Analyzing the Effectiveness and Applicability of Co-training

Kamal Nigam
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
knigam@cs.cmu.edu

Rayid Ghani
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
rayid@cs.cmu.edu

ABSTRACT
Recently there has been significant interest in supervised learning algorithms that combine labeled and unlabeled data for text learning tasks. The co-training setting [1] applies to datasets that have a natural separation of their features into two disjoint sets. We demonstrate that when learning from labeled and unlabeled data, algorithms explicitly leveraging a natural independent split of the features outperform algorithms that do not. When a natural split does not exist, co-training algorithms that manufacture a feature split may out-perform algorithms not using a split. These results help explain why co-training algorithms are both discriminative in nature and robust to the assumptions of their embedded classifiers.

Categories and Subject Descriptors
I.2.6 [**Artificial Intelligence**]: Learning; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering*

Keywords
co-training, expectation-maximization, learning with labeled and unlabeled data, text classification

1. INTRODUCTION
There has been much recent interest in supervised learning algorithms that combine information from labeled and unlabeled data. Such approaches include using Expectation-Maximization to estimate maximum a posteriori parameters of a generative model [16], using a generative model built from unlabeled data to perform discriminative classification [10], and using transductive inference for support vector machines to optimize performance on a specific test set [12]. Each of these results, and others, has shown that using unlabeled data can significantly decrease classification error, especially when labeled training data are sparse.

A related set of research uses labeled and unlabeled data in problem domains where the features naturally divide into two disjoint sets. For example, Blum and Mitchell [1] present an algorithm for classifying web pages that builds two classifiers: one over the words that appear on the page, and another over the words appearing in hyperlinks pointing to that page. Riloff and Jones [17] learn information extractors for geographic locations by a meta-bootstrapping process that builds a term-matching classifier over word tokens, and a context rule classifier over the neighboring words of the token. Yarowsky [18] performs word sense disambiguation by building a sense classifier using the local context of the word and a classifier based on the senses of other occurrences of that word in the same document. Finally, Collins and Singer [4] introduce the CoBoost algorithm to perform named entity classification which boosts classifiers that use either the spelling of the named entity or the context in which that entity occurs.

Datasets whose features naturally partition into two sets, and algorithms that use this division, fall into the *co-training* setting [1]. Blum and Mitchell [1] show that under the assumptions that (1) each set of features is sufficient for classification, and (2) the two feature sets of each instance are conditionally independent given the class, PAC-like guarantees on learning with labeled and unlabeled data hold.

This paper explores questions of why co-training algorithms are successful: do they actually leverage independent divisions of features, or do these algorithms use unlabeled data only as well as those that ignore the feature division? How sensitive are co-training algorithms to the correctness of their assumptions? Can co-training algorithms be applied to datasets without natural feature divisions?

We show that when an independent and redundant feature split exists, co-training algorithms outperform other algorithms using unlabeled data. For example, on a constructed text classification task based on UseNet newsgroups, Blum & Mitchell's co-training algorithm achieves 3.7% error using only 6 labeled documents and 1000 unlabeled documents, while an EM-based approach achieves a significantly higher 8.9% error. Even on real-world text data sets with no natural feature divisions, co-training algorithms outperform an EM approach by using random splits of the features. From these results, we are able to understand more about why co-training algorithms work, and give prescriptive suggestions for how to improve them.

2. THE CO-TRAINING SETTING

The co-training setting applies when a dataset has a natural division of its features. For example, web pages can be described by either the text on the web page, or the text on hyperlinks pointing to the web page. Traditional algorithms that learn over these domains ignore this division and pool all features together. An algorithm that uses the co-training setting may learn separate classifiers over each of the feature sets, and combine their predictions to decrease classification error. Co-training algorithms using labeled and unlabeled data explicitly leverage this split during learning.

Blum and Mitchell [1] formalize the co-training setting and provide theoretical learning guarantees subject to certain assumptions. In the formalization, each instance is described by two sets of features. Under certain assumptions Blum and Mitchell [1] prove that co-training algorithms can learn from unlabeled data starting from only a weak predictor. The first assumption is that the instance distribution is *compatible* with the target function; that is, for most examples, the target functions over each feature set predict the same label. For example, in the web page domain, the class of the instance should be identifiable using either the hyperlink text or the page text alone. The second assumption is that the features in one set of an instance are *conditionally independent* of the features in the second set, given the class of the instance. This assumes that the words on a web page are *not* related to the words on its incoming hyperlinks, except through the class of the web page, a somewhat unrealistic assumption in practice.

