Improving Multi-class Text Classification Performance by Combining Support Vector Machine and Random Forest Classifiers

Beom-Suk Chung, Kwanho Kim, Sungmin Lim, Jonghun Park

Information Management Lab., Department of Industrial Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul, 151-744, Republic of Korea {bumdo103, goalwisk, hovern, jonghun}@snu.ac.kr

Abstract. Multi-class classification is considered as an essential technique in various applications. While solving a website multi-class classification problem with one-versus-the-rest Support Vector Machine (SVM) classifier, we found a set of interesting instances, named Overly Confident Instances (OCIs), which exhibit overly confident classification results among multiple binary classifiers compared to normal instances. The OCIs can be attributed to noisy instances or errors of one-versus-the-rest SVM. We propose a novel approach that detects OCIs and leaves them unlabeled instead of labeling incorrectly. We employ a random forest model to learn specific patterns of OCIs from the results of SVM classifiers. The experiments on a real-world dataset show that the performance of multi-class classification was improved using the proposed approach.

Keywords: Support Vector Machine, Random Forest, Website Classification, Multi-class Classification

1 Introduction

The goal of multi-class text classification is to determine an appropriate class label for an unlabeled instance among the classes defined in advance. It is considered as an essential technique in various applications such as web page classification and spam filtering [1, 2, 3].

Support Vector Machine (SVM) [4], is widely used in classification problem. However, SVM cannot be directly used in solving a multi-class classification problem since it is essentially a binary classifier. There are several multi-class decomposition techniques available for SVM including one-versus-the-rest, pairwise comparison and error-correcting output coding.

We consider a one-versus-the-rest SVM classifier to solve a multi-class website classification problem. While solving the classification problem, we found an interesting set of instances, named Overly Confident Instances (OCIs), which refer to the instances classified incorrectly but have specific patterns. As the name implies, those instances show overly confident classification results compared to those of normal instances.

Table 1 shows example results of multi-class classification through one-versus-the-rest SVM classifiers. Each column shows the scores, each of which indicates how likely an instance belongs to the class. Instance 1, which is not an OCI, is assigned to class 3 according to the scores for each class, and its actual class is class 3. In contrast to the instance 1, instance 2 shows a different pattern. Instance 2 is classified as class 3 according to the abnormally high score for class 3, but in fact it belongs to class 2 which shows the second best score among the scores. We call the instance, which shows a pattern that the best score is unduly high compared to the best scores of normal instances and the best score is not associated with its actual class but on other class, as OCIs.

Table 1. Scores for classes when one-versus-the-rest SVM was applied. (*: actual class of an instance.)

	class 1	class 2	class 3	class 4
instance 1	-1.4	1.2	5.4*	0.3
instance 2	0.1	0.3*	20.12	-3.21

The occurrences of OCIs can be attributed to two reasons. First, it may be a result of the errors of one-versus-the-rest SVM classifiers. Figure 1 shows such results. In figure **1,** hyperplane A separates the triangles from the others and hyperplane B separates the squares from the others. Instances at the right side of hyperplane A are classified to the class of triangles while instances at the upper side of hyperplane B are classified to the class of squares. The OCIs can be observed in the dotted area which corresponds to the right side of hyperplane A and the upper side of hyperplane B.

When the i-th test instance, d_i, is given, which is unlabeled, this will be assigned to the class of squares if we consider the hyperplane B only. However, one-versus-the-rest SVM classifiers will assign d_i to the class of triangles because the distance between d_i and the hyperplane A is greater than that between d_i and the hyperplane B.

Another possibility is the existence of noisy instances in a dataset. Noisy instances, which are labeled incorrectly, can be frequently found in a real-world dataset. These noisy instances make it difficult to learn a classifier precisely and lead to decreasing the performance of the classifier.

In this paper, we take an approach that detects OCIs and leaves them unlabeled instead of labeling incorrectly. This approach may have advantages in many applications, since the false positive errors incurred by OCIs will degrade the quality of services. For instance, a falsely classified spam email, which is not a spam email, may lead to degrading the quality of an email service.

