
A Naive Bayes Classifier Using Transductive Inference for Text Classification

Kristin M. Branson
Dept. of Computer Science and Engineering
University of California, San Diego
kbranson@cs.ucsd.edu

Abstract

In text classification tasks, it is often the case that the set of test documents is known at the time the classifier is learned. This is the case, for example, when a large database of documents is searched for documents of a certain topic. The algorithm presented in this paper, transductive naive Bayes, uses both the training documents and the distribution of the unclassified test documents to learn a classifier. Transductive naive Bayes classifies the test documents using a naive Bayes classifier learned from the training documents, then learns another classifier from the test documents labeled with this classification. The classification used by this final classifier is returned. Transductive naive Bayes has the advantages that its classifier can be specific to the actual documents to be classified, as well as the increased information about the data distribution. The results of testing this algorithm have been that transductive naive Bayes significantly outperforms naive Bayes; on the Reuters-21578 data set, transductive naive Bayes classified 81.4% standard naive Bayes, which classified only 55.2% correctly. Results also showed that transductive naive Bayes gained mostly from being able to tailor the classifier learned to the actual distribution of the test data.

1 Introduction

As the amount of online text and email increases, the task of sifting through this information to find the desired text is becoming both more difficult and more important. One solution to these text classification tasks, which is both general and effective, is an algorithm that learns from input example documents and their classifications, then produces a classifier that can be used on unclassified documents. Most text classification algorithms assume that the training data is representative of the test data. These algorithms produce text classifiers that optimize an objective function on the training data, assuming that this classifier will also optimize the same objective function on the test data.

However, it is not always the case that the distribution of the training documents

is identical to the distribution of the test documents. For example, consider an application that searches for articles of interest to its user. If the articles the user has previously marked as interesting or uninteresting are from a very general database of documents, most of which are not of interest, and the application searches a database more specific to the topics of interest to the user, then the fraction of documents of interest is much larger in this database than in the initial database. In addition, the training data distribution might not be a good approximation of the test data distribution if the number of training example documents is very small, as each training document is a significant fraction of the total training data.

Potentially, a learner could use the unlabeled distribution of the test data, in addition to the labeled distribution of the training data, to learn its classifier. This is the idea of a learner that uses transductive inference in addition to inductive inference. “Transductive” is a term that implies the transfer of information. In the case of transductive inference, this is the transfer of information about the distribution of the test data to the learner generating the classifier for that test data. Transductive inference is in contrast to inductive inference, which involves a classifier being inferred as optimal on the test data because it is optimal on the training data. In many applications, using transductive inference is impractical because the application involves dynamic test data. However, as the test data sets in many text classification tasks are relatively static in comparison to the training data sets, it is feasible for the learning algorithm to learn a new classifier for each test data set.

Transductive inference has successfully been applied to Support Vector Machines for text classification [Joachims, 1999]. This paper explores whether transductive inference can be applied to naive Bayes with as much success. In this paper, I will first discuss the naive Bayes algorithm which I am augmenting with transductive inference. I will then go into more detail about transductive inference. Next, I will explain how transductive inference is used with naive Bayes and why this is beneficial. Next, I will give the details of the transductive naive Bayes implementation. Finally, I will discuss the preliminary experimental results.

2 Naive Bayes

Naive Bayesian learners attempt to maximize the probability of the classifier they generate by estimating its likelihood using the training data distribution. Therefore, example x is given the label $l = \operatorname{argmax}_c Pr(c|x)$. Using Bayes’ rule, this is rewritten as $l = \operatorname{argmax}_c Pr(x|c)Pr(c)$. For each class c , $Pr(x|c)$ and $Pr(c)$ are independently estimated from the training data.

Naive Bayes makes the naive assumption that all the attributes of the data are conditionally independent, given the example’s class. In text classification, each document is represented by a document vector, which has, for each word in the vocabulary, an entry that represents the number of times the word occurs in the document. That is, the i^{th} component of the document vector is the number of times word w_i in the vocabulary occurs in the document. Using the naive assumption,

$$Pr(x|c) = Pr(w_1, \dots, w_k|c) = \prod_{i=1}^k Pr(w_i|c).$$

In an attempt to optimize the likelihood of the classification it computes, naive Bayes models $Pr(x|c)$, the probability distribution of the training data for each possible class c . Using the naive assumption, $Pr(x|c)$ is approximated as $\prod_{i=1}^k Pr(w_i|c)$. Each $Pr(w_i|c)$ is approximated from the training data as the

number of times word w_i occurs in all the training data of this class over the total count of words in the training data of this class [Hamerly,2001].

