# MML Inference of Finite State Automata for Probabilistic Spam Detection

Vidya Saikrishna
Monash University
Clayton
Victoria 3800
Australia
Email: vidya.saikrishna@monash.edu

David L. Dowe
Monash University
Clayton
Victoria 3800
Australia
Email: david.dowe@monash.edu

Sid Ray
Monash University
Clayton
Victoria 3800
Australia
Email: sid.ray@monash.edu

***Abstract*—MML (Minimum Message Length) has emerged as a powerful tool in inductive inference of discrete, continuous and hybrid structures. The Probabilistic Finite State Automaton (PFSA) is one such discrete structure that needs to be inferred for classes of problems in the field of Computer Science including artificial intelligence, pattern recognition and data mining. MML has also served as a viable tool in many classes of problems in the field of Machine Learning including both supervised and unsupervised learning. The classification problem is the most common among them. This research is a two-fold solution to a problem where one part focusses on the best inferred PFSA using MML and the second part focusses on the classification problem of Spam Detection. Using the best PFSA inferred in part 1, the Spam Detection theory has been tested using MML on a publicly available Enron Spam dataset. The filter was evaluated on various performance parameters like precision and recall. The evaluation was also done taking into consideration the cost of misclassification in terms of weighted accuracy rate and weighted error rate. The results of our empirical evaluation indicate the classification accuracy to be around 93%, which outperforms well-known established spam filters.**
**Keywords: Finite State Automaton (FSA), Probabilistic Finite State Automaton (PFSA), Minimum Message Length (MML), Bayesian Information Theory, Spam Filtering.**

## I. Introduction

Finite State Automata are mathematical models of computation that can model a large number of problems amongst which describing the grammar of a language is the most common. They can be viewed as an effective way of representing regularities and patterns in a certain form. The class of Probabilistic Finite State Machines or Automata (PFSAs) represents a probability distribution of strings in a language where the problem of finding the most probable Finite State Machine for a distribution arises in a number of computational tasks in the field of Computer Science including artificial intelligence, pattern recognition and data mining [1]. Questions about the best inferred machine have served directly or indirectly as the predominant focus of research. From an information-theoretic view point, the best inference can be drawn from the explanation length of both theory and data although there is always a compromise between the complexity and accuracy. The best explanation of the facts is the shortest [2, sec. 1.1, pp. 1][3]. It is these kinds of questions that have led to the development of strategies that are aimed at inferring the best machine based on code-lengths computed using the Bayesian information-theoretic Minimum Message Length (MML) Principle.

At the same time, MML has emerged as a viable tool in many classes of problems in the field of machine learning including both supervised and unsupervised learning and there comes the most common problem, which is classification. Spam filtering is a classification problem, and more specifically it is a supervised learning approach owing to fact that we know the classes prior to classification - i.e., spam and non-spam (ham). Statistical data compression models like the one discussed in [4] are known to exist that have shown improved performance as far as the spam misclassification rate, accuracy or precision is concerned but the spam classification can also be tested with the MML technique for even better performance. This paper presents an effective approach to efficiently induce a Probabilistic Finite State Automaton (PFSA) which has its application in the Spam classification problem. The idea behind transforming the set of spam keywords detected by a Bit Parallel String Matching Algorithm [5] into a PFSA is the nature of the Finite State Automaton that results in deterministic state transitions on an input alphabet and this fact makes it suitable to be used in applications like Spam Classification.

We present a search algorithm that uses the greedy approach to efficiently induce a PFSA using MML (Minimum Message Length) [6]. The input to the search method is the prefix tree acceptor of the strings detected by the string matching algorithms. The prefix tree acceptor is a Finite State Automaton that uses the same states for the prefixes of the string generated by language L. At different stages, the method merges pair of nodes taking into consideration that the merging of pairs does not result in a Non-Deterministic Finite State Automaton. The pair merging is considered if the message length of the new machine is smaller than the message length of the machine before the merge. This kind of merge where determinism exists in the context of state transitions is not considered by the Beam Search Algorithm [7], which also aims at inducing PFSAs through a series of merge pairs in a random manner. The method of induction works effectively for large machines.

