Weights Space Exploration Using Genetic Algorithms for Meta-classifier in Text Document Classification*

Radu G. CREȚULESCU, Daniel I. MORARIU, Macarie BREAZU, Lucian N. VINȚAN

"Lucian Blaga" University of Sibiu,
10, Victoriei Street., 550024, Sibiu, Romania
radu.kretzulescu@ulbsibiu.ro, daniel.morariu@ulbsibiu.ro, macarie.breazu@ulbsibiu.ro,
lucian.vintan@ulbsibiu.ro

Abstract: Automatic document classification has become an important task because of the continually increasing number of text documents with the users have to deal with. The aim of this paper is to develop a non-adaptive meta-classifier for text documents that has an increased classification accuracy. The developed meta-classifier is based on combining some SVM classifiers and a Naïve Bayes classifier. We proposed a new meta-classification method which takes into consideration the corresponding positions and confidence degrees obtained for all the classes. In this work we have tried to find, using Genetic Algorithms, the optimal weighting factors for the values returned by each classifier separately. Consequently, it is possible for the meta-classifier to select as the winner class, a class that is not hierarchized as the first one by any of the compounded classifiers. The experimental results have showed that the classification accuracy can be improved through the proposed method.

Keywords: Text Classification and Performance Evaluation, SVM, Meta-classification, Genetic Algorithms

1. Introduction

While more and more textual information is available online, effective retrieval is difficult without good indexing and summarization of document content. Document categorization is one solution to this problem. The task of document categorization is to assign a user defined categorical label to a given document. In recent years a growing number of categorization methods and machine learning techniques have been developed and applied in different contexts.

Documents are typically represented as vectors in a features space. Each word in the vocabulary is represented as a separate dimension. The number of a certain word's occurrences in a document represents the value of the corresponding component in the document's vector.

In this paper we investigate a strategy for combining classifiers' results in order to improve the classification accuracy using genetic algorithms. We used classifiers based on Support Vector Machine (SVM) techniques and based on Naïve Bayes theory. They are less vulnerable to degrade with an increasing dimensionality of the feature space, and have been shown effective in many classification tasks. The SVM classifier is actually based on learning with kernels and support vectors.

We combine multiple classifiers hoping that the classification accuracy can be improved without a significant increase in response time. Instead of building just one highly accurate specialized classifier with much time and effort, we build and combine several simpler classifiers.

Several combination schemes have been described in the literature [5],[8] and [1]. A usual approach is to build individual classifiers and later combine their judgments to make the final decision. Another approach, which is not so commonly used because it suffers from the "curse of dimensionality" [7], is to concatenate features from each classifier to make a longer feature vector and use it for the final decision. Anyway, meta-classification is effective only if its component classifiers synergies can be exploited.

In previous studies combination strategies were used ad hoc and are strategies like majority vote, linear combination, winner-take-all [5], or Bagging and Adaboost [15]. Also, some rather complex strategies have been suggested. For example in [7] and [10] meta-classification strategies using SVM [14] are presented and are compared with probability based strategies.

Section 2 and 3 contains prerequisites for the main work developed in this research. In sections 4 we present the methodology used for developing our experiments. Section 5 presents the experimental framework and section 6 presents the main results of our experiments. Finally the last section debates and concludes on the most important results obtained and proposes some further work.

* A previous, shorter version of this paper was presented in "*The Second International Conference on Information Science and Information Literacy*", with the title "*Using Genetic Algorithms for Weight Space Exploration in an Eurovision-like weighted Meta-Classifier*".

2. Classifiers Used

Support Vector Machine

The Support Vector Machine (SVM) is a classification technique based on statistical learning theory that was applied with great success in many challenging non-linear classification problems and on large data sets ([12], [14]).