We propose a two-phased approach that aims to minimize false positive errors caused by OCIs through not labeling them from results of SVM classifiers. Specifically, at the first phase, one-versus-the-rest SVM classifiers produce the score for each class as illustrated in table 1. At the second phase, we employ a Random Forest (RF) [5] model to learn specific patterns of OCIs from the scores of SVM classifiers. During the test, the SVM classifiers compute scores of a new instance for each class, and then RF classifier decides whether the instance is an OCI or not. The

instances decided as OCIs, are not labeled and the others are labeled as the SVM classifiers predicted.

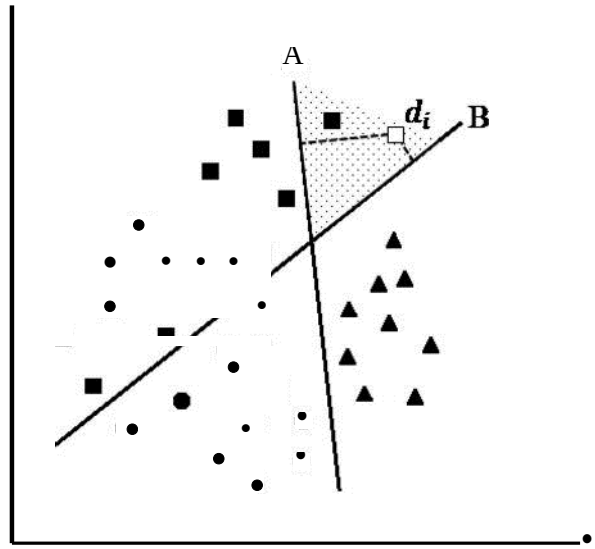

Fig. 1. Example of an OCI. OCIs can be observed in dotted area where a conflict occurs between two binary classifiers.

2 Proposed Approach

We introduce the notations for describing a multi-class text classification problem. The i-th instance, d_i, is represented by an L-dimensional vector of independent terms, $d_i = < t_{il}, \quad >$ where t_{it} is 1 if the /-th term appears in d_i, and zero, otherwise. We denote the j-th binary SVM classifier as m_i which separates the instances of j-th class from the others and the set of SVM scores of d_i for J classes as $S_i = \quad j = 1, \ldots, J\}$ where s_u is a computed score for d_i by m_i.

A multi-class classifier is denoted as M, which decide the class for the instance based on the set of scores, S_i. We denote the predicted class label for d_i by M as c_i. Finally, a binary variable for d_i, which indicates whether c_i is correct or not, is denoted as b_i where b_i is 1 if c_i is a correct class for d_i, and zero, otherwise.

We employ one-versus-the-rest SVM classifiers as a multi-class classifier for the first step. Each binary classifier, $= 1, \ldots, J$, is learned by using training instances, each of which is represented as vector d_i. Then the trained binary classifiers produce the set of SVM scores S_i for each training instance d_i. Finally, M predicts a class label for each instance.

At the second step, we employ an RF model as the second classifier which aims to predict whether the predicted instances are OCIs or not. The set of SVM scores, S_i, for each training instance is used as features for input of the second classifier. The values of b_i, for each training instance is used as a class label for input of the second

classifier. Subsequently, the second classifier is trained to detect the OCIs, which are a subset of the incorrectly classified instances, through learning specific patterns of OCIs from *S*.

To predict a class label for a test instance d_i, S_i for d_i is computed from binary classifiers m_i, and c_i is produced from M. Then S_i and c_i are handed over to the second classifier, RF. Next, the second classifier predicts whether the instance is an OCI or not by evaluating S_i. The OCIs that are detected by the second classifier are not labeled while the rests are assigned to their original class, c_i. In this way, we expect that the number of labeled instances decrease since OCIs detected by the second classifier are not labeled. However, the number of incorrectly labeled instances will also decrease, leading to improvements of the performance of the classifier.

