Comparison of a SOM Based Sequence Analysis System and Naive Bayesian Classifier for Spam Filtering

Xiao Luo, Nur Zincir-Heywood
Faculty of Computer Science
Dalhousie University
Halifax, NS, Canada, B3H1W5
E-mail: luo,zincir@cs.dal.ca

***Abstract*— The problem introduced by the unsolicited bulk emails, also known as ”spam” generates a need for reliable anti-spam filters. In this paper, we design and compare the performance of a newly designed SOM based sequence analysis(SBSA) system for the spam filtering task. The system is based on a SOM based sequential data representation combined with a kNN classifier designed to make use of word sequence information. We compare this system with the traditional baseline method Naive Bayesian filter. Three different cost scenarios and suitable cost-sensitive measurements are employed. The results show that the SBSA system is superior to the Naive Bayesian filter, particularly when the misclassification cost for non-spam message is high.**

I. Introduction

Electronic mail is an efficient and low cost communication medium. With the increasing of its popularity, it brings numerous benefits to private and business users, but also it gives unscrupulous marketers or advertisers potentials to send their advertisements to a large number of people for very low costs. These messages are called unsolicited bulk email, junk mail or spam. These unsolicited mails have already caused many problems such as filling mailboxes, consuming network resources, wasting users’ time and energy to sort through it, etc. Therefore, effective spam filters to identify spam messages are in demand.

In its simple form, spam filtering can be recast as text categorization task where the classes to be predicted are spam and legitimate. A variety of machine-learning algorithms have been successfully applied to mail filtering task. A non-exhaustive list includes: Naive Bayesian classifier [1] [12] [14], RIPPER rule induction algorithm [6], support vector machines [7] [9], memory-based learning [2], AdaBoost algorithm [5], and maximum entropy model [16]. All of them show appealing results. However, none of them considers the significance of the word sequence in a message and make use of the sequence information for the mail filtering task. In this work, our objective is to examine the performance of a newly designed SOM based sequence analysis system for the task of e-mail spam filtering. The SOM based sequence analysis system is based a new way of document representation, which keeps information regarding the temporal sequences of words, as well as their frequencies. A *k*-Nearest-Neighbour algorithm with a new cost function is applied on top of SOMs for classification.

SOM algorithm has been introduced for sequence processing and analysis by Carpinteiro [4] in 2000 and Barreto and Arujo [3] in 2001. In this paper, we made use of a hierarchical Self Organizing Feature Maps (SOMs) architecture to understand the character and word sequence by encoding the original information in a hierarchy by first considering the relationships between characters, then words. After the SOMs training process, each document is represented by a sequence of *best matching units* (*BMU*s) on the second level SOM and the Euclidean distances to the corresponding neurons (*BMU*s). To this end, the machine learning classification algorithm *k*-Nearest Neighbour (*k*NN) is employed for the stage of categorization. However, to make use of the sequence information that based on the hierarchical Self Organizing Feature Maps (SOMs) architecture, a new similarity function is designed by the authors to use with *k*NN. We compare this SOM based sequence analysis system with the Naive Bayesian filter which often serves as a baseline method for comparison in the anti-spam filtering area. The results show that the SOM based sequence analysis system is by far superior to the Naive Bayesian filter regarding the performance measurements widely used for the anti-spam filtering area.

The rest of the paper is organized as follows. Section 2 describes the corpus collection and feature selection method used in our experiments. The Naive Bayesian classifier is introduced in section 3. Section 4 presents the designed SOM based sequence analysis system. Section 5 introduced the performance measurements which is widely used for the anti-spam filtering area. Section 6 gives the experiments performed and the results. Finally, conclusions are drawn and future work is discussed in section 7.