Naive Bayes combines these submodels for each class c by weighting each model with the probability of that model, $Pr(c)$. $Pr(c)$ is estimated as the fraction of training documents of class c . It labels each test document, x^* , with the class that maximizes the probability of x^* :

$$label(x^*) = argmax_c Pr(x^*|c)Pr(c).$$

By Bayes' rule, this is equivalent to:

$$label(x^*) = argmax_c Pr(c|x^*).$$

In summary, naive Bayes computes the probability of an example having each possible class, then labels the example with the most likely class. The submodels for each class represent solely the distribution of the training data. If the distributions of the training and test data differ, then the estimates naive Bayes makes for these probabilities, as well as the probabilities of each class, will be inaccurate for the test data. This will cause naive Bayes classifier to be inaccurate. One possible solution to this problem is to use transductive inference to learn the distribution of the test data.

3 Transductive Inference

Transductive inference is a recently developed method that allows the distribution of the test data to be used in classification. The concept was developed by Vapnik and later applied to SVMs [Joachims,1999]. Learners that use transductive inference are given both a labeled training set and an unlabeled test set to learn the optimal classifier. The optimal classifier is that which minimizes the classification error on the test set. The classification error on the test set is the expected value of the probability that the classification assigned by the classifier, h , is not the true classification. That is,

$$error(h) = E_{\langle x,c \rangle} [Pr(h(x) \neq c)].$$

As transductive algorithms are only interested in the performance on the test set and not in producing the optimal general classifier, the classification error is:

$$error(h) = \frac{1}{|Test|} \sum_{\langle x,c \rangle \in Test} Pr(h(x) \neq c).$$

In the next section, the theoretical framework of a transductive naive Bayes algorithm is discussed.

4 Transductive Naive Bayes

Transductive inference is a natural addition to a naive Bayes learner. This is because, as discussed in section 2, naive Bayes uses a combination of unsupervised probability submodels, each of which represents the distribution of the set of data that submodel represents. These unsupervised probability submodels do not individually depend on the labels of the data they are modeling. If each submodel were given the correct set of unlabeled data, then it could accurately model the distribution of that data. Thus, given a set of unlabeled test data, each submodel could accurately represent the distribution of that set of test data. For each class c , we can

estimate the $Pr(x|c)$ using the distribution of the test data for class c , $Pr_{test}(x|c)$ (Pr_{test} represents the probability over the test set, while Pr_{train} represents the probability over the training set).

The only difficulty with this approach is that, if the labels of the test data are unknown, then which documents of the test data should be used in each model is also unknown. This follows from the fact that the documents for a given submodel must be of a certain class. If there were a method for accurately approximating the labels of the test data, then modeling the distribution of the test data hypothesized to have a certain label would more accurately represent the actual distribution of the test data.

The obvious method for estimating the labels of the test data is to classify the test documents using the naive Bayes classifier learned from the labeled training data. Using these hypothesized labels, each submodel can represent the distribution of the test data hypothesized to be of the class the submodel represents, $Pr_{test}(x|c)$. Using the naive Bayes assumption of the conditional independence of the words of a document given the document's class, $Pr_{test}(x|c)$ can be rewritten as $\prod_{w_i} Pr_{test}(w_i|c)$. Transductive naive Bayes estimates this, using the labeled test set, as the number of times w_i occurs in all test documents given the label c , divided by the total number of words in all test documents given the label c . Thus,

$$Pr_{test}(w_i|c) = \frac{\text{word_count}(w_i, Test_c)}{\text{word_count}(*, Test_c)},$$

where $Test_c$ is the set of all the test documents classified as c :

$$Test_c = \{x : x \in Test, c = \text{argmax}_{c'} Pr_{train}(x|c') Pr_{train}(c')\}.$$

Which documents are in $Test_c$ depends on the distribution of the training data. However, the actual distribution of the data, given the class, is dependent on the actual test data. Thus, the $Pr_{test}(w_i|c)$ is representative of both the classification assigned by the training data as well as the distribution of the test data.