The SVM algorithm finds a hyperplane that optimally splits the training set. The optimal hyperplane can be distinguished by the maximum margin of separation between all training points and itself. Looking at a two-dimensional problem we actually want to find a line that "best" separates points in the positive class from points in the negative class. The hyperplane is characterized by a decision function like:

$$f(x) = \text{sgn}\left(\left\langle \mathbf{w}, \mathbf{\Phi}(x) \right\rangle + b\right) \quad (1)$$

where **w** is the weight vector, orthogonal to the hyperplane, "*b*" is a scalar that represents the margin of the hyperplane, "*x*" is the current sample tested, "*Φ(x)*" is a function that transposes the input data into a higher dimensional feature space and $\langle \cdot, \cdot \rangle$ representing the dot product. *Sgn* is the sign function. If **w** has unit length, then <**w,** Φ*(x)*> is the length of Φ*(x)* along the direction of **w**. Generally **w** will be scaled by ||**w**||. In the training part the algorithm needs to find the normal vector "**w**" that leads to the largest "*b*" of the hyperplane.

Naïve Bayes

The Bayes classifier uses the Bayes Theorem which basically computes prior probabilities for a given class based on the probability for a given term to belong to the specified class. Thus the classifier computes the probability for a document to be into a given class.

Bayesian theory works as a framework for making decision under uncertainty - a probabilistic approach to inference [6] - and it is particularly suited when the dimensionality of the inputs data is high. Bayes theorized that the probability of future events could be calculated by determining their earlier frequency.

The Naive Bayes classifier is based on the simplified assumption that the attribute values are conditionally independent given the target value. In other words the assumption is that, given the target value of the instance, the probability of observing the conjunction y_1, y_2.....y_n is just the product of the probabilities of the individual attributes:

$$c_{map} = \arg\max_{1<j<m} \overline{P}(Y_j | X) = \arg\max_{1<j<m} \overline{P}(Y_j) \prod_{i=1}^{n} \overline{P}(x_i | Y_j), \quad (2)$$

where Y_j is the class *j*, with $j = \overline{1,m}$ and *X* is the vector set and x_i is the vector representation for the given document with *n* terms (words in our case), $x_i \in X$. Because, in almost cases the dataset is not complete we will compute the estimated probability that we will be noted with $\overline{P}$. For extending the SVM and the Naïve Bayes classifiers from two-class classification to multi-class classification typically one of two methods are used: "One versus the rest", where each topic is separated from the remaining topics, and "One versus the one", where a separate classifier is trained for each class pair [14**].** We selected the first method for two reasons. First, preliminary experiments show that this method gives better performance, which might be explained by the fact that the Reuter's database contains strongly overlapped classes and assigns almost all samples in more than one class. Second, overall training time is much shorter for the first method.

3. Genetic Algorithms

Genetic algorithms encode a potential solution to a specific problem on a simple chromosome-like data structure and apply genetic operators to these structures in order to preserve critical information [17]. In our weights selection problem the chromosome is considered to be of the following form:

$$c = (w_1, w_2, ..., w_k) \quad (3)$$

where w_j, $j = \overline{1,k}$ represents the weight for each class position returned by every classifier used into the meta-classifier. The training set has the form $\{\langle x_{ij}, y_j \rangle, i = \overline{1,m}$ *and* $j = \overline{1,k}\}$, where y_j represents the correct output for the sample x_{ij}, *k* represents the number of classes and *m* represents the number of classifiers used into the meta-classifier. The sample *x* is a matrix where the lines are classifiers and the

columns are the classes. For the genetic algorithm the training dataset consists of all outputs from all classifiers used into the meta-classifier. For simplifying the computing of the fitness function we choose a representation of the chromosome that uses the outputs of all classifiers from the meta-classifier. With this approach we take into consideration the position of each class in the classifier's output. The chromosome keeps only, for each class position, the weights that are used to compute the accuracy of meta-classifier. Thus, the potential solution of the problem encodes the weights that can be used to compute the final decision into the meta-classifier [3].

