Vertical Ensemble Co-Training for Text Classification

GILAD KATZ, Ben-Gurion University of the Negev
CORNELIA CARAGEA, University of North Texas
ASAF SHABTAI, Ben-Gurion University of the Negev

High-quality, labeled data is essential for successfully applying machine learning methods to real-world text classification problems. However, in many cases, the amount of labeled data is very small compared to that of the unlabeled, and labeling additional samples could be expensive and time consuming. Co-training algorithms, which make use of unlabeled data to improve classification, have proven to be very effective in such cases. Generally, co-training algorithms work by using two classifiers, trained on two different views of the data, to label large amounts of unlabeled data. Doing so can help minimize the human effort required for labeling new data, as well as improve classification performance. In this article, we propose an ensemble-based co-training approach that uses an ensemble of classifiers from different training iterations to improve labeling accuracy. This approach, which we call *vertical ensemble*, incurs almost no additional computational cost. Experiments conducted on six textual datasets show a significant improvement of over 45% in AUC compared with the original co-training algorithm.

CCS Concepts: • **Theory of computation → Semi-supervised learning**;

Additional Key Words and Phrases: Co-training, text classification, ensemble

ACM Reference format:
Gilad Katz, Cornelia Caragea, and Asaf Shabtai. 2017. Vertical Ensemble Co-Training for Text Classification. *ACM Trans. Intell. Syst. Technol.* 9, 2, Article 21 (October 2017), 23 pages.
https://doi.org/10.1145/3137114

1 INTRODUCTION

Supervised methods for inducing accurate text classifiers rely on the availability of large amounts of labeled data. However, in many applications, because of the high cost and effort involved in labeling the data, the amount of labeled data is small compared to the amount of unlabeled data. Because of this growing gap between the rate of acquisition of textual data and the rate of manual labeling, there is a significant interest in semisupervised algorithms that can exploit large amounts of unlabeled data together with limited amounts of labeled data in training text classifiers [58].

Co-training, originally developed by Blum and Mitchell [3], is a semisupervised learning method designed to tackle these types of scenarios. Specifically, in addition to a small labeled training set, the co-training algorithm assumes that a large, unlabeled training set is available. One of the

Authors' addresses: G. Katz and A. Shabtai, Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Beer Sheve 8410501, Israel; emails: {giladkz, shabtaia}@bgu.ac.il; C. Caragea, Department of Computer Science and Engineering, University of North Texas, Denton, TX 76203; email: ccaragea@unt.edu.

requirements for co-training to work is that the data can be represented using two different "views" of features on which two separate classifiers are trained. Unlabeled data are then labeled with these two classifiers. Iteratively, each classifier selects a few unlabeled samples for which it has the highest level of certainty in classification. These samples (a few from each class) are added to the labeled training set with the labels predicted by these classifiers in such a way that each classifier "trains" the other by providing it with samples that it may have difficulty classifying on its own. This process continues until some stopping criterion is met (e.g., until a predefined number of iterations is reached or all unlabeled data are used).

Since the introduction of the co-training approach, many extensions have been proposed in the literature. For example, the original co-training has been shown to work well when the two "views" or subsets of features satisfy certain assumptions on "sufficiency" and "independence" [3, 42]. The sufficiency assumption states that each feature subset is sufficient to predict the labels of instances in the test set. The independence assumption states that the views of each feature subset are independent of each other given the label. Recent research has proposed extensions that decompose the feature set into two views when an obvious split is not available [8, 15]. Moreover, recent works have shown that the independence criterion can be relaxed without much impact on the final performance [2].

Despite the many extensions, in our view, co-training methods are still lacking in two important ways. First, although it has been shown in many application domains that unlabeled data can help improve the classification performance [21, 29, 53], the addition of too many unlabeled samples to the labeled training set may very well hurt performance [43]. Three common causes for this are (1) the mislabeling of samples [28]; (2) the labeling of samples from a subgroup in the data, which causes the classifier to "ignore" the other groups; and (3) the instability of the classification model, caused by the small number of training samples.

The second shortcoming is that the final classification model produced by the co-training process does not utilize in any way the knowledge represented by the various classifiers that were created "along the way." This is problematic not only because of the loss of possibly useful information but also because these models have already been generated as part of the co-training process and using them incurs no additional cost.

To address the preceding limitations, we propose an improvement to the co-training algorithm. By using the classification models created in multiple training iterations, we propose creating an *ensemble classifier*. Not only does this approach produce a substantial improvement in classification performance (up to 45%), the combining of a large number of classification models also produces smoother, more robust results.

We evaluate our proposed improvements to co-training on six datasets from the text classification domain. This domain is particularly well suited for the use of co-training since the labeling of texts is both time consuming and expensive, as it requires human taggers. Our experiments on these datasets clearly show that our proposed approach significantly outperforms the "standard" co-training algorithm. In addition, the results clearly show that the *relative improvement* of the proposed methods increases as the number of labeled training samples decreases.

The remainder of this article is organized as follows. Section 2 introduces related work on co-training. In Section 3, we present the "standard" co-training algorithm. In Section 4, we present our proposed method, and in Section 5, we present the results of our evaluation. Last, Section 6 presents our conclusions and directions for future research.

2 RELATED WORK

In this section, we discuss the works most relevant to our study. In their ground-breaking article, Blum and Mitchell [3] introduced the co-training framework, which suggested the use of different perspectives of the same data to train a pair of classifiers. This bootstrapping approach, which was demonstrated on a Web page classification problem, was later applied to many additional domains. These domains include email classification [11, 29], academic home page classification [21], word-sense disambiguation [39], image object recognition [8], tag-based image search [47], sentiment analysis [53], and event identification in news [54].

Despite the diversity of the domains mentioned earlier, most works dealing with co-training closely follow the model described by Blum and Mitchell [3] in the sense that they all use a pair of classifiers trained on a small amount of labeled data. However, there are several works in which attempts were made to modify or improve the original co-training algorithm by Blum and Mitchell [3].

One such interesting attempt was proposed in Denis et al. [11], which applied the co-training algorithm when tagged samples from only one class are available, in addition to a pool of unlabeled samples. By modifying the naive Bayes (NB) algorithm, the authors proposed two versions of the algorithm capable of training the model. This approach may very well prove useful in cases where the dataset in use is highly imbalanced and the chances of obtaining a sufficient number of samples from both classes are small.

Multiview learning (of which co-training is a special case, with two views) has been studied by the machine learning community with great interest [9, 35, 48]. This problem is typically addressed by maximizing "consensus" or agreement among the different views. Previous solutions to multiview learning tend to frame the problem in terms of a global optimization problem and simultaneously learn classifiers for all underlying views. In some cases, the solutions depend on underlying classification algorithm used [4, 20].

Another interesting approach that has been proposed in several works [37, 41, 51, 52] is the use of co-training together with active learning [30]. In active learning, an oracle (e.g., a human expert) is used to analyze select samples and provide their true labels—an accurate but often time- and resource-consuming process. The utilization of these two opposite methods (one is "cheap" to apply but runs the risk of wrong labeling of samples, whereas the other is "expensive" yet accurate) attempts to boost performance by providing the co-training algorithm with a small number of "strategic" samples that can improve the classifiers' performance. One example of a scenario in which active learning may be very beneficial is when the two classifiers of the co-training algorithm are in a strong "disagreement" over the assumed label of a sample. Another approach for dealing with the uncertainty of adding initially unlabeled instances was proposed by Zhang and Zhou [55], using an approach called *COTRADE*. The classifiers' confidence is estimated on unlabeled data using editing techniques and then high-confidence predicted examples that satisfy certain constraints are added to the labeled set. In contrast, our approach uses classifiers that are trained and stored at each iteration.

Transductive learning [25] is a field that bears similarity to co-training. In addition to training a model based on a labeled training set, the said model also aims to derive insights from the entire population (labeled and unlabeled alike). This approach facilitates the generation of models that generalize better, especially when the size of the labeled training set is small. The approach was found to be highly useful in text classification [26] due to its high dimensionality. As shown by Li et al. [32], who used multiple views of the data together with transductive learning, this approach can be integrates with co-training.

Other works have proposed to use ensemble learning, instead of active learning, to improve the co-training process [34, 56, 57]. Ensemble classification consists of the use of multiple classifiers (perspectives on the data) to achieve greater accuracy. These methods depend on the diversity of their member classifiers with various methods being proposed to achieve this goal.

Two of the most common methods in ensemble classification are bagging and boosting. In bagging [5], different sets of instances and (reduced) attributes sets are chosen for each classifier, thus forcing diversity. The best known member of this family of algorithms is the random forest algorithm [6], which usually uses a simple majority vote to reach a final decision. In boosting [17, 18], sets of weak learner are iterative trained while giving additional weight to misclassified instances. Finally, the learners are linearly combined.

The common denominator of bagging and boosting, as well as most other ensemble methods, is the reliance on diversity: the different classifiers must be different from one another to provide different perspectives on the data. In this respect, ensemble methods are very similar to co-training, which also uses two (or more) different "views" of the data to improve classification. Because of this similarity, it is hardly surprising that several methods for the incorporation of co-training and ensemble methods have been proposed.

A successful attempt to incorporate these two methods was the tri-training algorithm [57], which used three classifiers instead of two as proposed in the original work. This approach offered a solution for scenarios in which the two co-training classifiers are in disagreement over the label of an instance. Other approaches successfully demonstrated the benefit of integrating the co-training and boosting methods [31, 34].

In this article, we also propose the use of ensemble learning together with co-training, however our proposed algorithm is easier to implement and less computationally costly, as it "reuses" classifiers from previous training iterations.

3 THE CO-TRAINING ALGORITHM

3.1 The Co-Training Algorithm

Before introducing our proposed co-training algorithm, we first provide some background on the original co-training [3].

The original co-training algorithm assumes that two independent views of the data are available and are used to train two classifiers. The classifiers are trained on a set of labeled training instances L and are used to classify a set of unlabeled training instances U. Then, iteratively, each classifier selects a few instances from U of which it is most certain (for each of the target classes) and adds them to L. For a classifier, the degree of certainty is determined by the probabilities assigned to each unlabeled instance for belonging to class c. The instances are ordered by their probabilities, and we select the top X ranking instances. We then add these instances to the labeled training instances set L. The label assigned to each instance is the one presumed by the classifier.