II. Corpus Collection and Feature Selection

In this work, we use the well known data set - Ling-Spam which is used for bench-marking spam filters and freely available from http://www.aueb.gr/users/ion/data/lingspam_public.tar.gz. The original corpus consists of 2893 messages. Among them, 2412 legitimate messages taken from a linguistics mailing list and 481 spam messages received by the first author of [1]. The messages in the corpus have header fields, attachments, and html tags removed, leaving only subject line and mail body text. Four versions of the corpus were created: a plain-text version; a version with word-stemming; a version with stop-words removed, and a version with stemming and stop-word removal. We used the stemmed version with stop-words removed.

Dimension reduction which is also called feature extraction or feature selection is employed to solve the high dimensionality problem and reduce the size of the feature space. Information Gain (IG) has been used in several anti-spam experiments under the name of ”Mutual Information” [2] [12] [14]. We use it as a feature selection method in our experiments. IG score of a feature w (in this case, a word) is computed by using the formula 1

$$IG(w,C) = \sum_{w\in\{0,1\}} \sum_{C\in\{spam,legitimate\}} P(w,C)\cdot \log \frac{P(w,C)}{P(w)P(C)} \quad (1)$$

N features which have the highest IG scores are usually selected for experiments. The probabilities $P(w|C)$, $P(C)$ and $P(w)$ are estimated from the relative frequencies of the training set.

III. Naive Bayesian Classifier

Naive Bayesian Classifier is a kind of probabilistic classifier [15]. It works as computing the probability that a document represented by a vector $d = \langle w_1, w_2, \ldots, w_n\rangle$of terms belongs to category C. The probability is computed by applying the Bayes' theorem, given by

$$P(C|d) = \frac{P(C)P(d|C)}{P(d)} \quad (2)$$

$P(C|d)$ is thus the probability that document d belongs to category C. $P(d)$ is the probability that a randomly picked document has vector $d = \langle w_1, w_2, \ldots w_n\rangle$as its representation, and $P(C)$is the probability that a randomly picked document belongs to C. $P(d|C)$is the probability that a document belongs to category C has the vector representation d. Because the possible value of d are too many, the estimation of $P(d|C)$is problematic. In order to alleviate the problem, it is common to assume that $w_1, w_2, \ldots w_n$ are conditionally independent given the category C. As a result, the $P(d|C)$is computed as:

$$P(d|C) = \prod_{i=1}^{n} P(w_i|C) \quad (3)$$

This is where the name ”Naive” comes from. $P(w_i|C)$ and $P(C)$ are easy to estimate from the relative frequencies of the training set. The Naive Bayesian classifier is a widely used classifier in text categorization task [10] [11]. It also enjoys a blaze of popularity in antispam researches [1] [12] [14], and often serves as baseline method for comparison with other approaches. Despite the fact that the ”Naive” assumption is often violated in real-world data, Naive Bayesian classifier is found to be surprisingly effective in the area of text categorization as well as on the anti-spam filtering task.

IV. SOM Based Sequence Analysis System

The SOM based sequence analysis system consists of two parts as mentioned before. The first part is the document representation part, where a two-level hierarchical SOM architecture is introduced to automatically encode word sequence of a document to a sequence of neurons on the map. The second part is the sequence analysis and categorization part where a newly designed sequence based *k*-Nearest Neighbour (*k*NN) classifier is employed.

A. Hierarchical SOM encoding architecture and document representation

Each word in the documents is pre-processed to provide a numerical representation for each character. SOMs may now be used to identify a suitable character encoding, then word encoding, and finally, encoded word sequence is constructed to represent a document. The hierarchical nature of the architecture is shown in figure 1.

1) Input for the First-Level SOMs: The assumption at this level is that the SOM forms a codebook for the patterns in characters that occur in a specific document category. In order to train an SOM to recognize patterns in characters, the document data must be formatted in such a way as to distinguish characters and highlight the relationships between them. Characters can easily be represented by their ASCII representations. However, for simplicity, we enumerated them by the numbers 1 to 26, i.e. no differentiation between upper and lower case. The relationships between characters are represented by a character's position, or time index, in a word. For example, in the word “*cost*”: “*c*” appears at time index 1, “*o*” appears at time index 2, “*s*” appears at time index 3, etc. It should be noted that it is important to repeat these words as many times as they occur in the documents, so that the neurons on the SOM will be more excited by those frequent words and their information will be more accurately encoded. The overall pre-processing process for the first-level SOM is therefore:

- Convert the word's characters to numerical representations between 1 and 26.
- Give the time index to the characters in a word. It is the actual time index plus 2, except the first character in the word.

