Combining Co-Training with Ensemble Learning for Application on Single-View Natural Language Datasets

Jelena Slivka, Aleksandar Kovačević, Zora Konjović

Computing and Control Department, Faculty of Technical Sciences
Trg Dositeja Obradovića 6, Novi Sad, Serbia
E-mail: slivkaje@uns.ac.rs, kocha78@uns.ac.rs, ftn_zora@uns.ac.rs

Abstract: In this paper we propose a novel semi-supervised learning algorithm, called Random Split Statistic algorithm (RSSalg), designed to exploit the advantages of co-training algorithm, while being exempt from co-training requirement for the existence of adequate feature split in the dataset. In our method, co-training algorithm is run for a predefined number of times, using a different random split of features in each run. Each run of co-training produces a different enlarged training set, consisting of initial labeled data and data labeled in the co-training process. Examples from the enlarged training sets are combined in a final training set and pruned in order to keep only the most confidently labeled ones. The final classifier in RSSalg is obtained by training the base learner on a set created this way. Pruning of the examples is done by employing a genetic algorithm that keeps only the most reliable and informative cases. Our experiments performed on 17 datasets with various characteristics show that RSSalg outperforms all considered alternative methods on the more redundant natural language datasets and is comparable to considered alternative settings on the datasets with less redundancy.

Keywords: semi-supervised learning; co-training; ensemble learning; genetic algorithm

1 Introduction

Semi-supervised learning is a class of machine learning techniques that employs both labeled and unlabeled data for training. The goal of semi-supervised learning is to achieve high accuracy while demanding less human effort. In this paper we focus on the major semi-supervised learning method called co-training [1]. Its successful application is guaranteed when the dataset possesses a natural division of the features in two disjoined subsets, called view, such that each view is sufficient for learning and conditionally independent to the other given the class label. This makes co-training application limited because the needed optimal feature split is unknown in the great majority of settings.

In this paper we propose a novel co-training based method, called the **R**andom Split **S**tatistic **alg**orithm (RSSalg), which exploits the advantages of co-training and is exempt from co-training requirement for the existence of adequate feature split. In our previous work [2, 3, 4] we published promising preliminary results for our RSSalg methodology. However, the disadvantage of our earlier method was the introduction of threshold parameters to the co-training setting, which greatly affects the performance of RSSalg and needs to be carefully manually tuned for its successful application. In this paper we develop a methodology for automatic determination of these thresholds.

This paper is organized as follows. Section 2 presents the related work. Section 3 describes our RSSalg methodology. Section 4 presents the conducted experiment and achieved results. The results are discussed in Section 5. Finally, Section 6 concludes the paper and outlines the possible directions of future work.

2 Related Work

One way to enable co-training application on the single view datasets is by designing a methodology that can provide a good approximation of the optimal feature split. This approach resulted with some promising results [5-10]. However, approximating the optimal feature split is a difficult task, as the relation between the characteristics of the views and the performance of co-training has not been sufficiently investigated. Moreover, research [10] indicates that in the real-world situations where co-training would be most useful, that is, in the situations where we posses only a small training set, the independence and sufficiency assumptions of the views cannot be reliably verified. In these situations, split methods can be unreliable, and thus the performance of co-training is uncertain as it may degrade the classification performance when its assumptions are not met [11].

In this paper, we compare our proposed method to a co-training algorithm that uses a random split of features, shown to be beneficial in case of redundantly sufficient feature sets [5, 6].

Also, we compare our method to a method called maxInd [7]. MaxInd approximates the optimal feature split for co-training by crating two maximally independent views, given the class label. This approach is based on the conditional independence requirement for the views [1]. Experimental results with maxInd have shown that the prediction accuracy of co-training does not always become better by simply choosing truly independent views, leading the authors to conclude that the relation between the characteristics of views and the performance of co-training should be investigated more in detail. These findings have also been confirmed in [8], where the authors randomly generated a number of splits and tested the most independent splits against the least independent ones.

Another way researchers approach the problem of enabling co-training application on single view datasets is by combining co-training with ensemble learning [12]. This approach is mainly based on using an ensemble of classifiers in the place of two single classifiers in the co-training algorithm [13-15]. Methods based on this approach are able to significantly boost the performance of co-training. However, they usually rely on having a relatively large initial training set in order to build the initial ensemble of diverse and accurate classifiers. Moreover, in the case of a high dimensional dataset, classifiers trained on small bootstrapped data samples using single feature view may face the 'large *p*, small *n* problem' [16, 17]. One of the goals of our setting is to combine the advantages of co-training style algorithms and ensemble learning while keeping the training set at the same size as would be used in the original co-training setting (just a few labeled examples). Instead of employing multiple classifiers inside the co-training algorithm, our methodology exploits different configurations of co-training in order to create an ensemble of classifiers from only a few labeled examples.

In [18] it is argued that ensemble learning and semi-supervised learning can be mutually beneficial. Based on this, some approaches exploit unlabeled data for ensemble diversity augmentation [19, 20]. As opposed to other ensemble approaches, our methodology combines multiple co-training classifiers. In this way, a hierarchical ensemble is constructed: each co-training consists of two base classifiers, and our RSSalg methodology combines multiple co-training classifiers. In [21] we have also explored an alternative way of combining multiple co-training classifiers by treating them as inconsistent and unreliable annotators in an unsupervised multiple-annotation setting.

3 Methodology

In our setting we are given a training set of labeled examples *L*, which is relatively small and a set of unlabeled examples *U*, which is relatively large. Our goal is to determine the unknown labels of the instances in the given test set *T*.

The first step in our Random Split Statistic algorithm (RSSalg) is to create an ensemble of *m* diverse co-training classifiers $\{CL\}_i^m$. This is achieved by running co-training until its termination *m* times on a given dataset, using a different configuration for each of the m runs in order to get a different classifier. Each time co-training is run independently of the other co-training runs and each run of co-training uses a different random split of features, thus producing a different co-training classifier. A random feature split is created by random selection of half of the features from the feature set as the first view, and the remaining half of the features is treated as the second view. Each independent run of co-training produces a different pair of base classifiers, as each time a different feature split is used. Based on their confidence, each pair of base classifiers selects different

instances from the unlabeled set for labeling. Thus, it may happen that in different runs of co-training algorithm different instances are selected for labeling. Also, it may happen that the pairs of base classifiers from different co-training runs give a different label to the same instance. Consequently, in each co-training process a disparate enlarged training set L (consisting of initial labeled data and data labeled in the co-training process) is formed. We will refer to an enlarged training set created during the i^{th} co-training run as co-training result set $Lres_i$.

