Spring 2016

Analysis on ALERGIA Algorithm: Pattern Recognition by Automata Theory

Xuanyi Qi
San Jose State University

Analysis on ALERGIA Algorithm: Pattern Recognition by Automata Theory

A Thesis
Presented to
The Faculty of the Department of Computer Science
San José State University

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science

By
Xuanyi Qi
May 2016

SAN JOSE STATE UNIVERSITY

The Designated Thesis Committee Approves the Thesis Titled

Analysis on ALERGIA Algorithm: Pattern Recognition by Automata Theory

By
Xuanyi Qi

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2016

Dr. T. Y. Lin, Department of Computer Science	Date
Dr. Thomas Austin, Department of Computer Science	Date
Dr. Ronald Mak, Department of Computer Science	Date

ABSTRACT

Based on Kolmogorov Complexity, a finite set x of strings has a pattern if the set x can be output by a Turing machine of length that is less than minimum of all |x|; this Turing machine, that may not be unique, is called a pattern of the finite set of string. In order to find a pattern of a given finite set of strings (assuming such a pattern exists), the ALERGIA algorithm is used to approximate such a pattern (Turing machine) in terms of finite automata. Note that each finite automaton defines a partition on formal language Σ^*, ALERGIA algorithm can be viewed as Granular Rough Computing based approximations. Any subset of Σ^*, such as DNA, can be approximated by equivalence classes. Based on this view, this thesis analyzes and improves the ALERGIA algorithm via minimization of deterministic finite automaton. Hypothesis testing indicates that the minimization does improve the ALERGIA. So the new method will have high usability in pattern recognition/data mining.

ACKNOWLEDGEMENTS

I would like to acknowledge and extend my heartfelt appreciation to the following people who have contributed to the success of this project. I thank my project advisor Dr. T.Y. Lin, for the significant encouragement and support. Throughout this challenging experience, you have always been a tremendous mentor, who kept giving advice and guidance. Your help will keep me accompany along my future career path as well as individual fulfillment.

I would also like to thank my committee members Dr. Austin and Dr. Mak for their support and patience. I also want to thank you for the being part of this exciting journey.

I also appreciate our department for providing the necessary academic recourse required in our project. I'm also grateful for the library for making all the critical books and materials needed to learn different concepts for our project.

A special thanks to Guoxin Li, a statistical machine learning scientist who has professional knowledge in statistics, and co-worker of the director of product data science at LinkedIn. You had made great contribution to help this project break through its very difficult stage. Your knowledge perfected the result and made the final outcome beyond our expectation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Introduction

1.1 What is Machine Learning?

Machine Learning (ML) is a multidisciplinary discipline, which involves the theory of probability, statistics, approximation theory, convex analysis, algorithm complexity theory, pattern recognition, and computational learning theory in artificial intelligence.

It is a study of computer algorithms that improve and optimize a performance criterion automatically through data examples or past experience. ML is also the core of artificial intelligence, a basic method to make intelligent computers. Its application mainly uses comprehensive induction, rather than deduction.

Machine learning has been widely used in natural language processing, data mining, DNA sequence analysis, search engine and robots.

1.2. What is Pattern Recognition?

Pattern recognition is a filed of machine learning that focuses on regularities in data and recognition of patterns. Pattern recognition algorithms commonly aim to approximate a reasonable pattern for all possible input data and to perform most likely matches of input data.

There are several typical applications of pattern recognition, such as automatic speech recognition, classification of text in to several categories, automatic recognition of handwritten postal codes, and automatic recognition of images of human faces.

1.3. Why use Statistical Method?

Statistical method is a powerful way to describe data. When data size is large, it is hard for human to read and judge the results. Through statistics, we can clearly present the results and make them easier for humans to understand. The most important thing is that, by using statistics, it can help summarize and analyze the data. Through this process, we can digitalize the useful information behind the data. In this paper, since the world languages can be infinite while the performance of computers is limited, it is hard to validate the results. Therefore, in order to help us analyze the data, statistics is introduced to generate the distribution of the results.

1.4. Use of Automata theory

This thesis generates a prefix tree acceptor (PTA) from incoming sentences of a specific regular language, then applies the ALERGIA algorithm to regenerate the PTA iteratively by merging all compatible states. This process leads to a finite automaton (FA) model that contains equivalent and useless states. A finite automaton minimization algorithm is implemented to improve the ALERGIA algorithm to find the minimum finite state automatons.

2. Approximation Theory

2.1 Thoughts of Integral in Turing Machine

In pattern recognition [1], approximation theory [2] is concerned with how a pattern can get best approximated with quantitatively characterizing the errors introduced thereby.

Based on Kolmogorov Complexity, a finite set *x* of strings come from one alphabet has pattern if the set x can be output by a Turing machine of length less than minimum of all $|x|$ [3]; this Turing machine, which may not be unique, is called a pattern of the finite set *x*. We consider a formal language *L*, which contains stings from one alphabet, has a Turing machine. In thoughts of Integral, we assume *L* represent an irregular figure. In order to calculate area of *L*, it will be cut into squares as shown in Figure 1; and we can approximate this area by adding all squares considering lower boundary, which includes all squares that have no interaction with the curve in this figure, and upper boundary, which includes squares in lower boundary and all others that the curve has interaction with. Both boundaries are used as rough sets to calculate the most accurate area of this figure.

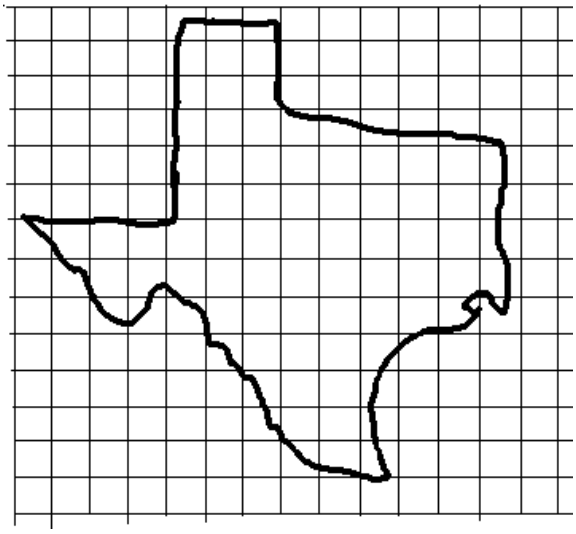

Figure 1: Integral example for area calculation

2. Thoughts of Integral in Pattern Recognition

Equivalently any Turing machine (enumerable recursive set) can be approximated by finite automata (equivalence classes [4]). Turing machine of a formal language Σ^* can be partitioned into small squares, which are equivalence classes with each can be presented as a pattern of one or more strings.