The hypothesized labels for the test data can also be used when estimating the weights of the class models, $Pr(c)$. $Pr_{test}(c)$ could potentially be a more accurate predictor of $Pr(c)$ than $Pr_{train}(c)$. Initially, the labels of the test data are hypothesized using the labeled training data. Thus, for each $x \in Test$, where $Test$ is the test data, the classification label of x is $\text{argmax}_{c' \in C} (Pr_{train}(x|c') Pr_{train}(c'))$. Let $Test_c$ be the set of all test documents that are labeled c by the classifier learned from the training data:

$$Test_c = \{x \in Test : c = \text{argmax}_{c' \in C} (Pr_{train}(x|c') Pr_{train}(c'))\}.$$

The probability of class c can then be estimated using the newly labeled test set as the number of examples in the test set labeled with class c divided by the total number of examples in the test set:

$$Pr_{test}(c) = \frac{|Test_c|}{|Test|}.$$

While $Pr_{test}(c)$ depends heavily on $Pr_{train}(c)$, it is also dependent on $Pr_{train}(x|c)$ for each x in the test set. If the distribution of the test set differs from the distribution of the training set, then incorporating $Pr_{train}(x|c)$ into the $Pr_{test}(c)$ will more closely model the actual distribution of the test data.

As an extreme example, consider the case in which all documents of class c contain the word w . Suppose the training set distribution is not the same as the test set distribution, and only a few documents in the entire training set are of class c , but

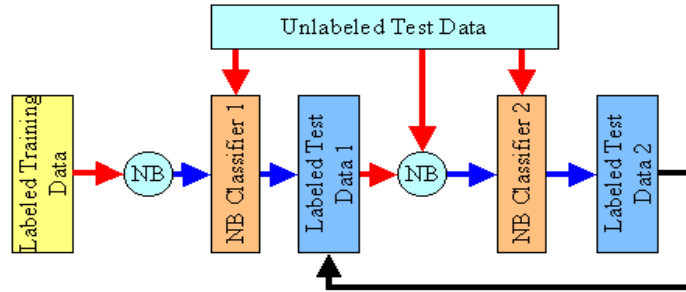


Figure 1: A visual depiction of the transductive naive Bayes algorithm.

nearly all of the documents in the test are of class c . Using the test data distribution while learning allows the algorithm both to better overcome less representative training data distributions and to tailor a classification to the actual test set. Approximating the $Pr_{test}(c)$ simply by the $Pr_{train}(c)$ leads to an extremely incorrect prediction of the probability of class c for the test data. However, by incorporating the $Pr_{train}(x|c)$, the fact that all documents of class c contain the word w will be used to increase the $Pr_{test}(c)$. In addition, the distribution $Pr_{test}(x|c)$ will also represent the fact that all documents of class c contain the word w .

By using the training data to label the test data and then using the resulting labeled test data to estimate the optimal classifier for the test data, transductive naive Bayes uses the distribution of the training and test data to account for any disparities between the distribution of the training and test data. It therefore allows an unrepresentative training data distribution to still be useful as well as allowing the classifier to be tailored to the actual test data. If the transductive naive Bayes classification is more accurate than the standard naive Bayes classifier, then perhaps using the initial transductive naive Bayes classification as a hypothesis of the test data classification, and estimating $Pr(c)$ and $Pr(x|c)$ for each class using this classification, then reclassifying the test data, would be even more accurate.

5 The Transductive Naive Bayes Implementation

As suggested in the last section, an iterative algorithm is implemented. The algorithm takes a mixture of the classifier based on only the training data and the classifier based on only the labeled test data. At each iteration, the weight of the classifier based on the labeled test data is increased. Figure 1 shows a visual depiction of this algorithm.

The algorithm begins by first labeling the test data based on the naive Bayes classifier learned from only the training data. This labeled test data is then used to train another naive Bayes classifier. A weighted sum of the classifications given by the training data classifier and the test data classifier is used to reclassify the test data. The pseudocode for this algorithm is shown in Figure 2. In step (1) of this algorithm, `TestWeight` is initialized. This parameter corresponds to the importance of the classification suggested by the test data distribution. It is in contrast to `TrainWeight`, which corresponds to the importance of the classification suggested by the training data distribution. In step (2), the naive Bayes classifier is learned from the training data. The function `solve_naive_bayes` learns the naive Bayes classifier from the data

Transductive Naive Bayes($Train, UTest, TrainWeight, MaxTestWeight$):

Train is the labeled training data, UTest is the unlabeled test data, TrainWeight is the weight of the training data examples, and MaxTestWeight is the maximum weight of the test examples.

