International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org ||Volume 4 Issue 11|| November 2015 || PP.33-37

Text Classification using Support Vector Machine

Anurag Sarkar[1], Saptarshi Chatterjee[2], Writayan Das[3], Debabrata Datta[4]
[1]*(Department of Computer Science, St. Xavier's College (Kolkata), India)*
[2]*(Department of Computer Science, St. Xavier's College (Kolkata), India)*
[3]*(Department of Computer Science, St. Xavier's College (Kolkata), India)*
[4]*(Department of Computer Science, St. Xavier's College (Kolkata), India)*

ABSTRACT : *Text-based classification is a technique which may be used to identify different types of data from the applications' point of view. Different researches are going on to identify ways of finding out the classes of data from a set of input data. In the present paper, a text-based classifier has been implemented and this classifier model can be used to classify input text into one of two categories, as defined by the user. The classifier is first trained with an initial dataset using the principle of supervised learning. After the training process is complete, the classifier makes use of the trained data in order to classify any new input text that may be provided. The proposed model also offers an incremental approach to text classification as it dynamically trains the classifier from a new set of data provided by the users.*

KEYWORDS - *Data Mining, Supervised Learning, Text Classification, Text Mining*

I. INTRODUCTION

Text classification is the process of classifying information in text format by its content, i.e., by the messages that may be conveyed by the words contained within it. Automating this process is crucial in order to be able to classify a large amount of text-based information in a time-critical manner. Due to the vast quantity of textual information that needs to be processed, automated text classification finds widespread application in a variety of domains such as text retrieval, summarization, information extraction and question answering, among others.

Text classification, which is also referred to as text categorization, falls under the broader domain of text mining, which is the general term for the process of deriving any information from a given text using a variety of text processing techniques. It usually involves processing the input text to make it suitable for mining with respect to the particular problem, finding patterns in the data and then evaluating the produced output. Aside from text classification, text mining also typically involves document summarization, sentiment analysis, text clustering and web mining.

Problems such as text classification, in which a labeled dataset is provided to the classifier for training purposes, are said to perform supervised learning. Supervised learning may be defined as the task of deducing information from data that has been labeled. In this approach, each data item in the dataset consists of a pair, in which the first item consists of the input data, and the second item is the associated label or desired output value for that input. The dataset consisting of these data pairs is then analyzed and a function is inferred that now allows the algorithm implementing supervised learning to produce output values or labels for new, unlabeled input data. This concept of supervised learning is in contrast to that of unsupervised learning in which the dataset used for training is not labeled.

In this research paper, the text classifier that has been implemented has utilized the idea of supervised learning in order to classify unseen data instances. Moreover, an incremental approach has been taken to text mining wherein the unseen data that has been classified by the algorithm proposed in this paper has been then added to the original labeled dataset used for training, so that in the future, for the same problem, the user can make use of a labeled dataset that has been augmented with the outputs produced in the previous iteration, thus allowing for more comprehensive training and increasing the performance of the classifier each time it is used for the same problem.

The paper is structured as follows. Section 2 provides a brief overview of related work in the field of text classification and cites a few examples of text classifiers implemented using various known techniques. Section 3 contains a detailed description of the text classifier that has been proposed here. Section 4 includes a discussion on the time complexity of the classifier and lists the results obtained upon using the classifier on a dataset. Section 5 concludes the paper with a summary of the work and lists ways in which the classifier may be improved in the future.

II. RELATED WORK

The paper in [2] had defined text categorization as "the task of automatically sorting a set of documents into categories (or classes, or topics) from a predefined set" and had stated that it was in the domain of both information retrieval and machine learning. As such, several techniques derived from these domains have found application in implementing text classifiers. Joachims [3] had proposed the use of Support Vector Machines (SVMs) for text classification and had also demonstrated that SVMs could offer better performance for text classifiers as compared to other well known machine learning techniques such as Naïve-Bayes classifiers and k-NN classifiers. Cristianini [4] had made a detailed discussion on the working of SVMs. A popular class of text classifiers are the Naïve Bayes classifiers based on the Bayes' theorem. In this regard, Leung had defined Bayesian classifiers as statistical classifiers that can predict the probability of a particular sample belonging to a particular class [5]. A variant of Naïve Bayes called Multinomial Naïve Bayes (MNB) is also often used in solving text categorization problems as evidenced in the work by Frank and Bouckaert [6] who had proposed a method that improves the efficiency of MNB classifiers used for text categorization by improving the performance of MNB in the context of unbalanced datasets. Dai, Xue, Yang and Yu [7] had shown an alternative way of Naïve-Bayes text classifier in which they had addressed the issue of classifying documents across different distributions. The k Nearest Neighbors (k-NN) class of classifiers also finds wide use in text classification problems. In the k-NN algorithm, for each data instance to be classified, its kth nearest neighbors are determined to form the neighborhood of the instance. Then, majority voting within the neighborhood is used to determine the class for the instance to be classified. In [8], Guo, Wang, Bell, Bi and Greer had developed a new text classification process which combines the strengths of the k-NN classifier and another type of classifier called the Rocchio classifier and had offered performance comparable to that of SVM-based text classifiers as discussed above. Similarly, Toker and Kirmemis [9] had projected a simple implementation of the k-NN method based text classifier which they use for developing a document organizing application. Li, Yu and Lu [10] had introduced a modified k-NN method for text classification which had used a suitable number of the nearest neighbors for prediction for different classes depending on the distribution of the class in the test documents.