The intuition behind this method is that the two classifiers are trained on two different views of the data, and therefore one classifier can improve the performance of the other. As one classifier feeds the other instances it may not be certain how to classify, the latter receives instances it has problems classifying on its own, and thus the overall performance improves.

The basic (original) co-training algorithm that we use as our baseline is shown in Algorithm 1. The input to the algorithm is a labeled set of instances (L), an unlabeled set of instances (U), a test set (T), a classification algorithm (I), and the number of iterations of the co-training (n). The product of the algorithm is a classification model whose training set consists of the extended training set (ET). This set consists of the initial labeled set (L) and a portion of the unlabeled set that was labeled throughout the training iterations and added to (L). Upon labeling, these newly-labeled instances are removed from the unlabeled set.

In addition, the final classification model produced by the co-training process completely ignores the knowledge represented by the various classifiers that were created "along the way." To address these limitations, next we present our proposed co-training algorithm.

ALGORITHM 1: The Original Co-Training Algorithm

```
Input:
  $L$ - labeled set
  $U$ - unlabeled set
  $T$ - test set
  $I$ - classification algorithm
  $n$ - number of iterations
Output:
  $ET$ - extended training set
  $h$ - classifier

Apply_Co_Training

$ET \leftarrow L$
for all iterations $i \leq n$ do
    $h_1 \leftarrow TrainClassifier(I, \pi_{\mathbf{x}_1}(ET))$
    $h_2 \leftarrow TrainClassifier(I, \pi_{\mathbf{x}_2}(ET))$
    // label all unlabeled instances with h1
    $L_1 \leftarrow ApplyClassifier(h_1, \pi_{\mathbf{x}_1}(U))$
    // label all unlabeled instances with h2
    $L_2 \leftarrow ApplyClassifier(h_2, \pi_{\mathbf{x}_2}(U))$
    // add instances with highest confidence in $L_1$ and remove them from the unlabeled set
    $ET \leftarrow ET \cup SelectSamplesWithHighestConfidence(L_1)$
    // add instances with highest confidence in $L_2$ and remove them from the unlabeled set
    $ET \leftarrow ET \cup SelectSamplesWithHighestConfidence(L_2)$
end for
$EvaluateClassifiers(h_1, h_2, T)$.
```

3.2 Ensemble of Co-Training Models

One of the prerequisites to the successful application of ensemble learning is diversity. The various classification models partaking in the ensemble need to offer varying "perspectives" (i.e., classifications or levels of confidence) regarding the labels of the instances of the training and validation sets [13]. The success of our proposed approach depends on the co-training models generated throughout the various iterations to possess this trait.

To determine whether this is the case, we review two previous studies examining this subject. Mihalcea [39] analyzed the performance of the co-training algorithm throughout the training process in an attempt to determine the optimal number of training iterations. The analysis presented in this study shows significant fluctuations in precision for consecutive iterations, a fact that leads us to conclude that the models are significantly different and therefore could be useful in an ensemble setting. Another study that supports this conclusion is that of Katz et al. [28], who demonstrate that different strategies for selecting additional instances have significant performance on the performance (i.e., the classification model) of the co-training algorithm.

Intuitively, this conclusion is to be expected. Given the fact that the classification models generated during the co-training process are created using only a small number of instances, the addition of even a few instances is very significant percentage -wise. These changes are likely to be even more significant for classifiers such as decision trees, which utilize a hierarchical ordering of features with specific "split points" for each attribute (e.g., "if $x < 4$. take the left branch and otherwise the right").

4 THE PROPOSED METHOD

As explained in Section 2, existing approaches to the use of ensembles with co-training have two drawbacks in our view: the added computational cost of using multiple classifiers in every iteration and the fact that there might not be enough "informative" features to make the different classifiers sufficiently different from one another (i.e., useful).

We present an ensemble solution designed to address the two shortcomings mentioned previously. Instead of using *horizontal* ensemble—the training of multiple classifiers on the same data at each iteration—we propose a *vertical* ensemble solution that utilizes the models created during the previous training iterations.

The proposed algorithm is presented in Algorithm 2. The two classifiers that are generated at each iteration are used in conjunction with the classifiers of previous iterations to rank the unlabeled samples. It is important to note that the splitting of the data into two "perspectives" remains, with each current model being integrated only with the previous models that were trained on the same data. The integration of the different classifiers is done by averaging the probabilities that they assign to the analyzed instances of belonging to each of the possible classes. This technique, known as distribution summation, is recommended by Menahem et al. [38]. The authors of this study have shown that averaging the probabilities obtained by the various classifiers is often superior to other methods (e.g., Bayesian combination [7]).

Once the training iteration is complete, the two models that were generated during the current iteration are saved for future use. Upon the termination of the training process, a set of the most recent classifiers are used in an ensemble to classify the test set.

The vertical ensemble solution offers significant advantages in running time and memory usage compared to previously proposed ensemble solutions while also addressing their possible performance shortcomings. By storing the models generated previously, the number of classification models that needs to be generated at each iteration remains two. This corresponds to the two views used by the algorithm and is identical to the original co-training algorithm. Moreover, the addition of new samples to the (very small) labeled dataset is likely to generate different models for different training iterations, thus providing the diversity required for a successful ensemble.

Although to the best of our knowledge this type of approach has not been previously used in the field of co-training, it has proven effective in another domain. In the field of hyperparameter optimization [16], the authors opted to utilize the multiple models generated in the search process instead of simply discarding them. However, their implementation can be considered "horizontal," as their models are not run on an evolving version of the same data.

5 EVALUATION

Our experiments were designed in an attempt to answer a set of questions: (1) How beneficial is the application of the proposed ensemble method to the co-training process? (2) What is the influence of varying training set sizes on the performance of algorithms? (3) Does the performance vary substantially with the use of different classification algorithms?

We begin by describing the evaluated datasets and their characteristics in Section 5.1. Section 5.2 reviews the experimental setting. In Section 5.3, we present the various configurations that were evaluated, and in Section 5.4, we analyze the results.

5.1 The Datasets

The proposed methods were tested on six datasets, all from the text classification domain: Spam, WebKB (which was used in the original co-training work by Blum and Mitchell [3]), Reuters RCV1, Cora, WebSpam, and Large Movie Reviews dataset [36]. All datasets are well known and publicly

Table 1. Five Datasets Used for the Evaluation and Their Characteristics

Dataset Name	Items (#)	Attributes (#)	Imbalance
Spam	4,601	57	1.53
WebKB	1,051	7,344	3.57
Reuters	657,399	47,236	1.1
Cora	3,191	3,774	2.43
WebSpam	350,000	254	1.54
Large Movie Reviews	100,000	89,527	1

ALGORITHM 2: The Co-Training Algorithm With the Use of Ensemble

```
Input:
  $L$ - labeled set
  $U$ - unlabeled set
  $T$ - test set
  $I$ - classification algorithm
  $n$ - number of iterations
  $e$ - number of iterations back to include in the ensemble
Output:
  $ET$ - extended training set
  $h$ - classifier

$Apply\_Co\_Training$

$ET \leftarrow L$
for all iterations $i \leq n$ do
    $h_1 \leftarrow TrainClassifier(I, \pi_{\mathbf{x}_1}(ET))$
    $h_2 \leftarrow TrainClassifier(I, \pi_{\mathbf{x}_2}(ET))$
    // label all unlabeled instances with $h_1$
    $L_1 \leftarrow ApplyClassifier(h_1, \pi_{\mathbf{x}_1}(U))$
    // label all unlabeled instances with $h_2$
    $L_2 \leftarrow ApplyClassifier(h_2, \pi_{\mathbf{x}_2}(U))$
    // add instances with highest confidence in $L_1$ and remove them from the unlabeled set
    $ET \leftarrow ET \cup SelectSamplesWithHighestConfidence(L_1)$
    // add instances with highest confidence in $L_2$ and remove them from the unlabeled set
    $ET \leftarrow ET \cup SelectSamplesWithHighestConfidence(L_2)$
    // save the first classifier for iteration $i$
    $SaveClassifierPerIteration(i, 1, h_1)$
    // save the second classifier for iteration $i$
    $SaveClassifierPerIteration(i, 2, h_2)$
end for
// create the ensemble by combining the classifications of the $e$ latest classifiers on the test set using distribution summation
return $ApplyDistributionSummation(T, e, n)$
```

accessible. The properties of the datasets are presented in Table 1. Multiclass datasets were transformed into binary datasets. For example, for WebKB, we used course versus noncourse (as in Blum and Mitchell [3]), whereas for Cora and RCV1, we used the most populous class as one class with the other class containing the remaining instances (i.e., neural networks vs. the rest, and CCAT vs. the rest, respectively).

We chose to focus on the analysis of text in our experiments, as it is one of the most challenging areas for collecting large amounts of high-quality data. The difficulty arises from the fact that accurate labeling of text still requires human taggers, who can identify elements such as cynicism, word games, and irregular expressions. Case in point: a previous study [27] by one of the authors required human annotation of automatically transcribed text. The tagging of approximate 500 conversation required close to 5 months of work.

It is also important to note that the datasets represent multiple domains of classification tasks: news, Web pages, spam, and reviews (opinions). This diversity enables us to evaluate the robustness of the proposed algorithm as well as its applicability in different domains.

5.2 Data Preparation and Experimental Configuration

To enable a full and informative evaluation of the proposed methods, we performed extensive evaluation with multiple configurations. It is also important to note that all experiments described in this section were run twice: once with the NB algorithm (which was used in the original co-training work) and once with the support vector machine (SVM) algorithm—a popular classifier that is often used in a co-training framework (e.g., see Chen et al. [8]).

Each dataset was preprocessed in the following way:

(1) Four different labeled training set sizes were used: 40, 60, 80, and 100 samples. In experiments where the validation sets were used, the labeled samples were evenly split between the training and validation sets.
(2) Each dataset was randomly partitioned into four disjoint sets: labeled (training set), unlabeled (training set), validation (which was left empty in some experiments), and test. This was done 20 times for each dataset and for each of the four labeled training set sizes. All algorithms presented in this article were run on the same random partitions to ensure the consistency of the results (if no validation was done, these samples were added to the labeled training set).
(3) During each training iteration, each of the two classifiers selected two samples from each of the possible classes and added them to the labeled training set.