978-1-4799-7458-0/15/$31.00 ©2015 IEEE

The experiment has been conducted on one of the six publicly available Enron Spam datasets [8] [9] and the results of our empirical evaluation indicate 93% classification accuracy. MML was not only used in the induction of PFSA but was also later used in classification of Spam and Non-Spam mails.

The remainder of the paper is organized as follows. Section II reviews the relevant theory related to Finite State Automata and PFSAs. In section III we review the concept of Minimum Message Length in context with PFSAs, which forms the basis of inducing Probabilistic Finite State Machines. In section IV, the approach used by the proposed search algorithm is discussed. In section V we discuss the experimental results on the spam dataset and finally section VI offers the conclusion and outline for future work.

## II. Finite State Machines and PFSAs

This section reviews the basic theory related to Finite State Automata and its extension in the form of PFSAs to include probabilistic transitions amongst the states given the data. A Finite State Automaton $M$ = <$Q$, $\sum$, $\delta$, $q_0$, $F$>, where $Q$ is a finite set of states, $\sum$ is a finite set of input alphabet symbols, $q_0$ is the initial state, $F$ is a set of final or accepted states and $\delta$ is a transition function mapping $Q \times \sum$ to $Q$.

A Finite State Automaton $M$, begins with the initial state $q_0$ and for each input alphabet, at each step, it undergoes a sequence of state transitions determined by the state transition function $\delta$. The process terminates at one of the final states. During the course of transitions, a string of symbols is produced as the output. This set of strings is the language generated by machine $M$ or alternatively accepted by machine $M$. The computation fails if there are no transitions from the current state with the given input alphabet.

If the language L can be represented as a set of finite strings, then there exists a Finite State Automaton that accepts the strings of the language L. Such an Automaton is called the Prefix Tree Acceptor. To illustrate this, we consider an example from Gains [10] where the set of strings is enumerated as $L = \{CAB/CAAAB/BBAAB/CAAB/BBAB/BBB/CB/\}$. The Prefix Tree Acceptor of the Language L is shown in the diagram below, Figure 1. In the Figure below, we follow the usual conventions of marking the states as circles, labelled arcs as transition between the states and double-circle as final states. The symbol '/' is the delimiter symbol and in the state diagram below, it is assumed that whenever symbol '/' is read, there is a transition back to the initial state.

A Probabilistic Finite State Automaton (PFSA) is essentially a class of Non-Deterministic Finite State Automata, with the degree of non-determinism reduced to a certain extent. The transitions of the Finite State Machines (FSMs) are marked with probability values indicating the probability of the next symbol generated during the transition. A PFSA determines the probability distribution over the strings in a language - the probability for a particular string is the product of the probabilities of transitions that generate it. We require for every state $q \in Q$ and every symbol $a \in \sum$ that one of the following holds:

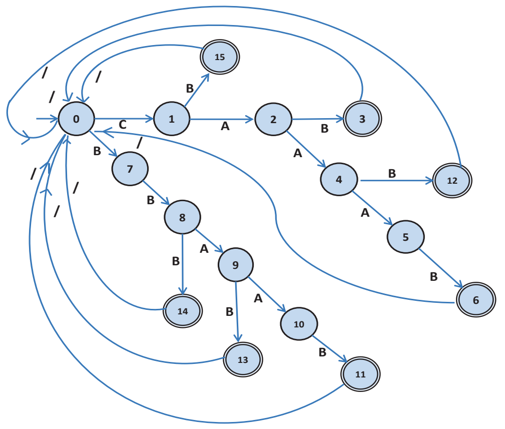

Fig. 1: Finite State Automaton accepting L

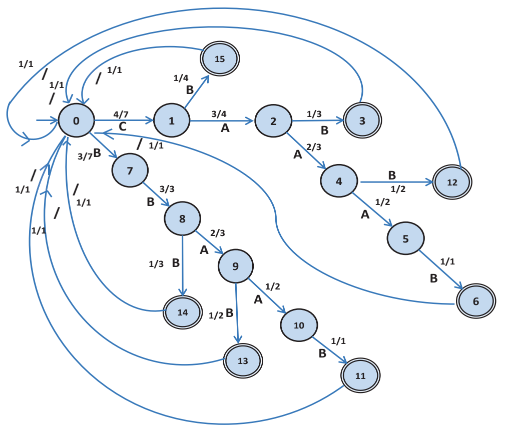

Fig. 2: Probabilistic Finite State Automaton accepting L

$$\text{either } \delta(Q,a) = \emptyset$$
$$\text{or } \textstyle\sum_{q \in \delta(Q,a)} P_r[q \xrightarrow{a} q'] = 1$$

When we construct an FSA from a set of sample strings, we can estimate the transition probabilities by keeping a count of the number of times each arc of the graph is traversed. The counts can be converted into probability estimates by dividing each count by the total count of all the arcs from that state. Doing this with the FSA in Figure 1 generates the PFSA in Figure 2.

