Multi-Label Text Classification

Jasneet Singh Sabharwal
301197190
Department of Computer Science
Simon Fraser University
8888 University Drive, Burnaby
BC, Canada V5A 1S6
jsabharw@sfu.ca

Abstract

In a standard machine learning classification problem, the classes are mutually exclusive. But, when they are overlapping then classification errors occur. The method proposed in this paper tackles such a problem in which the classes are not mutually exclusive. Such a problem is called multi-label classification and in this an instance can be classified into multiple classes. This paper presents a Support Vector Machine (SVM) based method to handle multi-label problems. This method is evaluated on a subset of Large Scale Hierarchical Text Classification Challenge 4 dataset and shows promising results.

1 Introduction

Standard machine learning classification problems are concerned with learning and classifying documents/examples into single classes from a set of mutually exclusive classes (Figure 1a). Classification error occurs when the classes are not mutually exclusive, that is when they overlap. Many classification algorithms to handle problem which are mutually exclusive have been developed, like Support Vector Machines, k-Nearest Neighbours, Logistic Regression, Decision Trees, Neural Networks, etc.

When the set of classes contains only two classes, then it is a *binary classification* problem and when there are more than two classes in the set, then it is called a multi-class problem (Figure 1b). But, both these problems only assign one class to an example. However there few problems in which the examples can belong to multiple classes, such a problem is called multi-label classification problem. In this the set of classes contains two or more than two classes and the examples can be assigned more than one class. Identifying multiple objects in an image is an example of this problem. Another problem occurring in the field of text classification is that of assigning categories to text. A document of text can belong to various categories, for example - Article on *Emperor Ping of Han* in Wikipedia [1] can categorized into *Western Han Dynasty emperors, 9 BC births, 1st-century BC monarchs, deaths by poisoning, 1st-century Chinese monarchs, etc.*

This paper focuses on a similar problem of text classification/categorization. Wikipedia is a large resource of text articles falling in thousands of categories and it was only natural to use such a resource to present a method on multi-label classification. Lot of work has been done in the field of binary/multi-class classification in order to improve their performance. So, is it possible to tackle the problem of multi-label classification using such algorithm? This paper presents an approach on how the problem can be transformed so that standard algorithms like Support Vector Machines can be utilized.

[1] Wikipedia article on Emperor Ping of Han - http:en.wikipedia.orgwikı/Emperor_Ping_of_Han

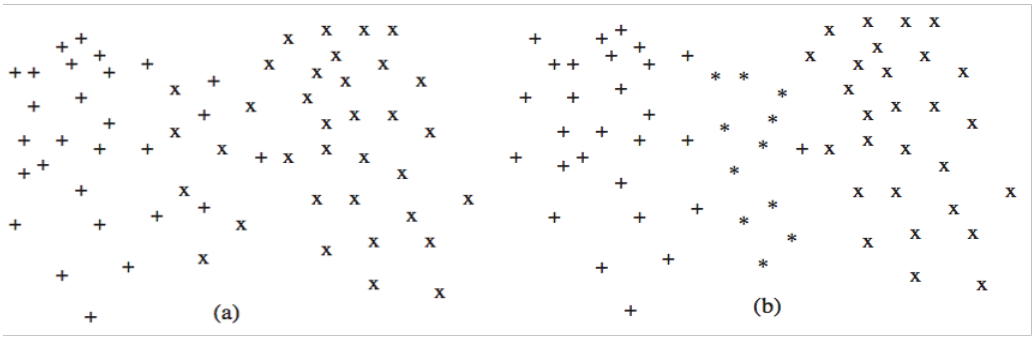

Figure 1: *(a) A standard classification problem in which classes are mutually exclusive. Though the two classes denoted by + and x contain non-separable data, but this can be handled by certain algorithm. (b)A multi-label problem, the * data belongs to both the classes (+,x) at the same time*

The structure of the paper is as follows: Section 2 gives an overview about the dataset used in this paper to evaluate the algorithm. Then the algorithm and experiments along with methods of evaluating the performance of the algorithm are presented in Section 3. The results of the algorithm on the dataset are given in Section4 and the paper then concludes with a discussion and suggestions for future work.

2 Dataset

The Large Scale Hierarchical Text Classification Challenge 4 dataset [5] used for this project is based on the data in Wikipedia. The datasets are multi-class and multi-label i.e each document can be classified into multiple classes at the same time. The original dataset consists of roughly 325,000 classes and 2,365,436 records. The dataset provided had been pre-processed by performing stemming of each word and removing stop-words. The records were represented in the form of term-frequencies. In the end, there were 2,085,161 features per document which are extremely sparse.