They argue that a weak initial hypothesis over one feature set can be used to label instances. These instances seem randomly distributed to the other classifier (by the conditional independence assumption), but have classification noise from the weak hypothesis. Thus, an algorithm that can learn in the presence of classification noise will succeed at learning from these labeled instances.

However, real-world data sets with a feature division will not completely satisfy the strict requirements of compatibility and conditional independence. It is thus an important empirical question to ask how sensitive are co-training algorithms to the correctness of these assumptions.

3. ALGORITHMS FOR LEARNING

In this section we present two algorithms that learn from labeled and unlabeled data, one that uses the co-training setting (the co-training algorithm), and one that does not (EM). We will then compare these algorithms, and others that use and ignore a given feature split, on a series of text classification tasks. We choose these algorithmic representatives, because (1) both have been experimentally successful on similar text classification domains, and (2) they can both be instantiated with the underlying representation of a naive Bayes classifier, which makes for a more direct comparison. We begin with a brief overview of naive Bayes text classification.

3.1 Naive Bayes

Naive Bayes is a simple but effective text classification algorithm for learning from labeled data alone [13, 14]. The parameterization given by naive Bayes defines an underlying generative model assumed by the classifier. In this model, first a class is selected according to class prior probabilities. Then, the generator creates each word in a document by drawing from a multinomial distribution over words specific to the class. Thus, this model assumes each word in a document is generated independently of the others given the class.

Naive Bayes forms maximum a posteriori estimates for the class-conditional probabilities for each word in the vocabulary, V, from labeled training data $\mathcal{D}$. This is done by counting the frequency that word w_t occurs in all word occurrences for documents d_i in class c_j, supplemented with Laplace smoothing to avoid probabilities of zero:

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

where $N(w_t, d_i)$ is the count of the number of times word w_t occurs in document d_i, and where $\mathrm{P}(c_j|d_i) \in \{0, 1\}$ as given by the class label.

The prior probabilities of each class are calculated in a similar fashion, counting over documents instead of words:

$$\mathrm{P}(c_j) = \frac{1 + \sum_{i=1}^{|\mathcal{D}|} \mathrm{P}(c_j|d_i)}{|\mathcal{C}| + |\mathcal{D}|}. \quad (2)$$

At classification time we use these estimated parameters by applying Bayes' rule to calculate the probability of each class label and taking the most probable class as the prediction. This makes use of the naive Bayes independence assumption, which states that words occur independently of each other, given the class of the document:

$$\begin{aligned} \mathrm{P}(c_j|d_i) &\propto \mathrm{P}(c_j)\mathrm{P}(d_i|c_j) \\ &= \mathrm{P}(c_j)\prod_{k=1}^{|d_i|} \mathrm{P}(w_{d_{i,k}}|c_j). \end{aligned} \quad (3)$$

The overly-strong word independence assumption causes naive Bayes to predict extreme (nearly 0 or 1) posterior class probabilities. However, while these estimates are poor, naive Bayes classification accuracy is typically high. This can be explained in part because classification is only a function of which class has the maximum posterior, and is not concerned with its actual value [8].

3.2 Expectation-Maximization

If we extend the supervised learning setting to include unlabeled data, the naive Bayes equations presented above are no longer adequate to find maximum a posteriori parameter estimates. The Expectation-Maximization (EM) technique can be used to find locally maximum parameter estimates.

EM is an iterative statistical technique for maximum likelihood estimation in problems with incomplete data [7]. Given a model of data generation, and data with some missing values, EM will locally maximize the likelihood of the parameters and give estimates for the missing values. The naive Bayes generative model allows for the application of EM for parameter estimation. In our scenario, the class labels of the unlabeled data are treated as the missing values.