The second step in RSSalg is to form the final classifier by training it on the best instances from gained m co-training result sets $\{Lres_i\}_1^m$. The statistic S, based on co-training result sets, is created: for each $Lres_{\text{i}}$ and each instance e which is a member of $Lres_i$, the number of times instance e occurs in all co-training result sets n_{e} is determined (equation 1), and for each class C_k we count the number of times instance e is labeled C_k (equation 2).

$$n_e = |\{Lres_i \mid e \in Lres_i\,,\ i \in \{1..m\}\}| \tag{1}$$

$$n_{eCk} = |\{Lres_i \mid e \in Lres_i \wedge label(e) = C_k\,,\ i \in \{1..m\}\}| \tag{2}$$

Each instance e is assigned its most voted label C, determined by a majority vote of co-training base classifiers $\{CL\}_{\text{i}}^m$ (equation 3).

$$n_{eC} = \max(n_{eCk} \mid k \in \{1..m\}) \tag{3}$$

Falsely labeled instances would introduce noise into the training set for the final classifier. Our assumption is that the instance is reliable (in terms of high probability of being assigned the correct label) if the majority of co-training base classifiers $\{CL\}_{\text{i}}^m$, created in the first step of RSSalg, agree on the label of that instance. Thus, for each instance e, the label agreement percent e_{agg} is calculated as follows: $e_{\text{agg}} = n_{eC}/n_e$. An instance is marked as reliable if its label agreement percent exceeds the defined label threshold l_{ts}.

In the extreme case, an instance might occur only once in co-training result sets (n_{e}=1), and label agreement percent of that instance would be 100%. However, we consider this instance uninformative, as there is no other co-training classifier which would contradict or agree on its label, and labeling an instance based on the prediction of only one co-training classifier, which can possibly be of poor performance, is unreliable. Therefore, another condition for selection of instances in the training set for the final classifier is defined: an instance must be informative. Our assumption is that instances that appear in most of co-training resulting sets are informative (in terms of having good bases to improve the learning process). To define which instances appear in most co-training resulting sets, an example threshold e_{ts} is used. Let n be the number of different instances noted in statistic S. For each instance e from S we calculate the occurrence percent e_{occ} as follows: $e_{occ} = n_e/n$. An instance is marked as an informative instance if its occurrence percent exceeds the defined example threshold e_{ts}.

Thus, if we had optimal values of label threshold and example threshold, we could train our final classifier on instances marked both informative and reliable, where each instance would be assigned its most voted label. The result of the RSSalg is this final classifier which can be used to label previously unlabeled data. Pseudo code of RSSalg is presented in Figure 2.

Given:
- A small set L of labeled training examples
- A much larger set U of unlabeled examples
- A set T of labeled test instances (used for model evaluation)
- Label threshold L_{ts} and example threshold E_{ts}

for $i = 1..m$ iterations:
- Create two feature sets (views) x_1 and x_2 describing the examples by randomly splitting feature set x in half.
- Run co-training algorithm using L, U, x_1 and x_2. Algorithm outputs $Lres_i$ (enlarged labeled set L consisting from initial L and examples from U labeled by co-training algorithm)
- Update statistic S - for each instance e which is a member of $Lres_i$:
  - Increase the occurrence number of instance e, n_e.
  - for each class C_k, we count the number of times instance e is labeled C_k (equation 2)

for each instance e occurring in statistic S:
- Calculate the most voted label of the instance, C: $n_{eC} = \max(n_{eCk} \mid k \in \{1..m\})$ and label instance e as C.
- Calculate the label agreement percent: $e_{agg} = n_{eC}/n_e$
- Calculate the example agreement percent: $e_{occ} = n_e/n$ (n -number of examples in S)

for each instance e occurring in statistic S:
- If $e_{agg} > L_{ts}$ and $e_{occ} > E_{ts}$ add e (labeled as C) to initial training set L

Train learner on the enlarged training set L to get the final classifier. Apply final classifier to T.

Figure 2
Pseudo-code of RSSalg

An example case of forming the training set for the final classifier in RSSalg is shown in Figure 3. In this example, the label threshold is set to 80%, which means that the example is considered reliable if it is in at least 80% cases assigned the same label. The example threshold is set to 70%, which means that the example is considered informative if it is contained in at least 70% of the co-training result sets. Figure 3 shows that all examples from the initial training set L are contained in the final training set, as these examples appear in all co-training result sets and are assigned the same label in all these sets (co-training algorithm only adds examples to the initial training set without modifying the initial data). Example U_1 is added to the final training set as it is contained in all the co-training result sets (it is contained in 100% co-training result sets, which is more than the value of 70% defined for the example threshold) and in 80% of the cases it was assigned the positive label (which is the same as the value of 80% defined for the label threshold), and thus in the final training set U_1 is labeled positive. Example U_2 is also added to the final training set and labeled negative as it is contained in 80% co-training result sets and it is 87.5% of the times labeled negative (in 7 out of 8 cases in which it is contained in the co-training result sets). Example U_3 passes the example threshold (it is contained in 80% co-training result sets), but is not

considered reliable as it was 50% labeled positive and 50% labeled negative, and thus it is not included in the training set of the final classifier. Example U_4 is not included in the final training set as it does not pass the example threshold (it is contained in only 50% co-training result set), although it has a high label agreement (100% - in the five cases that it was labeled, it was labeled negative). Finally, the last unlabeled example U_m is excluded from the final training set as it fails to pass both thresholds.