## III. MML in reference to PFSAs

The MML principle is a Bayesian information-theoretic metric used for comparing the different hypotheses that are the sources for generation of data. The Minimum Message Length principle [11] states that, given the data, the best possible conclusion about the data can be drawn from the theory that attempts to maximize the posterior probability. Maximizing the posterior probability equivalently means maximizing the product of the prior probability with the likelihood or probability of data given the theory or hypothesis.

Assuming the data to be $D$ and $H$ to be the hypothesis or the underlying theory that generates $D$, the probabilities are denoted as below.
$P(H)$ = Prior Probability of the hypothesis $H$
$P(H|D)$ = Probability of the hypothesis $H$ given the data $D$, also known as the Posterior Probability
$P(D)$ = Marginal Probability of the data $D$
$P(D|H)$ = likelihood of the data given the hypothesis $H$

In a Bayesian framework, $P(H|D) = \frac{P(H)P(D|H)}{P(D)}$, and the most probable $H$ given $D$ maximizes $P(H|D)$. As $D$ is a constant for all the hypotheses to be compared, maximizing $P(H|D)$ is equivalently maximizing the product of $P(H)$ and $P(D|H)$. An elementary information theory concept (related to Huffman coding) states that an event occurring with probability $P$ can be coded in -$\log_2 P$ bits. Highly probable events result in small code lengths. The code length for posterior probability can be represented as -$\log_2(P(H)P(D|H))$, which is equal to -$\log_2 P(H)$ -$\log_2 P(D|H)$. Therefore maximizing the posterior probability is equivalent to minimizing -$\log_2 P(H)$ -$\log_2 P(D|H)$, i.e. the length of a two-part message conveying the theory and the data in light of theory.

### A. Coding Scheme for PFSAs using MML

To develop a coding scheme for PFSAs as described by Wallace in Section 7.1 of [2], we need to find the code lengths for the theory $H$ and the data using or being generated by the theory. Therefore, given data $D$, we seek an FSM which minimizes the following sum.

$$Code - Length(H) + Code - Length(D|H) \qquad (1)$$

Enlisting the terms used in finding the code-lengths, we use the following terminology.

- $S$ is the number of states in the FSM
- $V$ is the cardinality of the alphabet set
- $t_i$ is the number of times a state $i$ is visited
- $n_{ij}$ is the number of transitions from state $i$ to state $j$
- $M$ is the total number of arcs from all the states
- $a_i$ is the number of arcs leaving state $i$

Note that $\sum_{i=1}^{S} a_i = M$.

We first describe the procedure for encoding the structure of PFSA using Wallace's assertion code for PFSAs [2], Section 7.1. This is followed by calculating the code-length of the PFSA with transition probabilities.

#### 1) Encoding the Structure of PFSA:

To encode the structure of a FSM, we consider the same example from Gains [10] where $L = \{CAB/CAAAB/BBAAB/CAAB/BBAB/BBB/CB/\}$.

The coding scheme is as follows [2]

(a) Encoding the number of arcs leaving a state $i$ gives a code length of $\log_2 V$ bits as the number of different possibilities for any arc $a_i$ is between 1 and V.
(b) Encoding the symbols labelling the arcs gives a code length of $\log_2 \binom{V}{a_i}$ as this set of symbols is some selection of $a_i$ symbols from the alphabet set of $V$ symbols.
(c) Encoding the destination states from state $i$, gives a code length of $a_i \log_2 S$ bits as the destination can be one of the $S$ states.
(d) A correction in the above coding scheme is obtained by subtracting $\log_2(S-1)!$ from the total code length. This happens because the above coding scheme permits $(S-1)!$ different, equal length, descriptions of the same FSM.

