LEARNING TO FILTER TEXT IN FORUM MESSAGE

Mohd Shamrie Sainin (shamrie@uum.edu.my)
Artificial Intelligence Special Interest Group (AISIG)
Department of Computer Science
Faculty of Information Technology, Universiti Utara Malaysia
06010 Sintok Kedah, Malaysia

Abstract: **Applying the basic filtering technique in forum application has been discussed in [1]. The paper explains about the use of the basic naïve Bayes algorithm to classify forum messages whether clean or bad where clean message has no bad words, while bad message contains one or more bad words. In this paper, the modification of the algorithm including pre-processing and classification will be discussed in the attempt to apply learning to filter forum messages.**

Keywords: Naïve Bayes, Filtering, Text Filtering, Online Forum

1.0 INTRODUCTION

The search from the internet about the text filtering and classification in forum message returns no exact application in this field. However, there are many sources that explain about learning element in text filtering especially in e-mail application. The examples of learning in text filtering using naïve Bayes are [2], [3], [4], and [5]. These papers discusses about naïve Bayes algorithm that is effectively construct a model of anti spam filtering.

Among these techniques, the current experiment of text filtering in forum application is using the standard naïve Bayes algorithm that adapted from [6]. The experiment includes algorithm for preprocessing, classifier model construction and finally forum message classifier. The next part of this paper explains about the basic naïve Bayes algorithm and experiment of the algorithm in forum text filtering.

2.0 NAÏVE BAYES

Bayesian learning method can be applied if manipulating probabilities are required. Naïve Bayes algorithm as a Bayesian learning is considered a simple approach and a competitive classifier among other method including neural network and in some cases outperforms the methods [7].

The learning task in naïve Bayes classifier includes building probability estimations from set of instance *x* described by conjunction of attribute values and some finite target function *f(x)*. Furthermore, naïve Bayes learner classification task is to predict the value for the new instance described by tuple of attributes *<a1, a2, …, an>*. Given with a set of target value *V*, Bayesian approach is to classify the new instance with the most probable target value, V_{MAP}. The value for V_{MAP} can be calculated using Equation 1.

$$\begin{aligned} V_{MAP} &= \max_{vj \in V} P(v_j \mid a_1, a_2 .... a_n) \\ &= \max_{vj \in V} \frac{P(a_1, a_2 .... a_n \mid v_j) P(v_j)}{P(a_1, a_2 .... a_n)} \\ &= \max_{vj \in V} P(a_1, a_2 .... a_n \mid v_j) P(v_j) \end{aligned} \quad (1)$$

Using (1), naïve Bayes classifier assumes the probability of observing a tuple of attributes conjunction *<a1, a2,…,an>* is a product of all individual attributes probability: $P(a_1, a_2,..,a_n|v_j) = \prod_I P(a_i|v_j)$.

Therefore naïve Bayes classifier approach will output the target value v_{NB} based on the maximum predicted value of $\prod_I P(a_i|v_j)$ as shown in Equation 2.

$$V_{NB} = \arg\max_{vj \in V} \prod_i P(a_i \mid v_j) \quad (2)$$

3.0 ANONYMOUS FORUM TEXT FILTERING USING NAÏVE BAYES

Artificial Intelligence Special Interest Group (AISIG) e-Community Portal is currently using an anonymous message submission in eJava Forum. The purpose of the the forum is to encourage student and visitors to ask Java related questions and in the same time response to the forum without registering their username and password. Based on this application, there was an idea to apply learning to classify the forum messages into two classes namely clean and bad.

A forum message database in eJava that was used for the experiment contains about 257 messages. From this total, 197 messages were pre-classified as ‘Clean’ and the remaining 60 are ‘Bad’ according to the content and words in the message. There is also

another 45 messages which were prepared for testing the performance of naïve Bayes algorithm in forum text filtering.

In the previous experiment using basic preprocessing and naïve Bayes algorithm, the result was 98.05 percent on training examples and 77.78 percent on test examples. There are two basic components which are preprocessing and classifier model that was developed to experiment the learning and filter the text in forum messages. The second experiment of the algorithms is the modified preprocessing algorithm to compare the performance of the application. The new preprocessing algorithm within the application is shown in Figure 1.

The preprocessing task is to clean the text messages from the forum database and can be further used to build the model for naïve Bayes learning. Preprocessing tasks include removing repeated string and collecting all words and punctuations. The new preprocessing algorithm was also applying a method to remove stop words and common words that normally used in English and Malay language for example "and", "the", "yang" and "itu".