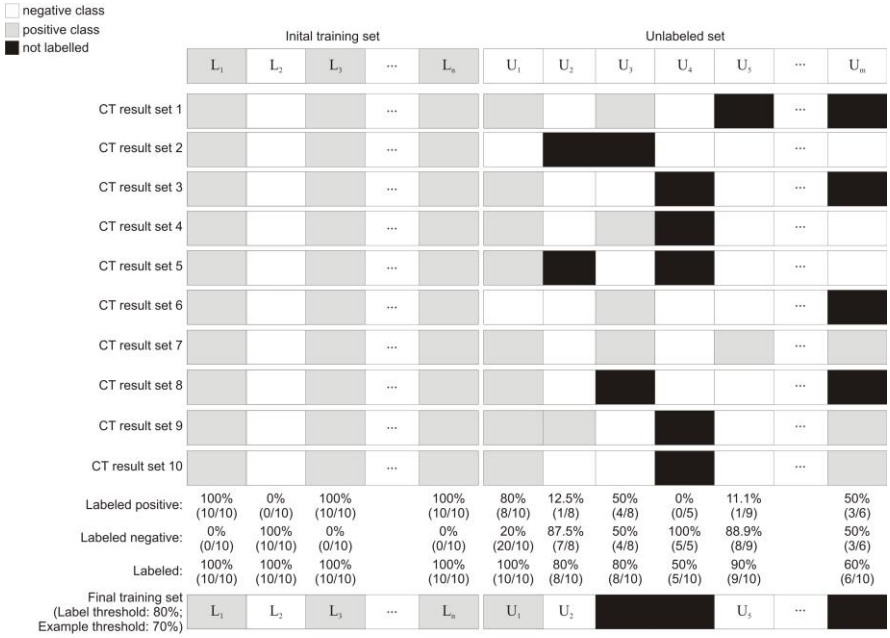

Figure 3
An example case of forming the training set for the final classifier

In order to find an optimal label threshold/example threshold pair (l_{ts} and e_{ts}) which maximizes the performance of the model output by RSSalg, a genetic algorithm is used.

3.1 Automatic Determination of Label Threshold/Example Threshold Pair

Defining label/example threshold pair is a complex optimization problem. Genetic algorithms [22] are a class of adaptive search and optimization techniques, extremely efficient at searching large solution spaces. They have been shown to be a robust and effective method for the optimization of complex problems characterized by multiple optima, nondifferentiability and other irregular features [23], and thus they are suitable for our problem of optimizing label/example threshold pair.

Genetic algorithm is a search heuristic applied in order to find approximate solutions to complex optimization and search problems. It mimics the behavior of natural selection in order to reach an optimum solution to any multi-dimensional problem [22].

In our methodology each individual has two chromosomes: one represents the label threshold and the other represents the example threshold. Both chromosomes have a binary string structure. The values of threshold range from 0% to 100%. Simply converting the threshold values to their binary values and handling them as binary strings would allow recombination operators to possibly change them in such a way that they fall out of specified range. For example, if we use 8-bit binary strings to code our values, a value of 1% encoded as 00000001 could fall out of range by modification of the first bit: 10000001 would encode 129%. Thus, if we use the n-bit encoding, we divide the number of all possible values (2^n-1) in 100 intervals and assign each interval one value from 0% to 100%. Thus, when converting the number x, which falls into range of (*minVal*, *maxVal*), to its n-bit binary encoding, we use equation 4. For example, the example threshold of 55% is encoded in 8-bit binary string as 10001100.

$$Binary[Round(\ (x - minVal) \cdot (2^n - 1) / (maxVal\text{-}minVal)\ )] \tag{4}$$

Each individual represents the label/example threshold pair. As we want to produce the model with the highest possible accuracy, the logical fitness function for the individual would be the accuracy of this classification model achieved on the set intended for model evaluation. However, in the co-training setting we are limited to only a few labeled examples and we lack the labeled test data necessary to evaluate the accuracy of our model.

Setting the label threshold/example threshold pair causes some of the data from co-training result sets that exceeds these thresholds to be selected as training data for the final model creation, and some of the data to be omitted from this selection. In the example shown in Figure 3, examples U_1, U_2 and U_5 are included in the final training set, while U_3, U_4 and U_m are shown to be omitted from the final training set. Based on the idea of out-of-bag estimation [24] to use the left-out examples from bootstrap samples to form accurate estimates of important quantities, we form the test set for model evaluation from the omitted data that did not fulfill the requirements to be included in training of the model, that is, from the data whose label agreement percent and example occurrence percent did not exceed label threshold and example threshold, respectively. Instances in thus formed test set are labeled by a majority vote of resulting co-training classifiers, in the same way as examples that did pass the thresholds. In the example shown in Figure 3, we evaluate the model trained on the examples $\{L_1,\ldots,L_n, U_1,U_2,U_5,\ldots\}$ on the test set $\{U_3,U_4,\ldots,U_m\}$.

However, some label threshold/example threshold pairs which might occur can cause all of the examples to be selected in the final training set, thus leaving no

examples left for evaluation of the individual. Also, too small test sets $\{U_3,U_4,\ldots,U_m\}$, with high possibility of noise, could impose a bad estimation of the performance of the created model. Thus, we define a testing threshold - a minimal number of examples needed in the test set for the evaluation of an individual. This testing threshold is dependent on the size of the total number of examples in statistic *S*. For example, if we use a testing threshold of 20%, each individual that uses more than 80% examples from the statistic *S* for the model, and less than 20% of the examples form *S* for testing the model, it is considered as poorly estimated. In these cases, we transfer examples from the training set $\{L_1,\ldots,L_n,\ U_1,U_2,U_5,\ldots\}$ to the test set $\{U_3,U_4,\ldots,U_m\}$ until we have enough examples in the test set for estimation. The examples that are transferred from the training set are those estimated to be the least confident ones. We estimate which examples are the least confident ones based on the label agreement percent and example agreement percent: those examples that have the smallest sum of these two values are considered to be the least confident ones.

In the selection stage individuals of the current generation are chosen from the population and allowed to reproduce. We used the proportional (roulette wheel) selection [25]. For recombination, we used bi-parental uniform crossover [26] and single point mutation operator [22] in order to produce new individuals. In our setting elitism is used to preserve the best individual (as determined from fitness evaluations) in each generation in order to increase the speed of the search. Elitism reserves two slots in the next generation for the highest scoring chromosome of the current generation.

3.2 The Motivation behind RSSalg Methodology

The motivation for our methodology is that unlabeled data can be helpful to ensemble learning [18]. Our methodology exploits ensemble learning in the terms of combining the result sets $Lres_i$, created in the co-training process by different co-training classifiers. Ensemble learning is only effective if the classifiers in the ensemble are both diverse and accurate [27], which can be achieved by applying an unstable base learning algorithm. Co-training possesses this feature of instability as it is sensitive to the two underlying assumptions on the views [5, 28], and therefore to the feature split division. Thus, by running co-training using different random splits of features, we should gain an ensemble of diverse classifiers. The empirical studies show that the co-training algorithm used with random split may be beneficial, provided that there is enough redundancy in the data [5]. A major characteristic of natural language datasets is the high level of feature redundancy [29]. Thus, by using co-training as the base learner, with different random splits of features on natural language datasets, we hope to gain an ensemble of both accurate and diverse classifiers. However, on the less redundant datasets, co-training using a random split is unreliable. In the experiment performed in this paper, we test RSSalg on both groups of datasets.