The indices of the characters is altered in this way so that when the list is input to an SOM, both data features (enumerated characters and indices) are spread out over a close range. There is no bias on either of the features. All the values are normalized before presentation to the SOM.

2) Input for the Second-Level SOMs: The assumption at this level is that the SOM forms a codebook for the patterns in words that occur in a specific document category. When a character and its index are run through a trained first-level SOM, the closest neurons (in the Euclidean sense), or *Best Matching Units* (*BMU*s), are used to represent the input space. A two-step process is used to create a vector for each word, k, which is input to the first-level SOM of each document:

- Form a vector of size equal to the number of neurons (*r*) in the first-level SOM, where each entry of the vector corresponds to a neuron on the first-level SOM, and initialize each entry of the vector to 0.
- For each character of word,
  - Observe which neurons $n_1, n_2, \ldots, n_r$ are affected the most.
  - Increase entries in the vector corresponding to the 3 most affected *BMU*s by $1/j, 1 \leq j \leq 3$.

Hence, each vector represents a word through the sum of its characters. The result given by the second-level SOM is clusters of words on the second-level SOM.

3)Training SOM - Learning algorithm:

The algorithm responsible for the formation of the SOM involves three basic steps after initialization: sampling, similarity matching, and updating. These three steps are repeated until the formation of the feature map has completed [8]. The algorithm is summarized as follows:

- Initialization: Choose random values for the initial weight vectors $W_j(0), j = 1, 2, \ldots, l,$ where l is the number of neurons in the map.
- Sampling: Draw a sample x from the input space with a uniform probability.
- Similarity Matching: Find the best matching neuron $i(x)$ at time step n by using the minimum distance Euclidean criterion:

$$i(x) = \arg\min \|x(n) - W_j\|, j = 1, 2, \ldots, l \tag{4}$$

- Updating: Adjust the weight vectors of all neurons by using the update formula:

$$W_j(n+1) = W_j(n) + \eta(n) \cdot h_{j,i}(n)(x(n) - W_j(n)) \tag{5}$$

- Continuation: Continue with sampling until no noticeable changes in the feature map are observed.

At the end of the training process, the observed average weight changes of the neurons on the SOM over the training period (in epochs) is analyzed to determine whether the size of the SOM is big enough to represent the pattern distribution of the input data. The sizes of the maps chosen for each level of SOM are shown in Table I.

TABLE I
SIZE OF THE SOMS

	Two-level SOMs architecture
Level-1	7 by 13
Level-2	12 by 12

After training the SOMs, each word of a document excites a corresponding *BMU* on the second level SOM, so that a document in terms of word sequence is transferred to a *BMU* sequence on the SOM. We propose a two dimensional vector to represent each word by using its corresponding *BMU*. The *BMU* sequence is presented as shown in Table II. The length of the sequence is the number of words in the document. The first dimension is the index of the *BMU*s on the second SOM; the second dimension is the Euclidean distances to the corresponding *BMU*s.