For each chromosome we can compute the fitness function as the final classification accuracy of the meta-classifier using the testing file. The classification accuracy is computed as the number of final correct classified classes by the meta-classifier.

For each test we start with a generation of 100 chromosomes, each of them having values randomly generated between -1 and 1. In the next step we generate the next population using genetic operators as: selection, crossover and mutation [17]. The evolutionary process stops after a predefined number of generations are taken or when in the last 20 generations no changes occur.

At the end of the algorithm, we obtain for each class position the "best" weight that will be used for computing the winner class. The general scheme of the genetic algorithm is presented in pseudo code in the Figure 1.

4. Meta-classifier Models

In our previous work [9] and [11] it is presented a meta-classifier, based on 8 SVM classifiers and one Bayes classifier that were used to improve the classification accuracy for text documents. In those works 3 meta-classifier models are used: majority vote, selection based on the Euclidian distance and selection based on the cosine angle. The first model was a non-adaptive model that had the advantage of speed, but obtained not so good results. The last two were adaptive models that used a training part before they were used in the classification. The training part is very time consuming and, unfortunately, this time increases when the meta-classifier learns more examples.

In this paper we propose a new non-adaptive meta-classifier model based on some pre-optimized weights that will increase the classification accuracy. In [3] we have presented a solution for this problem that uses static (pre-determined) values for the weights. Now we propose to use a genetic algorithm for pre-computing the optimal weights. Obviously, this adaptive optimization process will be done before the meta-classification process will start. The obtained weights' values will correspondingly weight all the results returned by every classifier for each document, as it is shown further.

All meta-classifiers presented in this article contain eight SVM type classifiers and one Naïve Bayes classifier [11].

```
Begin
 For each topic from a topics_set
  begin
   generate a population
   while not terminated condition
    For each chromosome from population
     compute the fitness functions
    make next population:
       select parents
       recombine pairs of parents
       apply mutation to offspring
   End while.
     Store the chromosome that obtain the best fitness
 End for.
     Take all stored chromosomes
End.
```

Figure 1. Pseudo code for GA algorithm

4.1 Non-adaptive Meta-classifier based on Majority Vote (M-MV)

This meta-classifier model, also presented in [9], is a non-adaptive model that obtains the same results for the same input every time. The idea is to use all selected classifiers to classify the current document. Each classifier will propose a class for the given document, and the meta-classifier will increment the corresponding class-counter. The winner class proposed by the meta-classifier is the class with the highest count. If we obtain two or more classes with identical maximal counts then we classify the current document in all the proposed classes. This method uses only the winner class from the classification returned by each classifier separately. The classes found on the subsequent positions are not taken into consideration.

4.2 Non-adaptive Meta-classifier based on sum (M-SUM)

Each classifier has as input a document that is represented as a vector having 1306 features and produces an output vector with 16 values. The output values represent the confidence degrees, given by the classifier. These values represent the "belonging degree" of the current document to each one of the 16 classes (see Figure 2). More precisely, each element from the output vector represents the classifier's decision function value for each class separately. So far, in [9], as it was already described in paragraph 4.1, the highest value from the output vectors has been chosen and the class corresponding to that value was considered to be the winner class. The methods proposed in the following paragraph, take into account both the values obtained for each class separately and the corresponding rating position. Since there are 9 classifiers in the meta-classifier for each document there will be 9 corresponding vectors, noted *V[k]*, each of them having 16 values (so-called confidence degrees) and $k = \overline{1,9}$.