Our RSSalg methodology has certain similarities with bagging [30]. The first phase in both bagging and our RSSalg methodology is creating the bootstrap replicas of the original training set. In order to perform bagging, we need a large training set. The set of bootstrap replicates is created by randomly drawing examples from that large training set. In bagging, a necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members is their accuracy and diversity [27]. Bagging relies on the available training data in order to create the diversity. In the setting where we dispose with only a small amount of training examples, the diversity among the ensemble classifiers would be limited. Also, ensemble classifiers would be trained on bootstrap samples, which omit some of the training data, and thus their performance would be even worse than the performance of a weak classifier trained on the small labeled training set. Ensemble methods do not use unlabeled data as an additional source of knowledge but are rather designed when there is sufficient source of labeled data but only week learning algorithms.

In RSSalg, we dispose of the few training examples and a sufficiently large set of unlabeled examples. The set of bootstrap replicas is created from both labeled examples and unlabeled examples by applying co-training in order to incorporate unlabeled data into the original training set. In such a way we hope to increase the amount of training data and gain a more accurate classifier then the one trained on the original labeled set. The difference is that labels appointed by co-training to examples in bootstrap data are less reliable compared to the labels of examples in bootstrap data created by bagging.

The second step in bagging is to apply a base learning algorithm on each bootstrap replicate in order to gain an ensemble of learning models. Then, we can decide on a label of a previously unseen example by a majority vote of the resulting learning models. In RSSalg, we use majority voting in order to determine the final labels of the examples from bootstrap replicas. Even if the requirements for the successful application of co-training are met, in the co training process some noise is added to the initial training set through non-perfect classification. Moreover, the hard requirements of co-training are rarely met in practical situations, thus the performance of co-training is degraded by using non-ideal split. This introduces noise in bootstrap replicas created in the first phase of algorithm. Compared to other ensemble methods, bagging tends to be less accurate but stable and robust to classification noise [27]. Thus, by employing majority voting as in bagging, we hope to gain more accurate labels of examples in bootstrap replicas.

4 Empirical Evaluation

In this section we describe the experimental procedures and report the obtained results.

4.1 Datasets and their Preprocessing

We have tested our settings on tree natural language datasets: WebKB-Course [1, 5], News2x2 [5, 7] and LingSpam corpus [6, 8]. In addition, we have performed experiments on 14 UCI datasets. All of these datasets were previously used for co-training evaluation [10, 13]. We have selected these datasets that vary in size, dimensionality, redundancy and other characteristics in order to gain better insight on the effectiveness of our method in the real-world situations.

In case of natural language datasets we adopt the preprocessing technique used in [7] in order to compare our algorithm to the performance of co-training run using their artificial maxInd feature split. For dimensionality reduction, an English stop-word filter that removes 319 frequent words is used, and stemming is performed with Porter's stemming algorithm [31][1]. Based on document frequency, 200 most important features are chosen for each view. After dimensionality reduction, datasets are represented using the bag of words model with *tfidf* measurement [32]. It should be noted that both document frequency and *tfidf* measurement do not require the knowledge of the class label and therefore they do not violate the co-training setting in which labels are known for just a few initial training examples.

4.2 Evaluation Methodology

For evaluation, we adopt a 10-fold-cross validation procedure used in [7] in order to compare these algorithms more accurately. In this setting, data is divided in 10 stratified folds and in each of the 10 rounds of the validation process, a different fold is selected for random selection of required number of labeled training examples (the required number of training examples for each dataset is listed in Table 1, in the column denoted by |L|). The remaining data from that fold as well as 5 adjacent folds are used as unlabeled training data. The remaining 4 folds are used as test data. Thus, in each round, 40% of the data is used for testing, and 60% of the data is used for training. Each fold is used exactly once for the selection of labeled data, five times it is included as unlabeled data and four times it is used as a part of the test set.

In our experiments accuracy is used as the measure of performance. Also, a statistical significance test, a pair-wise t-test with p-value of 0.05, is applied to see if the differences in accuracy of the tested algorithms are statistically significant.

Co-training itself has a list of parameters that need to be configured. The number of examples for each class in the initial training set is chosen proportional to the

[1] In case of WebKB dataset, each sample was also filtered in order to remove the phone numbers, digits sequences, dates and non–alphanumeric characters which give no significance in predicting the class of the document.

class distribution in the dataset (for the only exception is the LingSpam dataset where we use the same settings as those in [7]). Parameters *n* and *p* that represent the numbers of examples per each class labeled by co-training inner classifiers at each iteration are also chosen proportionally to the class distribution in the dataset. By following [7], the size of the unlabeled pool *u* was 50 and the number of iterations used in each of the setting of co-training algorithm was 20. The number of different random splits used in RSSalg *m* was 50 for the natural language datasets and 30 for the UCI datasets. As the base classifier in co-training algorithm, we have used Naïve Bayes as it displays both speed and accuracy when applied on benchmark datasets.

We have used the following settings for the genetic algorithm: the generation size was 50, the number of iterations for genetic algorithm[2] was 10, the fixed probability of swapping bits in uniform crossover was 0.3, the probability of bit mutation was 0.02, and elitism was used. All these parameters were empirically chosen. The value of testing threshold was set to 20%. The datasets and their basic properties are listed in Table 1. The first four datasets in Table 1 (WebKB, LingSpam, News2x2 and Spambase from UCI) are natural language datasets, and the remaining 13 are UCI datasets. Both groups are sorted by parameter Gap.

Table 1
A summary of the datasets used in the experiment. Notation - **Dim:** the number of features describing the dataset; **|L|:** the size of the initial training set *L*, in the format of number of positive examples/number of negative examples; $\mathbf{L_{acc}}$**:** accuracy achieved by a supervised Naive Bayes classifier trained on the initial set *L*; **|All|:** the size of the entire training set *All* (i.e., the sum of numbers of labeled and unlabeled examples), in the format of number of positive examples/number of negative examples; $\mathbf{All_{acc}}$**:** accuracy achieved by supervised Naive Bayes classifier trained on the entire training set *All* (i.e., labeled examples and unlabeled examples with correct label); **Gap:** performance gap (also called the optimal gain in [10]) computed as All_{acc} - L_{acc}.