Using the above coding scheme, the length of description of the machine constructed in Figure 1 is shown in Table 1. The first part of the total code-length that encodes the description of the machine is:

$$\begin{aligned} Code - Length(H) = \sum_{i=1}^{S} \log_2 \binom{V}{a_i} + M \log_2 S \\ + S \log_2 V - \log_2(S-1)! \end{aligned} \qquad (2)$$

TABLE I
DESCRIPTION LENGTH OF FSM FROM FIGURE 1 ACCEPTING L

| *State* | $a_s$ | *Cost* | *Label(s)* | *Cost* | *Dest.(s)* | *Cost* |
|---|---|---|---|---|---|---|
| 0 | 2 | $\log_2 V$ | $(C, B)$ | $\log_2 \binom{V}{2}$ | (1,7) | 2$\log_2 S$ |
| 1 | 2 | $\log_2 V$ | $(A, B)$ | $\log_2 \binom{V}{2}$ | (2,15) | 2$\log_2 S$ |
| 2 | 2 | $\log_2 V$ | $(B, A)$ | $\log_2 \binom{V}{2}$ | (3,4) | 2$\log_2 S$ |
| 3 | 1 | $\log_2 V$ | $(/)$ | $\log_2 \binom{V}{1}$ | (0) | $\log_2 S$ |
| 4 | 2 | $\log_2 V$ | $(A, B)$ | $\log_2 \binom{V}{2}$ | (5,12) | 2$\log_2 S$ |
| 5 | 1 | $\log_2 V$ | $(B)$ | $\log_2 \binom{V}{1}$ | (6) | $\log_2 S$ |
| 6 | 1 | $\log_2 V$ | $(/)$ | $\log_2 \binom{V}{1}$ | (0) | $\log_2 S$ |
| 7 | 1 | $\log_2 V$ | $(B)$ | $\log_2 \binom{V}{1}$ | (8) | $\log_2 S$ |
| 8 | 2 | $\log_2 V$ | $(A, B)$ | $\log_2 \binom{V}{2}$ | (9,14) | 2$\log_2 S$ |
| 9 | 2 | $\log_2 V$ | $(A, B)$ | $\log_2 \binom{V}{2}$ | (10,13) | 2$\log_2 S$ |
| 10 | 1 | $\log_2 V$ | $(B)$ | $\log_2 \binom{V}{1}$ | (11) | $\log_2 S$ |
| 11 | 1 | $\log_2 V$ | $(/)$ | $\log_2 \binom{V}{1}$ | (0) | $\log_2 S$ |
| 12 | 1 | $\log_2 V$ | $(/)$ | $\log_2 \binom{V}{1}$ | (0) | $\log_2 S$ |
| 13 | 1 | $\log_2 V$ | $(/)$ | $\log_2 \binom{V}{1}$ | (0) | $\log_2 S$ |
| 14 | 1 | $\log_2 V$ | $(/)$ | $\log_2 \binom{V}{1}$ | (0) | $\log_2 S$ |
| 15 | 1 | $\log_2 V$ | $(/)$ | $\log_2 \binom{V}{1}$ | (0) | $\log_2 S$ |

#### 2) Encoding the Transitional Probabilities of PFSA:

Having calculated the Code-Length for the discrete structure of the FSM above, next comes encoding the transition probabilities. For each state, the code length for the transition probability distribution over the transition arcs is calculated. The distribution appears to be multinomial and we assume a uniform prior over possible set of probabilities [3].

For each state $i$, if $t_i$ represents the number of times state $i$ is visited and $n_{ij}$ represents the frequency of the $j^{th}$ arc from

$i^{th}$ state, then the probability of all the transitions from state $i$ would be $\frac{(V-1)!\Pi(n_{ij}!)}{(t_i+V-1)!}$, using multinomial distribution for uniform prior [3].

Therefore the code length needed to encode all transitions out of some state $i$ is $\log_2 \frac{(t_i+V-1)!}{(V-1)!\Pi(n_{ij}!)}$ and for all the $S$ states, the code length is as follows.

$$Code - Length(D|H) = \sum_{i=1}^{S} \log_2 \frac{(t_i+V-1)!}{(V-1)!\Pi(n_{ij}!)} \quad (3)$$

**Total Two-Part Code-Length for Machine $H$**

The total code-length obtained for Machine $H$ is Code-Length$(H)$ + Code-Length$(D|H)$, which is approximately equal to the following, from expression (1) and equations (2) and (3).