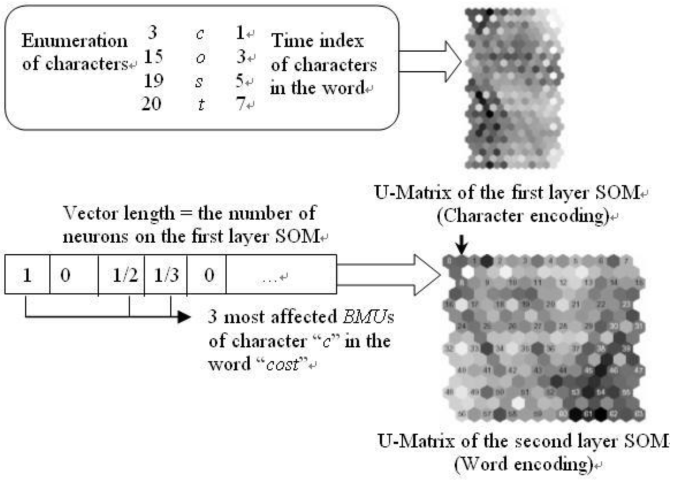
Enumeration of characters
3 c 1
15 o 3
19 s 5
20 t 7
Time index of characters in the word
U-Matrix of the first layer SOM (Character encoding)
Vector length = the number of neurons on the first layer SOM
1 0 1/2 1/3 0 ...
3 most affected BMUs of character "c" in the word "cost"
U-Matrix of the second layer SOM (Word encoding)

Fig. 1. An overview of the hierarchical SOMs encoding architecture

TABLE II
SEQUENCE PRESENTATION OF A DOCUMENT

Document	Word1	Word2	...
Index of BMUs on SOM:	56	7	...
Euclidean distance to the BMU (W):	1.7	0.9	...

B. Sequence based k-Nearest-Neighbour (kNN) classifier

k-Nearest-Neighbour (*k*NN) classifier is a memory-based or instance-based learning algorithm [15]. To classify a test document, the *k*-Nearest-Neighbour classifier algorithm finds the *k* nearest neighbours among the training documents, and uses category labels of the *k* nearest training documents to predict the category of the test document. The *k*NN classifier has been studied for e-mail spam filtering by Androutsopoulos et al. [2] for the vector space representation of a document.

In the *k*NN classifier, various metrics can be used to compute the distance between two instances. We designed a metric to fit to the word sequence (*BMU* sequence) presentation of a document. Given a test document T_i and a training document D_j the formula 6 is used to calculate the distance between them. The metric considers the length of the shared *BMU* sequence between two documents, the similarity degree of the *BMU*s in the shared the sequence, and the length in terms of the number of *BMU*s of the training document.

$$Sim(T_i, D_j) = \frac{\sum_{m=1}^{N} \frac{1}{1+dist(W_{im}, W_{jm})} \times N}{length(D_j)} \tag{6}$$

T_i : Test document to be categorized.

D_j : Document in the training set.

N : Total number of *BMU*s in common *BMU* sequences of T_i and D_j .

W : Euclidean distance of a word to the corresponding *BMU*

$dist(W_{im}, W_{jm})$: The distance between W in the common *BMU* sequences of T_i and D_j .

$length(D_j)$: Length of the document in the training set in terms of *BMU* sequence.

V. Performance Measurements

Similar to the performance measurements of text categorization, we calculate the spam recall(SR), and spam Precision (SP) as following:

$$SR = \frac{n_{S\rightarrow S}}{n_{S\rightarrow S} + n_{S\rightarrow L}} \tag{7}$$

$$SP = \frac{n_{S\rightarrow S}}{n_{S\rightarrow S} + n_{L\rightarrow S}} \tag{8}$$

$n_{S(L)\rightarrow S(L)}$: Number of spam (legitimate) messages classified to be spam (legitimate)

$n_{L\rightarrow S}$: Number of legitimate messages classified to be spam

$n_{S\rightarrow L}$: Number of spam messages classified to be legitimate

In anti-spam filtering, it is worth to notice that misclassifying a legitimate mail as spam is much more severe than letting a spam message pass the filter. Letting a spam to go through the filter generally does no harm while misblocking an important personal mail as spam can be a disaster. The precision, recall and F1-measure reflect little about the filter's performance from this aspect. Androutsopoulos et al. [1] introduced some cost-sensitive measurements, WAcc and WErr, which assign false positive a higher cost than false negative. These measurements are widely used in the anti-spam filtering area and defined as following:

$$WAcc = \frac{\lambda \cdot n_{L\rightarrow L} + n_{S\rightarrow S}}{\lambda \cdot N_L + N_S} \tag{9}$$

$$WErr = \frac{\lambda \cdot n_{L\rightarrow S} + n_{S\rightarrow L}}{\lambda \cdot N_L + N_S} \tag{10}$$

N_L : The total number of legitimate messages.