- (1) $TestWeight \leftarrow 1$.
- (2) $h_{train} \leftarrow solve_naive_bayes(Train)$.
- (3) $LTest \leftarrow \{(x, argmax_c(h_{train}(x, c))) | x \in UTest\}$.
- (4) While $TestWeight < MaxTestWeight$, do:
 - (5) $h_{test} \leftarrow solve_naive_bayes(LTest)$.
 - (6) $h \leftarrow \frac{(TrainWeight)h_{train} + (TestWeight)h_{test}}{TrainWeight + TestWeight}$.
 - (7) $LTest \leftarrow \{(x, argmax_c(h(x, c))) | x \in UTest\}$.
 - (8) $TestWeight = 2TestWeight$.

+ (9) Return $LTest$.

Figure 2: Algorithm for transductive naive Bayes.

input to it, as described in section 2. The naive Bayes classifier is learned using the “rainbow” naive Bayes package [McCallum, 1996]. In step (3), this classifier is used to label the test data. The program iterates over the loop started in step (4). In step (5) a naive Bayes classifier is learned from the test data. In step (6) a new classifier is defined to be the weighted sum of the classifier learned from the training data and the classifier learned from the labeled test data. In step (7), this classifier is used to relabel the test data. In step (8), the weight of the test data is increased, and the loop repeats. Finally, in step (9), the labeled test data is returned. Code for this algorithm is available at <http://www.cs.ucsd.edu/users/kbranson/tnb.pl>, and rainbow is available at <http://www.cs.cmu.edu/mccallum/bow>.

This algorithm was tested experimentally, and the results are reported in the next section.

6 Empirical Analysis

Experiments were performed to determine how transductive naive Bayes compared to other established text classification algorithms. In addition, experiments were performed in an attempt to determine how naive Bayes and transductive inference interact. Finally, experiments were performed to investigate the effects of the different parameters of transductive naive Bayes.

6.1 Experimental Data Set

The transductive naive Bayes algorithm was tested on the Reuters-21578 data set, a collection of documents from the 1987 Reuters newswire. This data set is available at <http://www.research.att.com/~lewis/reuters21578.html>. Reuters-21578 is an established text classification task in which the topic of a document is to be identified. Using the modApte split, each document of both the training and test data sets are initially labeled with any number of topics. Preprocessing of the documents included removing the topic label, word stemming, and stop word removal. The

documents were then converted into document vectors, as described in section 2, and input into the learner.

6.2 Experimental Measures

In my experiments, the task of the classifier was to determine if the document was of a certain topic or not. For example, the most common topic in the modApte split of the Reuters-21578 documents is “earn.” The training documents were each labeled as positive if their topic was “earn,” and negative if it was not. The task of the classifier was to label the test documents as either positive or negative. This task was performed with the 10 most common topics.

All results are described using the Precision/Recall-breakeven point. Precision is a measure of how accurate the classifier is. It is defined as the fraction of documents classified as positive that are actually positive:

$$Precision = \frac{truepositives}{truepositives + falsepositives}.$$

Recall is a measure of how well the classifier finds positive documents. It is defined as the fraction of actually positive documents that are classified as positive:

$$Recall = \frac{truepositives}{truepositives + falsenegatives}.$$

As naive Bayes and transductive naive Bayes both return probabilities of a class which can be used as a ranking, the labeling of the documents can be modified so that only some number of the highest ranked documents are labeled as that class. This number can be selected so that the number of positive documents incorrectly labeled is equal to the number of negative documents incorrectly labeled, $falsepositives = falsenegatives$. If this condition is true, then the precision and recall will be equal. The value of the precision and recall at this point is the Precision/Recall-breakeven point. High Precision/Recall-breakeven points indicate a better classifier.

6.3 Experimental Results

The first set of experiments described compare the performance of transductive naive Bayes on the Reuters-21578 data set with other classification algorithms. The results are summarized in Figure 3. The transductive naive Bayes (TNB) and the standard naive Bayes (NB) experiments are the average of 10 trials, each trial using 17 randomly selected training documents with the same ratio of positive to negative documents, and all 3299 test documents in the modApte split. The transductive naive Bayes results reported in this table are for the simplest possible implementation of the algorithm described in Figure 2: TrainWeight is set to 0 and MaxTestWeight is set to 1; there is no input from the classifier trained on just the training documents when producing the final classification, and only one iteration of the loop is performed. Thus, the algorithm is simply to label the test data using information learned from the training data, then relabel the test data using information learned from the labeled test data. This instantiation of the transductive naive Bayes algorithm is used unless otherwise specified. The Support Vector Machine (SVM) and Transductive Support Vector Machine (TSVM) results are taken from [Joachims, 1999]. The training sets in these results contain 17 documents, though the selection of these documents is unknown. However, the test data set for all the results in this table are the same.