In implementation, EM is an iterative two-step process. Initial parameter estimates are set using standard naive Bayes from just the labeled documents. Then we iterate the E- and M-steps. The E-step calculates probabilistically-weighted class labels, $P(c_j|d_i)$, for every unlabeled document using Equation 3. The M-step estimates new classifier parameters using all the documents, by Equations 1 and 2, where $P(c_j|d_i)$ is now continuous, as given by the E-step. We iterate the E- and M-steps until the classifier converges.

In previous work [16], we have shown this technique can significantly increase text classification accuracy when given limited amounts of labeled data and large amounts of unlabeled data. However, on datasets where the assumption correlating the classes with a single multinomial component is badly violated, basic EM performance suffers.

3.3 The co-training algorithm

The co-training algorithm[1] explicitly uses a feature split when learning from labeled and unlabeled data. Its approach is to incrementally build classifiers over each of the feature sets. Each classifier is initialized using just the few labeled documents on hand. At every round of co-training each classifier chooses one unlabeled document per class to add to the labeled set of examples.[2] The documents selected are those with the highest classification confidence as given by the underlying classifier. Then, each classifier rebuilds from the augmented labeled set, and the process repeats. In this paper we use naive Bayes for the underlying classifiers. The class probabilities (Equation 3) are the confidence estimates used by co-training. At classification time, the prediction of the underlying classifiers are combined by multiplying the posterior probabilities together, and renormalizing them so they sum to one. Table 1 outlines this process.

The intuition behind the co-training algorithm is that classifier A adds examples to the labeled set that classifier B will then be able to use for learning. If the conditional independence assumption holds, then on average each added document will as informative as a random document, and learning should progress, subject to adding many documents with the wrong class. If the independence assumption is violated, then on average added documents can be less informative and co-training may not be successful.

[1]We distinguish between *the co-training setting* and *the co-training algorithm*, both presented by Blum and Mitchell [1]. We use *co-training algorithms* to refer generically to algorithms using the co-training setting.

[2]If the class frequencies are not even, instead label documents according to the empirical frequencies.

- **Inputs:** An initial collection of labeled documents and one of unlabeled documents.
- Loop while there exist documents without class labels:
    - Build classifier A using the A portion of each document.
    - Build classifier B using the B portion of each document.
    - For each class C, pick the unlabeled document about which classifier A is most confident that its class label is C and add it to the collection of labeled documents.
    - For each class C, pick the unlabeled document about which classifier B is most confident that its class label is C and add it to the collection of labeled documents.
- **Output:** Two classifiers, A and B, that predict class labels for new documents. These predictions can be combined by multiplying together and then renormalizing their class probability scores.

Table 1: The co-training algorithm described in Section 3.3.

4. CO-TRAINING ON A REAL-WORLD DATASET

From the description of the EM and co-training algorithms in Sections 3.2 and 3.3, it is not clear whether co-training should do better than EM on data with divided feature sets, especially when each is based on a naive Bayes classifier. One assumption of naive Bayes is that words in a document occur independently of each other given the class. What benefit is to be gained by further asserting that one *set* of features is independent of another? Certainly, the naive Bayes assumption subsumes this. To answer this question, we compare the performance of these algorithms on a real-world dataset previously used in support of co-training.

4.1 The WebKB Course dataset

In their paper, Blum and Mitchell [1] present experimental results that compare the co-training algorithm with labeled and unlabeled data to the naive Bayes classifier with labeled data alone. Their experimental domain is the WebKB-Course dataset, a collection of 1051 web pages collected from computer science departments at four universities. The task is to identify those that are home pages of academic courses (22% fall into that category). Each example consists of both the words that occur on the web page, as well as words occurring in the anchor text of hyperlinks pointing to that page. The co-training algorithm uses this partition (page vs. hyperlinks) to define the feature split. EM pools these features together though for all algorithms, words that occur in hyperlinks are different features than words that occur in the body of a web page. Thus for the word "career" there are really two features, "career-body" and "career-hyperlink".

We run co-training, EM, and naive Bayes on this dataset for

Table 2: Classification error rates for co-training, EM and naive Bayes on the WebKB-Course dataset. This dataset does not demonstrate that co-training algorithms are better than other algorithms even when the features naturally divide.