Although Blum and Mitchell [3] used a dataset that had a clear division of features (textual features and links of Web pages), this is not the case for all of the other datasets used in our experiments. For this reason, we randomly and evenly split the features into two groups (for each dataset, the same partition was used in all experiments). We hypothesize that the large number of features in each group will provide the diversity required for the co-training process to succeed. In doing so, we rely on the work by Balcan et al. [1], who show that the independence assumption could be relaxed, and on a study by Gollapalli et al. [22], who demonstrate that random partitions are also effective and could even outperform "natural" splits in co-training.

The probability distribution of each classifier (i.e., the probability assigned to an instance for belonging to each class) is obtained in a manner consistent with each algorithm's inner workings. The NB algorithm is probabilistic by nature and operates under the assumption that all features are independent. Therefore, the probability for each class c is computed by multiplying the apriori probability of the class with the conditional probability of each feature given that the instance is of class c. The a priori and conditional probabilities are computed from the training set. For the SVM classifier, the probabilities are computed using the Platt scaling algorithm [45], which applies logistic transformation on the SVM scores to obtain the probability of each class. In our experiments, we used the Weka implementation of SVM with the option setBuildLogisticModels(true), which generates these probabilities.

A large majority of existing publications in the field of co-training use a predefined number of training iterations [40, 46], similar to what was done in the original work. This is a common approach that also enables easy comparison to works published previously. This approach, however, may not always produce the optimal results for the co-training algorithm; as shown in experiments conducted by Mihalcea [39], it may sometimes be better to use smaller number of training iterations (in the work mentioned earlier, the number was six to seven iterations). Based on these results (and empirical evaluation of larger iterations), we cap the number of training iterations for our algorithm at nine.

5.3 Evaluated Algorithms and Configurations

We conducted a thorough evaluation of the proposed method described in Section 3 with the following settings:

(1) **Single classifier** on all labeled instances and all features. This experiment is included as a baseline to determine whether the use of co-training improves performance.

(2) **Original co-training, stop after X iterations** (no validation). This set of experiments was based on Mihalcea [39], which showed that the performance of the co-training algorithm may "peak" after a small number of iterations. For this reason, we evaluated the performance of the original co-training algorithm after three, six, and nine iterations. In all experiments, we used the Weka ML platform [23] with the default parameters for all algorithms.

   It should be noted that although previous studies exist where parameters such as the number of iterations can be tuned [16, 50], the implementation and adaptation of these techniques to the domain of co-training is far from trivial. For this reason, we consider this subject beyond the scope of this work and plan to address it in the future.

(3) **Ensemble co-training** (the proposed algorithm). We evaluated multiple configurations:
   - With varying numbers of models making up the ensemble (always the models of the most recent iteration). The values of this parameter were set to 3, 5, 10, and all (all models generated up to that point). Since the best results were obtained by using the latter configuration (i.e., all previously generated models), we present these throughout this work. For full results, see Figure A1 in Appendix A.
   - With varying numbers of training iterations to make sure that the evaluation is correct, the same number of training iterations was used as in (3): three, six, and nine training iterations.

5.4 Experimental Results

The performance of the proposed algorithms is presented in Tables 2 and 3, where we present the AUC (area under the ROC curve [10]) of all proposed algorithms. To demonstrate that the improvement in AUC performance is reflected in other metrics as well, we present the precision-recall curves for one of our experiments (to not burden the reader, we do not present all possible dataset/algorithm/training set size configurations). The results for the SVM algorithm with a training size of 40 are presented in Figure 1(a) through (e).

As can be seen from the figures, the precision-recall curves also clearly show that the proposed method outperforms the "standard" co-training algorithm on all five datasets. In addition to showing higher precision, they also cover a much larger area of the "recall" axis. This reveals strong evidence that the classification models produced by the proposed methods are not only superior to the original co-training algorithm but are also more refined, offering a wider range of confidence level in the classification.

Table 2. AUC Results of the Various Co-Training Methods When Using the SVM Classifier

Dataset	Train Size	Single Classifier	CT 3 Iteration	CT 6 Iteration	CT 9 Iteration	EnsCT 3 Iteration	EnsCT 6 Iteration	EnsCT 9 Iteration
Cora	40	0.21	0.43	0.43	0.43	0.52	0.58	**0.60**
Cora	60	0.25	0.42	0.44	0.44	0.52	0.57	**0.60**
Cora	80	0.35	0.51	0.51	0.52	0.61	0.65	**0.67**
Cora	100	0.34	0.49	0.48	0.48	0.58	0.62	**0.63**
Spam	40	0.67	0.55	0.25	0.08	**0.78**	**0.78**	0.77
Spam	60	0.67	0.53	0.26	0.12	**0.75**	**0.75**	**0.75**
Spam	80	0.69	0.56	0.30	0.15	0.75	**0.76**	**0.76**
Spam	100	0.70	0.59	0.38	0.16	**0.76**	**0.76**	**0.76**
WebKB	40	0.51	0.63	0.66	0.68	0.73	0.76	**0.80**
WebKB	60	0.54	0.65	0.65	0.64	0.71	0.75	**0.77**
WebKB	80	0.56	0.65	0.71	0.71	0.72	0.79	**0.83**
WebKB	100	0.48	0.6	0.64	0.66	0.67	0.73	**0.77**
WebSpam	40	0.60	0.73	0.72	0.73	0.8	0.83	**0.84**
WebSpam	60	0.67	0.77	0.76	0.75	0.81	0.82	**0.83**
WebSpam	80	0.63	0.75	0.74	0.74	0.82	0.84	**0.85**
WebSpam	100	0.69	0.78	0.77	0.76	0.82	0.84	**0.85**
RCV1	40	0.47	0.68	0.69	0.69	0.68	0.74	**0.76**
RCV1	60	0.59	0.73	0.74	0.75	0.71	0.77	**0.80**
RCV1	80	0.67	0.78	0.78	0.79	0.77	0.81	**0.83**
RCV1	100	0.68	0.78	0.78	0.79	0.77	0.81	**0.83**
Movie reviews	40	0.28	0.42	0.43	0.47	**0.60**	0.59	**0.60**
Movie reviews	60	0.36	0.47	0.47	0.47	0.53	0.57	**0.63**
Movie reviews	80	0.42	0.49	0.51	0.53	0.59	0.62	**0.66**
Movie reviews	100	0.44	0.5	0.52	0.52	0.61	0.66	**0.69**

Note: The highest results in every experimental configuration are shown in bold.

The analysis of the AUC performance of the analyzed algorithms, shown in Tables 2 and 3, enabled us to conclude the following:

(1) The proposed method significantly outperforms both the basic co-training algorithm and the single classifier baseline. By using the paired *t*-test on the AUC values obtained for each experiment, we were able to determine that our approach outperforms the two baselines in all experiments for the NB algorithm and in five out of six datasets (all but Spam) for the SVM algorithm. The improvement was found to be statistically significant with a p-value of $p \leq 0.01$. In the Spam dataset, our approach is significantly better than the original co-training algorithm, but there is no statistically significant difference from the single classifier algorithm. This improvement is particularly significant given the fact that it comes at (almost) no additional computational cost compared to the original co-training algorithm.

(2) The ensemble co-training approach shows a higher relative improvement when the size of the labeled training set is smaller. This fact is very encouraging, as it shows that the proposed algorithm is useful in the more difficult learning scenarios.

(3) The performance of the co-training algorithm when using SVM produced a significantly higher performance than the use of the NB algorithm with co-training. This is consistent with the previous work by Kiritchenko and Matwin [29].

Table 3. AUC Results of the Various Co-Training Methods When Using the NB Classifier

Dataset	Train Size	Single Classifier	CT 3 Iteration	CT 6 Iteration	CT 9 Iteration	EnsCT 3 Iteration	EnsCT 6 Iteration	EnsCT 9 Iteration
Cora	20	0.37	0.52	0.49	0.49	0.63	**0.66**	**0.66**
Cora	30	0.47	0.55	0.54	0.54	0.66	0.69	**0.70**
Cora	40	0.52	0.57	0.58	0.57	0.68	0.71	**0.72**
Cora	50	0.52	0.57	0.57	0.56	0.69	0.7	**0.72**
Spam	20	0.78	0.6	0.52	0.57	**0.85**	0.83	0.80
Spam	30	0.80	0.7	0.64	0.63	**0.78**	0.75	0.73
Spam	40	0.80	0.8	0.70	0.63	**0.87**	**0.87**	0.85
Spam	50	0.79	0.8	0.73	0.61	**0.88**	**0.88**	0.86
WebKB	20	0.65	0.7	0.69	0.65	0.81	0.85	**0.86**
WebKB	30	0.68	0.7	0.70	0.68	0.82	0.84	**0.86**
WebKB	40	0.64	0.73	0.72	0.71	0.81	0.84	**0.85**
WebKB	50	0.70	0.71	0.70	0.69	0.81	0.84	**0.85**
WebSpam	20	0.61	0.66	0.65	0.66	0.71	0.73	**0.74**
WebSpam	30	0.66	0.66	0.66	0.65	0.71	0.72	**0.73**
WebSpam	40	0.65	0.66	0.65	0.63	0.72	**0.73**	**0.73**
WebSpam	50	0.65	0.66	0.63	0.62	0.7	**0.71**	**0.71**
RCV1	20	0.46	0.58	0.59	0.58	0.72	0.76	**0.77**
RCV1	30	0.51	0.65	0.67	0.66	0.78	0.8	**0.82**
RCV1	40	0.44	0.71	0.70	0.71	0.8	0.82	**0.84**
RCV1	50	0.59	0.72	0.72	0.71	0.79	0.81	**0.83**
Movie reviews	40	0.26	0.4	0.41	0.41	0.5	0.52	**0.53**
Movie reviews	60	0.35	0.43	0.45	0.42	0.53	0.55	**0.57**
Movie reviews	80	0.41	0.43	0.47	0.44	0.58	0.59	**0.62**
Movie reviews	100	0.41	0.43	0.45	0.46	0.61	**0.62**	**0.62**

Note: The highest results in every experimental configuration are shown in bold.