| topic | NB | TNB | SVM | TSVM |
|----------------|------|------|------|------|
| earn (1087) | .905 | .963 | .913 | .954 |
| acq (719) | .679 | .827 | .678 | .766 |
| crude (189) | .423 | .744 | .409 | .836 |
| money-fx (179) | .421 | .713 | .413 | .600 |
| grain (149) | .540 | .831 | .562 | .685 |
| interest (131) | .454 | .714 | .356 | .508 |
| trade (117) | .581 | .834 | .295 | .340 |
| ship (89) | .506 | .819 | .325 | .463 |
| wheat (71) | .505 | .878 | .479 | .544 |
| corn (56) | .506 | .821 | .413 | .437 |
| average (261) | .552 | .814 | .484 | .608 |

Figure 3: Precision/Recall-breakeven points for naive Bayes (NB), transductive naive Bayes (TNB), Support Vector Machines (SVM), and Transductive Support Vector Machines (TSVM). The number in parentheses next to the topic is the number of test documents of that topic out of all 3299 test documents

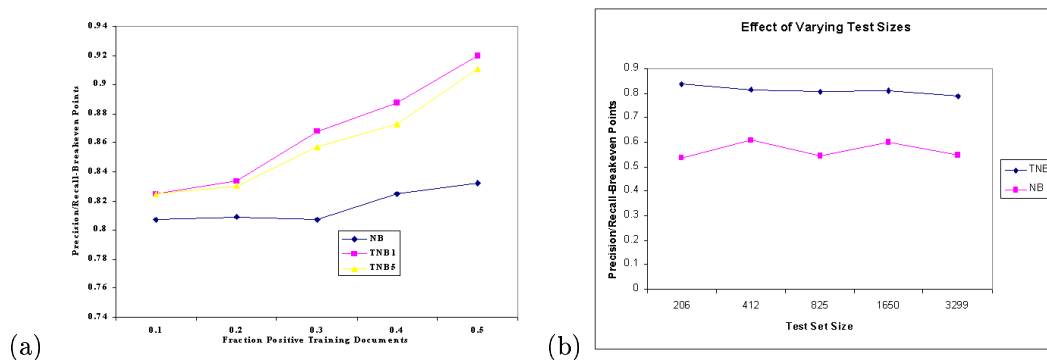


Figure 4: (a) Results of varying the amount the training and test data distributions differ for naive Bayes (NB), transductive naive Bayes with one iteration of transductive inference (TNB1), and transductive naive Bayes with 5 iterations of transductive inference (TNB5) (b) Results of varying the test set size.

As evidenced by these results, transductive naive Bayes outperforms standard naive Bayes in all tests, usually in a greater fraction than TSVMs outperform SVMs. This result suggests that, as it has been argued in sections 2 and 4, transductive inference is a natural addition to naive Bayes.

The next sets of experiments investigate why transductive naive Bayes outperforms naive Bayes. In Figure 3, the topics are listed in order of decreasing occurrence; “earn” occurs the most, 1087 times, while “corn” occurs the least, only 56 times. The results in Figure 3 indicate that naive Bayes’ performance degrades significantly as the number of positive test documents decreases, by nearly a factor of two. However, transductive naive Bayes’ performance degrades only slightly as the number of positive documents decreases. As each test reported in this table uses the same ratio of positive to negative training documents (8 positive, 9 negative), the smaller the number of positive test documents, the more the training data distribution differs from the actual distribution. This suggests that transductive naive Bayes is less affected by this difference in distribution.

To investigate this observation, experiments were performed in which the training distribution was varied, while the test distribution was held constant. Transductive naive Bayes was tested again on the same 10 topics listed in Figure 3, using a constant 56 positive and 56 negative test documents for each topic. The ratio of positive to negative training documents was varied and the results are shown in Figure 4(a). While a constant 100 training documents were used, results are shown for using 10 positive and 90 negative, 20 positive and 80 negative, 30 positive and 70 negative, 40 positive and 60 negative and 50 positive and 50 negative training documents. The greater the x value in the graph in Figure 4(a), the closer the test and training distributions. As in the last experiments, the simplest possible instantiation of the algorithm presented in Figure 2 is used: TrainWeight is set to 0 and MaxTestWeight is set to 1. Thus, the amount the prior probability of each class in the training data and the actual probability of each class in the test data differs is varied.