Datasets	Dim	|L|	L_{acc}	|All|	All_{acc}	Gap
WebKB	400	5/5	78.6	138/492	96.4	17.8
Spambase	57	2/1	67.7	1672/1087	79.6	11.9
LingSpam	400	5/5	80.1	288/1447	88.9	8.8
News2x2	400	5/5	81.1	600/600	89.6	8.5
Hepatitis	19	1/1	61.7	19/73	84.8	23.1
Kr-vs-kp	36	6/5	65.6	1001/916	87.2	21.6
Credit-g	20	1/1	53.6	420/180	74.1	20.5
Heart-statlog	13	3/2	65.7	90/72	80.5	14.8
Cylinder-bands	39	2/3	58.4	136/187	72.9	14.5
Sonar	60	1/1	55.5	66/58	68.8	13.3
Ionosphere	34	5/3	70.1	135/75	83.1	13.0
Breast-cancer	9	1/1	59.0	51/120	71.7	12.7
Credit-a	15	4/5	69.3	184/229	81.5	12.2

[2] Empirical experiments show that GA fastly converges after just a few iterations.

Tic-tac-toe	9	3/6	58.8	199/375	70.7	11.9
Breast-w	9	1/2	86.3	144/274	97.4	11.1
Mushroom	22	3/3	84.7	2524/2349	95.3	10.6
Diabetes	8	2/1	64.8	300/160	75.0	10.2

In our experiments the following co-training methods were considered as alternatives for solving the problem of co-training application on single-view datasets:

1) Co-training applied with a natural feature split (in the case where this split was known, i.e. WebKB, News2x2 and LingSpam dataset), hereinafter referred to as **Natural**.

2) Co-training applied with a random split of features, hereinafter referred to as **Random**. In order to give more realistic results, co-training is used with several random splits (the same *m* splits used in RSSalg) and the results are averaged.

3) To experiment with an alternative method of combining co-training classifiers gained in the first step of RSSalg, we use majority voting method, hereinafter referred to as **MV**.

4) Co-training applied with an artificial maxInd feature split introduced in [7]. MaxInd is not always successful in combination with Naïve Bayes and it displays better performance when applied with other base classifiers, such as RBF Nets and SVM [7]. Because of this, we report the best performance of maxInd achieved when using one of these three classifiers for each dataset, hereinafter referred to as $\mathbf{maxInd_{best}}$.

5) The RSSalg introduced in this paper, optimized with the genetic algorithm procedure presented in section 3.1., hereinafter referred to as **RSSalg**.

6) RSSalg with label threshold/example threshold pair optimized with genetic algorithm that uses the accuracy of the resulting classification model, achieved on the set intended for model evaluation, as the fitness function, hereinafter referred to as $\mathbf{RSSalg_{best}}$.

4.3 Experimental Results

The empirical studies [5] show that co-training algorithm using a random feature split may be beneficial if there is enough redundancy in the data. A major characteristic of natural language datasets is the high level of feature redundancy [29]. On the other hand, most of the UCI datasets are manually constructed with carefully chosen features and there should not be much redundancy in such created features. Thus, we will present and discuss the results achieved on natural language datasets and on the less redundant UCI datasets separately.

In Table 2 we show the accuracy and t-tests with 95% confidence obtained by each co-training method on all datasets.

Table 2
Comparisons of RSSalg and $RSSalg_{best}$ vs. four alternative co-training methods. The percent accuracy and standard deviation are reported based on 10-fold stratified cross-validation, and t-test with 95% confidence obtained by each co-training methods on all datasets. The Natural is applicable only for the first three datasets (WebKB, LingSpam, and News2x2) that have the known natural feature split. The results denoted by "v" indicate cases where RSSalg was significantly more accurate than the alternative co-training method, while a "*" indicates the significant difference in favor of the alternative method and no symbol represents a tie (no statistically significant difference). The overall results of the t-test are summarized in the rows "t-test" (separately for natural language datasets and UCI datasets) as the number of wins/ties/loses of RSSalg versus alternative co-training methods (e.g. 3/1/0 in the t-test row and the Random column means that RSSalg was statistically significantly more accurate than Random on 3 datasets, RSSalg and Random resulted in no statistical difference on 1 datasets, and zero (0) means no dataset indicated RSSalg was statistically significantly less accurate than Random in the experiment).

Datasets	Natural	Random	MV	$MaxInd_{best}$	RSSalg	$RSSalg_{best}$
WebKB	87.2±6.7	84.2±7.6	87.7±3.5	78.3±9.1 v	87.3±5.1	90.7±3.3
Spambase	-	67.8±15.8 v	77.4±4.7	68.9±8.2 v	78.2±7.7	81.5±4.1
LingSpam	70.3±13.3 v	76.6±8.5 v	81.1±7.4 v	83.9±1.1	88.5±7.0	91.1±5.9
News2x2	82.9±4.6 v	80.0±7.0 v	86.3±2.4 v	76.2±12.8 v	89.1±3.1	90.6±1.8
t-test	2/1/0	3/1/0	2/2/0	3/1/0	-	0/4/0
Hepatitis	-	80.3±8.0	83.3±4.3	80.8±7.9	82.6±3.5	86.5±3.4 *
Kr-vs-kp	-	54.4±5.0 v	55.3±4.5	60.1±6.2	58.3±5.7	67.1±4.2 *
Credit-g	-	62.0±5.5	64.4±5.5	68.1±1.8 *	62.7±6.8	70.2±0.7 *
Heart -statlog	-	79.4±8.2	81.8±2.0	80.8±4.5	81.1±3.2	83.3±2.2
Cylinder -bands	-	52.5±5.2	52.9±6.5	56.3±5.7	54.3±4.8	61.6±2.5 *
Sonar	-	54.9±6.0	56.5±5.5	56.7±9.1	56.7±8.0	61.2±5.8
Ionosphere	-	69.4±12.6	73.1±4.9	78.3±7.7	74.6±7.3	79.6±5.8
Breast -cancer	-	66.7±6.1	68.2±4.5	67.5±5.4	67.0±6.8	70.4±5.3
Credit-a	-	69.2±15.0	73.4±11.0	76.1±2.6	72.0±8.4	77.6±4.8
Tic-tac-toe	-	61.5±3.2	63.2±2.5	62.0±1.7	61.6±3.0	64.1±2.9
Breast-w	-	96.8±0.8	96.9±0.7	96.7±0.7	96.5±1.0	97.5±0.4 *
Mushroom	-	88.2±3.2	89.1±1.0	88.4±1.3	88.6±1.4	89.2±0.9
Diabetes	-	61.4±7.3	64.1±3.3	65.3±1.1	63.9±3.7	67.7±1.8 *
t-test	-	1/12/0	0/13/0	0/12/1	-	0/7/6

The aggregated t-test results obtained by each method against each other in terms of wins-ties-losses on natural language datasets and less redundant UCI datasets are reported at Tables 3 and 4, respectively. Methods include not only the 6 co-training methods, but also 2 additional Naive Bayes classifiers trained on the small labeled set *L* and on a much larger training set *All* respectively.