The values of the decision functions for the SVM type classifiers are in the range (-∞,∞) but usually close to the value 0, and the values of the Naive Bayes classifier are in the range (-∞, 0). Considering these differences, and for making a correct sum of those values, we have transposed the values of all vectors into the interval [1,∞). The formula used for transposing the values is:

$$\vec{V}'_i = \vec{V}_i + \left|\min(\vec{V})\right| + 1, \ for\ i = \overline{1,16} \tag{4}$$

This transposing formula doesn't change the differences between the elements values from a vector even if it contains positive and negative values or only positive ones. Thus, for each vector the differences between its values remain unchanged. In order to calculate the sum of these vectors in the next step we have normalized the vectors bringing their values in the (0, 1] interval (see equation 5). This normalization ensures that the value from the first position, of the descended arranged vector, is always 1. We avoid also that the value from the last position to be 0.

$$\vec{V}''_i = \frac{\vec{V}'_i}{\max(\vec{V})} \tag{5}$$

In the current meta-classifier, that only makes the un-weighted sums, (called here M-SUM), we have calculated the sum of corresponding scalars from these 9 vectors. Thus we have obtained, for the current document, a single vector with 16 values. The meta-classifier will decide the winning class as the class with the highest obtained value. If there are two or more classes with identical maximal values, the meta-classifier will propose all those classes.

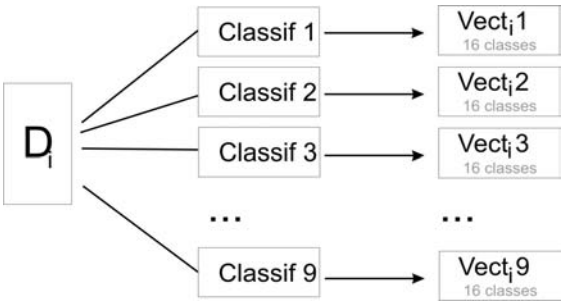

Figure 2. Creating entry data for meta-classifiers

$$Class = \max_{c_i, i=\overline{1,16}} \left(\sum_{k=1}^{9} V_i''[k]\right) \quad (6)$$

where c_i represents the final vector of classes. This approach gives the possibility that, even if a class that was never selected as a winner by the classifiers but always obtained a value very close to the winner class, such a class should have the chance that, after summing, to be selected by the meta-classifier as the winner class. The idea started from the fact that in many articles like [4] and [11] we observed the tendency to use methods for choosing the second or the third class as the winner class mostly when it was obvious that the meta-classifier would not provide the correct winner class.

4.3 Non-adaptive Meta-classifier based on weighted sum (M-GSUM)

In this section we will introduce a new non-adaptive meta-classifier based on the weighted sum (we will call it as M-WSUM).

In this meta-classifier the values for the classes will be multiplied with a value, called weight. In [3] are presented some meta-classifiers using different weight values and their obtained results. Since choosing the optimum weight values for each class position is a difficult task, in this article we propose a method based on a genetic algorithm for computing the weights.

Starting from the test dataset [3] we have created a proper test file which is used for the genetic algorithm. In this new file, for each document entry we have saved all outputs from all classifiers separately. Thus, for a document we obtained as output 9 vectors (because we have 9 classifiers), each vector having a number of 16 scalars descending ordered (because we have 16 classes). In the genetic algorithm used one chromosome represents the weights value for each class separately, depending by the class position. The first value from the vector represents the weight for the first position class from the classifier; the second value from the vector represents the weight for the class from the next position, and so on.

$$c = (w_1, w_2, ..., w_j), \text{ with } j = \overline{1,16} \quad (7)$$

where w_j represents the value of weight for the j^{th} class position. We have fulfilled the following inequalities: $w_1 > w_2 > \ldots w_{16}$. For computing the fitness function the following steps are taken:

1. For each document from the testing set the algorithm obtains the output vector corresponding to each classifier;
2. Each output vector is weighted with the corresponding chromosome value;
3. The winner class is obtained by summing all results for each class and selecting the highest score (formula 8);

$$Class_j = \sum_{i=1}^{9} w_j c_{ij}, \text{ for } j = \overline{1,16} \text{ and}$$
$$WinClass = \arg\max_{j=\overline{1,16}} (Class_j) \quad (8)$$

where w_j represents the value of the chromosome for the j^{th} class position (called weight), c_{ij} represents the value computed by the i^{th} classifier for the j^{th} class position (called confidence degree) and $Class_j$ represents the value computed for the class from the j^{th} position. Each classifier can return a different order of the classes. For a classifier the class returned at j^{th} position is multiplied with the value w_j.