Algorithm	# Labeled	# Unlabeled	Error
Naive Bayes	788	–0–	3.3%
Co-training	12	776	5.4%
EM	12	776	4.3%
Naive Bayes	12	–0–	13.0%

ten paired trials of randomly selected train/test/unlabeled splits. Three course documents and 9 non-course documents form the training set. 25% of the documents are held aside as a test set, and the remaining documents are unlabeled. The co-training algorithm proceeds identically as in Blum and Mitchell [1], except that we run co-training until it gives labels to all the unlabeled documents.[3] The performance of co-training continues to improve as it labels more documents. EM proceeds according to Section 3.2. We run EM for seven iterations; EM convergence for text classification is rapid as naive Bayes gives very extreme probability estimates.

4.2 Experimental Results

Table 2 show classification error rates for co-training and EM, in comparison to baselines provided by naive Bayes. Both co-training and EM lower classification error considerably compared to a naive Bayes classifier using just the labeled data. However, EM results in a smaller error than co-training, 4.3% compared to 5.4%. If all the data were labeled, naive Bayes would achieve error of 3.3%.

From these results, we certainly cannot conclude that co-training successfully uses the feature splits provided in the WebKB-Course data, as the performance of EM is better than co-training. Three possibilities could explain why the co-training performance on this dataset is disappointing. First, the WebKB-Course task could be too easy and thus suffers from ceiling effects that make it hard to compare classification algorithms. The EM and co-training error rates are close to the naive Bayes error when all the labels are known. Another possible explanation is that the feature split of the WebKB-Course dataset is not sufficiently independent to allow co-training to perform well. Although hyperlink text and web page body text will typically be authored by different people, it is certainly unreasonable to expect them to be completely independent, as they both refer to the same web page. Co-training algorithms may be quite sensitive to the correctness of this assumption. A final and more serious possibility is that co-training algorithms do not adequately benefit from the existing independence of the feature split and thus do not perform better than EM. In the next section we will explicitly test this hypothesis by constructing a dataset for which this assumption does hold.

[3] There are slight variations between their algorithm and the one presented in Section 3.3. They perform feature selection at each iteration of co-training and we follow Section 3.3. They select from and replenish a limited pool of unlabeled documents. We follow their protocol in this section for strict comparability. In later sections we do not; preliminary experiments on our datasets indicated that the pool did not provide extra benefit, and its removal reduces the number of tunable parameters.

Table 3: The setup of the News 2x2 dataset. This data has class-conditional independence and redundancy between its two feature sets.

Class	Feature Set A	Feature Set B
Pos	comp.os.ms-windows.misc	talk.politics.misc
Neg	comp.sys.ibm.pc.hardware	talk.politics.guns

5. CO-TRAINING WITH INDEPENDENT FEATURE SPLITS

The previous section raises the interesting question of whether co-training algorithms sufficiently leverage a feature split to provide a benefit over algorithms for learning with labeled and unlabeled data that do not explicitly use this knowledge. In this section we show that on datasets for which there truly is independence between the two feature sets, co-training provides a significant improvement over EM.

5.1 The News 2x2 dataset

To test this question, we create a semi-artificial dataset that has the independence properties we seek. We select four newsgroups from the 20 Newsgroups dataset [11]. We create a two-class problem with class-conditional independence by joining together randomly selected documents from each of the first two newsgroups to make positive examples, and joining together randomly selected documents from each of the second two newsgroups to make negative ones. This joining is done such that the words in the first and third newsgroups come from the same vocabulary, while words from the second and fourth newsgroups come from a separate vocabulary. Thus, the word "career" from the first newsgroup is a distinct feature from the word "career" in the second. Table 3 shows the setup of this dataset.

This dataset has some features that make it appropriate for testing the efficacy of co-training. First, and most importantly, there is true class-conditional independence between the words in the two feature sets. This represents an ideal situation for co-training. Second, each feature set taken on its own is sufficient for accurate classification ensuring high compatibility. When taken separately each feature set can be used to train a naive Bayes classifier that reaches error rates of less than 10% with a large amount of training data. Finally, it is appropriate that each of the two subtasks itself consists of real-world data. It keeps the artificiality of the dataset at a minimum, and allows us to draw our conclusions more confidently.

