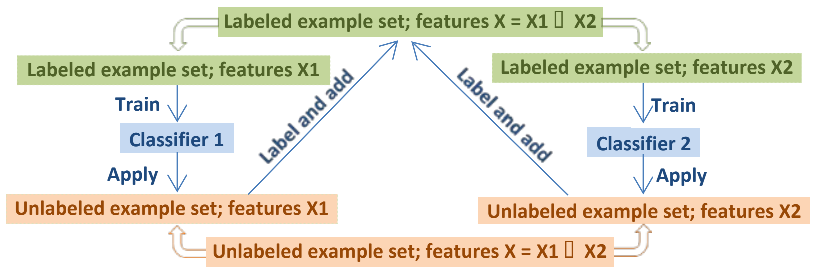
Appendix to RSSalg software: a tool for flexible experimenting with co-training based semi-supervised algorithms

1 Supplementary material for section 2.2: A high-level overview of co-training

A high-level overview of co-training is illustrated in figure 1. In co-training two different views are used to train two different classifiers using the same available labeled data. The trained classifiers are applied on the set of unlabeled data with the goal of selecting and labeling the most confident instances. These instances are added to the initial labeled set and both classifiers are retrained in the supervised fashion on the enlarged training set. This process is repeated for a predefined number of iterations.



Labeled example set; features X = X1 X2
Labeled example set; features X1
Labeled example set; features X2
Train
Classifier 1
Apply
Label and add
Label and add
Train
Classifier 2
Apply
Unlabeled example set; features X1
Unlabeled example set; features X2
Unlabeled example set; features X = X1 X2

Figure 1.A high-level overview of co-training algorithm

2 Supplementary material for section 2.2: A high-level overview of RSSalg

The first step in RSSalg is to create an ensemble of diverse co-training classifiers. The ensemble is formed by generating a predefined number of random feature splits. Each split is used for running a separate co-training process on the same data. Since each split is different, each run of co-training produces a different enlarged training set, consisting of initially labeled data and data labeled during the co-training process. Examples from all acquired enlarged training sets are integrated in one unique training set. If the same instance was labeled in multiple enlarged training sets, its label is determined by majority vote. The integrated set is then filtered in order to keep only the most confidently labeled instances. Instance is considered confidently labeled if: (1) it appears in most of the produced enlarged training sets and (2) it was assigned the same label in the majority of those sets. Two thresholds are defined for these two conditions: each instance that does not meet the thresholds is omitted from the dataset. After filtering, the integrated training set is used in order to train a supervised classification model which can be used to predict the label for previously unseen examples. The high-level overview of RSSalg is shown in figure 2.

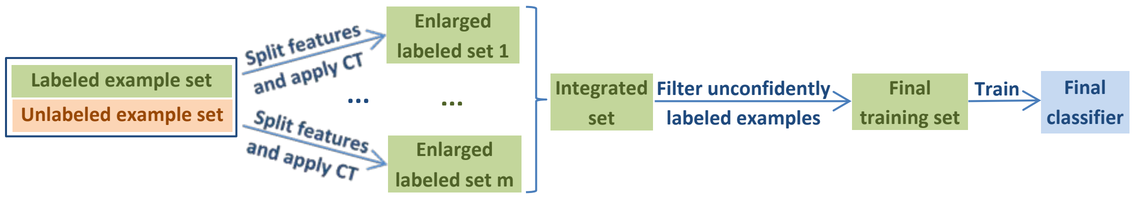


Figure 2. A High-level overview of RSSalg

3 Supplementary material for section 4: Illustrative Examples

In supervised learning, we rely on sufficient amounts of quality training data in order to build classification models with high predictive power [1]. Training sets are formed by experts and their creation involves tedious manual labeling. Thus, obtaining a sufficiently large training set can be expensive in terms of both time and money.

This problem can be alleviated by using semi-supervised learning (SSL). SSL techniques can induce high quality models using a small amount of labeled data and a sufficiently large amount of unlabeled data. Unlike labeled data, unlabeled data is abundant. Employing unlabeled data in the process of learning can result with a considerable improvement in classifier performance. This way, SSL techniques are able to achieve higher accuracy while demanding less human effort in the terms of manual data labeling [2].

A number of semi-supervised learning methods have been proposed, but their efficiency largely depends on the assumptions they impose on the data. One of the major semi-supervised learning techniques is co-training [3]. Co-training application assumes that the dataset can be described by two disjoined sets of attributes, called views. This feature split should be the consequence of the existence of two independent sources of information describing the same data. However, many real-world problems that would benefit from co-training lack this property. In order to address this problem, in our previous work [4] we have introduced RSSalg, a co-training based technique that can be applied on single-view datasets.