III. PROPOSED METHOD

3.1 Methodology

The text classifier algorithm that has been implemented in the present research paper has utilized the concepts of supervised learning and incremental data mining. To perform the classification, two files have been taken as the inputs, the first containing the training data and the second containing the corresponding data labels. The data from these two files has been used to train the classifier. A crucial aspect of designing a practical text classifier is to reduce the dimensionality of features, which in this case are the different words in the training data. To facilitate this, two key preprocessing steps have been performed. At first, all the words in the data have been converted to the lower case since case sensitivity does not aid in the classification process and as the next step, all the stop words have been removed from the training data. Stop words are the commonly occurring words in the language that do not contribute any meaning that would benefit the text classification process. To make the text classification process efficient, a hash table has been used and the keys of the hash table are the words that appear in the training data and whose corresponding values constitute the information that will be used to classify the text. The text classification process consists of two main stages – training and classification.
The training process involves loading the aforementioned hash table with the required information. In the hash table, for each word (i.e. key) in the entire training dataset, two values have been stored – a category 1 value and a category 2 value. The category 1 value is a count of the number of times that the word appears in a data instance labeled as category 1. Similarly, the category 2 value is a count of the number of times that word appears in a data instance labeled as category 2. The hash table thus acts as a training matrix consisting of data used to train the classifier. Once the hash table has been loaded, the training process is complete. Now, the classification process begins which makes use of the hash table, containing the trained data, to classify new data instances provided by the user. The classification task is similar to the task of loading the hash table. The classifier parses each data instance into its individual words and computes the category 1 and category 2 values, as defined earlier, for each word. It then computes the category 1 and category 2 values of the data instance itself by summing the category 1 and category 2 values of its constituent words. If the category 1 value of the data instance is higher than the category 2 value, it is classified as a category 1 data instance and vice-versa. The

classifier then writes this newly classified data instance and its corresponding label to disk so that this information can be used in future iterations of the classifier for the same problem. This implements incremental data mining. The classifier then asks the user for the next input and the process continues until the user quits. The situation, in which the category 1 and category 2 values of a particular data instance are the same, indicates that the classifier has insufficient information to classify the data instance and cannot determine whether it belongs to category 1 or category 2. In this case, the classifier displays an appropriate message to the user and does not write the data instance to disk since an unclassified instance does not provide any useful information to the classification process.

3.2 Algorithms

The proposed method consists of three connected sections, viz., Initialize, Train and Classify. Corresponding to each section, the algorithm is stated below:

3.2.1 Algorithm: Initialize

```
BEGIN
    Read file containing the data instances for training
    Read file containing the corresponding labels
    Convert all data instances to lowercase
    Remove all punctuation marks and other non-alphanumeric characters from the data instance
    Split each data instance to its constituent words and remove stop words
END
```

3.2.2 Algorithm: Train

```
BEGIN
    LOOP through each data instance and label pair in training data
        FOREACH word w in the data instance
            IF w is not in WordList hash table THEN
                Add w to WordList
                IF the corresponding label is category 1 THEN
                    Set category 1 value of w to 1 and category 2 value to 0
                ELSE
                    Set category 1 value of w to 0 and category 2 value to 1
                END IF
            ELSE
                IF the corresponding label is category 1 THEN
                    Add 1 to the category 1 value of w
                ELSE
                    Add 1 to the category 2 value of w
                END IF
            END IF
        END FOR
    END LOOP
END
```

3.2.3 Algorithm: Classify

```
BEGIN
    LOOP while user wants to classify data
        Read data entered by the user
        Preprocess the data as in the training process
        Compute the sum of the category 1 and category 2 values of the words in the data instance
        Let these sums be sum1 and sum2 respectively
        IF sum1 > sum2 THEN
            Classify data instance as belonging to category 1 by assigning the appropriate label
        ELSE IF sum2 > sum1 THEN
            Classify data instance as belonging to category 2 by assigning the appropriate label
        END IF
        Add information about the newly classified data instance and its label to the hash table
        Write the newly classified data instance and its corresponding label to disk
    END LOOP
```

END

IV. ANALYSIS AND RESULTS

The runtime performance of the classifier is bounded by the time complexity of the training phase. To determine this complexity, let the number of data instances used for training the classifier be N (i.e. |Training Dataset| = N). Also, let the number of words in the largest data instance in the training dataset be M.

In order to train the classifier, each of the N data instances must be looped through in the training data set and for each data instance; each of its constituent words must be looped through. Thus, the time complexity of the entire training process is O(M×N), which is its performance in the worst case of training the classifier with the largest training instance. For each training instance, the data must be processed before training the classifier by removing the stop words. In order to do this, the constituent words in each data instance must be looped through, each of which must be compared with each of the stop words. Let the number of stop words that can be detected by the classifier be c. Then, the overall complexity of the stop word removal process for each of the N data instances may be given by O(c×M). However, c is a constant as it is fixed for the classifier, thus the complexity may be rewritten as c×O(M), which reduces to O(M).