N_S : The total number of spam messages.

λ : The cost-sensitive factor. Classifying legitimate messages to spam messages is λ times more costly than classifying spam messages to legitimate messages.

Three different values of λ :1,9, and 999 were introduced by Androutsopoulos et al. [1]. When λ is set to be 1, spam and legitimate messages are weighted equally; when λ is set to be 9 or 999, a false positive is penalized 9 or 999 times more than a false negative. $\lambda = 999$ is suitable for the scenario where messages marked as spam are deleted directly. In practice, the value of $WAcc$is often misleadingly high. Especially, when λ is set to be 999. In order to avoid this problem, Androutsopoulos et al. [1] suggest to compare the weighted accuracy or error rate to those of a simplistic baseline, where no filter is used legitimate messages are never blocked, and spam messages always pass. The weighted accuracy and weighted error rate of the baseline are given as following:

$$WAcc^b = \frac{\lambda \cdot N_L}{\lambda \cdot N_L + N_S} \tag{11}$$

$$WErr^b = \frac{N_S}{\lambda \cdot N_L + N_S} \tag{12}$$

The *total cost ratio* (TCR) shown as formula 13 allows easy comparison with the baseline. The greater the TCR value is, the better the performance. An effective spam filter should be able to achieve a TCR value higher than 1 in order to be useful in real world applications.

$$TCR = \frac{WErr^b}{WErr} = \frac{N_S}{\lambda \cdot n_{L\rightarrow S} + n_{S\rightarrow L}} \tag{13}$$

VI. Experimental Results

We implemented the Naive Bayesian filter (NB) and the SOM based sequence analysis system (SBSA) as described in section 3 and 4. Experiments are done on a balanced data set and an unbalanced data set to see if the spam rate of the training set has effect on the final performance on both systems. The unbalanced data set is the original Ling-Spam data set, where spam rate is 16.6%, while the balanced data set is a subset of original Ling-Spam data set, where we keep all the spam messages and randomly selected the same amount of legitimate messages so that the spam rate is increased to 50%. We randomly split all messages into training set and test set of ratio 3:1, and keep the spam rate the same in both the training and test sets.

In order to fully explore the performance of both systems, we varied the number of selected features (features with highest IG score) from 50 to 750 by a step of 100, and experimented on TCR value set to 1,9, and 999. As for the SBSA system, we varied the *k* value from 1 to 10 for the stage of classification. Figure 2- 4 show the results of the SBSA system and the Naive Bayesian filter for each λ value on the balanced data set. Although we carried out experiments for every *k* value from 1 to 10, we present only some representative curves because of the limited space.

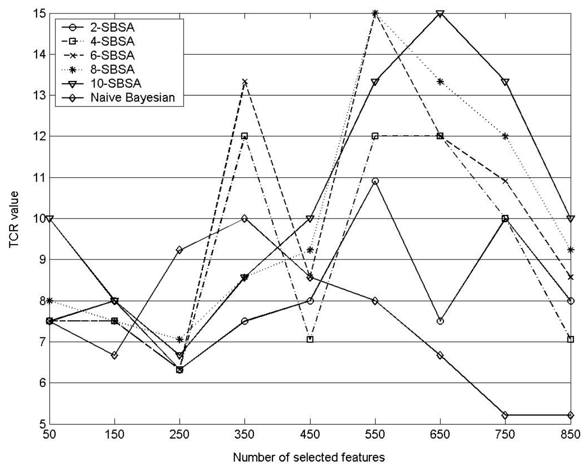

Fig. 2. TCR curves for λ =1 on balanced data set

In figure 2, both learning systems improve significantly on the baseline where TCR=1. The trend for SBSA system is: when the number of selected features is 550 or 650, the highest TCR value obtained for all *k* values, and the overall TCR value seems to be increasing with the increase of the *k* value. As for the Naive Bayesian, it peaks when the number of selected features is 350. But the SBSA system is better than the Naive Bayesian filter even in the worst case *k*=2.