The winner class (*WinClass*) will be compared with the real class proposed by Reuters and if the classes are identically we consider that the document was correctly classified;

After processing all documents the fitness value is the accuracy obtained on the whole testing set.

The best chromosome is that with the highest fitness. For building a population we have used 100 chromosomes and the algorithm was applied for 1000 generations.

As selection operator for choosing the chromosome from the current population to be used for the next population we have used two methods: Roulette method and Gaussian method. For the Gaussian method we have used the following formula:

$$Gauss(c) = e^{-\frac{1}{2}\frac{(fitness(c)-m)^2}{2\sigma^2}} \quad (9)$$

where c is the current chromosome, *fitness(c)* is the result obtained for the corresponding chromosome c, m represents the average and in our case is equal to 1 and σ is the standard deviation (here we have used a value equal with 0.5).

For creating the new population we have used all 3 genetic operators as it follows: 30% from the new population is created using the *selection* operator (selecting the best chromosome form the current population and copy them into the new population - elitism); the rest of the population is created by using the *mutation* operator in 40% and respectively *crossover* operator in 30%. After applying this operators the condition $w_1>w_2>\ldots w_{16}$ needs to be respected. Thus, for crossover operator we search after a cut point (i) for which the condition $w_i^{first-parent} > w_{i+1}^{second\_parent}$ is valid and there are created two new children each of them with a part from one parent and a part from the other parent. For mutation operator, after selecting randomly the mutation point (i), the new value for the child is selected randomly in the interval $w_i \in (w_{i-1}, w_{i+1})$.

5. Experimental Frameworks

The Dataset

Our experiments were performed on the Reuters-2000 collection [13], which has 984 Mb of newspapers articles in a compressed format. The collection includes a total of 806,791 documents, with news stories published by Reuters Press covering the period from 20.07.1996 through 19.07.1997. The articles have 9822391 paragraphs and contain 11522874 sentences and 310033 distinct root words. Documents are pre-classified according to 3 categories: by the Region (366 regions) the article refers to, by Industry Codes (870 industry codes) and by Topics proposed by Reuters (126 topics, 23 of them contain no articles). Due to the huge dimensionality of the database we will present here results obtained using a subset of data. From all documents we have selected the documents for which the industry code value is equal to "System software". We obtained 7083 files that are represented using 19038 features and 68 topics. We have represented a document as a vector of words, after applying a stop-word filter (from a standard set of 510 stop-words) and extracting the word stem [2]. From these 68 topics we have eliminated those topics that are poorly or excessively represented. Thus we have eliminated those topics that contain less than 1% documents from all 7083 documents in the entire set. We have also eliminated topics that contain more than 99% samples from the entire set, as being excessively represented. After doing so we have obtained 24 different topics and 7053 documents, that were split randomly in a training set (4702 samples) and a testing set (2351 samples). In the feature extraction phase we take into consideration both the article and the title of the article. In the feature selection phase we have selected only 1306 features for each vector.

6. Experimental Results

In [3] were presented first time the results obtained using static (pre-determined) weights values. The best classification accuracy obtained was 87.20% (i.e. 301 incorrectly classified documents from a number of 2351 documents). These results were obtained with a model of meta-classifier called "M-05W" that has weighting values that linearly decrease from 12 with a step of 0.5.

In the meta-classifier with genetic algorithm, for each chromosome we compute the fitness function using the training dataset. After selecting the best chromosome in this manner, we compute the fitness function obtained only by this chromosome using the testing dataset and in the figures only the values computed on the testing dataset are presented. Therefore, sometimes, the evolution of the best chromosome in the presented figures is not always ascending.