Total Two-Part Code-Length =

$$\sum_{i=1}^{S} \left\{ \log_2 \frac{(t_i+V-1)!}{(V-1)!\Pi(n_{ij}!)} + \log_2 \binom{V}{a_i} \right\} \quad (4)$$
$$+M\log_2 S + S\log_2 V - \log_2 (S-1)!$$

If it is desired to treat the probabilities as an essential feature of the inferred model, a small correction (from [2, sec. 5.2.13]) can be added to the above expression, as detailed in [2, sec. 7.1.6, p313], given by approximately $(\frac{1}{2}(\pi(a_i-1))-0.4)$ for each state.

## IV. Induction of a PFSA using MML

This section is concerned with finding the best PFSA using the information-theoretic metric MML. The problem of inferring the best such Automaton by enumeration is computationally intractable since for a given sequence of strings the number of possible automata with $n$ states is exponential in $n$. Therefore an approach that generates a near optimal automaton fairly quickly is used. In order to make the problem tractable, we trade solution optimality versus time.

The method that has been used along these lines is a greedy method and it begins the induction process by considering the input in the form of a Prefix Tree Acceptor. The node pairs are merged in stages satisfying the constraint that merging remains deterministic as far as transitions on input symbols are concerned and the code-length of the new machine is smaller than the code-length of the machine before merge.

In the first stage of the merge process, the final states of the PFSA are merged. The merging process might result in more than one final state if the merging does not produce a machine with least code-length. Applying the above kind of merge for the machine in Figure 2, the following PFSA in Figure 3 is obtained. The code-length is calculated using the formula derived in Section III. The intermediate steps have been omitted and finally the machine after having performed the merge on all final states is constructed.

In the second stage of the merge process, a list of states that are connected to the final states is found. Merging on the above list is applied and if the code-length improves the

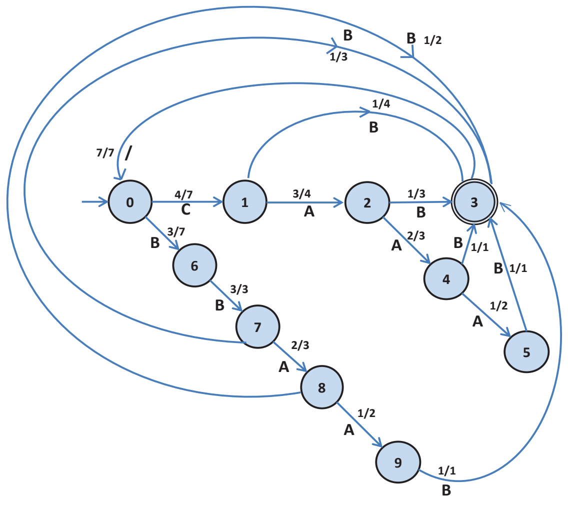

Fig. 3: PFSA with Final states Merged

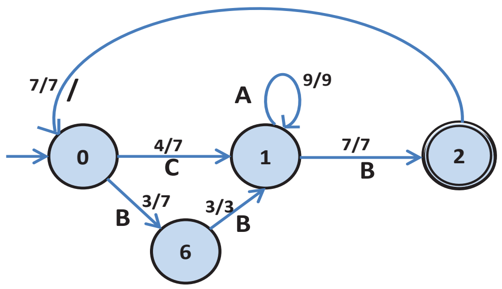

Fig. 4: PFSA with states merged in Stage 2

merge is considered otherwise rejected. That is, the set $K = \{q_1, q_2, ..., q_n\}$ is a subset of states in $Q$ such that there is an input from $\sum$ which directly leads to one of the final states in $F$. We loop through the above process until a machine with least code-length is found. Figure 4 shows the MML machine obtained after applying sequence of merges in stage 2.

In the third stage of the merge process, pair of states are merged where one state is the final state and the other state is the state connected to it. We will loop until we reach a machine whose code-length is the minimum. This machine can now be termed as the MML machine. This kind of stage-wise merging is more systematic as opposed to the random merge pairs in the Beam Search Algorithm [10]. Merging states in stage 3 also generates the same machine in Figure 4.