Figure 3 presents the scenario λ =9, we show the worst case(*k*=1) and the trend of most other cases(*k*=3,4,7,9) for

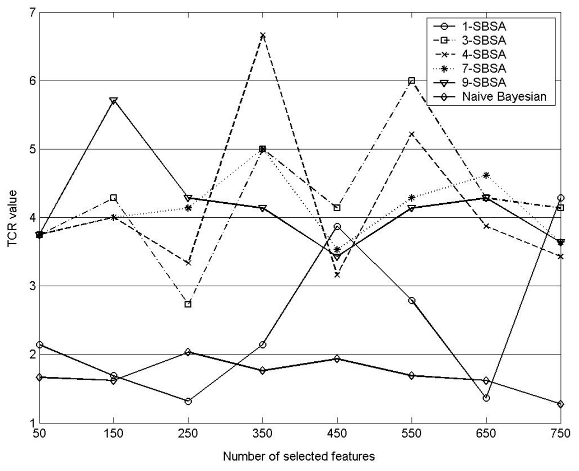

Fig. 3. TCR curves for λ =9 on balanced data set

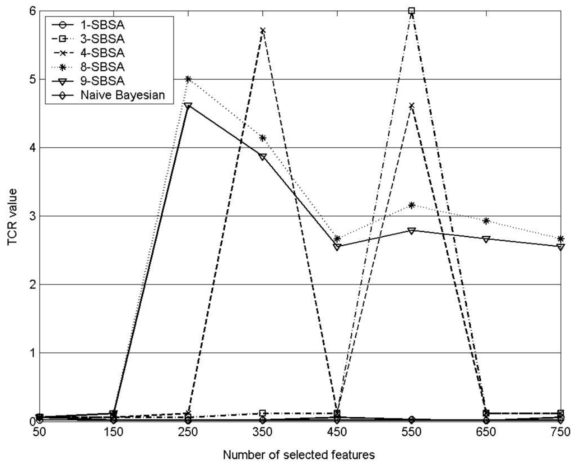

Fig. 4. TCR curves for λ =999 on balanced data set

SBSA system. The behaviors of both of the systems are different here. The performance of SBSA system does not increase as the *k* value increases. It performs best when *k*=4, and deteriorate a little when *k*=7. The performance of Naive Bayesian is relatively stable. It peaks when the number of selected features is 250. But still, its performance is worse than the worst case of the SBSA system.

Finally, in figure 4, we observed some steep fluctuations in the TCR curve of the SBSA system for some *k* values. It indicates the transitions from below baseline to above and back below the baseline. *k*=3 achieves the highest TCR value for 550 selected features. However, it is not useful, as its performance is not steadily above the baseline. When $k >= 5$, such as k=8,9, there is no steep fluctuation in the TCR curve. The performance peaks at 250 selected features. We conclude that in this scenario, the SBSA system can obtain a reliable performance by using a larger *k* value. The performance of the Naive Bayesian is similar to the case *k*=1 of SBSA system, which is below the baseline performance.

Except the TCR values, we also analyzed the performance of the two systems by other measurements such as spam recall(SR), spam precision(SR), and weighted accuracy(WAcc) as described in section 5. The best results of both methods for each λ value is given in table III. Apparently, when λ=9 or 999, with the SBSA system, the spam precision raised to 1, which means that no legitimate message has been misclassified to spam. This is a crucial issue to the anti-spam filtering, because in scenarios λ=9 or 999, it is safe to not block the legitimate messages.