5.4.1 Co-Training for Sentiment Analysis Tasks. The results presented here are particularly significant for the Large Movie Reviews dataset. Of all domains evaluated in this study, sentiment analysis is often considered to be the most challenging [44]. Our approach's ability to outperform the standard co-training algorithm by 30% and the single classifier by 50% to 200% is particularly significant given the difficulty of amassing high-quality datasets in this domain.

5.4.2 Runtime and Complexity Analysis. Our evaluation has shown that the average runtime of our proposed approach was less than 5% longer than that of the corresponding original co-training algorithm (this translates to additional few minutes). The reason for this is that the only additional computational cost incurred by our algorithm is the need to combine the predictions of the classification models generated throughout the experiment. This cost has $O(n)$ complexity, as it is dependent on the number of instances in the test set.

6 DISCUSSION

This section is comprised of two parts. We begin by exploring the reasons for our approach's superior performance over the standard co-training algorithm. We focus on the subject of *classification model stability* and demonstrate how its absence harms the standard co-training algorithm while enabling the vertical ensemble's more robust model. In the second part of this section, we explore

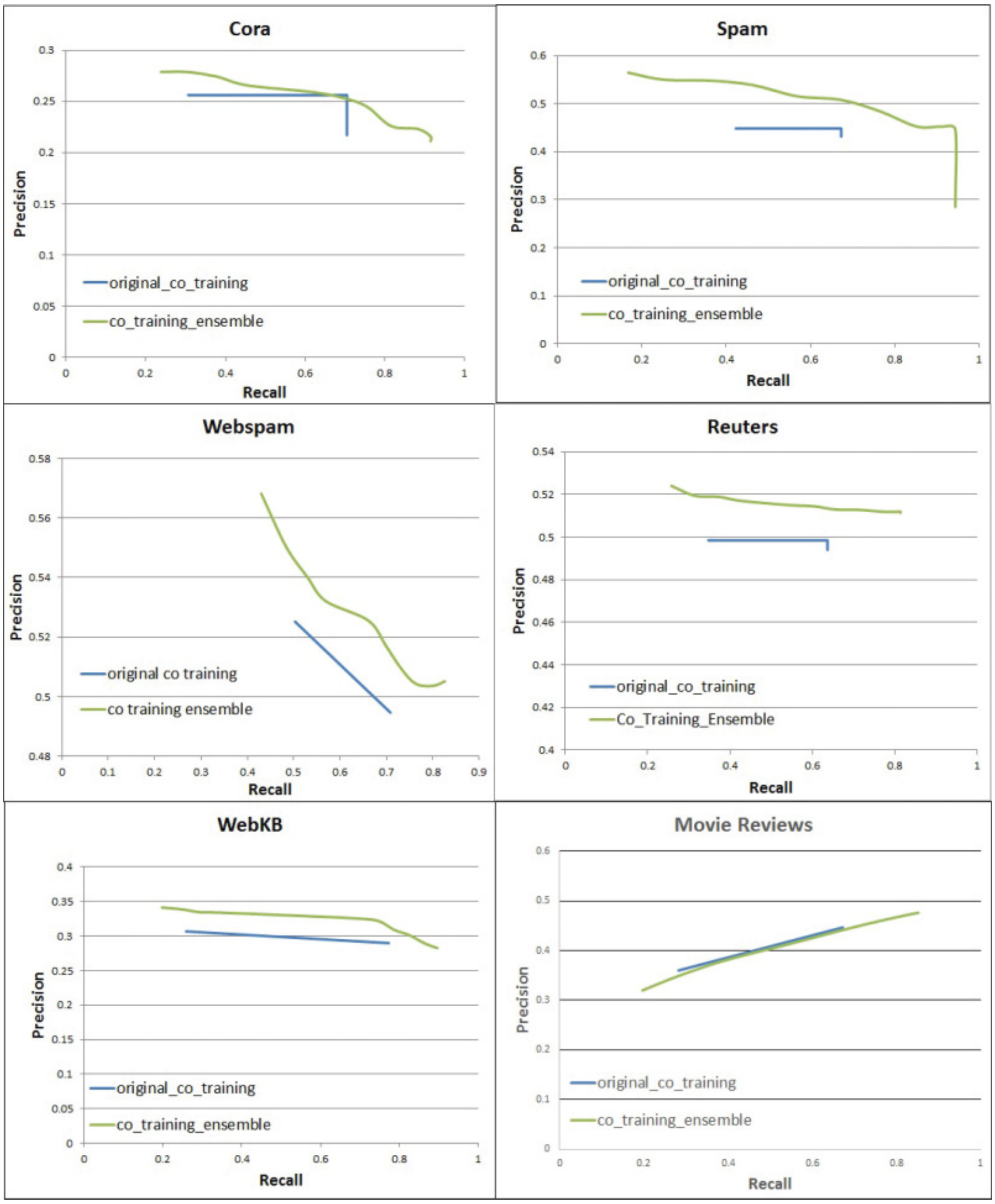


Fig. 1. Precision-recall curves, using the SVM algorithm and a labeled training set of 40 samples for the following datasets: Cora (a), RCV1 (b), Spam (c), WebKB (d), WebSpam (e), and Movie reviews (f).

the use of validation sets in an attempt to mitigate the lack of model stability by using validation sets and assess its contribution to our vertical ensemble approach.

6.1 Analyzing Classification Model Stability

The main reason for the ability of ensemble learning algorithms to outperform single classifiers in many cases is that they use a diverse group of classification models [12]. Intuitively, combining the

ALGORITHM 3: Co-Training With a Validation Set

```
Input:
  $L$ - labeled set
  $V$ - validation set
  $U$ - unlabeled set
  $T$ - test set
  $I$ - classification algorithm
  $n$ - number of iterations
Output:
  $ET$ - extended training set
  $h$ - classifier

Apply_Co_Training

$ET \leftarrow L$
$CI \leftarrow 0$ // The index of the iteration that will be chosen by the algorithm
for all iterations $i \leq n$ do
    $CI \leftarrow i$
    $h_1 \leftarrow TrainClassifier(I, \pi_{\mathbf{x}_1}(ET))$
    $h_2 \leftarrow TrainClassifier(I, \pi_{\mathbf{x}_2}(ET))$
    // assess the performance of current classifiers on validation set
    $VAL_i \leftarrow EvaluateClassifiers(h_{i_1}, h_{i_2}, V)$
    // if the performance on the validation set is worse than in the previous iteration, stop
    if $VAL_i < VAL_{i-1}$ then
        $CI - -$; // revert to the previous training iteration
        break; // terminate the training process
    end if
    // label all unlabeled instances with $h_1$
    $L_1 \leftarrow ApplyClassifier(h_1, \pi_{\mathbf{x}_1}(U))$
    // label all unlabeled instances with $h_2$
    $L_2 \leftarrow ApplyClassifier(h_2, \pi_{\mathbf{x}_2}(U))$
    // add instances with highest confidence in $L_1$ and remove them from the unlabeled set
    $ET \leftarrow ET \cup SelectSamplesWithHighestConfidence(L_1)$
    // add instances with highest confidence in $L_2$ and remove them from the unlabeled set
    $ET \leftarrow ET \cup SelectSamplesWithHighestConfidence(L_2)$
end for

// classify the test set using classifiers of iteration $CI$
$EvaluateClassifiers(h_1^{CI}, h_2^{CI}, T)$
```

predictions of three identical classification models will produce the same results as using either one. It is the combining of multiple and different perspectives of the data that makes ensemble learning such an effective approach.

We hypothesized that the reason for the superior performance of the vertical ensemble co-training approach over the standard co-training is the large variance in the classification models generated over the course of the training process. To test this hypothesis, we compared the performance of the standard co-training algorithm to that of our approach on each iteration for each of the datasets evaluated in Section 5.

Our analysis has shown the same trends in all datasets: whereas the original co-training algorithm demonstrated a large degree of volatility in its performance, the ensemble approach presented monotonous improvement. Interestingly, this was not only the case for the average of all

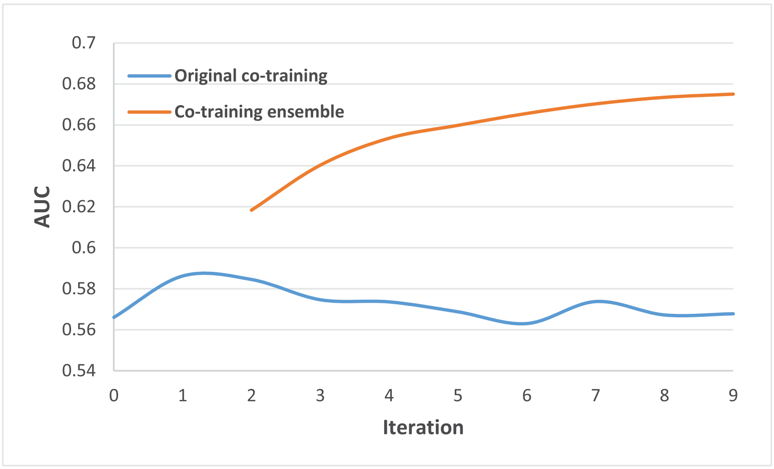


Fig. 2. Comparison of the average performance of the original and ensemble co-training algorithms over the test set in the Cora dataset.

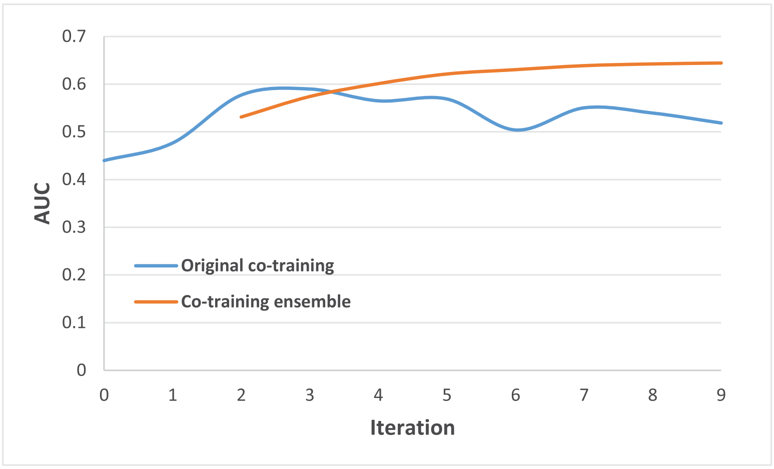


Fig. 3. Comparison of the performance of the original and ensemble co-training algorithms over a single experiment on the Cora dataset.

experiments on a given dataset but in most individual datasets. In Figures 2 and 3, we present an example of these trends (as all datasets behave in a similar manner, we only present the results for one dataset to not burden the reader).