When tokenizing this data, the UseNet headers (including the subject line) are discarded. Words on a stoplist are removed, but no stemming is performed. The word counts of each document are scaled such that each document has constant length, with possibly fractional word counts. Each experiment is run on ten paired randomly-selected test/train/unlabeled splits. Three documents per class form a training set, one thousand documents are left unlabeled, and the remaining 976 documents form the test

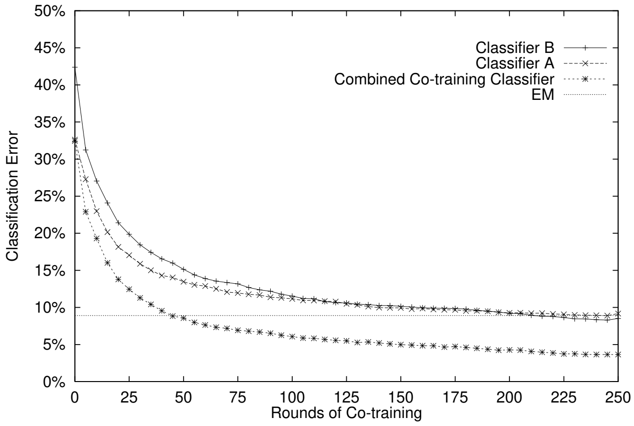

Figure 1: Performance of co-training as it gives labels to more and more unlabeled documents. The combined co-training classifier does better than either of its embedded classifiers because of the independence of their features. The performance of EM is shown as the horizontal line.

set. Based on the initial training set, mutual information feature selection is used to prune the vocabulary to 4000 words. Initial experiments indicated that this feature set size gave best performance for both EM and co-training.

5.2 Co-training outperforms EM

Figure 1 shows the average performance of co-training as it gives labels to more and more documents. At each round, co-training labels four documents (each classifier labels one from each of two classes). Note that the combination of the two embedded classifiers gives significantly lower error than either of the two individually, because of the independence of the features each sees. The question of how co-training performs when this independence does not strictly hold is addressed in the next section.

Table 4 presents the classification errors obtained by co-training and EM. On this dataset with conditional feature set independence, co-training has significantly lower error rates. Starting from a baseline error of 34.0%, co-training uses the unlabeled data to reduce error down to 3.7%, even slightly below the 3.9% achieved by knowing all the labels of the unlabeled data. EM also reduces the error, but only to 8.9%. These results suggest that the co-training algorithm performs better than EM when there is indeed an independent division of the feature space.

However, this result does not allow us to argue that co-training algorithms perform better than non-co-training algorithms in general. Algorithmically, co-training differs from EM in more ways than just its use of the feature split. The other significant difference between them is that co-training labels just a few documents per round; that is, co-training *incrementally* uses the unlabeled data. In contrast, EM probabilistically labels all the data at each round; EM *iteratively* uses the unlabeled data. This difference could account for the benefit of the co-training algorithm instead of its use of the feature split.

Table 4: Classification error rates on the News 2x2 dataset. On a dataset with true class-conditional independence between the two feature sets, co-training outperforms EM, which does not explicitly use the feature split.

Algorithm	# Labeled	# Unlabeled	Error
Naive Bayes	1006	–0–	3.9%
Co-training	6	1000	3.7%
EM	6	1000	8.9%
Naive Bayes	6	–0–	34.0%

5.3 Hybrid algorithms

In order to tease apart the effects of the feature splitting from the effects of the labeling process, we specify two hybrids of EM and co-training to fill the space of algorithms along these dimensions. The first, *co-EM*, is an iterative algorithm that uses the feature split. It proceeds by initializing the A-feature-set naive Bayes classifier from the labeled data only. Then, A probabilistically labels all the unlabeled data. The B-feature-set classifier then trains using the labeled data and the unlabeled data with A's labels. B then relabels the data for use by A, and this process iterates until the classifiers converge. A and B predictions are combined together as co-training embedded classifiers are. In practice, co-EM converges as quickly as EM does, and experimentally we run co-EM for 10 iterations.