TABLE III
BEST PERFORMANCE FOR BALANCE(B) AND UNBALANCE (U) DATA SETS

Methods	λ	No. of features	SR	SP	WAcc	TCR
6-SBSA(B)	1	550	0.975	0.959	0.967	15.0
NB(B)	1	350	0.958	0.943	0.946	10.0
5-SBSA(U)	1	450	0.933	0.933	0.978	7.5
NB(U)	1	450	0.925	0.941	0.976	7.5
4-SBSA(B)	9	350	0.850	1.0	0.985	6.667
NB(B)	9	100	0.917	0.957	0.947	2.182
7-SBSA(U)	9	750	0.741	1.0	0.994	3.871
NB(U)	9	450	0.925	0.941	0.985	1.667
8-SBSA(B)	999	250	0.800	1.0	0.999	5.0
NB(B)	999	50	0.917	0.982	0.975	0.06
7-SBSA(U)	999	250	0.675	1.0	0.999	3.08
NB(U)	999	450	0.925	0.957	0.990	0.024

Figure 5- 6 show the results of the SBSA system and the Naive Bayesian filter for λ =1, 9 or 999 on the unbalanced data set. The results show that the performances of both systems decreased on the unbalanced data set. We hypothesis the reasons behind this are: first, the size of the unbalanced data set is larger than the balanced data set, so that it introduces more noise and difficulty for the classification. Second, with the SBSA system, the newly designed data representation is based on the machine-learning algorithm to represent the characteristic words for the categories, so the more frequent the word is, the more easily it can be caught and represented by the neurons of the second level SOM. In the unbalanced data set, percentage of legitimate messages is much more than that of spam; there are a larger number of words from this category. This results in more neurons are well trained to represent the characteristic words of this category. Thus, some spam messages may be distortedly represented and be misclassified. This returns lower spam recalls shown in table III, especially, when λ =9 and 999.

However the SBSA system still perform better than Naive Bayeisan on the unbalanced data set. Table III gives the best results of both methods for each λ value.

Further more, it is worth mentioning that the performance of the SBSA system is better than that of the vector space model based kNN spam filter developed by Androutsopoulos et al. [2] [13] on the different ratio of training and test sets.