This volatility in performance can be explained as follows. Given an instances set I with each $i \in I$ containing a feature set F, the goal of the classifier is to identify a subset $\{f_1, f_2, ..\} \in F$ that best describes I. If $|I| \ll |F|$, this task becomes far more difficult due to the curse of dimensionality [19] and overfitting [14]. This is particularly true when I consists of no more than 100 instances, as is the case in our experiments—the addition of even a single instance may significantly change the indicative value of a given feature and may lead to the selection of a different subset.

To verify the correctness of this explanation, we have done the following for each of our analyzed datasets:

Table 4. AUC Results of the Standard Co-Training and Ensemble Co-Training Methods When Using Validation Sets and the SVM Classifier

Dataset	Train Size	Original CT	CT + VS	EnsCT + VS
Cora	40	0.43 (1)	0.39 (0.88)	0.56 (1.29)
Cora	60	0.44 (1)	0.40 (0.90)	0.55 (1.25)
Cora	80	0.51 (1)	0.42 (0.83)	0.56 (1.11)
Cora	100	0.48 (1)	0.46 (0.94)	0.60 (1.25)
Spam	40	0.25 (1)	0.60 (2.32)	0.67 (2.60)
Spam	60	0.26 (1)	0.63 (2.41)	0.73 (2.77)
Spam	80	0.30 (1)	0.65 (2.14)	0.72 (2.36)
Spam	100	0.38 (1)	0.69 (1.81)	0.73 (1.91)
WebKB	40	0.66 (1)	0.62 (0.94)	0.77 (1.17)
WebKB	60	0.65 (1)	0.65 (0.99)	0.80 (1.22)
WebKB	80	0.71 (1)	0.64 (0.91)	0.83 (1.18)
WebKB	100	0.64 (1)	0.60 (0.93)	0.81 (1.24)
WebSpam	40	0.72 (1)	0.68 (0.94)	0.78 (1.08)
WebSpam	60	0.76 (1)	0.71 (0.93)	0.81 (1.06)
WebSpam	80	0.74 (1)	0.69 (0.93)	0.79 (1.06)
WebSpam	100	0.77 (1)	0.73 (0.94)	0.84 (1.08)
RCV1	40	0.69 (1)	0.59 (0.86)	0.76 (1.10)
RCV1	60	0.74 (1)	0.64 (0.86)	0.81 (1.07)
RCV1	80	0.78 (1)	0.70 (0.89)	0.83 (1.05)
RCV1	100	0.78 (1)	0.72 (0.92)	0.85 (1.09)
Movie reviews	40	0.43 (1)	0.41 (0.95)	0.50 (1.17)
Movie reviews	60	0.47 (1)	0.47 (1)	0.53 (1.13)
Movie reviews	80	0.51 (1)	0.51 (1.01)	0.58 (1.13)
Movie reviews	100	0.52 (1)	0.56 (1.07)	0.62 (1.20)

Note: Relative performance to the original co-training algorithm is shown in parentheses.

- For each of the 20 experiments conducted on the dataset, we obtained the *labeled sets* of the two first consecutive training iterations (iteration 0 and 1). This was done for the experiments in which the original size of the labeled training set was 40 instances.
- We used Weka's *InfoGainAttributeEval* filter to select to top 10 ranking features for both labeled instances sets.
- We calculated the overlap percentage of the two sets.
- We averaged the results of all experiments for each dataset.

The goal of this analysis is to determine whether the most indicative features can vary to a large degree in consecutive training iteration. Intuitively, if the usefulness of the features vary, different features will be used to create the classification model. Our results show a relatively low overlap: just 66% on average. The full results are presented in Table 4.

The results of analysis clearly indicate that the classification models generated during the various training iterations are not only different in their performance but also in the composition of the features used to generate them. This diversity explains the robustness of our proposed ensemble approach, as it meets the major prerequisite of a successful ensemble.

6.2 Co-Training With Validation Sets

As shown by Mihalcea [39], a larger number of training iterations does not always contribute to the performance of the co-training algorithm. The solution proposed by our work, which has proven beneficial even for the more robust ensemble solution presented in this article, was to terminate the training process after a small number of iterations (six to nine in our case).

Despite the fact that this rule has been proven beneficial on average, our analysis of the co-training algorithm's performance has shown large variance across different datasets and runs. This has led us to try to implement one of the most common "best practices" that is often in use in other ML domains—validation sets.

The use of validation sets consists of setting aside a certain percentage of the labeled training set to evaluate the performance of the classifier that is generated using the remaining labeled instances. It is often used to prevent overfitting [24], where the generated model is too "tailored" to the training set to generalize well to the test set.

The obvious hurdle to the use of a validation set in conjunction to co-training is the small number of available labeled samples. Given this limitation, it seems almost counterintuitive to further reduce the number of available samples by allocating some of them to the validation set. However, we hypothesize that our proposed ensemble method would be able to perform well even with a smaller training set while reaping the benefits of the validation set.

We tested two versions of the co-training with validation set algorithm. The first, presented in Algorithm 3, was applied on the "regular" co-training algorithm. The second, presented in Algorithm 4, was applied on the proposed ensemble co-training algorithm. The same principles were applied to both algorithms: once a training iteration is completed, the performance of the new model is evaluated on the validation set. If the performance of the current training iteration is inferior to that of the one before, we revert to the classification model of the previous iteration and terminate the training (the chosen index is designated by CI in both algorithms). To avoid scenarios where the performance declines and then "picks up," we also tested configurations that required a decline over a consecutive number of iterations.

The results of our experiments are presented in Tables 5 and 6, and we can clearly see that both for the standard co-training example and for our proposed ensemble approach, the use of validation sets causes a slight relative decrease in performance.

Despite the decreased performance, we find it noteworthy that it is surprisingly small given the fact that only half of the labeled set is used for training in this setting. This suggests, in our opinion, that the use of validation sets has significant potential in the context of co-training. We plan to explore this area further in future work by exploring options such as k-fold cross validation and varying validation set sizes.

7 CONCLUSIONS

In this article, we presented a novel type of ensemble approach for the co-training algorithm, which enables a significant improvement in results (almost) without any added computational complexity. We evaluated our proposed methods on six textual datasets, some with extremely high dimensionality and sparsity. The results of the evaluation showed that the proposed methods clearly outperform the standard co-training algorithm.

The effectiveness of our approach in the field of text analysis in general and sentiment analysis in particular is significant, as it is a challenging field for the creation of large, high-quality

Table 5. AUC Results of the Standard Co-Training and Ensemble Co-Training Methods When Using Validation Sets and the NB Classifier

Dataset	Train Size	Original CT	CT + VS	EnsCT + VS
Cora	40	0.49 (1)	0.45 (0.91)	0.60 (1.22)
Cora	60	0.54 (1)	00.48 (0.88)	0.63 (1.16)
Cora	80	0.58 (1)	0.52 (0.88)	0.67 (1.14)
Cora	100	0.57 (1)	0.53 (0.93)	0.66 (1.16)
Spam	40	0.52 (1)	0.65 (1.25)	0.68 (1.31)
Spam	60	0.64 (1)	0.73 (1.13)	0.76 (1.17)
Spam	80	0.70 (1)	0.75 (1.07)	0.80 (1.13)
Spam	100	0.73 (1)	0.79 (1.08)	0.84 (1.16)
WebKB	40	0.69 (1)	0.71 (1.01)	0.80 (1.15)
WebKB	60	0.70 (1)	0.68 (0.94)	0.83 (1.19)
WebKB	80	0.72 (1)	0.68 (0.94)	0.81 (1.12)
WebKB	100	0.70 (1)	0.68 (0.96)	0.83 (1.17)
WebSpam	40	0.65 (1)	0.64 (0.97)	0.75 (1.14)
WebSpam	60	0.66 (1)	0.65 (0.98)	0.74 (1.12)
WebSpam	80	0.64 (1)	0.67 (1.03)	0.73 (1.13)
WebSpam	100	0.63 (1)	0.66 (1.04)	0.73 (1.15)
RCV1	40	0.59 (1)	0.60 (1.02)	0.63 (1.06)
RCV1	60	0.67 (1)	0.62 (0.92)	0.70 (1.04)
RCV1	80	0.70 (1)	0.74 (1.05)	0.78 (1.11)
RCV1	100	0.72 (1)	0.75 (1.04)	0.84 (1.17)
Movie reviews	40	0.41 (1)	0.40 (0.99)	0.50 (1.22)
Movie reviews	60	0.45 (1)	0.41 (0.929)	0.51 (1.14)
Movie reviews	80	0.47 (1)	0.44 (0.93)	0.58 (1.24)
Movie reviews	100	0.45 (1)	0.44 (0.99)	0.53 (1.19)

Note: Relative performance to the original co-training algorithm is shown in parentheses.

Table 6. Overlap Percentage of the Top Indicative Features for the Two First Consecutive Training Iterations for Each Dataset

Dataset Name	Overlap (%)
Spam	75
WebKB	64
Reuters	61
Cora	73
WebSpam	68
Large Movie Reviews	58

datasets. The need to rely on human taggers, capable of understanding the intricacies of human communication, causes the process of corpus creation to be slow and expensive. On this task, the proposed approach outperforms a single classifier by 200% and the standard co-training by 50%.