The co-EM algorithm can be thought of as a closer match to the theoretical argument of Blum and Mitchell [1] than the co-training algorithm. The essence of their argument is that an initial A classifier can be used to generate a large sample of noisily-labeled data to train a B classifier. The co-EM algorithm does exactly this using one learner to assign labels to all the unlabeled data, from which the second classifier learns from. In contrast, the co-training algorithm learns from only a single example at a time.

The second EM/co-training hybrid is *self-training*. Self-training is an incremental algorithm that does not use the split of the features. Initially, self-training builds a single naive Bayes classifier using the labeled training data and all the features. Then it labels the unlabeled data and converts the most confidently predicted document of each class into a labeled training example. This iterates until all the unlabeled documents are given labels.

Self-training can be viewed as the classification analog of pseudo relevance feedback [6, 2] in information retrieval. In pseudo relevance feedback, an initial query is performed on a corpus. Then the initial query is refined by adding new terms from its best matching documents. Self-training is similar in that it adds its most confident document as a training example and repeats.

Table 5 shows the setup of these four algorithms. Note that co-EM is an algorithm using the co-training setting while self-training is not.

Table 6 shows classification error rates for the four algorithms above. Both algorithms that explicitly use the fea-

Table 5: The space of algorithms using labeled and unlabeled data. The co-training algorithm is an incremental labeling algorithm that utilizes the feature split explicitly. EM iteratively relabels the unlabeled data, but not directly use any feature splits. Co-EM and self-training are hybrid algorithms that combine these properties.

Method	Uses Feature Split? Yes	No
Incremental	co-training	self-training
Iterative	co-EM	EM

Table 6: Classification error rates for four algorithms using labeled and unlabeled data on the News 2x2 dataset. Both algorithms that explicitly using the feature sets are more accurate than algorithms ignoring the feature split.

Method	Uses Feature Split? Yes	No
Incremental	3.7%	5.8%
Iterative	3.3%	8.9%

ture split have lower error rates than either algorithm that does not. Co-training and co-EM have error rates of 3.7% and 3.3%, where self-training and EM have error rates of 5.8% and 8.9% respectively. Given these results, we can finally argue that algorithms that explicitly use an independent and redundant division of the features can be expected to perform better than algorithms that do not. Section 7 discusses why this finding holds, and what it suggests about the nature of co-training.

6. APPLYING CO-TRAINING TO REGULAR DATASETS

Although co-training is a powerful paradigm, it is not widely applicable. Relatively few datasets come with a known, natural division of the features which can reasonably be expected to help for classification. The large majority of datasets have a single set of features with no obvious or natural way to divide them. Yet, if there were sufficient redundancy among the features, and we could identify a fairly reasonable division of them, then co-training algorithms may show similar advantages to those seen in the previous section. To test this idea, we present some initial experiments on applying co-training to regular text datasets. One straightforward way of splitting a feature set is to randomly divide it in two. We use this as our initial step in applying co-training to regular datasets.

6.1 Datasets and Protocol

The first dataset we use is the News 2x2 dataset, but without the knowledge of the natural split. Since we know there is a natural split, we can evaluate how much we lose by choosing a random split instead.

The second dataset we use is the News5 dataset, used previously by McCallum and Nigam [15]. This dataset is the subset of the 20 Newsgroups dataset consisting of the approximately 5000 articles in the 5 comp.* newsgroups. Unlike News 2x2 there is no natural way of splitting these features. When tokenizinig this data, we skip the UseNet headers, use a standard stoplist, perform no stemming and normalize word counts by document length.

Table 7: Classification error rates for four algorithms using labeled and unlabeled data on the News 2x2 dataset. Both algorithms explicitly using the feature splits are more accurate than algorithms ignoring the feature split.

Method	Uses Random Feature Split? Yes	No
Incremental	5.5%	5.8%
Iterative	5.1%	8.9%

Table 8: Classification error rates for four algorithms using labeled and unlabeled data on the News5 dataset.

Method	Uses Random Feature Split? Yes	No
Incremental	28.0%	27.0%
Iterative	29.9%	31.2%

Again, experimental results are shown as averages of ten paired randomly selected train/unlabeled/test splits. For the News5 dataset, ten documents per class are training, 3000 documents are unlabeled, and the remaining are held aside for testing. Feature selection is performed by selecting the top 4000 words by mutual information.