Because we start with a randomly generated population, and the obtained results depend of the starting points, we make four successively runs for the same method. Here we present the outputs for each method and also the average value.

In Figure 3 we present the classification accuracy obtained on the testing dataset, when using the genetic algorithm with Gaussian method for selecting the chromosomes.

The average accuracy obtained is 88.37%. Thus we obtained an average growth of 1.17% for the classification accuracy comparing with the case of the static values of weights presented in [3].

In case of using genetic algorithm with the Roulette method for selecting the chromosomes in the new population we have obtained an average classification accuracy of 88.36%, with only 1.16% greater the in M-SUM method. The results are presented in Figure 4.

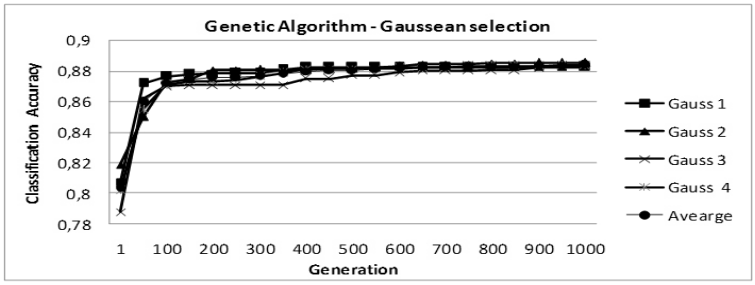

Figure 3. Classification accuracy - Gaussian method for selecting the best chromosomes for new generation

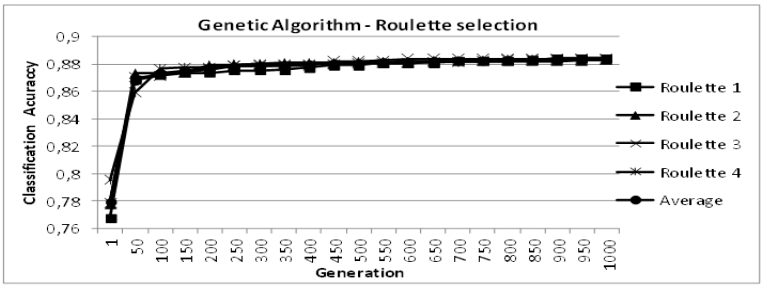

Figure 4. Classification accuracy with Roulette method for selecting the best chromosome for new generation

The evolution for the average accuracy obtained in 1000 generation using Gaussian and Roulette method for selecting chromosomes is presented in Table 1.

Table 1. Average classification accuracy for Gaussian and Roulette selection method

Generation	Average Gaussian selection	Average Roulette selection
1	0.803701	0.779561
50	0.859634	0.867822
100	0.872288	0.873458
150	0.874202	0.874840
200	0.875691	0.876648
250	0.87601	0.878243
300	0.876648	0.878562
350	0.877818	0.878881
400	0.879945	0.879413
450	0.880051	0.880582
500	0.880689	0.880795
550	0.880902	0.881327
600	0.881646	0.881752
650	0.882284	0.882177
700	0.882284	0.882496
750	0.882497	0.882603
800	0.882816	0.882922
850	0.882816	0.883134
900	0.883348	0.883453
950	0.883667	0.883560
1000	0.883773	0.883666

The best accuracy obtained by us in all four performed tests was obtained using Gaussian method for chromosome selection where the maximum accuracy value was 88.55%.

7. Conclusions and Further Work

In this article we have presented a non-adaptive meta-classifier used for classifying text documents. This meta-classifier uses the outputs of eight independent SVM classifiers and one Naive Bayes classifier. The output of each classifier is a vector of values, where each position represents the confidence given by the classifier that the current document belongs to the corresponding class. The meta-classifier developed in this article uses a genetic algorithm for calculating the best values that could be used to weight the outputs of each classifier so that the final classification accuracy to be improved.