The results show that transductive naive Bayes was more affected by differences in the training and test distribution than standard naive Bayes, a result which is contrary to my initial hypothesis. However, these are the results for using just one iteration of the transductive inference. With just one iteration, transductive naive Bayes is still heavily dependent on the distribution of the training data. If the fraction of positive training documents is smaller than the fraction of positive test documents, then only a small fraction of the test documents will initially be labeled positive by the classifier learned from the training data (h_{train} in Figure 2). If this is the case, then some words that should be associated with the positive class will instead initially be associated with the negative class, thus causing the difference in training and test data distribution to in fact be amplified. As transductive naive Bayes, with just one iteration, still outperforms naive Bayes, it is likely that key words that should be associated with the positive class are being discovered. With multiple iterations of transductive inference, these key words will be accumulated and transductive naive Bayes will become more resistant to differences in the training and test distributions. Figure 4(a) shows also that transductive naive Bayes with 5 iterations of transductive inference is more resistant to differing training and test distributions. Further testing is required to see if increased iterations will improve this result.

The next set of experiments investigates the information retrieved from the test data by transductive naive Bayes. In particular, these experiments investigate whether or not a larger test data set allows transductive naive Bayes to relay more information than a smaller test data set. If this were the case, then transductive naive Bayes would perform better on larger test data sets than on smaller ones. Experiments were performed to investigate this, in which the training data and the fraction of positive test data were held constant, but the size of the test data set was varied. These experiments were performed on each of the 10 topics listed in Figure 3 and the average results are shown in Figure 4(b). While TSVMs were experimentally shown to perform better on larger test sets, transductive naive Bayes was shown not to be affected by the size of the test data. As transductive naive Bayes did outperform standard naive Bayes, it is likely that transductive naive Bayes retrieves as much information from small test data sets as large ones. This implies that transductive naive Bayes is able to tailor its classification to a small data set, instead of using the additional information from the test data distribution to create a more general model, as TSVMs do.

The next sets of experiments described investigate the importance of the parameters to the transductive naive Bayes algorithm, i.e. TestWeight and TrainWeight, described in section 5. The results shown, up until now, have been for instanti-

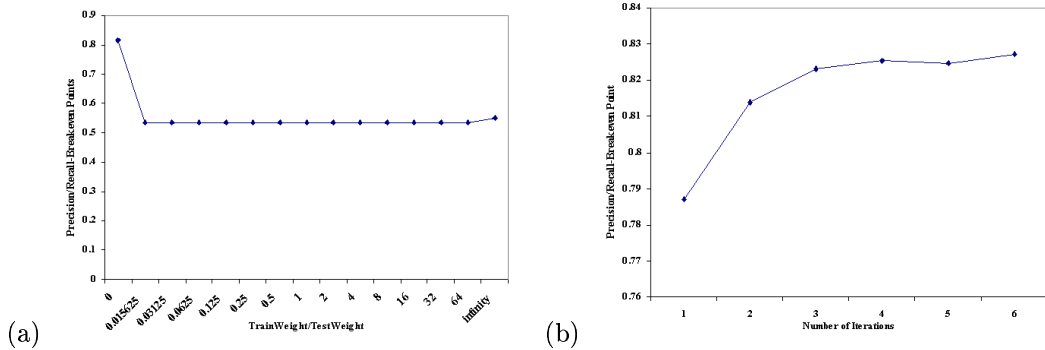


Figure 5: (a) Results of varying the ratio of TrainWeight to TestWeight (b) Results of performing multiple iterations of transductive inference.

ations of the algorithm in Figure 2 in which TestWeight is 1 and TrainWeight is 0. I performed one set of experiments to determine the effect of varying the ratio of TrainWeight to TestWeight. The results of these experiments, using only one iteration of the loop, are shown in Figure 5(a). The graph in Figure 5(a) represents the average Precision/Recall-breakeven points for tests on the Reuters-21578 data set, using the 10 topics as earlier, 17 training documents, and 3299 test documents. These results indicate that the algorithm performs much better when no input from the classifier learned from the training documents is used. In fact, if both TrainWeight and TestWeight are nonzero, then transductive naive Bayes performs slightly worse than standard naive Bayes. This suggests that perhaps there is already a significant amount of input from the training data in the classification. In addition, perhaps the combining of the two classifiers as a weighted sum does not provide accurate probabilities or rankings. One solution to this problem would be to learn a classifier from both the training and test data, where each document has a weight depending on whether it is from the test or training data.