## V. Experimental Setup and Results

An experiment on one of the six publicly available Enron spam datasets was conducted by building spam and non-spam models. The spam model was constructed from the spam class of mails which is an induced PFSA of the spam keywords detected in the collection of spam mails. The PFSA was induced using the induction procedure described in section IV. Similarly we constructed the non-spam model from non-spam class of mails. The theory was tested on 200 mails on the

Enron spam dataset with equal collection of spam and non-spam mails. The mails in the dataset were mixed randomly and the total random collection was divided into 10 groups of 20 mails each. In general for k groups, the following training and testing procedure was followed.

1. For i = 1 to k-1
2. if (i<k)
3. Incrementally build the spam and non-spam model for the number of mails upto the ith group.
4. Test the mails in (i+1)th group with the model built in step 2
5. else
6. Test the mails in group 1 for the last batch trained.
7. End For

Table II summarizes the code-lengths of the MML machines for both spam and non-spam models in bits for each group along with spam to non-spam ratio in each group. The code-lengths are represented in thousands of bits or Kilo bits.

To test the group of mails with the models built, the target mail was input to both spam and non-spam models. The model that generates least increase in the code-length is more likely to have generated the message and thus becomes the classification class for the new mail. Basically, the MML spam classifier classifies a mail according to the following steps.

1. Extract the spam keywords using Bit Parallel String Matching Algorithm [5]. Let the words extracted in the new mail $m$ be denoted as $\{t1, t2, ..., tn\}$.
2. Calculate the increase in code-length for each of the $n$ terms extracted in the new mail $m$. This is calculated in the following manner:
   Let $L(spam)$ and $L(non-spam)$ denote the code-lengths for the spam and non-spam models respectively. These models are the MML inferred models and the code-length is calculated using the expression derived in Section III. $L_m(spam)$ and $L_m(non-spam)$ denote the new code-lengths when each of the $n$ terms is input to both the models. At this moment we don't re-infer the model but only calculate the increase in code-lengths as a result of adding them to already inferred models. The increase in code-lengths is then calculated as
   $\Delta_m(spam) = L_m(spam) - L(spam)$
   $\Delta_m(non-spam) = L_m(non-spam) - L(non-spam)$
3. If $\Delta_m(spam) < \Delta_m(non-spam)$, then $m$ is classified as spam; otherwise $m$ is classified as non-spam.

In the evaluation procedure, spam recall ($SR$), non-spam recall ($NSR$), spam precision ($SP$) and non-spam precision ($NSP$) were used as the measures of performance evaluation. $TP$ denotes true positives which is equal to the number of mails correctly classified as spam, $TN$ denotes true negatives which is equal to the number of mails correctly classified as non-spam, $FP$ denotes number of non-spam mails misclassified as spam and $FN$ denotes number of spam mails misclassified as non-spam. $SR$ is calculated as $\frac{TP}{TP+FN}$, $NSR$ is calculated as $\frac{TN}{TN+FP}$, $SP$ is calculated as $\frac{TP}{TP+FP}$ and $NSP$ is calculated as $\frac{TN}{TN+FN}$ [12]. Weighted accuracy rate $WAcc$ and weighted error rate $WErr$ have also been used as measures for cost sensitive evaluation because precision and recall do not takes into account the cost of misclassification done. The penalty for classifying a non-spam mail as spam is more severe than letting a spam mail pass the filter. Therefore Androutspoulos et al. [13] introduced these cost sensitive measurements and they are defined as $WAcc = \frac{\lambda.TN+TP}{\lambda.N_l+N_s}$ and $WErr = \frac{\lambda.FP+FN}{\lambda.N_l+N_s}$ where $N_l$ and $N_s$ are the number of spam and non-spam messages respectively.

Three different values of $\lambda$ :1, 9 and 999 were introduced by Androutspoulos et al. [13]. A value of $\lambda$ equal to 1 denotes a scenario where classifying a non-spam mail as spam and classifying a spam mail as non-spam are equally penalized. Value of $\lambda$ equal to 9 or 999 denotes a scenario where classifying a non-spam message as spam is 9 or 999 times more severe. In our experiments the value of $\lambda$ as 1 has been considered. Tables III show precision, recall, weighted accuracy rate and weighted error rate for each group in the Enron spam dataset.