In order to present the Turing machine of a formal language Σ^*, pattern recognition is needed to get all patterns for upper and lower boundaries as rough patterns [3]. In order to find a pattern of a partitioned square, which is a given finite set of strings; assume such a pattern exists. The ALERGIA algorithm is used to approximate such a pattern (Turing machine) by stochastic deterministic finite automata [5]; note that each finite automaton defines a partition on formal language Σ^*; and such approximation is based on Granular Rough Computing (GRC). ALERGIA algorithm can be viewed as GRC based approximation theory, any subset of Σ^*, such as DNA, can be

approximated by equivalence classes. Once pattern recognition is done for all subsets from Σ^* by the ALERGIA algorithm, the Turing machine of Σ^* can be indicated by rough sets of finite automatons.

3. Deterministic Finite Automaton

3.1 Definition of Deterministic Finite Automaton

A deterministic finite automaton [5] M is represented as

$$M = (Q, \Sigma, \delta, q_0, F)$$

- Q: A finite set of states with symbols q.
- Σ : A finite set of input symbols.
- δ : A transition function that takes a state and a symbol returning a state.
- q_0: The initial state.
- F: A set of final/accepting states.

Roles:

- $q_0 \in Q$
- $F \subseteq Q$
- $Q \times \Sigma \rightarrow Q$
- $Q \times \Sigma$ is the set of 2-tuples (q, a) with $q \in Q$ and $a \in \Sigma$

3.2 Example of Deterministic Finite Automaton

A DFA example is denoted $M = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{0, 1\}$, $F = \{q_0\}$, and δ is shown in Figure 2.

States	0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Figure 2: State transition table

The transition diagram of M is shown in Figure 3.

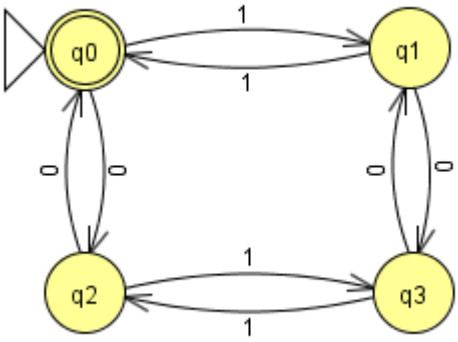

Figure 3: State transition diagram

3.3 Stochastic Deterministic Finite Automaton

A stochastic finite state automaton (SFA) [7] provides transition probabilities to each of the next states for a set of strings as the given input. Consider input symbols 0 and 1, for example, there are possibilities of two transitions δ(q, 0) or δ(q, 1) for a state q. SFA helps us in analysis and evaluation on the probabilities of the transitions to each of the states.

The probability function to calculate arbitrary transitions is given by,

$$p_{if} + \sum_{q_j \in Q} \sum_{a \in A} p_{ij}(a) = 1$$

4. ALERGIA Algorithm

4.1 Introduction of ALERGIA Algorithm

The ALERGIA algorithm [7] is a learning algorithm which specializes in merging the states of a generated automaton from a probabilistic perspective. For example, the algorithm is able to learn the Deterministic Frequency Finite Automata (DFFA) [5] and the Deterministic Probabilistic Finite Automata (DPFA) [5] of a sample set that contains duplicate strings.

When the probability of appearance of a string follows a well approximated distribution, ALERGIA has the ability to merge states when the resulting automatons are compatible with the observed frequency of strings.

Firstly, the algorithm generates a Prefix Tree Accepter (PTA) from the input strings and analyzes the relative frequency of outgoing transactions at every node. The PTA captures all of this information.

Let n_i be the number of strings arriving at node q_i.

$f_i(a)$: Number of strings following arc $\delta_i(a)$

$f_i(\#)$: Number of strings terminating at node q_i

Calculate the following probabilities:

$p_i(a) = f_i(a)/n_i$

$p_{if} = f_i(\#)/n_i$

The algorithm compares corresponding nodes (q_i, q_j). In algorithm compatible shown in Figure 4, the value of j varies from 2 to number of states in PTA and i varies from 1 to j-1.

If the probabilities of two states are equal, then they are considered compatible and their corresponding children are going to be checked recursively. If the difference between the probabilities of the two states is less than the acceptance range α, these states are considered as compatible. The formula to detect whether two states are compatible given is by:

$$\left|\frac{f}{n} - \frac{f'}{n'}\right| < \sqrt{\frac{1}{2} log \frac{2}{\alpha}} \left(\frac{1}{\sqrt{n}} + \frac{1}{\sqrt{n'}}\right)$$

Algorithm to check **COMPATIBLE** is shown in Figure 4.

```
algorithm compatible
Boolean compatible(i,j)
Input:
        i, j nodes
Output:
        Boolean
Begin
        If different(n_i,f_i(#),n_j,f_j(#))
                Return false
        End if
        Do (∀a ∈ A)
                If different(n_i,f_i(a),n_j,f_j(a))
                        Return false
                End if
                If not compatible(δ(i,a),δ(j,a))
                        Return false
                End if
        End do
        Return true
End algorithm
```

Figure 4: Algorithm compatible

Algorithm for merging in **ALERGIA** is shown in Figure 5.

```
algorithm ALERGIA
Input:
        S: sample set of strings
        α: 1 – confidence level
Output:
        Stochastic DFA
Begin
        A = stochastic prefix tree acceptor from S
        Do (for j = successor(first node(A)  to last node(A))
                Do (for i = firstnode(A) to j)
                        If compatible(i,j)
                                Merge(A,i,j)
                                Determinize(A)
                                Exit (i loop)
                        End if
                End for
        End for
        Return A
End algorithm
```

Figure 5: Algorithm ALERGIA

4.2 Example of ALERGIA Algorithm

CT: Termination Probability Check which is P(#)/n

C0: Transition 0 check which is P(0)/n

C1: Transition 1 check which is P(1)/n

Sequence = {110, λ, λ, λ, 0, λ, 00, 00, λ, λ, λ, 10110, λ, λ, 100}

α= 0.8

Build the Prefix Tree Acceptor tree as shown in Figure 6:

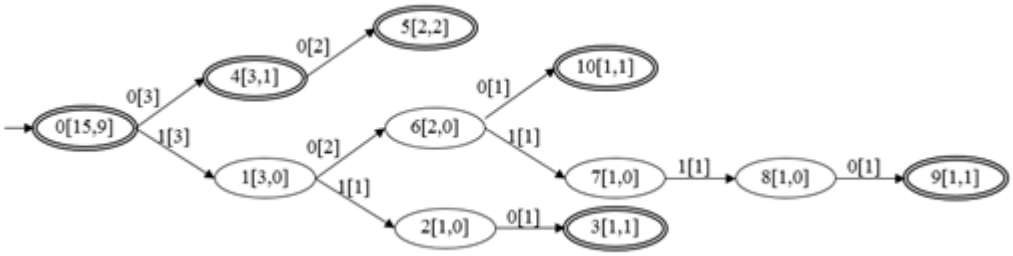

Figure 6: PTA tree of ALERGIA example

Start to merge:

1. Check node 0 and 4. Results are shown in Table 1:

CT	C0	C1
Input n: 15 Input f: 9 Input n': 3 Input f': 1 Diff is: 0.26666668 Range is: 0.5655534492421921 Merge: True	Input n: 15 Input f: 3 Input n': 3 Input f': 2 Diff is: 0.4666667 Range is: 0.5655534492421921 Merge: True	Input n: 15 Input f: 3 Input n': 3 Input f': 0 Diff is: 0.2 Range is: 0.5655534492421921 Merge: True

Table 1: ALERGIA algorithm example step1-1

Check their children by transition 0 (node 4 and 5). Results are shown in Table 2:

CT	C0	C1
Input n: 3 Input f: 1 Input n': 2 Input f': 2 Diff is: 0.6666666 Range is: 0.869403203239481 Merge: True	Input n: 3 Input f: 2 Input n': 2 Input f': 0 Diff is: 0.6666667 Range is: 0.869403203239481 Merge: True	Input n: 3 Input f: 0 Input n': 2 Input f': 0 Diff is: 0.0 Range is: 0.869403203239481 Merge: True

Table 2: ALERGIA algorithm example step1-2

Node 0 and 4 are compatible.