Since the dataset that is available is extremely large, it then reduced for the scope of the project. The documents which contain only 2 labels are selected. Also, each label and combination of labels should contain at least 100 documents in the selected dataset. This process brings down the dataset to contain 25,759 documents and a total of 184 labels.

The dataset is then split into two sets using. A training data set and testing data set containing 80% (20,563) and 20% (5,196) documents respectively. Both the sets containing the same ratio of data for each label and for each combination of labels. Also, the records were split randomly to create the sets.

3 Approach

This section first presents the algorithm and how the experiments were conducted. Then the methods for evaluating the performance of the algorithm are presented.

3.1 Algorithm and Experiments

In order to tackle the multi-label classification problem, the *Problem Transformation Approach*, which is based on the Cross-Training approach proposed by Boutell et. al., 2004 [1] is utilized. The idea behind this approach is to transform the original multi-label classification problem into multiple

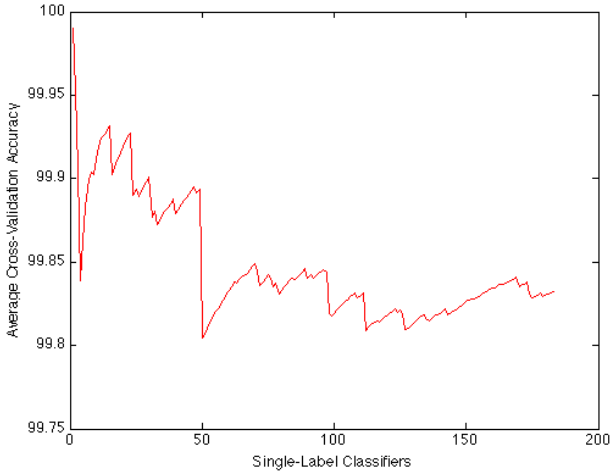

Figure 2: *Average cross-validation accuracy for all single-label classifiers. This plot is generated by calculating cross-validation accuracy for all the tried C parameters and then taking the average for each of the classifiers.*

single-label classification problems. This approach has made the problem an algorithm independent approach and due to this any classification algorithm can now be used to tackle the problem. *Support Vector Machines* and specifically LibSVM by Chang and Lin, 2011 [2] is used for this work..

In this work, single-classifiers for each label were created, i.e classifiers which classify whether a document belongs to that class or not (One-vs-All method) were created. For this, 184 binary classifiers were created for each of the labels. To create the data for each of these classifiers, multiple labels for each record were stripped and two copies of the record were created and were added to the training dataset of both the labels. Also, when adding to the training set, the labels were set as 1. All the other documents which did not belong to either of the 2 labels were added also to the training set with label -1.

For each of the 184 single-label classifiers, Support Vector Machines with Linear Kernels were used and also grid search for selecting the best *C* parameter was done for each of the classifier. Parameter selection was performed by doing 5-fold cross-validation runs. *C* value was chosen separately for each of the 184 classifiers and then each of the classifiers was trained using the *C* value which gave the best performance. Figure 2 shows the average cross-validation accuracy for all the single-label classifiers.

Once the models were created, we ran tests on test data which was separated earlier and which did not take part during any of the training process. Each test document was passed through each of the 184 classifiers and the classifiers which gave positive prediction were recorded and they acted as the predicted labels for that record.

3.2 Evaluation Measures

The standard classification algorithm evaluation methods like accuracy, precision, recall and F1-measure would not be sufficient to understand the performance of multi-label problem. To understand the performance of a multi-label problem, multiple evaluation measures are presented:

3.2.1 Average Accuracy

To calculate the *Average Accuracy*, first the accuracy of each individual classifier is calculated and then the average of all the accuracies are taken. It is given as follows:

$$Average\ Accuracy = \frac{1}{N}\sum_{i=1}^{N}\frac{|T_i| - |Y_i|}{|T_i|} \tag{1}$$

In the above equation $|T_i|$ is the number of true labels for classifier i, $|Y_i|$ is the number of predicted labels for classifier i and N is the number of classifiers.

3.2.2 Average Precision

To calculate the *Average Precision*, first the precision of each each individual classifier is calculated and then average of all precisions is taken. It is given as follows:

$$Average\ Precision = \frac{1}{N}\sum_{i=1}^{N}\frac{tp_i}{tp_i + fp_i} \tag{2}$$

In the above equation tp_i is the number of true positives for classifier i, fp_i is the number of false positives for classifier i.