The sorted and cleaned text from the preprocessing task will be passed through LEARN_BAYES_TEXT function to learn and calculate all probabilities. Probabilities that will be calculated are probability for each target value *v* given the number of document with target is *v* and probability for each word in the vocabulary. Figure 2 shows the related algorithm for naïve Bayes learning in this experiment.

4.0 RESULTS

Similar to the first experiment on the algorithms, naïve Bayes text classifier was trained using 257 messages and another 45 messages as a performance measure data set. In the training phase, there are total of 2153 words produced in the *Vocabulary* compared to 3143 words in the previous experiment. The reduction in the number of words in Vocabulary is due to the stop words that were removed from the message. Training examples that were used to train naïve Bayes produce 92.22 percent accuracy as shown in the confusion matrix (see Figure 3).

	Classified as		
--->	Clean	Bad	| Total
Clean	180	17	| 197
Bad	3	57	| 60

Figure 3: Confusion matrix for test with training examples

Based on the result, 180 of the clean messages were correctly classified as Clean and 17 other messages were classified as Bad. Among of the bad messages, three were classified as Clean out of 60. True

PREPROCESS_DATA(*Database*)
Database is a file which the source for preprocessing.
Build *Examples* from the data
  For each record in *Database*
    RawText ← remove stop words, common words
    Text ← collects all words, punctuations and other tokens from *RawText*
    Examples ← add *Text* to *Examples*

Build Vocabulary
  For each example in Examples
  Text ← remove repeated from *example*
  Vocabulary ← add *Text* to *Vocabulary*

Figure 1: New preprocessing algorithm

LEARN_NAIVE_BAYES(*Examples*)
Examples is a set of text documents along with their target values and contains the set of all possible values V found in the *Examples*. This function will learn the probability terms $P(w_k|v_j)$ and prior probability $P(v_j)$.

Calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms
  For each target value v_j in *V* do
    docsj ← the subset of document from *Examples* for which the target value is v_j
    $P(v_j) \leftarrow \frac{|docsj|}{|Examples|}$
    $Text_j$ ← a single document from *Examples* for which the target value is v_j
    n ← total number of distinct word positions in $Text_j$

    For each word w_k in *Vocabulary*
      n_k ← number of times word w_k occurs in $Text_j$
      $P(w_k|v_j) \leftarrow \frac{n_k + 1}{n + |Vocabulary|}$

CLASSIFY_NAIVE_BAYES(*Text*)
Returns the estimated target value for the text Text. a_i is the word found in the *i*th position within *Text*.
  positions ← all word positions in *Text* that contain tokens found in *Vocabulary*
  Return v_{NB}, where

$$v_{NB} = \arg\max_{vj \in V} \prod_i P(a_i \mid v_j)$$

Figure 2: Algorithm for learning and classifying text

classification of the messages is 237 (180+57) and the training accuracy is computed as follows:

Training Accuracy = 237 / 257 * 100
= 92.22%

The performance of this second experiment using the modified preprocessing algorithm is slightly reduced compared to 98.05 percent when stop word is not applied. This is due to many of the words in the vocabulary that estimate the probabilities of the message have been reduced.

The naïve Bayes algorithm then applied to the remaining 45 unseen messages and the accuracy achieved by the algorithm was 66.77 percent. Confusion matrix shown in Figure 4 describes about naïve Bayes prediction over the performance measure data set.

	Classified as		
-->	Clean	Bad	| Total
Clean	16	7	| 23
Bad	8	14	| 22

Figure 4: Confusion matrix for test with validation message

The result in Figure 4 shows that seven of the clean messages and eight of the bad messages were misclassified by naïve Bayes. Stop words is supposed to increase the performance of naïve Bayes classification when it was applied, however, due to the small number of training examples and furthermore reduced by stop words, search space for probabilities are also reduced. This concludes that naïve Bayes algorithm is strongly dependent to the word probability that if more words are ignored (no probability in the vocabulary), then the message will produce lower classification towards certain class value. The example message that was classified as bad by the naïve Bayes is shown in Figure 5.

Sue where can I get documentation for that

Figure 5: Example message

The text in Figure 5 consists of an author name and message which obviously has no bad words. However, naïve Bayes has predicted the message as bad because there were not enough probability terms that predict the message as clean. Based on the message, stop words such as 'where', 'can', 'I', 'get', 'for' and 'that' were removed from the message that leaving 'Sue' and 'documentation' to determine whether the message is clean or bad. The naïve Bayes learning model specifies the probability of class Clean and Bad are 0.77 and 0.23 respectively while the remaining word probabilities in the example message is presented in Table 1. Further calculation of v_{NB} is then computed where naïve Bayes assigns 0.494481767438259 for class Clean and 0.505518232561741 for class Bad. Using (2), the value for *P(Bad|message)* is maximum and therefore selected as a classification for the message.