Break link between node 4 and parent.

Link node 0 to node 4's parent which is 0 itself.

Add node 4 to node 0.

Fold node 4's child to node 0's child

Diagram after step 1 is shown in Figure 7:

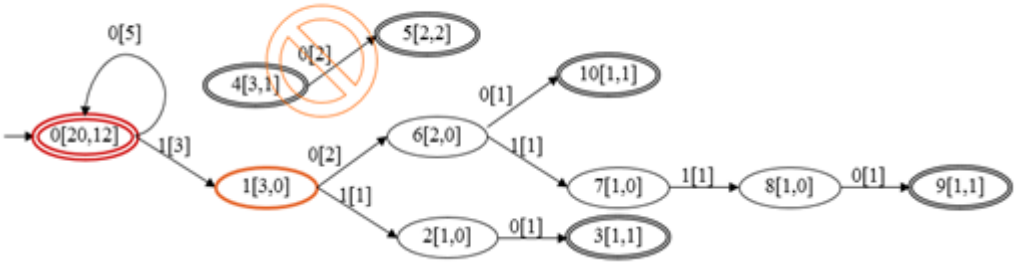

Figure 7: ALERGIA algorithm example step1 diagram

2. Check node 0 and 1. Results are shown in Table 3:

CT	C0	C1
Input n: 20 Input f: 12 Input n': 3 Input f': 0 Diff is: 0.6 Range is: 0.5421392949267191 Merge: False	CT False	CT False

Table 3: ALERGIA algorithm example step2

Node 1 and 0 are not compatible.

Set node 1 to RED.

Add node 1's children to BLUE set.

Diagram after step 2 is shown in Figure 8:

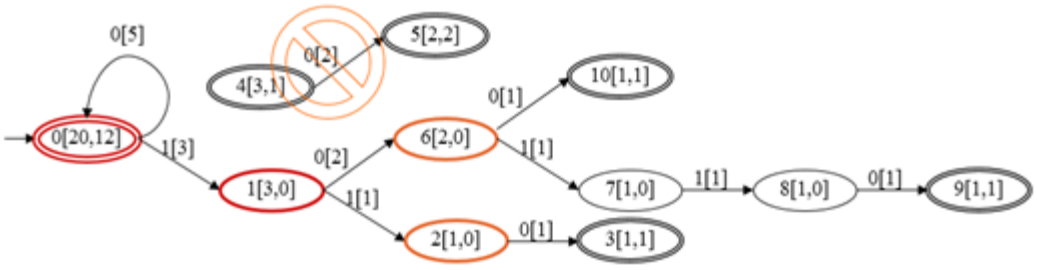

Figure 8: ALERGIA algorithm example step2 diagram

3. Check node 0 and 6. Results are shown in Table 4:

CT	C0	C1
Input n: 20 Input f: 12 Input n': 2 Input f': 0 Diff is: 0.6 Range is: 0.6299668537682291 Merge: True	Input n: 20 Input f: 5 Input n': 2 Input f': 1 Diff is: 0.25 Range is: 0.6299668537682291 Merge: True	Input n: 20 Input f: 4 Input n': 2 Input f': 1 Diff is: 0.3 Range is: 0.6299668537682291 Merge: True

Table 4: ALERGIA algorithm example step3-1

Check their children by transition 0 (node 0 and 10). Results are shown in Table 5:

CT	C0	C1
Input n: 20 Input f: 12 Input n': 1 Input f': 1 Diff is: 0.39999998 Range is: 0.8282158357555691 Merge: True	Input n: 20 Input f: 5 Input n': 1 Input f': 0 Diff is: 0.25 Range is: 0.8282158357555691 Merge: True	Input n: 20 Input f: 3 Input n': 1 Input f': 0 Diff is: 0.15 Range is: 0.8282158357555691 Merge: True

Table 5: ALERGIA algorithm example step3-2

Check their children by transition 1 (node 1 and 7). Results are shown in Table 6:

CT	C0	C1
Input n: 3 Input f: 0 Input n': 1 Input f': 0 Diff is: 0.0 Range is: 1.067652185226821 Merge: True	Input n: 3 Input f: 2 Input n': 1 Input f': 0 Diff is: 0.6666667 Range is: 1.067652185226821 Merge: True	Input n: 3 Input f: 1 Input n': 1 Input f': 1 Diff is: 0.6666666 Range is: 1.067652185226821 Merge: True

Table 6: ALERGIA algorithm example step3-3

Recursively check successors: (2 and 8). Results are shown in Table 7:

CT	C0	C1
Input n: 1 Input f: 0 Input n': 1 Input f': 0 Diff is: 0.0 Range is: 1.3537287260556712 Merge: True	Input n: 1 Input f: 1 Input n': 1 Input f': 1 Diff is: 0.0 Range is: 1.3537287260556712 Merge: True	Input n: 1 Input f: 0 Input n': 1 Input f': 0 Diff is: 0.0 Range is: 1.3537287260556712 Merge: True

Table 7: ALERGIA algorithm example step3-4

Recursively check successors: (3 and 9). Results are shown in Table 8:

CT	C0	C1
Input n: 1 Input f: 1 Input n': 1 Input f': 1 Diff is: 0.0 Range is: 1.3537287260556712 Merge: True	Input n: 1 Input f: 0 Input n': 1 Input f': 0 Diff is: 0.0 Range is: 1.3537287260556712 Merge: True	Input n: 1 Input f: 0 Input n': 1 Input f': 0 Diff is: 0.0 Range is: 1.3537287260556712 Merge: True

Table 8: ALERGIA algorithm example step3-5

Node 0 and 6 are compatible.

Break link between node 6 and parent.

Link node 0 and node 6's parent which is 1.

Add node 6 to node 0.

Fold node 4's child which is 10 to node 0's child which is 0 itself and recursively 9 to 3, 8 to 2 and 7 to 1.