RSSalg software is a tool intended for flexible experimenting with co-training based algorithms. It encompasses the implementation of co-training and RSSalg algorithm. In the following, we present the empirical experiments obtained using algorithms implemented in *RSSalg software* on multiple datasets of different properties. The experiments presented here are completely reproducible by using the listed parameter values. For the details on supplying the chosen property values, please refer to RSSalg software online documentation[1]. Prepared property files needed for running the experiments presented here are also available on *RSSalg software* code repository[2].

3.1 The summary of the datasets used in the experiments

The experiments presented here are performed on multiple publicly available datasets of various size, dimensionality and feature redundancy: three natural language datasets (WebKB-Course [3], LingSpam [5] and News2x2 [6]) and 12 UCI datasets. The datasets and their basic properties are listed in table 1.

The natural language datasets are preprocessed as proposed in [7]: an English stop-word filter is applied in order to remove 319 frequent words; stemming is performed using Porter's stemmer [8]; for each of the views separately, 200 most frequent words are used. After preprocessing, the text is represented using the bag of words model in combination with tfidf measurement [9]. The used text preprocessing and text representation techniques do not depend on the class labels and therefore do not violate the co-training setting (in which the class labels are available for only the few labeled instances). The described preprocessed versions of WebKB-Course, LingSpam and News2x2 datasets are available for download on the RSSalg software code repository[3].

Table 1 A summary of the datasets used in experiments. Notation: **Dataset:** datasets marked with ***** can be downloaded from *RSSalg software* repository https://github.com/slivkaje/RSSalg-software/tree/master/dist/data. Datasets marked with **ƚ** are part of the Weka distribution of 37 classification problems originally obtained from the UCI repository http://www.cs.waikato.ac.nz/ml/weka/datasets.html; Datasets marked with **×** can be downloaded from UCI machine learning repository http://archive.ics.uci.edu/ml/; **Dim**: dimensionality (number of features); **Pos class**: the class considered to be "positive" in the experiment[4]; **|L|**, **|All|**: size of the labeled training set *L* and the total number of instances used in training (both labeled and unlabeled instances), respectively. The number of instances is given as number of positive instances/number of negative instances; **Baseline**: accuracy achieved by classifying all instances to the majority class; **p/n**: the dataset specific growth size parameter used in co-training: the number of instances most confidently labeled as positive and negative, respectively, that will be added to labeled set *L* in each iteration of co-training

Dataset	Dim	Pos class	|L|	|All|	Baseline	p/n
WebKB *	400	non-course	5/5	492/138	78.1	1/3
LingSpam *	400	spam	5/5	288/1447	83.4	2/10
News2x2 *	400	positiveClass	5/5	600/600	50.0	5/5
Spambase ×	57	0	2/1	1672/1087	60.6	1/1
Hepatitis ƚ	19	LIVE	1/1	73/19	79.4	11/3
Kr-vs-kp ƚ	36	won	6/5	1001/916	52.2	1/1
Credit-g ƚ	20	good	1/1	420/180	70.0	5/2
Heart-statlog ƚ	13	present	2/3	72/90	55.6	5/6
Sonar ƚ	60	Rock	1/1	58/66	53.4	6/7
Ionosphere ƚ	34	g	5/3	135/75	64.1	2/1
Breast-cancer ƚ	9	no-recurrence-events	1/1	120/51	70.3	5/2
Credit-a ƚ	15	+	4/5	184/229	55.5	1/1
Breast-w ƚ	9	malignant	1/2	144/274	65.5	1/2
Mushroom ƚ	22	e	3/3	2524/2349	51.8	1/1
Diabetes ƚ	8	tested_positive	1/2	160/300	65.1	1/2

[1] https://github.com/slivkaje/RSSalg-software/blob/master/docs/IllustrativeExample.pdf

[2] https://github.com/slivkaje/RSSalg-software/tree/master/dist/data

[3] The preprocessed versions of the WebKB-Course, News2x2 and LingSpam datasets are available at https://github.com/slivkaje/RSSalg-software/tree/master/dist/data

[4] In *RSSalg software,* if all class probabilities for the instance are equal, tie is broken by assigning the example to the class specified as "positive"

3.2 Applied algorithms

In the experiments presented here we will apply the following algorithms implemented in *RSSalg software*:

1. Supervised settings:
    - *L_{acc}*: Supervised learner trained on the labeled portion of the data. The goal for the semi-supervised settings is to exceed this performance while only using additional unlabeled examples for training.
    - *All_{acc}*: Supervised learner trained on both labeled and unlabeled instances with correct label assigned. This can be seen as the goal performance for semi-supervised settings – it is the performance we could achieve if all unlabeled instances would be assigned the correct label during the learning process.
2. Semi-supervised settings:
    - *Natural*: co-training run with natural feature split. This setting is applicable only on the datasets where the "natural" feature split can be defined: *WebKB-course*, *News2x2* and *LingSpam*.
    - *Random*: co-training run with random feature split obtained by randomly assigning each feature to one of the two views of roughly equal size. We will report the average performance co-training of *m* different random feature splits.
    - *MajorityVote* (*MV*): a majority vote of *m* co-training classifiers trained using *m* different random splits (same classifiers used in *Random* setting)
    - *RSSalg*: *RSSalg* as introduced in [4]. *RSSalg* is applied on the same *m* co-training classifiers created by *Random* setting.
    - *$RSSalg_{best}$*: *RSSalg* with thresholds optimized on test data (*$RSSalg_{best}$* in [4]. *$RSSalg_{best}$* serves as the indication of upper bound performance of *RSSalg*. It is used to test the threshold optimization procedure in *RSSalg*

In our experiments we use the following underlying co-training parameters (table 2, 'co-training.properties'): the number of co-training iterations is 20; a small unlabeled pool of size 50 is used for the selection of most confidently labeled examples in each iteration; the number of instances most confidently labeled as 'positive' (p) and the number of instances most confidently labeled as 'negative' (n) that are added to the initial labeled set L in each co-training iteration for each dataset are listed in table 1. The numbers p and n are chosen to reflect the original dataset distribution; the classification model used for underlying view classifiers in co-training and for the final model in RSSalg is Naïve Bayes (table 2, 'data.properties'). These are the standard settings used in co-training evaluation [3][7].

The number of different random splits (m) used in Random, MV, RSSalg and RSSalgbest settings is 100 with the exception of Diabetes dataset where the maximal number of balanced random feature splits is 35 (table 2, common experiment properties).

In RSSalg and RSSalgbest settings, a genetic algorithm is used for automatic threshold optimization. We have used the following genetic algorithm settings (table 2, 'GA.properties'): the generation size was 50; the maximal number of iterations was 50, but the process was stopped if there was no improvement in 5 successive generations; the crossover threshold was 0.3; the probability of bit mutation was 0.02; elitism was used; the testing threshold in RSSalg was set to 20%; All these parameters were empirically chosen.

3.3 Performance evaluation

In order to evaluate the performance of the applied settings a following 10-fold-cross evaluation procedure is performed [7]: the dataset is divided in 10 equal folds; in each iteration of the fold-cross validation procedure, one fold is used for selecting a predefined number of labeled training instances; the remaining data from that fold as well as five adjacent folds are used as unlabeled data (the class-label information for these instances is disregarded); the remaining 4 fold are used as test data. In RSSalg software, user is able to set these experimental parameters through the file named 'cv.properties', table 2. In our experiments accuracy is used as the measure of performance (table 2, common experiment properties). The random number generator seed used in the experiments is 2016 (table 2, 'data.properties');

The distribution of labels in the initial labeled example set L is roughly the same as the distribution of labels in the original datasets L is chosen from as in [3] (with the exception of LingSpam and WebKB datasets where we kept the settings recommended in [7]). Also, L is chosen so that the difference Allacc - Lacc is large enough in order to give space for possible improvement by SSL techniques (10-20 %). The exact size of the initial set L for each dataset is listed in table 1. User can define the initial set L through className and noLabeled properties in ‘cv.properties’. For example, in order to form an initial labeled set L from Spambase dataset consisting of two randomly chosen instances belonging to class 0 and one randomly chosen instance belonging to the class 1, the following property values should be set: className=”0” “1” and noLabeled=2 1.

Table 2 A summary of the property values common for all presented experiments

Property file	**Property name in *RSSalg software***	**Used value**
data.properties	randomGeneratorSeed classifiers combinedClassifier	2016 weka.classifiers.bayes.NaiveBayes weka.classifiers.bayes.NaiveBayes
cv.properties	noFolds noUnlabeled noTest	10 6 4
co-training.properties	labelAllUnlabeledData coTrainingIterations poolSize	false 20 50
GA.properties	generationSize Iterations noImprovalGenerations crossoverTS mutationTS elitism testingTS optimizationMeasure optimizationMeasureClass	50 50 5 0.3 0.02 true 0.2 classificationResult.measures.AccuracyMeasure avg
Commom experiment properties	measures measuresForClass noSplits (This property is used in all settings that require multiple random splits for co-training: *Random*, *MV*, *RSSalg* and *RSSalg_best*)	classificationResult.measures.AccuracyMeasure avg 100 (with the exception of Diabetes dataset where the maximal number of different balanced random splits of 35 is used)

3.4 Experimental results

Table 3 presents the accuracy and standard deviation obtained by each tested method on all considered datasets. Note that the results slightly differ from those presented in [4] due to latter software refactoring which affected the sequence of numbers generated by Java random number generator and the usage of different random number generator seed.