ALGORITHM 4: The Ensemble Co-Training Algorithm With a Validation Set

```
Input:
  $L$ - labeled set; $U$ - unlabeled set; $T$ - test set; $I$ - classification algorithm; $n$ - number of iterations; $e$ -
number of iterations back to include in the ensemble
Output:
  $ET$ - extended training set
  $h$ - classifier

Apply_Co_Training

$ET \leftarrow L$
$CI \leftarrow 0$ // The index of the iteration that will be chosen by the algorithm
for all iterations $i \leq n$ do
    $CI \leftarrow i$
    $h_1 \leftarrow TrainClassifier(I, \pi_{\mathbf{x}_1}(ET))$
    // save the first classifier for iteration $i$
    $SaveClassifierPerIteration(i, 1, h_1)$
    $h_2 \leftarrow TrainClassifier(I, \pi_{\mathbf{x}_2}(ET))$
    // save the second classifier for iteration $i$
    $SaveClassifierPerIteration(i, 2, h_2)$
    // evaluate the performance of the ensemble (by distribution summation) of the latest $e$ classifiers on the validation set
    $VAL_i \leftarrow ApplyDistributionSummation(V, e, i)$
    // if the performance on the validation set is worse than in the previous iteration, stop
    if $VAL_i < VAL_{i-1}$ then
        $CI - -$; // revert to the previous training iteration
        break; // terminate the training process
    end if
    // label all unlabeled instances with $h_1$
    $L_1 \leftarrow ApplyClassifier(h_1, \pi_{\mathbf{x}_1}(U))$
    // label all unlabeled instances with $h_2$
    $L_2 \leftarrow ApplyClassifier(h_2, \pi_{\mathbf{x}_2}(U))$
    // add instances with highest confidence in $L_1$ and remove them from the unlabelled sett
    $ET \leftarrow ET \cup SelectSamplesWithHighestCofidence(L_1)$
    // add instances with highest confidence in $L_2$ and remove them from the unlabelled set
    $ET \leftarrow ET \cup SelectSamplesWithHighestCofidence(L_2)$
end for
// create the ensemble by combining the classifications of the $e$ latest classifiers from iteration $CI$ on the test set using distribution
summation
return $ApplyDistributionSummation(T, e, CI)$.
```

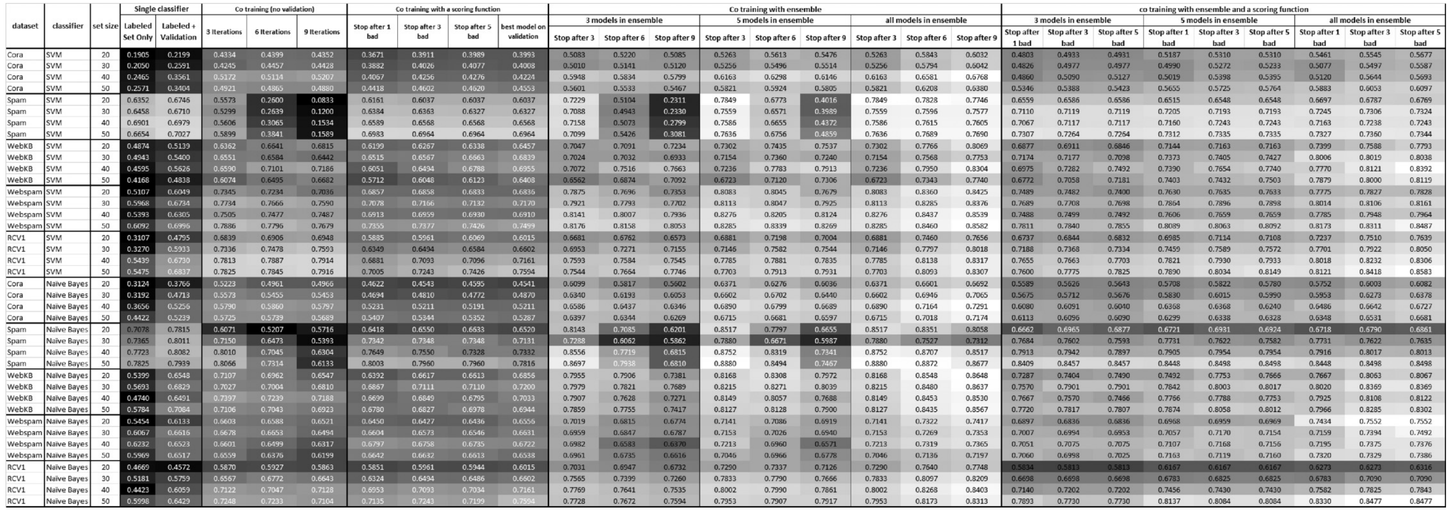
We believe that the work presented here can be expanded in several interesting directions:

(1) *Exploration of additional stopping criteria.* Since the use of validation sets, although providing valuable insights into the inner workings of the co-training algorithm, do not fully produce the desired improvement, we are interested in exploring other methods to achieve this goal. We plan to apply metalearning and deep reinforcement learning [49] to train a neural net capable of automatically determining when to terminate the training process. Additional advantages of this approach may be the ability to better train the ensemble model given a fixed training "budget" [33].

(2) *A more intelligent selection of classifiers for the ensemble.* Currently, the last X classifiers are chosen without regard to their similarity. A more advanced approach that takes one (or several) representatives from each group may produce better results.
(3) *Labeling different numbers of items per class during training.* We currently add the same number of instances from each class during the training phase. A different approach would be adding a different number of instances per each class based on criteria such as dataset imbalance or level of certainty by each classifier.
(4) *The evaluation of the proposed methods in other domains.* In this study, we focused on text data. It would be interesting to apply the proposed approaches to other types of data for which multiple modalities are available, such as image data.
(5) *Applying our approach to additional versions of the co-training algorithm.* As our approach has proven itself effective when applied on the standard co-training algorithm, evaluating it on additional variations may provide additional improvement. We believe that one promising "candidate" for such an enhancement is the tri-training algorithm [57].