Diagram after step 3 is shown in Figure 9:

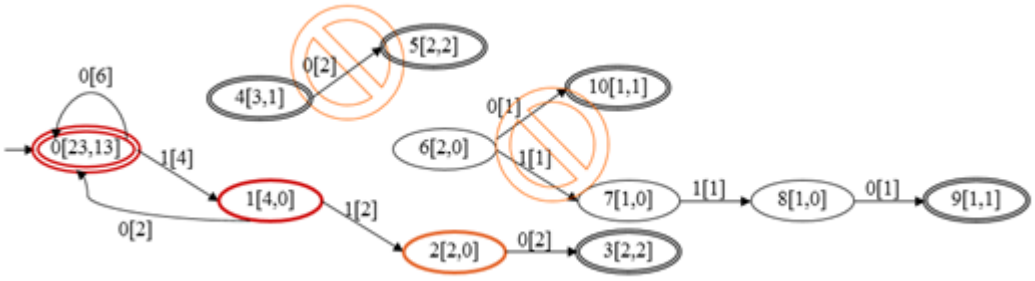

Figure 9: ALERGIA algorithm example step3 diagram

4. Check node 0 and 2. Results are shown in Table 9:

CT	C0	C1
Input n: 23 Input f: 13 Input n': 2 Input f': 0 Diff is: 0.5652174 Range is: 0.6197513570933598 Merge: True	Input n: 23 Input f: 6 Input n': 2 Input f': 2 Diff is: 0.73913044 Range is: 0.6197513570933598 Merge: False	C0 False

Table 9: ALERGIA algorithm example step4

Node 0 and 2 are not compatible.

5. Check node 1 and 2. Results are shown in Table 10:

CT	C0	C1
Input n: 4 Input f: 0 Input n': 2 Input f': 0 Diff is: 0.0 Range is: 0.8170475625544134 Merge: True	Input n: 4 Input f: 2 Input n': 2 Input f': 2 Diff is: 0.5 Range is: 0.8170475625544134 Merge: True	Input n: 4 Input f: 2 Input n': 2 Input f': 0 Diff is: 0.5 Range is: 0.8170475625544134 Merge: True

Table 10: ALERGIA algorithm example step5-1

Check their children by transition 0 (node 0 and 3). Results are shown in Table 11:

CT	C0	C1
Input n: 23 Input f: 13 Input n': 2 Input f': 2 Diff is: 0.43478262 Range is: 0.6197513570933598 Merge: True	Input n: 23 Input f: 6 Input n': 2 Input f': 0 Diff is: 0.26086956 Range is: 0.6197513570933598 Merge: True	Input n: 23 Input f: 4 Input n': 2 Input f': 0 Diff is: 0.17391305 Range is: 0.6197513570933598 Merge: True

Table 11: ALERGIA algorithm example step5-2

Node 1 and 2 are compatible.

Break link between node 2 and parent.

Link node 1 and node 1's parent which is 1 itself.

Add node 2 to node 1.

Fold node 2's child which is 3 to node 1's child which is 0.

Diagram after step 5 is shown in Figure 10:

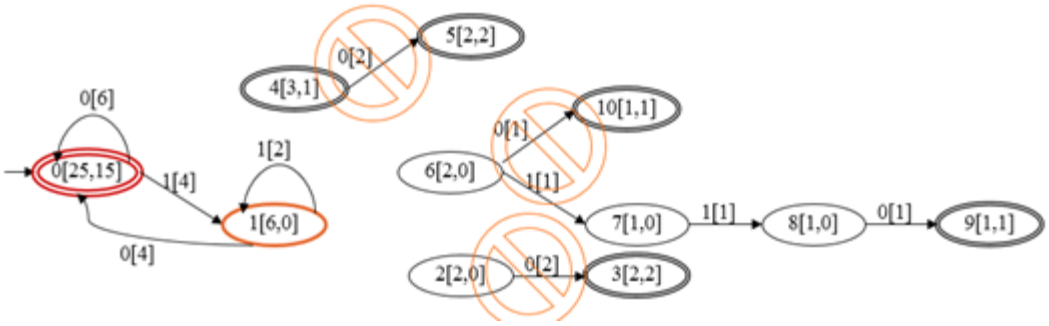

Figure 10: ALERGIA algorithm example step5 diagram

Final diagram is shown in Figure 11:

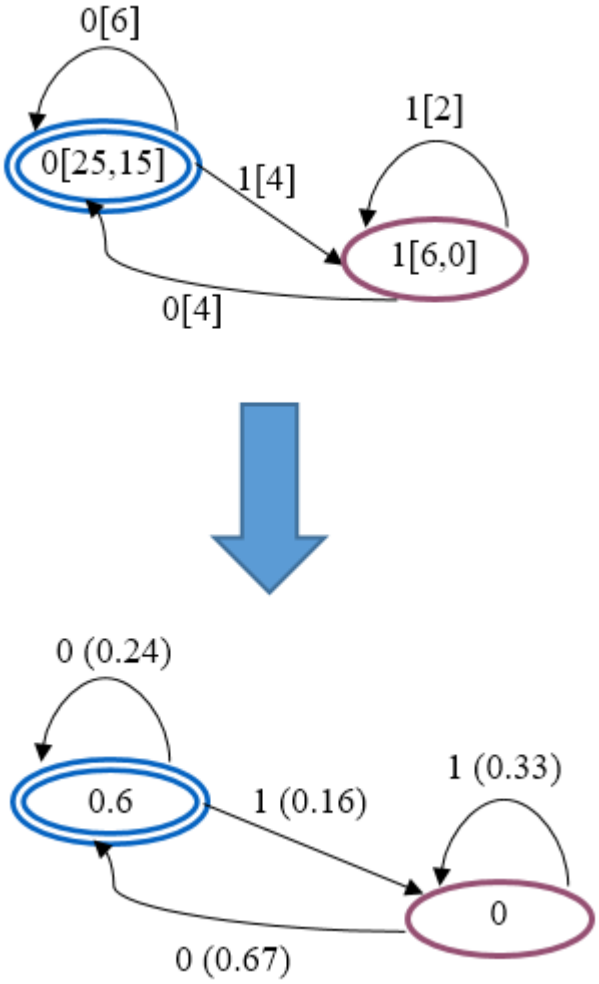

Figure 11: ALERGIA algorithm example final diagram

5. Minimization of Finite Automatons

5.1 Equivalent States

The following three statements are equivalent [4]:

- The set $L \subseteq \Sigma^*$ is accepted by some finite automaton.
- L is the union of some of the equivalence classes of a right invariant equivalence relation of finite index.
- Let equivalence relation R_L be defined by: xR_Ly if and only if for all z in Σ^*, xz is in L exactly when yz is in L. Then R_L is of finite index.

The minimum state automaton M accepting a set L is unique and is given by M'.

5.2 Proof of Equivalent States

Statement 1 to 2 [4]**:**

L: Accepted by DFA $M = (Q, \Sigma, \delta, q_0, F)$

R_M: Equivalence relation xR_Ly if and only if $\delta(q_0, x) = \delta(q_0, y)$

R_M is right invariant since, for $\forall z \subseteq \Sigma^*$, if $\delta(q_0, x) = \delta(q_0, y)$, then $\delta(q_0, xz) = \delta(q_0, yz)$.