TABLE 1: PROBABILITY ESTIMATION FOR WORDS IN THE EXAMPLE MESSAGE (FIGURE 6)

Word	P(Word|Clean)	P(Word|Bad)
Sue	6.76E-03	1.34E-03
documentation	1.62E-04	2.23E-04

The probability estimation for word 'documentation' with class Bad is greater than class Clean. This is somehow contributes the false classification of the message as bad. The trace through the algorithm shows that the function to estimate $P(w_k|v_j)$ for 'documentation' gives larger probability value despite none of the word appear in the bad message. This due to the small number of words positions in all training examples whose target value is v_j that uniformly estimates the priors. The *m*-estimate in Equation 3, shows how each of the word in Vocabulary gets its probability terms.

$$P(w_k|v_j) \leftarrow \frac{n_k + 1}{n + |Vocabulary|} \quad (3)$$

In (3), the divisor ($n+|Vocabulary|$) for clean message is 12328 (10175 + 2153) and bad message is 2731 (578 + 2153). In addition, the word 'documentation' appears two times in clean messages and none in bad messages. The *m*-estimates for 'documentation' using (3) for both classes are:

$$P(documentation|Clean) = \frac{2+1}{10175+2153} = 2.43E\text{-}04$$

$$P(documentation|Bad) = \frac{0+1}{578+2153} = 3.66E\text{-}04$$

The calculation for *m*-estimate in the LEARN_NAIVE_BAYES function is obviously giving higher estimates to the word as bad because of the small divisor value in (3) when the training example is small. However, if training examples is large, we could expect that the word 'documentation' will appear more in the clean message and will overcome the problem of small training examples.

In terms of processing time, the stop word has slightly decreased the time to classify messages compared to the previous experiment. In the online classification, message shown in Figure 5 was processed in 0.1914 seconds compared to 0.4102 seconds if stop word is not applied. The processing time will increase depending on the text size where abstract section of this paper is processed in 1.4102 seconds and introduction section in 3.0625 seconds. Active Server Pages (ASP) produces error when the

number of minimum floating point reaches 1.03449816861596E+306, where value is near to absolute zero. In this situation, setting for minimum floating point is set to 2.56914135837448E-322 to avoid the probability overflow computation. This setting will be useful when classifying large text where the whole content of this paper is classified in approximately 32.6289 seconds.

The experimental result was produced on Pentium III 933Mhz machine with 256MB RAM with stop words or common words ignored. The naïve Bayes computation for prediction assumes only a subset of words in the text is included and words that were not found in the vocabulary are simply ignored.

5.0 CONCLUSION AND FUTURE WORKS

The new modification on the preprocessing algorithm contributes relatively small drop in the accuracy, however if the number of training examples is big, the search space for priors probability is expected to increase the accuracy. Again, Naïve Bayes algorithm will perform better if number of training examples is enough to cover words which are normally used in forums. There is a need to find another method to estimate the priors when the number of training examples is small and the difference between clean and bad message size is big.

The experiment to apply naïve Bayes learner in forum message is interesting and the performance of the classifier including classification and processing can be improved with high performance machine and better algorithm for text filtering.

References

[1] M.S. Sainin, "Applying Learning to Filter Text in Forum Message". To appear in Proceeding of Seminar Sosio Ekonomi Ke 3 (SEIT03), Universiti Utara Malaysia, Perlis, 2005.

[2] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos, "An Evaluation of Naive Bayesian Anti-Spam Filtering". Proceedings of the workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning, Barcelona, Spain, 2000, pp. 9-17.

[3] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, and, C. D. Spyropoulos, "An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering", In Proc. of the 23 rd Annual International ACM SIGR Conference on Research and Development in Information Retrieval, 2000, pp. 160-167.

[4] E. Michelakis, I. Androutsopoulos, G. Paliouras, G Sakkis and P. Stamatopoulos, "Filtron: A Learning-Based Anti-Spam Filter", Proceedings of the 1st Conference on Email and Anti-Spam (CEAS 2004), Mountain View, CA, USA, 2004.

[5] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian Approach to Filtering Junk E-Mail", In AAA1 Workshop, 1998, pp. 55-62.

[6] T. Mitchell, Machine Learning, McGraw Hill, 1997, pp. 180-184.

[7] I. H. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufman, 1999, pp. 88-89.