APPENDIX

A FULL RESULTS OF ALL EXPERIMENTS



dataset	classifier	set size	Single classifier		Co training (no validation)			Co training with a scoring function				Co training with ensemble									co training with ensemble and a scoring function								
												3 models in ensemble			5 models in ensemble			all models in ensemble			3 models in ensemble			5 models in ensemble			all models in ensemble		
			Labeled Set Only	Labeled + Validation	3 Iterations	6 Iterations	9 Iterations	Stop after 1 bad	Stop after 3 bad	Stop after 5 bad	best model on validation	Stop after 3	Stop after 6	Stop after 9	Stop after 3	Stop after 6	Stop after 9	Stop after 3	Stop after 6	Stop after 9	Stop after 1 bad	Stop after 3 bad	Stop after 5 bad	Stop after 1 bad	Stop after 3 bad	Stop after 5 bad	Stop after 1 bad	Stop after 3 bad	Stop after 5 bad
Cora	SVM	20	0.1905	0.2199	0.4334	0.4399	0.4352	0.3671	0.3911	0.3989	0.3993	0.5083	0.5220	0.5085	0.5263	0.5613	0.5476	0.5263	0.5843	0.6032	0.4803	0.4933	0.4931	0.5187	0.5310	0.5310	0.5461	0.5545	0.5677
Cora	SVM	30	0.2050	0.2591	0.4245	0.4457	0.4428	0.3882	0.4026	0.4077	0.4008	0.5010	0.5141	0.5120	0.5256	0.5496	0.5514	0.5256	0.5794	0.6042	0.4826	0.4977	0.4977	0.4990	0.5272	0.5233	0.5077	0.5497	0.5587
Cora	SVM	40	0.2465	0.3561	0.5172	0.5114	0.5207	0.4067	0.4256	0.4276	0.4224	0.5948	0.5834	0.5799	0.6163	0.6298	0.6146	0.6163	0.6581	0.6768	0.4860	0.5090	0.5127	0.5019	0.5398	0.5395	0.5120	0.5644	0.5693
Cora	SVM	50	0.2571	0.3404	0.4921	0.4865	0.4880	0.4418	0.4602	0.4620	0.4553	0.5601	0.5533	0.5467	0.5821	0.5924	0.5805	0.5821	0.6208	0.6380	0.5346	0.5388	0.5423	0.5655	0.5725	0.5764	0.5883	0.6053	0.6097
Spam	SVM	20	0.6352	0.6746	0.5573	0.2600	0.0833	0.6161	0.6037	0.6037	0.6037	0.7229	0.5104	0.2311	0.7849	0.6773	0.4016	0.7849	0.7828	0.7746	0.6559	0.6586	0.6586	0.6515	0.6548	0.6548	0.6697	0.6787	0.6769
Spam	SVM	30	0.6458	0.6710	0.5299	0.2639	0.1200	0.6384	0.6363	0.6327	0.6327	0.7088	0.4943	0.2330	0.7559	0.6571	0.3989	0.7559	0.7596	0.7577	0.7110	0.7119	0.7119	0.7205	0.7193	0.7193	0.7245	0.7306	0.7324
Spam	SVM	40	0.6901	0.6979	0.5606	0.3065	0.1534	0.6589	0.6568	0.6568	0.6568	0.7158	0.5073	0.2799	0.7586	0.6655	0.4372	0.7586	0.7615	0.7605	0.7067	0.7117	0.7117	0.7160	0.7243	0.7243	0.7163	0.7238	0.7243
Spam	SVM	50	0.6654	0.7027	0.5899	0.3841	0.1589	0.6983	0.6964	0.6964	0.6964	0.7099	0.5426	0.3081	0.7636	0.6756	0.4859	0.7636	0.7689	0.7690	0.7307	0.7264	0.7264	0.7312	0.7335	0.7335	0.7327	0.7360	0.7344
WebKB	SVM	20	0.4874	0.5139	0.6362	0.6641	0.6815	0.6199	0.6267	0.6338	0.6457	0.7047	0.7091	0.7234	0.7302	0.7435	0.7537	0.7302	0.7766	0.8069	0.6877	0.6911	0.6846	0.7144	0.7163	0.7163	0.7399	0.7588	0.7793
WebKB	SVM	30	0.4943	0.5400	0.6551	0.6584	0.6442	0.6515	0.6567	0.6663	0.6839	0.7024	0.7032	0.6933	0.7154	0.7360	0.7240	0.7154	0.7568	0.7753	0.7174	0.7177	0.7098	0.7373	0.7405	0.7427	0.8006	0.8019	0.8038
WebKB	SVM	40	0.4595	0.5626	0.6590	0.7101	0.7186	0.6051	0.6494	0.6788	0.6955	0.7072	0.7516	0.7663	0.7236	0.7783	0.7913	0.7236	0.7950	0.8304	0.6975	0.7282	0.7492	0.7390	0.7654	0.7740	0.7770	0.8121	0.8392
WebKB	SVM	50	0.4168	0.4838	0.6074	0.6495	0.6682	0.5712	0.6048	0.6123	0.6408	0.6562	0.6874	0.7092	0.6723	0.7120	0.7306	0.6723	0.7343	0.7740	0.6772	0.7058	0.7181	0.7403	0.7432	0.7503	0.7879	0.8000	0.8119
Webspam	SVM	20	0.5107	0.6049	0.7345	0.7234	0.7036	0.6857	0.6858	0.6833	0.6836	0.7875	0.7696	0.7353	0.8083	0.8045	0.7679	0.8083	0.8360	0.8425	0.7489	0.7482	0.7400	0.7630	0.7635	0.7633	0.7775	0.7827	0.7828
Webspam	SVM	30	0.5968	0.6734	0.7734	0.7666	0.7590	0.7078	0.7166	0.7132	0.7170	0.7921	0.7793	0.7702	0.8113	0.8047	0.7925	0.8113	0.8285	0.8376	0.7689	0.7708	0.7698	0.7864	0.7896	0.7898	0.8014	0.8106	0.8161
Webspam	SVM	40	0.5393	0.6305	0.7505	0.7477	0.7487	0.6913	0.6959	0.6930	0.6910	0.8141	0.8007	0.7936	0.8276	0.8205	0.8124	0.8276	0.8437	0.8539	0.7488	0.7499	0.7492	0.7606	0.7659	0.7659	0.7785	0.7948	0.7964
Webspam	SVM	50	0.6092	0.6996	0.7886	0.7796	0.7679	0.7355	0.7377	0.7426	0.7499	0.8176	0.8158	0.8053	0.8285	0.8339	0.8269	0.8285	0.8460	0.8582	0.7811	0.7840	0.7855	0.8089	0.8063	0.8092	0.8173	0.8311	0.8487
RCV1	SVM	20	0.3107	0.4795	0.6839	0.6906	0.6948	0.5885	0.5961	0.6069	0.6015	0.6681	0.6762	0.6573	0.6881	0.7198	0.7004	0.6881	0.7460	0.7656	0.6737	0.6844	0.6832	0.6985	0.7114	0.7108	0.7237	0.7510	0.7639
RCV1	SVM	30	0.3270	0.5933	0.7336	0.7478	0.7593	0.6349	0.6494	0.6584	0.6602	0.6953	0.7271	0.7155	0.7146	0.7582	0.7544	0.7146	0.7797	0.8018	0.7188	0.7368	0.7334	0.7459	0.7589	0.7572	0.7701	0.7922	0.8050
RCV1	SVM	40	0.5439	0.6730	0.7813	0.7887	0.7914	0.6881	0.7093	0.7096	0.7161	0.7593	0.7584	0.7545	0.7785	0.7881	0.7835	0.7785	0.8138	0.8317	0.7655	0.7663	0.7703	0.7821	0.7930	0.7933	0.8018	0.8232	0.8306
RCV1	SVM	50	0.5475	0.6837	0.7825	0.7845	0.7916	0.7005	0.7243	0.7426	0.7594	0.7544	0.7664	0.7746	0.7703	0.7913	0.7931	0.7703	0.8093	0.8307	0.7600	0.7775	0.7825	0.7890	0.8034	0.8149	0.8121	0.8418	0.8583
Cora	Naïve Bayes	20	0.3124	0.3766	0.5223	0.4961	0.4966	0.4622	0.4543	0.4595	0.4541	0.6099	0.5817	0.5602	0.6371	0.6276	0.6036	0.6371	0.6601	0.6692	0.5589	0.5626	0.5643	0.5708	0.5822	0.5780	0.5752	0.6003	0.6082
Cora	Naïve Bayes	30	0.3192	0.4713	0.5573	0.5455	0.5453	0.4694	0.4810	0.4772	0.4870	0.6340	0.6193	0.6053	0.6602	0.6702	0.6440	0.6602	0.6946	0.7065	0.5675	0.5712	0.5676	0.5830	0.6015	0.5990	0.5953	0.6273	0.6378
Cora	Naïve Bayes	40	0.3656	0.5256	0.5790	0.5860	0.5797	0.5231	0.5211	0.5191	0.5211	0.6586	0.6437	0.6346	0.6890	0.6799	0.6689	0.6890	0.7164	0.7291	0.6080	0.6091	0.6040	0.6368	0.6368	0.6240	0.6486	0.6642	0.6727
Cora	Naïve Bayes	50	0.4422	0.5239	0.5725	0.5739	0.5689	0.5407	0.5344	0.5352	0.5287	0.6397	0.6344	0.6269	0.6715	0.6681	0.6597	0.6715	0.7018	0.7174	0.6113	0.6096	0.6090	0.6299	0.6338	0.6328	0.6348	0.6531	0.6681
Spam	Naïve Bayes	20	0.7078	0.7815	0.6071	0.5207	0.5716	0.6418	0.6550	0.6633	0.6520	0.8143	0.7085	0.6201	0.8517	0.7797	0.6655	0.8517	0.8351	0.8058	0.6662	0.6965	0.6877	0.6721	0.6931	0.6924	0.6718	0.6790	0.6861
Spam	Naïve Bayes	30	0.7365	0.8011	0.7150	0.6473	0.5393	0.7342	0.7348	0.7348	0.7131	0.7288	0.6062	0.5862	0.7880	0.6671	0.5987	0.7880	0.7527	0.7312	0.7684	0.7602	0.7593	0.7731	0.7622	0.7582	0.7731	0.7622	0.7635
Spam	Naïve Bayes	40	0.7723	0.8082	0.8010	0.7045	0.6304	0.7649	0.7550	0.7328	0.7332	0.8556	0.7719	0.6815	0.8752	0.8319	0.7341	0.8752	0.8707	0.8517	0.7913	0.7942	0.7897	0.7905	0.7954	0.7954	0.7916	0.8017	0.8013
Spam	Naïve Bayes	50	0.7825	0.7939	0.8066	0.7314	0.6133	0.8003	0.7960	0.7960	0.7816	0.8697	0.7938	0.6810	0.8880	0.8494	0.7467	0.8880	0.8872	0.8677	0.8409	0.8457	0.8457	0.8448	0.8498	0.8498	0.8448	0.8498	0.8498
WebKB	Naïve Bayes	20	0.5399	0.6548	0.7107	0.6962	0.6547	0.6392	0.6617	0.6613	0.6856	0.7955	0.7906	0.7381	0.8168	0.8308	0.7972	0.8168	0.8548	0.8648	0.7287	0.7404	0.7490	0.7492	0.7753	0.7666	0.7667	0.8063	0.8067
WebKB	Naïve Bayes	30	0.5693	0.6829	0.7027	0.7004	0.6810	0.6867	0.7111	0.7110	0.7200	0.7979	0.7821	0.7689	0.8215	0.8271	0.8039	0.8215	0.8480	0.8637	0.7570	0.7901	0.7901	0.7842	0.8003	0.8017	0.8020	0.8369	0.8369
WebKB	Naïve Bayes	40	0.4740	0.6491	0.7397	0.7239	0.7188	0.6699	0.6849	0.6795	0.7033	0.7907	0.7628	0.7271	0.8149	0.8057	0.7688	0.8149	0.8453	0.8530	0.7667	0.7570	0.7466	0.7766	0.7788	0.7753	0.7925	0.8108	0.8122
WebKB	Naïve Bayes	50	0.5784	0.7084	0.7106	0.7043	0.6923	0.6780	0.6827	0.6978	0.6944	0.7859	0.7755	0.7417	0.8127	0.8128	0.7900	0.8127	0.8435	0.8567	0.7720	0.7817	0.7807	0.7874	0.8058	0.8012	0.7966	0.8285	0.8302
Webspam	Naïve Bayes	20	0.5454	0.6133	0.6603	0.6588	0.6521	0.6450	0.6427	0.6436	0.6556	0.7019	0.6815	0.6774	0.7141	0.7086	0.6919	0.7141	0.7322	0.7417	0.6897	0.6836	0.6836	0.6968	0.6959	0.6969	0.7434	0.7552	0.7552
Webspam	Naïve Bayes	30	0.6067	0.6616	0.6678	0.6653	0.6494	0.6604	0.6573	0.6546	0.6631	0.6959	0.6847	0.6787	0.7153	0.7026	0.6940	0.7153	0.7269	0.7353	0.7007	0.6994	0.6953	0.7057	0.7170	0.7154	0.7159	0.7394	0.7492
Webspam	Naïve Bayes	40	0.6232	0.6523	0.6601	0.6499	0.6317	0.6797	0.6758	0.6735	0.6722	0.6982	0.6583	0.6370	0.7213	0.6960	0.6571	0.7213	0.7319	0.7365	0.7051	0.7075	0.7075	0.7107	0.7168	0.7156	0.7195	0.7375	0.7376
Webspam	Naïve Bayes	50	0.5969	0.6517	0.6559	0.6376	0.6199	0.6642	0.6632	0.6613	0.6538	0.6961	0.6735	0.6616	0.7046	0.6966	0.6778	0.7046	0.7136	0.7197	0.7060	0.6998	0.7025	0.7163	0.7119	0.7160	0.7320	0.7329	0.7386
RCV1	Naïve Bayes	20	0.4669	0.4572	0.5870	0.5927	0.5863	0.5851	0.5961	0.5944	0.6015	0.7031	0.6947	0.6732	0.7290	0.7337	0.7126	0.7290	0.7640	0.7748	0.5834	0.5813	0.5813	0.6167	0.6167	0.6167	0.6273	0.6273	0.6316
RCV1	Naïve Bayes	30	0.5181	0.5759	0.6567	0.6772	0.6643	0.6324	0.6494	0.6486	0.6602	0.7565	0.7399	0.7260	0.7833	0.7790	0.7666	0.7833	0.8097	0.8209	0.6698	0.6698	0.6698	0.6783	0.6825	0.6825	0.6783	0.7090	0.7090
RCV1	Naïve Bayes	40	0.4423	0.6059	0.7122	0.7047	0.7128	0.6953	0.7093	0.7034	0.7161	0.7769	0.7641	0.7535	0.8002	0.7990	0.7861	0.8002	0.8268	0.8403	0.7140	0.7202	0.7202	0.7456	0.7430	0.7430	0.7582	0.7825	0.7843
RCV1	Naïve Bayes	50	0.5998	0.6429	0.7248	0.7233	0.7104	0.7135	0.7243	0.7199	0.7594	0.7728	0.7672	0.7594	0.7953	0.7907	0.7917	0.7953	0.8173	0.8313	0.7893	0.7730	0.7730	0.8137	0.8084	0.8084	0.8330	0.8477	0.8477

Fig. 4. Full AUC results for all experiment configurations.