Number of states in Q is limited so index of R_M is finite.

There is a string w such that $\delta(q_0, w)$ is in F and L is the union of some of the equivalence classes that include w.

Statement 2 to 3 [4]**:**

We show that any equivalence relation E satisfying statement 2 is a *refinement* of R_L; that is, each equivalence class of E is entirely contained in some equivalence class of R_L. Thus the index of R_L is lower or equal to that of E and so is finite.

Suppose that ***xEy***. Since ***E*** is right invariant, for $\forall z \subseteq \Sigma^*$, ***xzEyz***, thus ***yz*** is in ***L*** if and only if ***xz*** is in ***L*** and xR_Ly and hence equivalence class of ***x*** in ***E*** is contained in equivalence class of x in R_L.

So each equivalence class of E is contained within one equivalence class of R_L.

Statement 3 to 1 [4]**:**

Assume xR_Ly and $w \subseteq \Sigma^*$. For $\forall\ z \subseteq \Sigma^*$, $xwz \subseteq L$ when $ywz \subseteq L$. Since xR_Ly, for $\forall\ v \subseteq \Sigma^*$, $xv \subseteq L$ when $yv \subseteq L$. R_L need be proved as right invariant with letting $v = wz$.

Then [4]**:**

Q': Finite set of equivalence classes of R_L

[***x***]: The element of ***Q'*** containing ***x***

Since R_L is right invariant, δ'([***x***], ***a***) = [***xa***] is defined consistently. If ***y*** was chosen instead of ***x***, result would be δ'([***x***], ***a***) = [***ya***]. Since xR_Ly, $xz \subseteq L$ when $yz \subseteq L$. In particular, if $z = az'$, $xaz' \subseteq L$ when $yaz' \subseteq L$, so xaR_Lya, and [***xa***] = [***ya***].

Let $q'_0 = [\varepsilon]$, $F' = \{[x] \mid x \subseteq L\}$. Since δ' ($q'_0$, ***x***) = [***x***], $M' = (Q', \Sigma, \delta', q'_0, F')$ accepts ***L*** and thus ***x*** is in ***L***(***M'***) if and only if [***x***] is in ***F'***.

5.3 Deterministic Finite Automaton Minimization Algorithm

This algorithm finds the minimum finite state automaton ***M'***, which is equivalent to a given DFA $M = (Q, \Sigma, \delta, q_0, F)$, ***given*** by proof of theorems in section 3.1 equivalent to.

Let ≡ be the equivalence relation on the states of ***M*** such that ***p*** ≡ ***q*** if and only if for each input string ***w***, δ(***p***, ***w***) is an accepting state if and only if δ(***q***, ***w***) is an accepting

state.

If ***p*** ≡ ***q***, we say ***p*** is equivalent to ***q***. We say that ***p*** is *distinguishable* from ***q*** if there exists a ***w*** such that δ(***p***, ***w***) is in F and δ(***q***, ***w***) is not, or vice versa.

Algorithm is shown in Figure 12.

```
    Begin
1)    for p in F and q in Q - F do mark (p, q);
2)    for each pair of distinct states (p, q) in F ⨯F or (Q - F) ⨯(Q - F) do
3)     if for some input symbol w, (δ(p, w), δ(q, w)) is marked then
         begin
4)        mark (p, q);
5)        Recursively mark all unmarked pairs on the list for (p, q) and on the lists
            of other pairs that are marked at this step.
         end
       else /* no pair (δ(p, w), δ(q, w)) is marked */
6)       for all input symbols a do
7)           put (p, q) on the list for (δ(p, w), δ(q, w)) unless
             δ(p, w) = δ(q, w)
    End
```

Figure 12: Algorithm for marking pairs of inequivalent states

5.4 Example of Deterministic Finite Automaton Minimization Algorithm

Let *M* be the finite automaton of Figure 13.

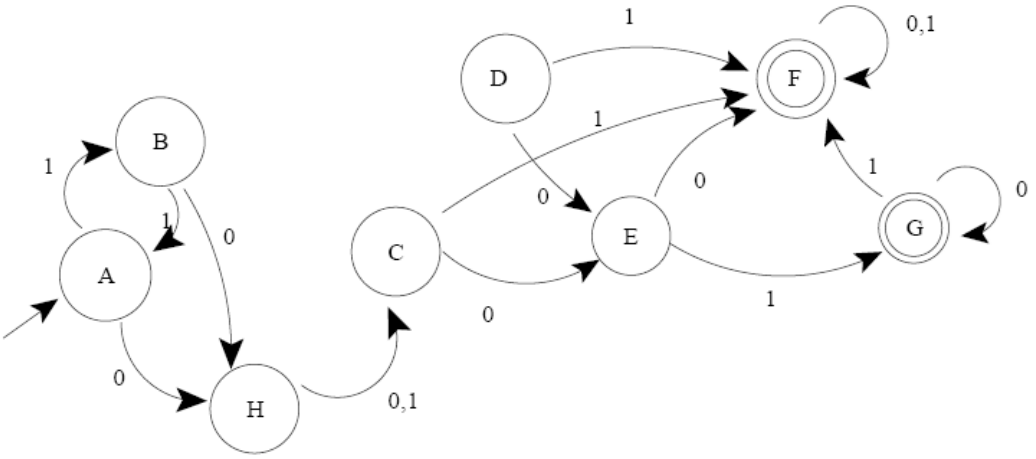

Figure 13: Finite automaton minimization example diagram

Based on algorithm in Figure 12, a simple process can be defined as:

Two states *p* and *q* are distinct if

- *p* in F and *q* not in F or vice versa, or
- for some α in Σ, δ(*p*, α) and δ(*q*, α) are distinct

For every pair of states (*p*, *q*):

- If p is final and q is not, or vice versa
- DISTINCT(*p*, *q*) = ε

Loop until no change for an iteration:

- For every pair of states (*p*, *q*) and each symbol α
    - If DISTINCT(*p*, *q*) is blank and

      DISTINCT(δ(*p*, α), δ(*q*, α)) is not blank

        - DISTINCT(*p*, *q*) = α

In Figure 14 it has constructed a table with an entry for all pairs of states for *Step 1*. A character will be placed in the table each time a pair of states that are

distinguishable is discovered.

b							
c							
d							
e							
f	ϵ	ϵ	ϵ	ϵ	ϵ		
g	ϵ	ϵ	ϵ	ϵ	ϵ		
h						ϵ	ϵ
	a	b	c	d	e	f	g

Figure 14: Finite automaton minimization example Step 1

From Figure 15 to Figure 17, minimization is done.

b							
c	1	1					
d	1	1					
e	0	0	0	0			
f	ϵ	ϵ	ϵ	ϵ	ϵ		
g	ϵ	ϵ	ϵ	ϵ	ϵ		
h			1	1	0	ϵ	ϵ
	a	b	c	d	e	f	g