Table 3
Aggregate number of wins/ties/loses of each method against other methods over four natural language datasets. Comparisons versus NB_L (i.e., Naive Bayes classifier trained on the initial set *L*) and NB_All (i.e., Naive Bayes classifier trained on a much larger training set *All*) are reported in the last two rows/columns. For example, 1/2/0 in the row MV and column Natural means that MV was statistically significantly more accurate than Natural on 1 dataset, MV and Natural resulted in no statistical difference on 2 datasets, and MV was statistically significantly less accurate than Natural on 0 datasets.

	Natural	Random	MV	$maxInd_{best}$	RSSalg	$RSSalg_{best}$	NB_L	NB_All
Natural	-	0/2/1	0/2/1	1/1/1	0/1/2	0/1/2	1/2/0	0/0/3
Random	1/2/0	-	0/3/1	1/2/1	0/1/3	0/0/4	1/3/0	0/0/4
MV	1/2/0	1/3/0	-	3/1/0	0/2/2	0/2/2	3/1/0	0/1/3
$maxInd_{best}$	1/1/1	1/2/1	0/1/3	-	0/1/3	0/0/4	0/4/0	0/0/4
RSSalg	2/1/0	3/1/0	2/2/0	3/1/0	-	0/4/0	4/0/0	0/3/1
$RSSalg_{best}$	2/1/0	4/0/0	2/2/0	4/0/0	0/4/0	-	4/0/0	0/3/1
NB_L	0/2/1	0/3/1	0/1/3	0/4/0	0/0/4	0/0/4	-	0/0/4
NB_All	3/0/0	4/0/0	3/1/0	4/0/0	1/3/0	1/3/0	4/0/0	-

Table 4
Aggregate number of wins/ties/loses of each method against other methods over 13 UCI datasets. Comparisons versus NB_L (i.e., Naive Bayes classifier trained on the initial set *L*) and NB_All (i.e., Naive Bayes classifier trained on a much larger training set *All*) are reported in the last two rows/columns. For example, 1/12/0 in the row RSSalg and column Random means that RSSalg was statistically significantly more accurate than Random on 1 dataset, RSSalg and Random resulted in no statistical difference on 12 datasets, and RSSalg was statistically significantly less accurate than Random on 0 datasets.

	Random	MV	$maxInd_{best}$	RSSalg	$RSSalg_{best}$	NB_L	NB_All
Random	-	0/13/0	0/9/4	0/12/1	0/4/9	7/4/2	0/2/11
MV	0/13/0	-	0/13/0	0/12/0	0/7/6	7/4/2	0/2/11
$maxInd_{best}$	4/9/0	0/13/0	-	1/11/0	0/8/5	6/6/1	0/2/8
RSSalg	1/12/0	0/13/0	0/12/1	-	0/7/6	4/7/1	0/3/10
$RSSalg_{best}$	9/4/0	6/7/0	5/8/0	6/7/0	-	12/1/0	0/5/8
NB_L	2/4/7	2/4/7	1/6/6	1/7/4	0/1/12	-	0/0/13
NB_All	11/2/0	11/2/0	10/3/0	10/3/0	8/5/0	13/0/0	-

In summary, in our conducted experiments the proposed method RSSalg had the following properties as compared to alternatives on natural language datasets:

1) On two datasets the proposed method outperformed Natural and they resulted in a statistical tie on one remaining dataset (WebKB). These results indicate that RSSalg is generally better then Natural.

2) On three datasets RSSalg outperformed Random and they resulted in a statistical tie on one remaining dataset (WebKB). These results indicate that RSSalg is generally better then Random.

3) On two datasets RSSalg outperformed MV (LingSpam and News2x2) and they resulted in a statistical tie on two remaining datasets. These results indicate that, RSSalg is generally better then MV.

4) On three datasets RSSalg outperformed $MaxInd_{best}$ and they resulted in a statistical tie on one remaining dataset (LingSpam). Therefore, it is generally better than maxInd because RSSalg outperformed maxInd's best performance on the majority of datasets.

5) On all four datasets RSSalg and $RSSalg_{best}$ result in a statistical tie. As $RSSalg_{best}$ represents the upper bound of performance of RSSalg on the test set, we can conclude that our method for automatic threshold determination was successful.

6) On all four datasets RSSalg outperformed the Naive Bayes classifier trained on the small labeled set *L*. In contrast, none of the alternative co-training methods outperformed the Naive Bayes to that level.

7) On three datasets RSSalg resulted with a statistical tie with Naive Bayes trained using a much larger set of labeled data *All*, and it lost to NB_All on one dataset (WebKB). None of the alternative methods were able to achieve this on the majority of datasets (only MV achieved a statistical tie with Naive Bayes classifier trained on a large labeled set on one dataset, Spambase).

In our experiments co-training using a natural feature split outperformed Naive Bayes classifier trained on the small labeled set *L* on only one dataset (WebKB).

In our experiments on the group of natural language datasets, maxInd was not beneficial. Its best setting ($maxInd_{best}$) has resulted with a statistical tie with Naive Bayes classifier trained on the small labeled set *L*.

On natural language datasets, in our experiments MV outperformed maxInd on three datasets and resulted with a statistical tie with maxInd on the remaining dataset (LingSpam). MV resulted with a statistical tie with Natural and Random on most datasets. However, as opposed to Natural, Random and maxInd it outperformed Naive Bayes classifier trained on the small labeled set *L* on the majority of datasets (it is only tied with NB_L on LingSpam dataset).

Generally, in our experiments on natural language datasets, RSSalg performed the best, followed by MV, and finally Natural, Random and maxInd which turned out to be of similar performance.

In the summery of our experiments on UCI datasets, the proposed method RSSalg had the following properties as compared to alternatives:

1) On one dataset (Kr-vs-kp) RSSalg outperformed Random and they resulted in a statistical tie on the remaining 12 datasets. These results indicate that RSSalg performs similarly to Random.

2) On all 13 datasets RSSalg and MV resulted in a statistical tie. Thus, RSSalg and MV have a similar performance.