Summarizing the results obtained by averaging the results obtained for each group for each of the performance parameters, 74.80% spam recall, 94.89%, 94.23% spam precision, 88.83% non-spam precision, 93.00% weighted accuracy and 10.97% weighted error rate, was obtained.

## VI. Conclusion

In this paper the MML-based induction approach to infer PFSAs for spam and non-spam classes of mails was presented. The induction approach used generates a near optimal PFSA fairly quickly as generating PFSA quickly is critical to the application in concern. With the PFSA inferred by merging pairs of states in stages, the spam and non-spam models are built with the known classes from the dataset. MML is used again in the classification process, where the model that minimally increases the code length becomes the classification class for the new mail.

An experiment on 200 mails of both classes with equal spam and non-spam proportion on the well-known publicly available spam dataset was conducted. This size of learning dataset is too small at the moment to validate our theory. This is just a preliminary testing of the theory of induction and classification process using MML and we attained satisfactory results. With the increase in size of the learning set, it is expected that the classification accuracy will improve because the trained models will now represent the entire data.

We are conducting the experiments on all the six Enron spam datasets considering the complete collection of mails. We will also consider other publicly known datasets such as Spam-Assign to validate our theory. Comparison with other well-known established spam filters based on statistical data compression techniques such as Minimum Description Length (MDL) and other machine learning techniques such as Naive Bayesian Classifiers, will also be considered in future. The techniques mentioned above are known to report better classification accuracy at the moment. We believe that accuracy reported by using the MML classifier will atleast be competitive against the established spam filters because of the strong technical strength of MML. The relationship

TABLE II
Code-Lengths (Kilobits) for Spam and Non-Spam models from training data in Enron-1 spam dataset

| no. of mails | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 | 180 | 200 |
|---|---|---|---|---|---|---|---|---|---|---|
| spam model | 1.73 | 1.87 | 3.70 | 5.13 | 8.97 | 9.81 | 10.56 | 10.98 | 11.34 | 11.90 |
| non-spam model | 1.33 | 2.46 | 3.83 | 4.51 | 4.81 | 4.98 | 5.12 | 5.74 | 5.81 | 5.52 |
| spam:non-spam ratio | 4:16 | 3:17 | 8:12 | 6:14 | 7:13 | 7:13 | 18:20 | 19:10 | 19:10 | 4:16 |

TABLE III
Classification accuracy in test data using Enron-1 training model from Table II

| | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 | 180 | 200 |
|---|---|---|---|---|---|---|---|---|---|---|
| $SR(\%)$ | 33.00 | 75.00 | 62.50 | 50.00 | 85.70 | 100.00 | 89.00 | 91.67 | 86.66 | 75.00 |
| $NSR(\%)$ | 100.00 | 100.00 | 100.00 | 100.00 | 92.30 | 87.50 | 75.00 | 100.00 | 100.00 | 94.11 |
| $SP(\%)$ | 100.00 | 100.00 | 100.00 | 100.00 | 85.70 | 87.50 | 94.11 | 100.00 | 100.00 | 75.00 |
| $NSP(\%)$ | 89.40 | 93.50 | 80.00 | 85.00 | 92.30 | 100.00 | 50.00 | 100.00 | 100.00 | 94.11 |
| $WAcc(\%)$ | 90.00 | 90.00 | 85.00 | 100.00 | 90.00 | 95.00 | 90.00 | 100.00 | 100.00 | 90.00 |
| $WErr(\%)$ | 10.50 | 5.00 | 15.00 | 15.00 | 10.00 | 5.00 | 15.00 | 10.00 | 14.28 | 10.00 |

between MML and (two-part) Solomonoff-Kolmogorov complexity [6][2, chap. 2][14, sec. 1.2] gives it the expressibility of any language that can be expressed by a universal Turing machine (UTM), with accompanying convergence and statistical consistency results [15][16, p241][2, chap. 3.4.5, pp. 190-191][17] and corresponding conjectures [18, p 93] [6, p 282] [19, sec. 0.2.5] [20, p 945]. This is testament to MML's being resistant both to noise and to (model misspecification or) incorrect distributional assumptions. A range of applications of MML is surveyed in (e.g.) [2], [19], [20].