Figure 15: Finite automaton minimization example Step 2

b							
c	1	1					
d	1	1					
e	0	0	0	0			
f	ϵ	ϵ	ϵ	ϵ	ϵ		
g	ϵ	ϵ	ϵ	ϵ	ϵ		
h	1	1	1	1	0	ϵ	ϵ
	a	b	c	d	e	f	g

Figure 16: Finite automaton minimization example Step 3

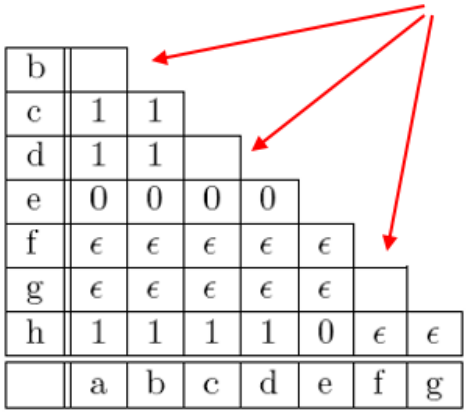

Figure 17: Finite automaton minimization example Step 4

In Figure 17 final iteration makes no changes and blank cells are equivalent pairs of states. Combine equivalent states for minimized DFA in Figure 18.

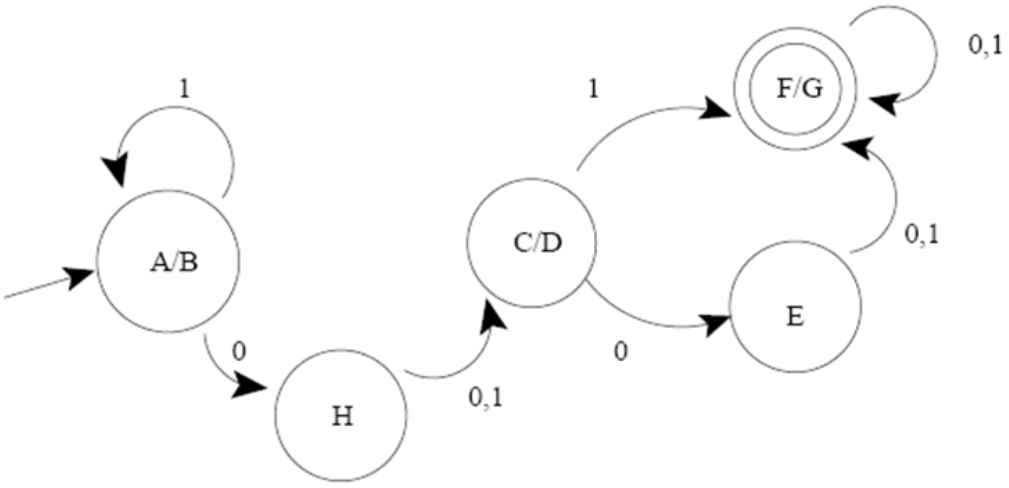

Figure 18: Minimized finite automaton example diagram

5.5 Use of Minimization in Stochastic DFA

In the ALERGIA algorithm, the output is a stochastic DFA. Figure 11 demonstrates the ALERGIA output with string frequencies, which are used to calculate the probabilities in the automaton output. Since there is no such a minimization method that can be proved to be used in deterministic finite automatons with probabilities, states in a stochastic DFA will be merged only with their string frequencies using the DFA minimization algorithm, and the probabilities will be calculated for the final automaton.

In Figure 19, a string frequency based stochastic DFA is given. Then DFA minimization algorithm is applied to find out that states S1 and S2 can be merged, and the corresponding numbers of stings are merged as:

- Number of strings go through merged states is 8+12=20.
- Number of strings terminating at merged states is 2+3=5.
- Number of strings go through transition 0 of merged states is 4+6=10.
- Number of strings go through transition 1 of merged states is 2+3=5.

Then, probabilities will be easily calculated as in Figure 19.

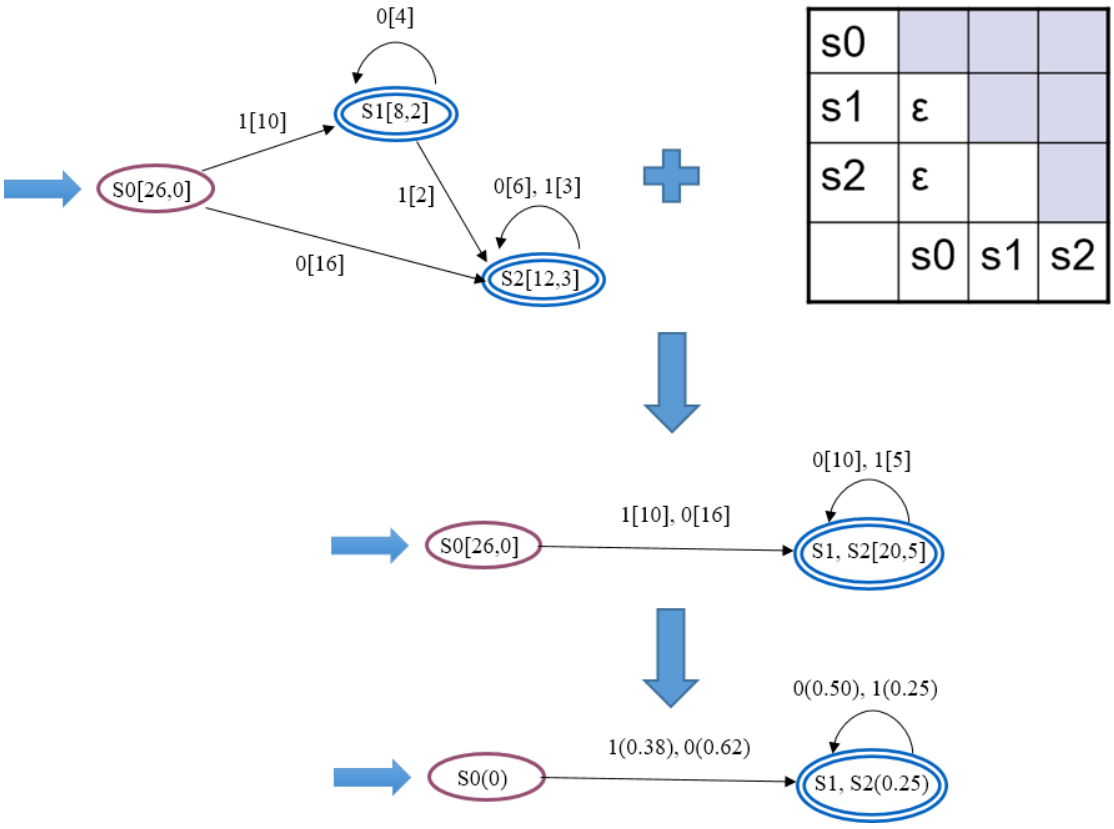

Figure 19: Example of minimization on stochastic DFA

Next chapter, we will improve ALERGIA using minimization of deterministic finite automatons algorithm on the stochastic results and analyze performance of this improvement with linear regression [8] through statistical inference.