6.2 Experimental Results

Table 7 shows classification error rates for the News 2x2 dataset. The results for EM and self-training are identical to those in Section 5. Notice that again, the two co-training algorithms perform better than the non-co-training ones, but by a smaller margin that if the correct feature division were known. Interestingly, each embedded classifier performs more accurately here than with the ideal feature split of the previous section, indicating a higher degree of compatibility in the random feature split. These results show promise for applying co-training algorithms to flat data. However, recall that this dataset is constructed to be redundant in that it contains twice as much data as is needed for classification. The results indicate that, with enough redundancy in the text, there may be enough natural independence of words to allow co-training to flourish.

Table 8 shows classification errors for News5. Naive Bayes with access to all the labels gets 16.7% accuracy, while with just the initially labeled data it gets 50.1% error. The results for this dataset do not set such a clear trend. The worst performer is EM—the algorithm with a strong probabilistic foundation. The best performer is self-training, an algorithm that does not use feature splitting. However, the co-training algorithm does reduce error over EM by 10%. This seems like encouraging evidence to support our hypothesis that splitting regular datasets into disjoint feature sets and running co-training-like algorithms on it can result in a decrease in classification error compared to regular algorithms like EM.

Note that we split our feature sets in a random fashion; the next step is to develop a splitting algorithm that results in feature sets that are maximally independent. An ideal feature split is one with class-conditional independence of the two sets of features; that is, the *conditional mutual information* between the feature sets is zero. With text data though, there are too many features to reasonably calculate the mutual information between sets of features for a candidate split. However, we can approximate the conditional mutual information criteria between two feature sets by the sum of the pairwise conditional mutual informations for all pairs of words that are in different sets. With this criteria, we can define the following algorithm for splitting a vocabulary of size V into approximately independent parts:

- Calculate the conditional mutual information between every pair of words in the vocabulary.
- Create a V-regular undirected weighted graph with the words as nodes and the weights on the edges being the conditional mutual information between the two nodes that the edge connects.
- Make a 2-way balanced cut in the graph so as to minimize the sum of the weights of the edges that are cut.

The two resulting sets of nodes (words) then form the feature split to which co-training can be applied. However, step 3 of the algorithm is NP-hard so we need to use efficient approximation algorithms. Fortunately, much is known about efficient approximate min-cut graph partitioning algorithms [9]. Experiments using this approach, and ideas in similar directions, are an area of ongoing research.

7. DISCUSSION AND FUTURE WORK

Given the results from the previous sections, what can we conclude about the behavior of co-training? Certainly, results on the News 2x2 dataset show that co-training performs better than EM when the feature set independence assumption is valid. But *why* does co-training do well? Perhaps some insight can be gained by considering instead the question of why EM does *not* do so well. As discussed by Nigam et al. [16], EM is expected to do well when its underlying assumptions about the data are valid. When these assumptions are strongly violated, the performance of EM suffers because it depends on these assumptions when using the unlabeled data. Since our observed performance of EM is poor in comparison to the other algorithms, we argue that the violated naive Bayes assumptions are one cause of this performance.

Yet, the co-training algorithm in this paper also makes the same assumptions (as it too has underlying naive Bayes classifiers), but does not suffer from the violations. Thus we hypothesize that the co-training algorithm succeeds in part because it is more robust to the assumptions made by its underlying classifiers. This can be understood by looking at the differences in how EM and co-training use the underlying assumptions.

EM uses the naive Bayes classifier to assign posterior class probabilities to each unlabeled document. However, as discussed in Section 3.1, these probabilities are poorly estimated because the word independence assumption is violated by text data. Co-training, on the other hand, makes limited use of the assumptions of the underlying classifier. It uses the classifier to rank the documents by confidence, but does not directly use the actual posterior probabilities. This ranking use is a a much weaker use of the independence assumption than EM makes, but still a stronger use than classification makes. Empirical evidence shows that the ranking of naive Bayes scores is well correlated with the correctness of classification [5], and thus co-training's use of the naive Bayes assumptions is not harming performance as it does for EM.