The classification process depends on the number of instances that the user wants to classify, with each instance requiring a runtime of O(M), where M is the number of words in the instance. Since, the number of data instances for classification is usually much less than the number of instances in the training set, it may be concluded that the overall complexity of the classifier is dependent on the training phase.

To test the classifier, a dataset consisting of 500 book titles with 250 relating to computer science and 250 relating to biology or nature have been used. The instances in each set have been numbered from 1 to 250 and 5 rounds of testing have been performed, using 200 titles from each set to train and the remaining 50 from each to test. Thus, for each round, the training dataset consisted of a total of 400 titles and the testing dataset consisted of 100 titles. With these, the following results have been obtained:

Table I: Test Results

Training Dataset	**Testing Dataset**	**Prediction Accuracy**
Nos. 1-200	Nos. 201-250	89%
Nos. 1-150, 201-250	Nos. 151-200	80%
Nos. 1-100, 151-250	Nos. 101-150	83%
Nos. 1-50, 101-250	Nos. 51-100	81%
Nos. 51-250	Nos. 1-50	88%

Thus, satisfactory prediction accuracy has been obtained given the fact that a simple text classifier has been used without implementing any advanced techniques. It may further be concluded that higher prediction accuracy would have been obtained had a larger training dataset been used, such as one consisting of more than 500 book titles of each class. One drawback of incremental classification is that if a data instance is incorrectly classified, the accuracy of the classifier is lessened for future classifications. This problem is reduced as bigger datasets are used to train the classifier.

V. CONCLUSION AND FUTURE WORK

In this paper, a simple text classifier has been implemented and it supports incremental data mining. The classifier, as has been demonstrated, achieves a reasonable rate of accuracy even though it has been implemented using simple techniques as discussed previously. Also, since only two values for each data instance in the training set have been stored, no dimensionality reduction technique needs to be used. The support for incremental data mining, i.e., adding newly classified data instances to the training data set to train the classifier for future classifications, allows the classifier to achieve better classification results than would otherwise be possible.

Its advantages aside, in the future, the functionality of the classifier may be extended in the following ways:

- At present, the classifier can only classify instances into one of two classes. Future versions of the classifier may be extended to support classification into greater than two classes.
- Our classifier has a quadratic-time complexity in the worst case. Application of more complex machine

learning and information retrieval techniques may help in achieving a better runtime complexity.

- Higher prediction accuracy may be obtained by using more advanced weighting factors such as tf-idf instead of simple frequency of occurrence as have been employed.
- The classifier could also be improved by incorporating more advanced preprocessing techniques such as word stemming.

REFERENCES

[1] M. Ikonomakis, S. Kotsiantis, V. Tampakas, Text Classification Using Machine Learning Techniques, *WSEAS Transactions on Computers, Issue 8, Vol. 4,* August 2005, pp. 966-974.

[2] F. Sebastiani, Text Categorization, 2005: 683-687.

[3] T. Joachims, Text Categorization with Support Vector Machines: Learning with Many Relevant Features, *Technical Report 23,* Universitat Dortmund, LS VIII, 1997.

[4] N. Cristianini, Support Vector and Kernel Machines, *Tutorial at the 18th International Conference on Machine Learning*, June 28, 2001.

[5] K. Ming Leung, Naive Bayesian Classifier, *Polytechnic University Department of Computer Science/Finance and Risk Engineering,* 2007.

[6] E. Frank, and R. R. Bouckaert, Naive bayes for text classification with unbalanced classes, *Knowledge Discovery in Databases: PKDD* 2006, pp 503-510.

[7] W. Dai. et al., Transferring naive bayes classifiers for text classification, *Proceedings of the national conference on artificial intelligence, Vol. 22 No. 1,* Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[8] G. Guo et al., Using kNN model for automatic text categorization, *Soft Computing 10.5*, 2006: 423-430.

[9] G. Toker and O. Kirmemis, Text Categorization using k Nearest Neighbor Classification, *Survey Paper*, Middle East Technical University.

[10] Baoli Li, Shiwen Yu, and Qin Lu., An improved k-nearest neighbor algorithm for text categorization, *arXiv preprint cs/0306099*, 2003.

[11] D. D. Lewis, and W. A. Gale, A sequential algorithm for training text classifiers, *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, (Springer-Verlag New York, Inc., 1994).

[12] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, Learning to classify text from labeled and unlabeled documents, *AAAI/IAAI792*, 1998.

[13] P. Soucy, G. Mineau, Feature Selection Strategies for Text Categorization, *AI 2003, LNAI 2671*, 2003, pp. 505-509.

[14] A. Kehagias, V. Petridis, V. Kaburlasos, P. Fragkou, A Comparison of Word- and Sense-Based Text Categorization Using Several Classification Algorithms, *JIIS, Volume 21, Issue 3,* 2003, pp. 227-247.