Since we have started for the genetic algorithm with random initial weights we conducted four separate experiments. Comparing the average results obtained using a genetic algorithm with the results obtained when the classifiers outputs were weighted with pre-defined values [3] we have obtained an improvement of 1.17%. As classification accuracy, the best value obtained is 88.55% in case of using Gaussian method for selection of chromosomes for the new population.

In the future, an interesting natural extension of our work may be an adaptive and intelligent meta-classifier that uses a neural network for choosing the classifier that will be used in classifying the current document.

Acknowledgment

This work was partially supported by CNCSIS-UEFISCSU, project number PN II-RU code PD_670/2010.

REFERENCES

1. ANDREI, N., **On Quadratic Internal Model Principle in Mathematical Programming**, Studies in Informatics and Control, Vol. 18, No. 4, 2009, pp. 337-348.
2. CHAKRABARTI S., **Mining the Web-Discovering Knowledge from Hypertext Data**, Morgan Kaufmann Press, 2003.
3. CRETULESCU, R., D. MORARIU, L. VINTAN, **Eurovision-like weighted Non-Adaptive Meta-classifier for Text Documents**, The 8th RoEduNet International Conference, Galati, Romania, 2009.
4. CHEN, Q., D. ZHENG, T. ZHAO, S. LI, **A Fusion of Multiple Classifiers Approach Based on Reliability Function for Text Categorization**, 5th International Conference on Fuzzy Systems and Knowledge Discovery, IEEE, 2008.
5. DIMITROVA, N., L. AGNIHOTRI, G. WEI, **Video Classification Based on HMM Using Text and Face**, Proceedings of the European Conference on Signal Processing, Finland, 2000.
6. LEWIS, D., **Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval**. In Proceedings of the 10th European Conference on Machine Learning, 1998.
7. LIN, W.-H., A. HAUPTMANN, **News Video Classification Using SVM-based Multimodal Classifier and Combination Strategies**, In Proceedings of the Tenth ACM international Conference on Multimedia, 2002.
8. LIN, W.-H., R. JIN, A. HAUPTMANN, A **Meta-classification of Multimedia Classifiers,** International Workshop on Knowledge Discovery in Multimedia and Complex Data, Taiwan, 2002.
9. MORARIU, D., L. VINTAN, V. TRESP, **Meta-Classification using SVM Classifiers for Text Documents**, The 3rd International Conference on Neural Computing and Patter Recognition, Barcelona, October 2006.
10. MORARIU, D., **Text Mining Methods based on Support Vector Machine**, MatrixRom, Bucharest, 2008.
11. MORARIU, D., R. CRETULESCU, L. VINTAN, **Improving a SVM Meta-classifier for Text Documents by using Naive Bayes**, International Journal of Computers, Communications & Control, Vol. V, No. 3, 2010, pp. 351-361, ISSN 1841-9836, E-ISSN 1841-9844.
12. NELLO, C., J. SWAWE-TAYLOR, **An introduction to Support Vector Machines**, Cambridge University Press, 2000.
13. Reuters Corpus: http://about.reuters.com/ researchandstandards/ corpus/. Released in November 2000.
14. SCHOELKOPF, B., A. SMOLA, **Learning with Kernels. Support Vector Machines**, MIT Press, London, 2002.
15. SIYANG, G., L. QUINGRUI, M. LIN, **Meta-classifier in Text Classification**, http://www. comp.nus.edu.sg/~zhouyong/papers/cs5228project.
16. SUDUC, A. M., L. DUTA, G. GORGHIU, **Interface Architecture for a web-Based Group Decision Support System**, Studies in Informatics and Control, Vol. 18, no. 3, 2009, pp. 241-246.
17. GHOSH, A., L. C. JAIN, **Evolutionary Computation in Data Mining**, Springer Verlag Berlin Heidelberg, 2005.