Furthermore, a cost matrix, which associates different costs with a false positive error and a false negative error, is employed in the learning process of the second classifier. The second classifier is trained to minimize the false positive errors by setting the cost matrix to have a higher cost with a false positive error than a false negative error. By controlling the cost matrix, the number of false positive errors can be decreased as needed, depending on how a service is sensitive to false positive errors.

3 Experiment results

To evaluate our proposed approach, we conducted a website multi-class classification task. We used a real-world dataset gathered from Somansa (http://www.somansa.com), which is one of the biggest security companies in Korea. About 80,000 websites were gathered, and the number of classes was 39. We used 90% of the dataset as training data, and the rest as test data.

We considered each website as an instance. To represent each instance as a vector, we gathered the terms from plain text, title, meta keywords, and meta description of each website. The plain text was obtained by removing HTML tags from the front page of each website. The title of each website was extracted, which is enclosed by the <title> and </title> tags. Meta keywords and meta description for a website were extracted from <meta name="keywords" content="..."> tag and <meta name="description" content="..."> tag, respectively.

To compare the effectiveness of combining second classifier with SVM, we compared the performances of SVM alone and SVM + RF which employed RF as a second classifier. And SVM + RF + cost matrix, which applied a cost matrix to the second classifier, was also compared. To compare the performances of RF as a second classifier, we also employed the partial decision trees based classifier (PART) [6] as a second classifier. We compared each method in terms of accuracy and coverage which are defined as follows.

$$\text{Accuracy} = \frac{\text{the number of correctly labeled instances}}{\text{the number of labeled instances}}$$

$$\text{Coverage} = \frac{\text{the number of labeled instances}}{\text{the number of instances to be labeled}}$$

Table 2 shows the accuracy and the coverage of each method.

Table 2. The accuracy and the coverage of the methods considered.

Method	Coverage	Accuracy
SVM	100%	69.29%
SVM + RF	90.58%	72.8%
SVM + RF + cost matrix	42.72%	90.1%
SVM + PART	97.99%	70.12%
SVM + PART + cost matrix	40.97%	90.09%

As expected, the accuracy obtained by combining RF and SVM classifier was improved compared to the accuracy of SVM alone. Furthermore, employing a cost matrix for RF has decreased the coverage while successfully improving the performances in terms of accuracy. Employing PART as the second classifier improved the accuracy, however, not as much as employing RF as the second classifier.

4 Conclusion

The proposed approach successfully enhanced the performance of multi-class classification by not labeling OCIs instead of labeling them incorrectly. The experiment results based on a real-world dataset showed that the proposed approach was able to reduce false positive errors. Although the proposed approach resulted in decreased coverage, it achieved significantly improved accuracy results and was helpful for the website classification services which are sensitive to false positive errors. We expect that the proposed approach can be also applied to Spam filtering or genre classification problems.

Acknowledgement

This research was supported by the National Research Foundation of Korea (NRF) grants funded by the Korea Government (MEST) (Nos. 2011-0004423 and 2011-0030814), and partly by Engineering Research Institute at Seoul National University and Somansa.

References

1. Sun, A., Lim, E.P., Ng, W.K.: Web Classification Using Support Vector Machine. In: 4th International Workshop on Web Information and Data Management, pp. 96--99. ACM, New York (2002)
2. Hu, W.C., Chang, K.H., Ritter, G.X.: Web Document Classification Using Modified Decision Trees. In: 38th Annual on Southeast Regional Conference, pp. 262--263. ACM, New York (2000)
3. Hidalgo, J.M.G., Bringas, G.C., Sanz, E.P., Garcia, F.C.: Content Based SMS Spam Filtering. In: 2006 ACM Symposium on Document Engineering, pp. 107--114. ACM, New York (2006)
4. Vapnik, V.: The Nature of Statistical Learning Theory. Springer-Verlag, New York (1995)
5. Briman, L.: Random Forests, Mach. Learn. 45, 5--25 (2001)
6. Frank, K. and Witten, I.H.: Generating Accurate Rule Sets without Global Optimization. In: 15th International Conference on Machine Learning, pp. 144--151. Morgan Kaufmann, San Francisco (1998)