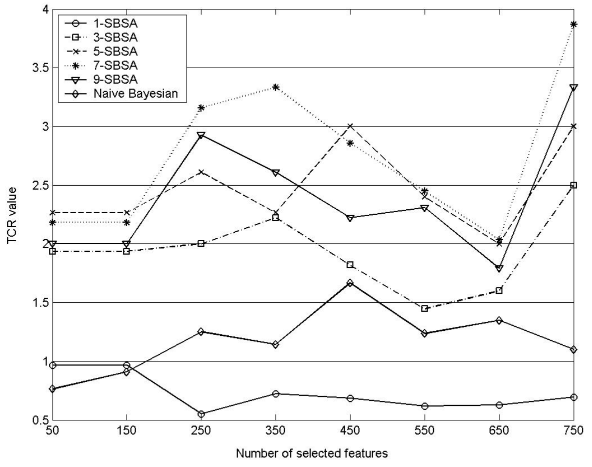

Fig. 5. TCR curves for λ =9 on unbalanced data set

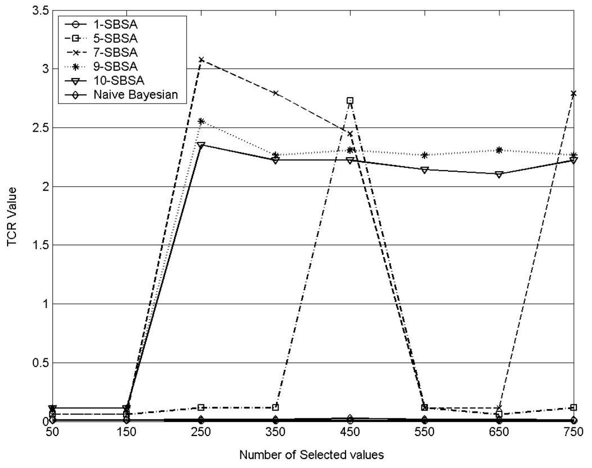

Fig. 6. TCR curves for λ =999 on unbalanced data set

VII. Conclusions

In this work, we explored a SOM based sequence analysis system for anti-spam filtering, and we compared its performance with the Naive Bayesian filter. The returned results show that the SOM based sequence analysis system successfully encode the word sequence of a document and make use of it for the message classification and it out perform the Naive Bayesian filter for all λ scenarios, especially for the scenarios λ =9 and 999, which consider the weight of two types of error (misclassify spam as legitimate message and wrongly mark legitimate message as spam). In general, these scenarios are what people more interested in for anti-spam filtering where wrongly labelling a legitimate mail is penalized much more than letting a spam pass the filter.

The efficiency of this SOM based sequence analysis system for the spam filtering presented in this paper is still not completely elaborated, so we definitely consider this as a work in progress. Future work will include performing more extensive evaluation on the sequence analysis system on other data sets and compare it against other systems and moreover, we plan to make use of the sequence of special characters, such as spaces or '$' that can be relevant for identifying spam messages. Finally, other classifiers which fit more to the sequences representation will also be analyzed and utilized in the future.

References

[1] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos and C. D. Spyropoulos, "An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Personal E-mail Messages," *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* pp. 160-167, 2000.

[2] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C.D. Spyropoulos and P. Stamatopoulos, "Learning to Filter Spam E-Mail: A Comparison of a Naive Bayesian and a Memory-Based Approach," *Proceedings of the Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases,* pp. 1-13, 2000.

[3] G. A. Barreto and A. F. R. Arujo, "Time in Self-Organizing Maps: An Overview of Models," International Journal of Computer Research, vol. 10, no. 2, pp. 139-179, 2001

[4] O. A. S. Carpinteiro, "A Hierarchical Self-organizing Map Model for Sequence Recognition." Pattern Analysis Application, vol. 3, no. 3, pp. 279-287, 2000.

[5] X. Carreras and L. Marquez, "Boosting Trees for Anti-Spam Email Filtering," *Proceedings of the 4th International Conference on Recent Advances in Natural Language Processing,* pp. 58-64, 2001.

[6] W. Cohen,"Learning rules that classify e-mail," *Spring Symposium on Machine Learning in Information Access,* 1996.

[7] H. Drucker, D. Wu, and V. N. Vapnik, "Support vector machines for spam categorization," *IEEE Transactions on Neural networks,* vol. 10, no. 5, pp. 1048-1054, 1999

[8] S. Haykin, "Neural Networks - A comprehensive foundation," Second Edition, Prentice Hall, 1999.

[9] A. Kolcz and J. Alspector, "SVM-based Filtering of E-mail Spam with Content-specific Misclassification Costs," *Proceedings of the ICDM-2001 Workshop on Text Mining* , 2001.

[10] D. Koller and M. Sahami, "Hierarchically classifying documents using very few words," *Proceedings of the 14th International Conference on Machine Learning,* pp. 170-178, 1997.

[11] D. D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval," *Proceedings of the 10th European Conference on Machine Learning,* pp. 4-15,1998.

[12] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian approach to filtering junk e-mail," *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization,* 1998.

[13] G. Sakkis, I. Androutsopoulos, G. Paliouras, P. Stamatopoulos, "A Memory-Based Approach to Anti-Spam Filtering for Mailing Lists," Information Retrieval, vol. 6, pp. 49-73, 2003.

[14] K. Schneidera, "A comparison of event models for naive bayes anti-spam e-mail filtering," *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics,* pp. 207-314, 2003.

[15] F. Sebastiani, "Machine learning in automated text categorization." ACM Computing Surveys, vol. 34, no. 1, pp. 1-47, 2002.

[16] L. Zhang and T. Yao, "Filtering Junk Mail with A Maximum Entropy Model," *Proceedings of the 20th International Conference on Computer Processing of Oriental Languages,* pp. 446-453, 2003