While co-training may be more robust to the violated assumptions of its underlying classifiers, that does not make it immune to violations of its own assumptions of compatibility and feature set independence. In particular, the comparison of ideal feature splits and random splits on the News 2x2 dataset show the sensitivity of co-training to the validity of the feature set independence assumption. As discussed in Section 6.2, we can approximately measure the amount of feature set independence empirically for a given feature split. In future work, we plan to explicitly measure the independence of different feature splits to evaluate the extent to which co-training depends on the correctness of these assumptions.

The robustness of co-training to the underlying classifier assumptions can also be understood in another way. EM is a likelihood-based approach, and nothing about the technique is geared specifically towards classification. Thus, as EM fits the generative model to the unlabeled data, if the natural clustering of the unlabeled data does not correspond to class-based clusters, EM will suffer. Co-training, on the other hand, is a more discriminative approach, in that it tries to add documents to its labeled set that will help with classification. Most incremental co-training algorithms [1, 17, 18] approximate this by adding documents about which it is most confident.

This selection criteria can be improved by making it more directly focused towards the classification task at hand. For example, instead of always adding the most confident examples, one could balance this confidence (which minimizes the risk of adding a misclassified example) with a measure of how much will be learned from the other half of the document. McCallum and Nigam [15] use a prototypicality measure in an active learning setting that approximately measures the benefit of labeling a particular example. More formally, one might quantify the expected reduction in classification error for adding a single document, in the style of Cohn, Ghahramani, and Jordan [3], that mathematically balances the cost of misclassifying an example with the benefit of correctly adding it. These two suggested improvements should allow co-training algorithms to behave even more discriminatively, requiring fewer documents for good performance, and presenting lower error rates than likelihood-based parameter estimation.

As an interesting side point, note that the self-training algorithm outperforms EM on the data sets used in this paper. One possible explanation for this difference is that EM is suffering from getting trapped in local maxima in parameter likelihood space. Self-training may be more resistant to local maxima, because at each round of the algorithm, a new document is added to the labeled training data. By contrast, EM works with the same data at each iteration, and thus can get stuck more easily in a local maxima. This suggests that incremental algorithms may outperform iterative algorithms, so long as they are not led astray by a few mislabeled documents in the early rounds of using the unlabeled data.

In addition to ongoing work on constructing feature splits and making co-training more discriminative, other areas of future work remain. We will investigate the behavior of co-training with underlying classifiers other than naive Bayes. We plan to examine the performance of co-training algorithms on more challenging real-world text datasets drawn from the Web. We hope to use co-training to combine text and non-text features for mixed-media datasets in a natural way. Finally, we plan further a empirical and theoretical examination of the sensitivity of co-training to the assumption of conditional feature set independence.

Acknowledgements

We thank Andrew McCallum for clarifying discussions and suggestions, Tom Mitchell for helpful discussion and Tommi Jaakkola for insights about discriminative classification.

8. REFERENCES

[1] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT '98*, 1998.

[2] C. Buckley, A. Singhal, M. Mitra, and G. Salton. New retrieval approaches using SMART: TREC 4. In *Proceedings of the TREC 4 Conference*, 1996.

[3] D. Cohn, Z. Ghahramani, and M. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[4] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proceedings of EMNLP*99*, 1999.

[5] M. Craven and S. Slattery. Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning*. To appear.

[6] W. B. Croft and D. J. Harper. Using probabilistic models of document retrieval without relevance information. *Journal of Documentation*, 35:285–295, 1979.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[8] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[9] P.-O. Fjallstrom. Algorithms for graph partitioning: A survey. *Linkoping Electronic Atricles in Computer and Information Science*, 3, 1998.

[10] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in NIPS 11*, 1999.

[11] T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of ICML '97*, 1997.

[12] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of ICML '99*, 1999.

[13] D. D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of ECML-98*, 1998.

[14] A. McCallum and K. Nigam. A comparison of event models for naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998. Tech. rep. WS-98-05, AAAI Press. http://www.cs.cmu.edu/∼mccallum.

[15] A. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In *Proceedings of ICML '98*, 1998.

[16] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[17] E. Riloff and R. Jones. Learning dictionaries for information extraction using multi-level boot-strapping. In *Proceedings of AAAI-99*, 1999.

[18] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL-95*, 1995.