3) RSSalg and $MaxInd_{best}$ resulted in a statistical tie on 12 datasets and RSSalg loses to $MaxInd_{best}$ on one remaining dataset (Credit-g). Thus, RSSalg performs similar to $MaxInd_{best}$.

4) On 7 datasets RSSalg results with a statistical tie with $RSSalg_{best}$ and RSSalg loses to $RSSalg_{best}$ on 6 datasets. As $RSSalg_{best}$ represents the upper bound of performance of RSSalg on the test set, we can conclude that our automatic threshold determination was successful on the majority of datasets. However, it is less effective compared to its performance on natural language datasets.

5) On 4 datasets RSSalg outperformed the Naive Bayes classifier trained on the small labeled set *L*, resulted in a statistical tie with it on 7 datasets, and lost to Naive Bayes classifier trained on small labeled set *L* on two datasets. Thus, given less redundant datasets, it is unreliable weather RSSalg will be beneficial.

6) On three datasets RSSalg resulted with a statistical tie with Naive Bayes trained using a much larger set of labeled data *All*, and it lost to NB_All on 10 datasets.

Generally, in our experiments on UCI datasets, Random, maxInd, and MV show similar performance. RSSalg showed a slightly worse performance in that it outperformed the Naive Bayes classifier trained on the small labeled set *L* on fewer datasets, compared to other benchmark algorithms.

It should be noted that $RSSalg_{best}$ has the best performance of all benchmark algorithms on UCI datasets:

1) $RSSalg_{best}$ outperformed Random on 9 datasets and they resulted in a statistical tie on the remaining 4 datasets.

2) $RSSalg_{best}$ outperformed MV on 6 datasets and they resulted in a statistical tie on 7 datasets.

3) $RSSalg_{best}$ outperformed $maxInd_{best}$ on 5 datasets and they resulted in a statistical tie on 8 datasets.

4) On 12 datasets $RSSalg_{best}$ outperformed the Naive Bayes classifier trained on the small labeled set *L* and resulted in a statistical tie with it on one dataset. In contrast, none of alternative co-training methods outperformed the Naive Bayes to that level, and some co-training methods even degraded the performance of the Naive Bayes classifier on certain datasets.

5) On 5 datasets RSSalg resulted with a statistical tie with Naive Bayes trained using a much larger set of labeled data *All*, and it lost to NB_All on 8 datasets.

Thus, a better method for automatic determination of thresholds for RSSalg on the less redundant datasets would greatly improve its performance.

The limitation of RSSalg is its time complexity. However, it runs completely off-line without any human interaction, which makes this issue less of a problem. Also, RSSalg can be easily executed in parallel, as multiple runs of co-training in RSSalg are completely independent.

Random should have the lowest time complexity of all benchmark algorithms. The time complexity of the MaxInd algorithm should increase as the number of features in the dataset increases, as it requires calculating pair wise *CondMI*s for

all pairs of words that are in different sets in order to determine maximally independent views [7]. Finally, MajorityVote is less complex then RSSalg as it does not require genetic algorithm optimization or training of the final classifier as in RSSalg. However, MV requires an additional step that is not performed in RSSalg, namely applying each of the obtained co-training classifiers on the test set. As RSSalg, MV requires multiple independent co-training runs, which can easily be executed in parallel.

Conclusion

In this paper we have proposed RSSalg, a methodology designed enable successful application of co-training on single-view datasets and boost its performance. Our method is most successful on datasets with enough feature redundancy, such as natural language datasets. RSSalg relies on co-training applied with different random feature splits in order to form the accurate enlarged training set.

We have compared the accuracy of the proposed method to several alternative co-training methods applicable to single-view datasets. In addition, the new method is compared to co-training with a natural feature split in cases where such a split was known. Our algorithm outperformed all considered alternative methods for co-training on the more redundant natural language datasets, while it was comparable to the considered alternative settings on the less redundant UCI datasets.

Our genetic algorithm technique of determination of most accurate and informative examples would perform its best if we could define the fitness function as actual accuracy achieved on the test set. However, in the co-training setting we are limited to only a few labeled examples and we lack the labeled test data necessary for this fitness function. We have used this model $RSSalg_{best}$ as indication of the upper bound on the performance of our RSSalg. Our results show that the performance of RSSalg reaches the performance of $RSSalg_{best}$ as its upper bound accuracy on natural language datasets, while it is less successful on the less redundant UCI datasets. $RSSalg_{best}$ has outperformed all considered co-training methods on UCI datasets; thus, a better method for automatic determination of thresholds for RSSalg on the less redundant datasets would greatly improve its performance, and this remains a task for the future.

Our empirical experiments with $RSSalg_{best}$ suggest that, given optimal parameters, RSSalg is consistent on various classification domains and does not require any specific requirements about the dataset to be met. However, the determination of optimal parameter values, without using labeled examples, remains to be improved in the future.

The downfall of genetic search is that it suffers from serious variance (i.e., very different results may be obtained by different runs). In the future we plan to undertake an experimental study in order to analyze the difference caused by different runs of genetic search.

Another limitation is the introduction of an additional parameter in the co-training setting, represented as the number of co-training classifiers created in the first step of RSSalg. In the future we plan to experiment with different parameter settings in order to investigate their impact on RSSalg.

In the experiments presented in this paper we have considered only binary classification problems, and in future research we plan on extending our method to multiple-category label problems. Also, we plan on undertaking more experimental studies in order to compare our method to ensemble approaches that exploit unlabeled data, such as co-forest [14]. Finally, we hope to integrate our solution into the information system for monitoring the scientific research activity of the University of Novi Sad (CRIS UNS)[3]. Our solution would serve as an additional support of system for automatic extraction of metadata from scientific publications [33]. The goal is to overcome the problem of manual annotation of a large number of scientific papers. Finally, we hope to apply our algorithm as the solution for automatic mining of methodologies from scientific articles [34].

Acknowledgement

Results presented in this paper are a part of the research conducted within the Grant No. III-47003 provided by the Ministry of Education and Science of the Republic of Serbia. The authors are thankful to Dr. Felix Feger and Dr. Irena Koprinska for sharing the details of their work.