6. Analyzing ALERGIA Algorithm in Statistical Inference

6.1 Introduction to R

R is a free software for statistical computing and graphics [9]. R is similar to S language [10], and can be considered as a different implementation of S. Although they have some difference, most code written for S can run unaltered by R.

R provides a lot different of statistical (linear and nonlinear models, statistical tests, time-series analysis, classification) and graphical functions, and is highly extensible. R provides an Open Source; it is a free software. It compiles and runs on a different of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and Mac OS X.

6.2 Exponential Distribution

As more words fit into the patterns, more complexity is expected. However, if the complexity is convergent or constant as more words fit into the patterns, we can assume that the more words are put into patterns, a more stable pattern recognition will be found.

To measure the convergence of the complexity of the PTAs, we introduce the exponential distribution [11]. The probability density function (PDF) of an exponential distribution and cumulative distribution function (PDF) are:

$$f(x;\lambda) = \lambda e^{-\lambda x}, x \geq 0$$

$$F(x;\lambda) = 1 - e^{-\lambda x}, x \geq 0$$

PDF graph example is shown in Figure 20. CDF graph example is shown in Figure 21.

The exponential distribution is normally used for describing the Poisson process. It is a particular case of the gamma distribution. The distribution is supported on the interval [0, ∞].

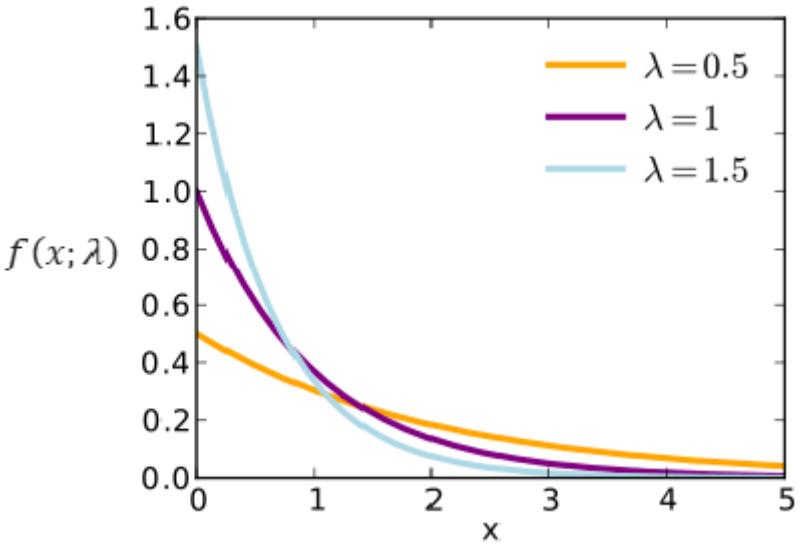

Figure 20: PDF of Exponential function

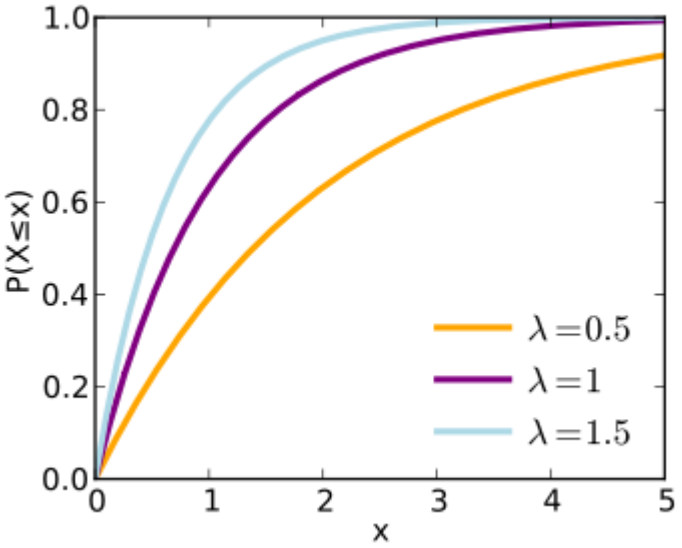

Figure 21: CDF Exponential function

According to the figures, the PDF of exponential distribution is convergent as x increase. This character can be applied to the data. If the complexity of the model is proved to follow exponential distribution, then the model has good performance on modeling language.

6.3 Analysis on Test Results in Hypothesis Test

In order to analyze improved ALERGIA algorithm, continuous training has been used on growing data that is randomly generated from a regular language for each case. As a result, stochastic DFA outcomes from ALERGIA algorithm are improved with DFA minimization to get the minimum automatons. Number of nodes in models is used to construct a mathematical function.

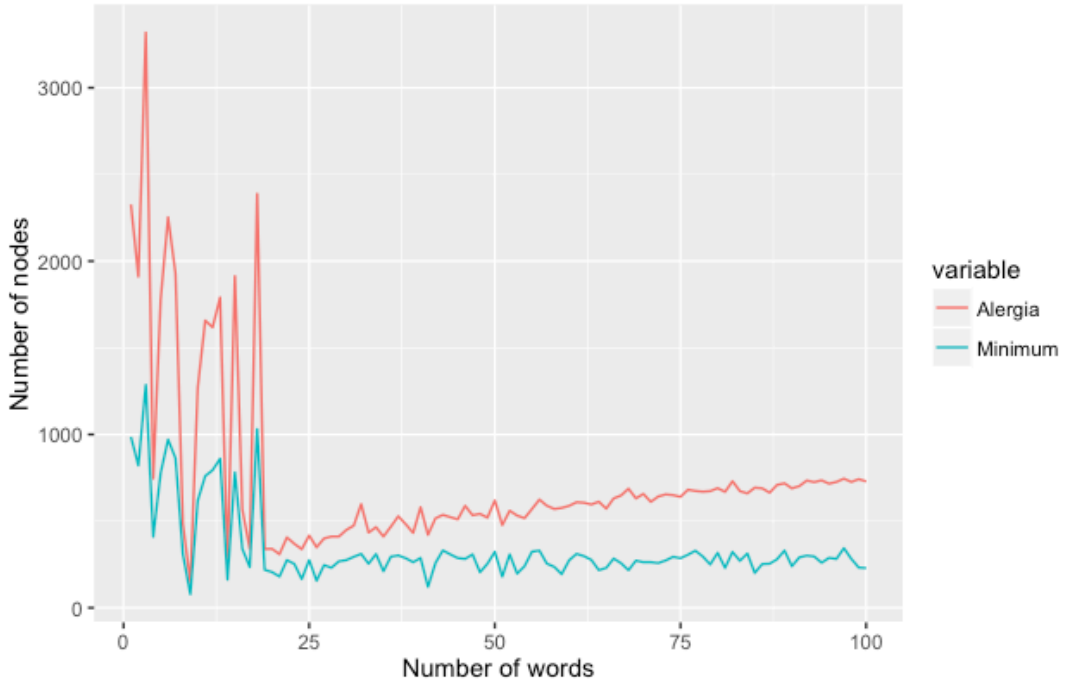

Figure 22: Program result example

Comparing two curves in Figure 22, we suppose both curves represent the same

behavior. To confirm this assumption, linear regression and hypothesis test are introduced in this section.