Finally, to test the effects of performing multiple iterations of the loop in the algorithm discussed in the previous section, a set of experiments was performed, using MaxTestWeight set to 16 and TrainWeight set to 0; thus 6 iterations of training are performed. The training and test documents were the same as for the experiments described before this. The results are shown in Figure 5(b). These results indicate that a greater number of iterations is advantageous. This is intuitive, because if the classification provided by one iteration of transductive naive Bayes is more accurate than that provided by standard naive Bayes, the hypothesized labels will also be more accurate and thus the final classification. Further investigation could be made into the benefits of even more iterations of transductive inference.

7 Conclusions and Future Work

This paper presents a technique for adding information about the distribution of the test data set to a naive Bayes learner. This information about the test data distribution allows transductive naive Bayes to work well despite different distributions for the training and test data sets. Allowing for a different distribution for the data set lets the classifier be specific to the test data set. The experimental results presented indicate that transductive naive Bayes is a worthwhile idea, but further investigation needs to be made into its precise benefits. Also, transductive naive Bayes needs to be tested on other text classification tasks to see if it performs

Error Minimizing TNB($Train, UTest, TrainWeight, MaxTestWeight$):

Train is the labeled training data, *UTest* is the unlabeled test data, *TrainWeight* is the weight of the training data examples, and *MaxTestWeight* is the maximum weight of the test examples.

- (1) $TestWeight \leftarrow 1$.
- (2) $h_{train} \leftarrow solve_naive_bayes(Train)$.
- (3) $LTest \leftarrow \{(x, argmax_c(h_{train}(x, c))) | x \in UTest\}$.
- (4) While $TestWeight < MaxTestWeight$, do:
 - (5) $h_{test} \leftarrow solve_naive_bayes(LTest)$.
 - (6) $h \leftarrow \frac{(TrainWeight)h_{train} + (TestWeight)h_{test}}{TrainWeight + TestWeight}$.
 - (7) $LTest \leftarrow \{(x, argmax_c(h(x, c))) | x \in UTest\}$.
 - (8) While there exists $(x, c) \in LTest$ s.t. $Error(Train, LTest - (x, c) + (x, c)) < Error(Train, LTest)$, $LTest \leftarrow LTest - (x, c) + (x, c)$.
 - (9) $TestWeight = 2TestWeight$.
- (10) Return $LTest$.

Figure 6: Algorithm for transductive naive Bayes which minimizes classification error on the training set on each iteration.

equally well. In addition, further investigation can be made into ways to improve transductive naive Bayes, for example, by increasing the number of iterations of transductive inference used.

A separate idea for improving transductive naive Bayes is to minimize classification error on the training set on each iteration of the loop in Figure 2. The original idea of transductive inference was to use the distribution of the test data to minimize error. The implementation of transductive naive Bayes described in this paper does not minimize error. One simple method of minimizing error would be to try swapping the labels of the test document in the loop with the goal of minimizing the error on the training set. An algorithm for this is shown in Figure 6. Whether or not error minimization helps transductive naive Bayes achieve its goals, as well as whether the added work for each classification is worth the trouble, has yet to be investigated.

References

- [Hamerly, 2001] Hamerly, Greg, and Elkan, Charles. (2001) Bayesian Approaches to Failure Prediction for Disk Drives. *Proceedings of the Eighteenth International Conference on Machine Learning (ICML'01)*.
- [Joachims, 1999] Joachims, Thorsten. (1999) Transductive Inference for Text Classification using Support Vector Machines. *Proc. 16th International Conf. on Machine Learning*.
- McCallum, Andrew. (1996) Bow: A toolkit for Statistical Language Modeling, Text Retrieval, Classification and Clustering. <http://www.cs.cmu.edu/mccallum/bow>.
- [Mitchell, 1997] Mitchell, Tom M. (1997) Machine Learning. Boston, Massachusetts, McGraw Hill.
- Zhou, Wei and Greiner, Russell. (2001) Supervised Learning of Belief Net Classifiers. *Technical Report, University of Alberta*.