Table 3 Algorithm performance (accuracy ± standard deviation) of all considered algorithms. The performance of *Natural* setting is reported only for WebKB, LingSpam and News2x2 datasets where it was applicable.

Datasets	**L_{acc}**	**All_{acc}**	**Natural**	**Random**	**MV**	**RSSalg**	**RSSalg best**
WebKB	76.1±8.2	96.3±1.1	87.7±4.7	83.9±5.6	88.1±5.1	87.7±3.7	90.2±4.0
LingSpam	78.2±9.4	89.3±3.1	71.1±14.7	77.0±9.1	79.6±7.2	86.7±3.5	88.1±3.8
News2x2	76.8±5.1	89.7±1.7	77.7±10.8	78.4±8.0	84.1±5.2	87.1±4.0	89.2±2.0
Spambase	67.4±3.8	79.6±1.2		75.0±4.4	79.6±6.0	76.6±7.1	78.0±5.7
Hepatitis	59.6±17.9	84.2±2.8		80.6±8.3	83.0±4.5	82.6±2.8	85.7±3.6
Kr-vs-kp	65.0±4.8	87.7±0.6		53.2±3.2	55.7±4.1	56.6±3.6	65.4±5.2
Credit-g	55.8±10.0	73.2±2.8		60.3±5.4	65.8±4.8	64.4±5.0	70.2±1.7
Heart-statlog	66.2±7.5	85.0±2.0		80.0±4.4	82.1±4.5	80.4±4.2	83.1±4.4
Sonar	54.9±6.3	67.9±4.7		51.1±5.2	54.9±4.6	54.7±4.2	58.4±3.2
Ionosphere	70.2±6.8	83.3±5.3		73.9±4.7	71.8±5.7	72.0±7.3	77.8±4.1
Breast-cancer	56.4±11.6	74.1±1.9		68.0±5.8	69.7±3.9	68.8±4.4	72.2±3.3
Credit-a	68.0±3.6	78.1±2.5		71.3±14.2	78.7±3.9	74.6±4.5	79.3±3.5
Breast-w	80.5±11.8	96.0±1.2		93.8±3.1	95.1±1.6	94.3±2.6	96.1±1.6
Mushroom	80.5±10.0	95.4±0.4		82.6±12.2	89.0±0.8	88.3±1.4	89.2±1.5
Diabetes	64.6±5.4	75.2±1.4		54.1±9.4	64.7±4.9	64.5±4.8	68.2±2.2

4 References

[1] Figueroa, R.L., Zeng-Treitler, Q., Kandula, S. and Ngo, L.H., 2012. Predicting sample size required for classification performance. BMC medical informatics and decision making, 12(1), p.1.

[2] Zhu, X., 2005. Semi-supervised learning literature survey.

[3] Blum, Avrim, and Tom Mitchell. "Combining labeled and unlabeled data with co-training." In Proceedings of the eleventh annual conference on Computational learning theory, pp. 92-100. ACM, 1998.

[4] Slivka, J., Kovačević, A. and Konjović, Z., 2013. Combining Co-Training with Ensemble Learning for Application on Single-View Natural Language Datasets. Acta Polytechnica Hungarica, 10(2).

[5] Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Paliouras, G. and Spyropoulos, C.D., 2000. An evaluation of naive bayesian anti-spam filtering.arXiv preprint cs/0006013.

[6] Nigam, K. and Ghani, R., 2000. Understanding the behavior of co-training. InProceedings of KDD-2000 workshop on text mining (pp. 15-17).

[7] Feger, F. and Koprinska, I., 2006, July. Co-Training using RBF nets and different feature splits. In Neural Networks, 2006. IJCNN'06. International Joint Conference on (pp. 1878-1885). IEEE.

[8] Porter, M.F., 1980. An algorithm for suffix stripping. Program, 14(3), pp.130-137.

[9] Salton, G. and Buckley, C., 1988. Term-weighting approaches in automatic text retrieval. Information processing & management, 24(5), pp.513-523.