engineering*, *2*, 15–60.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. *Proceedings of ECML '98.*

McCallum, A., & Nigam, K. (1998). A comparison of event models for naive Bayes text classification. *Proceedings of AAAI '98.*

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, *24*, 513–523.

Webb, G. I., & Pazzani, M. J. (1998). Adjusted probability naive Bayesian induction. *Proceedings of AI '01.*

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. *Proceedings of SIGIR '99.*

Zhang, T., & Oles, F. J. (2001). Text categorization based on regularized linear classification methods. *Information Retrieval*, *4*, 5–31.

Appendix

For our experiments, we use three well–known data sets: 20 Newsgroups, Industry Sector and Reuters-21578. Industry Sector and 20 News are single-label data sets: each document is assigned a single class. Reuters is a multi-label data set: each document may have many labels. Since Reuters is multi-label, it is handled differently than described in the paper. For MNB, we use the standard one-vs-all-but-one (usually misnamed "one-vs-all") on each binary problem. For CNB, we use all-vs-all-but-one, thus making the amount of data per estimate more even.

Industry Sector and Reuters-21578 have widely varying numbers of documents per class, but no single class dominates. The distribution of documents per class for 20 Newsgroups is even at about 1000 examples per class. For 20 Newsgroups, we ran 10 random splits with 80% training data and 20% testing data per class. There are 9649 Industry Sector documents and 105 classes; the largest category has 102 documents, the smallest has 27. For Industry Sector, we ran 10 random splits with 50% training data and 50% testing data per class. For Reuters-21578 we use the "ModApte" split and use only topics with at least one training document and one testing document. This gives 7770 training documents and 90 classes; the largest category has 2840 training documents.

For the SVM experiments, we used SvmFu[1] and set $C = 10$. We use one-vs-all to produce multi-class labels for the SVM. We use the linear kernel since it performs as well as non-linear kernels in text classification (Yang & Liu, 1999).

[1]SvmFu is available from http://fpn.mit.edu/SvmFu.