References

[1] A. Blum and T. Mitchell: "Combining Labeled and Unlabeled Data with Co-Training", *Proc. 11th Annual Conf. Computational Learning Theory*, pp. 92-100, 1998

[2] J. Slivka, A. Kovačević and Z. Konjović: "Co-Training-based Algorithm for Datasets without the Natural Feature Split", *Proc. IEEE Int'l Symp. Intelligent Systems and Informatics*, pp. 279-284, Serbia, 2010

[3] J. Slivka, A. Kovačević and Z. Konjović: "Multi-Label Classification Experiments with Co-Training-based Algorithm" *Int'l Conf. Internet Society Technology and Management*, Serbia, 2011

[4] J. Slivka, A. Kovačević and Z. Konjović: "Co-Training-based Algorithms Applied to Subjectivity Detection Task" *Int'l Conf. Internet Society Technology and Management*, Serbia, 2012

[5] K. Nigam and R. Ghani: "Understanding the Behavior of Co-Training", *Proc. Workshop on Text Mining at KDD*, 2000

[6] J. Chan, I. Koprinska and J. Poon: "Co-Training with a Single Natural Feature Set Applied to Email Classification", *Proc. IEEE/WIC/ACM Int'l Conf. Web Intelligence*, pp. 586-589, 2004

[3] http://cris.uns.ac.rs/

[7] F. Feger and I. Koprinska: “Co-Training Using RBF Nets and Different Feature Splits”, *Proc. Int’l Joint Conf. Neural Network*, pp. 1878-1885, 2006

[8] M. Terabe and K. Hashimoto: “Evaluation Criteria of Feature Splits for Co-Training”, *Proc. Int’l MultiConference of Engineers and Computer Scientists*, 2008

[9] A. Salaheldin and N. El Gayar: “New Feature Splitting Criteria for Co-Training using Genetic Algorithm Optimization” Lecture Notes in Engineering and Computer Science, 5997/2010, pp. 22-32, 2010

[10] J. Du, C. Ling and Z. Zhou: “When Does Co-Training Work in Real Data?”, *IEEE Trans. on Knowledge and Data Engineering*, 23(5), pp. 788-799, 2010. ISSN 1041-4347, 2010

[11] D. Pierce and C. Cardie: “Limitations of Co-Training for Natural Language Learning from Large Datasets”, *Proc. 2001 Conf. on Empirical Methods in Natural Language Processing*, 2001

[12] Z.-H. Zhou: "Ensemble Methods: Foundations and Algorithms" In: Boca Raton, FL: Chapman & Hall, 2012

[13] Z.-H. Zhou and Ming Li: “Tri-training: Exploiting unlabeled data using three classifiers”, *IEEE Trans. Knowledge and Data Engineering* 17, pp. 1529-1541, 2005

[14] M. Li and Z. H. Zhou: “Improve Computer-aided Diagnosis with Machine Learning Techniques using Undiagnosed Samples” *IEEE Trans. on Systems, Man and Cybernetics*, 37(6), pp. 1088-1098, 2007

[15] M. F. Abdel Hady and F. Schwenker: “Co-Training by Committee: A New Semi-supervised Learning Framework” *Proc. IEEE Inte’l Conf. Data Mining Workshops*, pp. 563-572, 2008

[16] M. West: “Bayesian Factor Regression Models in the “Large p, Small n” Paradigm”, In J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith, and M. West, editors, *Bayesian Statistics 7*, pp. 723-732, Oxford University Press, 2003

[17] Y. Yaslan and Z. Cataltepe: “Co-Training with Relevant Random Subspaces”, Neurocomputing, Vol. 73(10-12), pp. 1652-1661, 2010

[18] Z-H. Zhou: “When Semi-supervised Learning Meets Ensemble Learning”, *Proc. Int’l Workshop on Multiple Classifier Systems*, pp. 529-538, 2009

[19] M-L. Zhang and Z-H. Zhou: “Exploiting Unlabeled Data to Enhance Ensemble Diversity”, Data Mining and Knowledge Discovery, pp. 1-32, 2011

[20] L. Izsó and P. Tóth: “Applying Web-Mining-Methods for Analysis of Student Behaviour in VLE Courses”, Acta Polytechnica Hungarica, Vol. 5, No. 4, pp. 79-92, 2008

[21] J. Slivka, P. Zhang, A. Kovačević, Z. Konjović and Z. Obradović: "Semi-Supervised Learning on Single-View Datasets by Integration of Multiple Co-Trained Classifiers", 11th International Conference on Machine Learning and Applications (ICMLA 2012), *December 12-15, Boca Raton, Florida, USA*, 2012

[22] D. Goldberg: "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, 1989

[23] R. E. Dorsey and W. J. Mayer: "Genetic Algorithms for Estimation Problems with Multiple Optima, Nondifferentiability, and Other Irregular Features," Journal of Business and Economic Statistics, 13(1), 53-66, 1995

[24] L. Breiman: Out-of-Bag Estimation, Technical Report, Statistics Department, University of California, 1996

[25] T. Back and F. Hoffmeister: "Extended Selection Mechanisms in Genetic Algorithm", In: Belew, R. K. & Brooker, L. B., editors, *Proc. 4th Int'l Conf. on Genetic Algorithm*, pp. 92-99, 1991

[26] G. Syswerda: "Uniform Crossover in Genetic Algorithms", *Proc. 3rd Int'l Conf. Genetic Algorithms*, pp. 2-9, 1989

[27] T. G. Dietterich: "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization", Machine Learning, 40(2), pp. 139-157, 2000

[28] I. Muslea, S. Minton and C. Knoblock: "Active + Semi-supervised Learning = Robust Multi-View Learning", *Proc. ICML*, 2002

[29] T. Joachims: "A Statistical Learning Model of Text Classification for Support Vector Machines" *Proc. ACM-SIGIR Int'l Conf. Research & Development in Information Retrieval*, pp. 128-136, 2001

[30] L. Breiman: "Bagging Predictors", Machine Learning, Vol. 24(2), pp. 123-140, 1996

[31] M. F. Porter: "An Algorithm for Suffix Stripping", Program: Electronic Library and Information Systems, 14(3):130-137, 1980

[32] Gerard Salton and Chris Buckley: "Term-Weighting Approaches in Automatic Text Retrieval", Information Processing and Management, Vol. 24(5) pp. 513-523, 1988

[33] Kovačević, A., Ivanović D., Milosavljević B., Konjović Z., Surla D., 2011. "Automatic Extraction of Metadata from Scientific Publications for CRIS Systems" Program: Electronic Library and Information Systems, 45(4), pp. 376-396, 2011

[34] Kovačević, A., Konjović Z., Milosavljević B., Nenadic G., "Mining Methodologies from NLP Publications: A Case Study in Automatic Terminology Recognition", Computer Speech & Language, 26(2), pp. 105-126, 2011