3.2.3 Average Recall

To calculate the *Average Recall*, first the recall of each each individual classifier is calculated and then average of all recalls is taken. It is given as follows:

$$Average\ Recall = \frac{1}{N}\sum_{i=1}^{N}\frac{tp_i}{tp_i + fn_i} \tag{3}$$

In the above equation fn_i is the number of false negatives for classifier i.

3.2.4 Average F-Measure

To calculate the *Average F-Measure*, first the *F1 Scores* for each individual classifier are calculated and then all the *F1 Scores* are divided by the number of classifiers. By definition, F-Measure is a harmonic mean of Precision and Recall. The *Average F-Measure* is given as follows:

$$Average\ F - Measure = \frac{1}{N}\sum_{i=1}^{N}\frac{Precision_i * Recall_i}{Precision_i + Recall_i} \tag{4}$$

Where $Precision_i$ is precision for classifier i and $Recall_i$ is recall for classifier i and N is the number of classifiers.

3.2.5 Hamming Loss

Schapire and Singer, 2000 [4] proposed using *Hamming Loss* for evaluating the performance of multi-label classification. *Hamming Loss* is a loss function which calculates the percentage of miss-classified labels to the total number of labels and is given by the formula:

$$Hamming\ Loss = \frac{1}{K}\sum_{i=1}^{K}\frac{Predicted_i\ \Delta\ True_i}{L} \tag{5}$$

Where K is the number of examples/instances, L is the number of classes, Δ is the XOR operation or difference between the two sets $Predicted_i$ and $True_i$ where $Predicted_i$ is the prediction for instance i and $True_i$ is the true/target value for instance i. Since it is a loss function, smaller the value, better would be the performance of the algorithm.

3.2.6 Subset Accuracy

Subset Accuracy [3] is a strict accuracy measurement which consider a classification as correct if and only if all the labels have been correctly classified. It is given as follows:

$$Subset\ Accuracy = \frac{1}{K}\sum_{i=1}^{K} I(Predicted_i\ =\ True_i) \tag{6}$$

Where, *I(true) = 1* when both the predicted labels match with the true labels and *I(false) = 0* when any or all of the predicted labels do not match with the true labels.

Table 1: *Results*

Evaluation Metric	Single-Labelled Classifiers
Average Accuracy	99.8519%
Average Precision	0.9162
Average Recall	0.8982
Average F-Measure	0.9012
Hamming Loss	0.1481%
Subset Accuracy	81.1778%

4 Results

Table 1 shows Average Accuracy, Average Precision, Average Recall, Average F-Measure, Hamming Loss and Subset Accuracy calculated on testing data set. From the table, it is evident that the average accuracy is extremely high, but it doesn't denote that the performance of the algorithm is that high. The reason for exceptionally high average accuracy is because the data (both testing and training) for each classifier were skewed. On an average for each classifier training there were approximately 100+ records as positive and rest 20,000 as negative. Even the testing data was similar. That is why the f-measure scores were calculated for the positive class.

5 Conclusion

This paper presented an approach in which the problem is transformed to smaller single-label classification problems to handle multi-label classification problem. This approach performed well with low Hamming Loss and Average F-Measure of 0.90. Along with this approach, various evaluation methods for multi-label classification were also presented.

Remarks

Another approach of solving multi-label problem called Algorithm Adaptation was also tried. The algorithms which were tried are Multi-Label k-Nearest Neighbours [6] and and Rank-SVM [7] but the results from these algorithm could not be formulated in time for the deadline due to the large data set and high dimensionality of data which required large computational resources.

References

[1] Boutell, Matthew R., Jiebo Luo, Xipeng Shen, and Christopher M. Brown. "Learning multi-label scene classification." *Pattern recognition* 37, no. 9 (2004): 1757-1771.

[2] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm

[3] Ghamrawi, Nadia, and Andrew McCallum. "Collective multi-label classification." In Proceedings of the *14th ACM international conference on Information and knowledge management*, pp. 195-200. ACM, 2005.

[4] Schapire, Robert E., and Yoram Singer. "BoosTexter: A boosting-based system for text categorization." *Machine learning* 39, no. 2-3 (2000): 135-168.

[5] Large Scale Hierarchical Text Classification Challenge (LSHTC4), Track 1: Very Large Scale Supervised Learning. Available at: http://lshtc.iit.demokritos.gr/

[6] Zhang, Min-Ling, and Zhi-Hua Zhou. "ML-KNN: A lazy learning approach to multi-label learning." *Pattern Recognition* 40, no. 7 (2007): 2038-2048.

[7] Elisseeff, Andr, and Jason Weston. "A kernel method for multi-labelled classification." In *Advances in neural information processing systems*, pp. 681-687. 2001.