In the future we also wish to establish our theory with the concept of a single machine which would be space and time efficient. The single machine which has both spam final states and non-spam final states would do the classification. We hope to use MML to do that work.

## References

[1] Philip Hingston, "Inference of Regular Languages using Model Simplicity," in *Proc. 2001 Australian Computer Science Conference*, 2001, pp. 69–76.

[2] C. S. Wallace, *Statistical and Inductive Inference by Minimum Message Length*, ser. Information Science and Statistics. Spring Street, NY, USA: Springer Science and Business Media, 2005.

[3] C. S. Wallace and M. P. Georgeff, "A General Objective for Inductive Inference," *Technical Report No. 32, Department of Computer Science, Monash University, Australia*, 1983.

[4] Andrej Bratko, Gordon V. Cormack, Bogdan Filipic, Thomas R. Lynam, Blaz Zupan, "Spam Filtering using Statistical Data Compression Models," *Journal of Machine Learning Research*, vol. 7, pp. 2673–2698, 2006.

[5] Leena Salmela, J. Tarhio and J. Kytojoki, "Multiple Pattern String Matching with Q Grams," *ACM Journal of Experimental Algorithmics*, vol. 11, 2006.

[6] C. S. Wallace and D. L. Dowe, "Minimum Message Length and Kolmogorov Complexity," *The Computer Journal*, vol. 42, no. 4, pp. 270–283, 1999.

[7] A. Raman, P. Andreae and J. Patrick, "A Beam Search Algorithm for PFSA Inference," *Pattern Analysis and Applications*, vol. 1, pp. 121–129, 1998.

[8] http://www.iit.demokritos.gr/skel/i-config/.

[9] http://www.aueb.gr/users/ion/publications/.

[10] B. R. Gains, "Behaviour/Structure Transformation under Uncertainity," *International Journal of Man-Machine Studies*, vol. 8, pp. 337–365, 1976.

[11] C. S. Wallace and D. L. Dowe, "Intrinsic Classification by MML-the Snob Program," in *Proc. Seventh Australian Joint Conf. Artificial Intelligence*. World Scientific, 1994, pp. 37–44.

[12] Vangelis Metsis, Ion Androutsopoulos, Georgis Paliouras, "Spam Filtering with Naive Bayes - Which Naive Bayes?" in *Third Conference on Email and Anti-Spam CEAS*, 2006.

[13] 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," in *Proc. of 23rd Annual International ACM SIGIR Conference on Research and Developement in Information Retrieval*, 2000, pp. 160–167.

[14] D. L. Dowe, "Introduction to Ray Solomonoff 85th Memorial Conference," in *Proceedings of Ray Solomonoff 85th memorial conference (Algorithmic probability and friends. Bayesian prediction and artificial intelligence.) - LNAI/LNCS*, vol. 7070. Springer, 2013, pp. 1–36.

[15] A. R. Barron and T. M. Cover, "Minimum Complexity Density Estimation," *IEEE Transactions on Information Theory*, vol. 37, no. 4, pp. 1034–1054, 1991.

[16] C. S. Wallace and P. R. Freeman, "Estimation and Inference by Compact Coding," *Journal of Royal Statistical Society*, vol. 49, no. 3, pp. 240–265, 1987.

[17] C. S. Wallace and D. L. Dowe, "MML mixture modelling of Multi-state, Poisson, von Mises circular and Gaussian distributions," in *Proc. of 6th International Workshop on Artificial Intelligence and Statistics*, 1997, pp. 529–536.

[18] D. L. Dowe, R. A. Baxter, J. J. Oliver and C. S. Wallace, " Point estimation using the Kullback-Leibler Loss Function and MML," in *Proc. of 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD98), Lecture Notes in Artificial Intelligence (LNAI) 1394*, 1998, pp. 87–95.

[19] D. L. Dowe, "Foreword re C. S. Wallace," *The Computer Journal*, vol. 51, no. 5, pp. 523–560, 2008.

[20] D. L. Dowe, "MML, hybrid Bayesian network graphical models, statistical consistency, invariance and uniqueness," in *Handbook of the Philosophy of Science*, ser. Philosophy of Statistics, In P. S. Bandyopadhyay, M. R. Forster (Eds.), Ed. Elsevier, 2011, vol. 7, pp. 901–982.