REFERENCES

[1] Maria-Florina Balcan, Avrim Blum, and Ke Yang. 2004. Co-training and expansion: Towards bridging theory and practice. In *Advances in Neural Information Processing Systems*. 89–96.

[2] Maria F. Balcan, Avrim Blum, and Ke Yang. 2005. Co-training and expansion: Towards bridging theory and practice. In *Advances in Neural Information Processing Systems*.

[3] A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT'98)*. ACM, New York, NY, 92–100. DOI: http://dx.doi.org/10.1145/279943.279962

[4] Ulf Brefeld and Tobias Scheffer. 2004. Co-EM support vector learning. In *Proceedings of the 21st International Conference on Machine Learning (ICML'04)*. 16.

[5] Leo Breiman. 1996. Bagging predictors. *Machine Learning* 24, 2, 123–140.

[6] Leo Breiman. 2001. Random forests. *Machine Learning* 45, 1, 5–32.

[7] Wray Buntine. 1992. Learning classification trees. *Statistics and Computing* 2, 2, 63–73.

[8] Minmin Chen, Kilian Weinberger, and Yixin Chen. 2011. Automatic feature decomposition for single view co-training. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*. 953–960.

[9] C. M. Christoudias, R. Urtasun, and T. Darrell. 2008. Multi-view learning in the presence of view disagreement. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI'08)*. 88-96.

[10] Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 7, 1–30. http://dl.acm.org/citation.cfm?id=1248547.1248548.

[11] Francois Denis, Anne Laurent, Razmi Gilleron, and Marc Tommasi. 2003. Text classification and co-training from positive and unlabeled examples. In *Proceedings of the ICML 2003 Workshop: The Continuum from Labeled to Unlabeled Data*. 80–87.

[12] Thomas G. Dietterich. 2000. Ensemble methods in machine learning. In *Multiple Classifier Systems*. Springer, 1–15.

[13] Thomas G. Dietterich. 2002. Ensemble learning. In *The Handbook of Brain Theory and Neural Networks* (2nd ed.). MIT Press, Cambridge, MA, 110–125.

[14] Pedro Domingos. 2012. A few useful things to know about machine learning. *Communications of the ACM* 55, 10, 78–87.

[15] J. Du, C. X. Ling, and Z.-H. Zhou. 2011. When does cotraining work in real data? *IEEE Transactions on Knowledge and Data Engineering* 23, 5, 788–799.

[16] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*. 2944–2952.

[17] Yoav Freund and Robert E. Schapire. 1995. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the 2nd European Conference on Computational Learning Theory (EuroCOLT'95)*. 23–37. http://dl.acm.org/citation.cfm?id=646943.712093.

[18] Yoav Freund and Robert E. Schapire. 1999. A short introduction to boosting. *Journal of the Japanese Society for Artificial Intelligence* 14, 5, 771–780.

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

[20] Rayid Ghani. 2002. Combining labeled and unlabeled data for multiclass text categorization. In *Proceedings of the 19th International Conference on Machine Learning (ICML'02)*. 187–194.

[21] Sujatha Das Gollapalli, Cornelia Caragea, Prasenjit Mitra, and C. Lee Giles. 2013. Researcher homepage classification using unlabeled data. In *Proceedings of the 22nd International Conference on World Wide Web (WWW'13)*. 471–482. http://dl.acm.org/citation.cfm?id=2488388.2488430.

[22] Sujatha Das Gollapalli, Cornelia Caragea, Prasenjit Mitra, and C. Lee Giles. 2015. Improving researcher homepage classification with unlabeled data. *ACM Transactions on the Web* 9, 4, 17.

[23] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter* 11, 1, 10–18.

[24] Douglas M. Hawkins. 2004. The problem of overfitting. *Journal of Chemical Information and Computer Sciences* 44, 1, 1–12.

[25] Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning(ICML'99)*. 200–209.

[26] Thorsten Joachims. 2006. Transductive support vector machines. In *Semi-Supervised Learning*, O. Chapelle, B. Scholkopf, and A. Zieneds (Eds.). MIT Press, Cambridge, MA, 105–118.

[27] Gilad Katz, Nir Ofek, and Bracha Shapira. 2015. ConSent. *Knowledge-Based Systems* 84, C, 162–178.

[28] Gilad Katz, Asaf Shabtai, and Lior Rokach. 2014. Adapted features and instance selection for improving co-training. In *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics*. Springer, 81–100.

[29] Svetlana Kiritchenko and Stan Matwin. 2001. Email classification with co-training. In *Proceedings of the 2001 Conference of the Centre for Advanced Studies on Collaborative Research (CASCON'01)*. 8. http://dl.acm.org/citation.cfm?id=782096.782104.

[30] Anders Krogh and Jesper Vedelsby. 1995. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems*. 231–238.

[31] Anat Levin, Paul Viola, and Yoav Freund. 2003. Unsupervised improvement of visual detectors using co-training. In *Proceedings of the 9th IEEE International Conference on Computer Vision, Volume 2 (ICCV'03)*. IEEE, Los Alamitos, CA, 626–633. http://dl.acm.org/citation.cfm?id=946247.946615.

[32] Guangxia Li, Steven C. H. Hoi, and Kuiyu Chang. 2010. Two-view transductive support vector machines. In *Proceedings of the 2010 SIAM International Conference on Data Mining*. 235–244.

[33] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2016. Hyperband: A novel bandit-based approach to hyperparameter optimization. arXiv:1603.06560.

[34] Rong Liu, Jian Cheng, and Hanqing Lu. 2009. A robust boosting tracker with minimum error bound in a co-training framework. In *Proceedings of the 2009 IEEE 12th International Conference on Computer Visiong (ICCV'09)*. 1459–1466.

[35] Bo Long, Philip S. Yu, and Zhongfei (Mark) Zhang. 2008. A general model for multiple view unsupervised learning. In *Proceedings of the 8th SIAM International Conference on Data Mining*. 822–833.

[36] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 142–150. http://www.aclweb.org/anthology/P11-1015.

[37] Ching-Hao Mao, Hahn-Ming Lee, Devi Parikh, Tsuhan Chen, and Si-Yu Huang. 2009. Semi-supervised co-training and active learning based approach for multi-view intrusion detection. In *Proceedings of the 2009 ACM Symposium on Applied Computing (SAC'09)*. ACM, New York, NY, 2042–2048. DOI: http://dx.doi.org/10.1145/1529282.1529735

[38] Eitan Menahem, Lior Rokach, and Yuval Elovici. 2009. Troika—an improved stacking schema for classification tasks. *Information Sciences* 179, 24, 4097–4122. DOI: http://dx.doi.org/10.1016/j.ins.2009.08.025

[39] Rada Mihalcea. 2004. Co-training and self-training for word sense disambiguation. In *Proceedings of the HLT-NAACL 2004 Workshop: 8th Conference on Computational Natural Language Learning (CoNLL'04)*. 33–40.

[40] Christoph Müller, Stefan Rapp, and Michael Strube. 2002. Applying co-training to reference resolution. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL'02)*. 352–359. DOI: http://dx.doi.org/10.3115/1073083.1073142

[41] Ion Muslea, Steven Minton, and Craig A. Knoblock. 2002. Active + semi-supervised learning = robust multi-view learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML'02)*. 435–442. http://dl.acm.org/citation.cfm?id=645531.655845.

[42] Kamal Nigam and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM'00)*. 86–93.

[43] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning* 39, 2–3, 103–134. DOI: http://dx.doi.org/10.1023/A:1007692713085

[44] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, Volume 10 (EMNLP'02)*. 79–86.

[45] John Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers* 10, 3, 61–74.

[46] Anoop Sarkar. 2001. Applying co-training methods to statistical parsing. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies (NAACL'01)*. 1–8. DOI: http://dx.doi.org/10.3115/1073336.1073359

[47] Aayush Sharma, Gang Hua, Zicheng Liu, and Zhengyou Zhang. 2008. Meta-tag propagation by co-training an ensemble classifier for improving image search relevance. In *Proceedings of the 2008 IEEE Computer Society Conference on Computer Visiona and Pattern Recognition (CVPRW'08)*.

[48] Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. 2005. A co-regularization approach to semi-supervised learning with multiple views. In *Proceedings of the ICML Workshop on Learning with Multiple Views*.

[49] Bradly C. Stadie, Sergey Levine, and Pieter Abbeel. 2015. Incentivizing exploration in reinforcement learning with deep predictive models. arXiv:1507.00814.

[50] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2013. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 847–855.

[51] Katrin Tomanek and Udo Hahn. 2009. Semi-supervised active learning for sequence labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*. 1039–1047.

[52] Simon Tong and Daphne Koller. 2002. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research* 2, 45–66.
[53] Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 (ACL'09)*. 235–243. http://dl.acm.org/citation.cfm?id=1687878.1687913.
[54] William Yang Wang, Kapil Thadani, and Kathleen McKeown. 2011. Identifying event descriptions using co-training with online news summaries. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP'11)*.
[55] Min-Ling Zhang and Zhi-Hua Zhou. 2011. CoTrade: Confident co-training with data editing. *IEEE Transactions on Systems, Man, and Cybernetics: Part B (Cybernetics)* 41, 6, 1612–1626.
[56] Zhi-Hua Zhou. 2009. When semi-supervised learning meets ensemble learning. In *Proceedings of the 8th International Workshop on Multiple Classifier Systems (MCS'09)*. 529–538. DOI: http://dx.doi.org/10.1007/978-3-642-02326-2_53
[57] Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering* 17, 11, 1529–1541.
[58] X. Zhu and A. B. Goldberg. 2009. *Introduction to Semi-Supervised Learning*. Morgan & Claypool.

Received February 2017; revised June 2017; accepted August 2017