Linear regression is a common statistical model that is generally used in different fields and industries. Linear regression is modeling the relationship among continuous numeric variables and one or more independent variables. Simple linear regression usually refers to one explanatory variable. The linear regression model is linearly depended on the unknown parameters. It is easier to fit comparing to a model that has non-linearly relationship between the explanatory variables. Because the statistical properties of the estimators are easier to determine, linear regression was the first regression model that was applied to most cases in the real world.

Here is a detailed linear regression model. Given a data set

$$\{y_i, x_{i1} \ldots, x_{ip}\}, \quad i = 1, \ldots, n$$

n identical independent distributed (iid) variable x_i; y is a dependent variable. The relationship between y and x is modeled through a *disturbance term* or *error variable* ε_i, a random variable that the model couldn't catch, which is usually treated as noise for the linear relationship. Thus, the model can be written as:

$$y_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

Where $^{\mathrm{T}}$ denotes the transpose, and $\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}$ is the inner product between vectors $\boldsymbol{x}_i$ and $\boldsymbol{\beta}$.

A statistical hypothesis test [12] is a method of statistical inference. Usually it is used to compare two statistical data sets, or the sampling distribution and the true

distribution. A hypothesis test is to compare an alternative hypothesis to an idealized null hypothesis, assuming no relationship between two data sets exists. The purpose of hypothesis tests is to determine whether to reject the null hypothesis for a pre-specified level of significance, or to accept the null hypothesis, based on the test outcomes. The comparison is decided by statistically significance. If the statistical metric of the relationship between the two data sets reaches the threshold probability (the significance level), the alternative hypothesis should be accepted. The process of distinguishing the null hypothesis from the alternative hypothesis is aided by identifying two conceptual types of errors (type 1 & type 2). Type 1 error (significant level, alpha) is more important and cared, while type 2 is prefixed; usually type 1 error is 0.01 or 0.05. In the statistics literature, statistical hypothesis testing plays a fundamental role. The usual line of reasoning is as following:

1. Set up the research with the truth of the hypothesis unknown.
2. The first step is to set the null and alternative hypotheses. If the hypotheses were misunderstood, it would mislead the rest of the processes.
3. The second step is to consider the statistical assumptions that are applied in the hypothesis testing. Assume the data observed from the sampling distribution is independent.
4. Select the appropriate testing, such as normal test, student T test [13], and F test [14].
5. Derive the distribution of the test statistic under the null hypothesis, such as Student's T-distribution [15] or normal distribution.
6. Select a significance level (alpha). It is a threshold; if the test statistics is below the threshold, the null hypothesis should be rejected.

7. Compute from the observations to observe test statistic.

8. Decide either to reject the null hypothesis, or to accept null hypothesis.

The decision rule is to reject the null hypothesis H_0 if the observed test statistics is in the critical region, and to accept the hypothesis otherwise.

How could we apply linear model to fit nonlinear model? Here we do the log transformation for the exponential distribution.

$$\boldsymbol{f}(x;\lambda) = \lambda e^{-\lambda x}$$

$$\mathrm{Log}\big(f(x;\lambda)\big) = \log(\lambda) + \log\big(e^{-\lambda x}\big)$$

$$= log(\lambda) - \lambda x$$

After logging the exponential distribution, it transfers to the linear function respect to lambda. So we do log transformation to our data, and fit it to linear model; here are the results that we get.

In order to get more convincing results, regular languages with more stars in their regular expressions are selected to keep low repeatability of all test files with different numbers of strings for every regular language. For instance, the regular expression of test case 1 is 0*10*(0+1)*. Each star comes as a random number from 0 to 20, and the total number of strings is $21*21*2^{21}$ =924,844,032. Since the number of strings in all test files is from 500 to 50,000, which is way less than 924,844,032, the repeatability of strings in test files will be extremely low. Then Patterns will be recognized from subsets samples of strings from their corresponding regular language.

Test Case 1:

Regular language : 0*10*(0+1)*

Date result in curve graphic as shown in Figure 23:

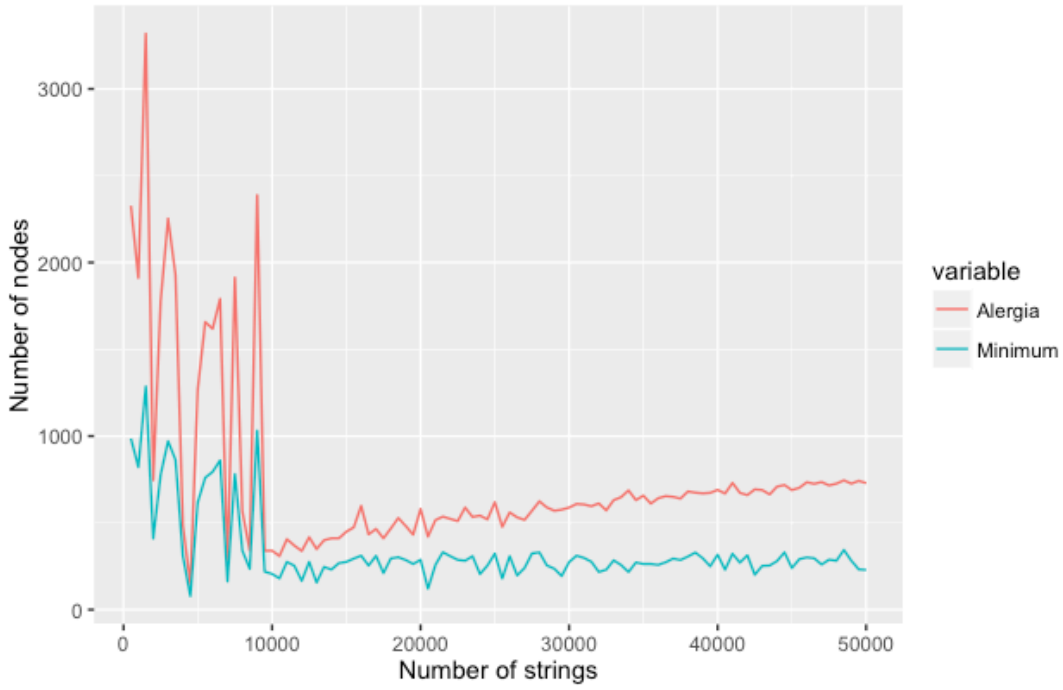

Figure 23: Test case1 R program result

Analysis results in R as shown in Table 12 and Table 13:

Summary Table

Attribute	ALERGIA	Improved ALERGIA
Formula	$log(y) = log(\lambda) - \lambda x$	$log(y) = log(\lambda) - \lambda x$
Coefficients Estimate: Intercept, x	6.540e+00, -3.219e-06	5.964772, -0.005018
Coefficients Std Error: Intercept, x	1.007e-01, 3.461e-06	0.088490, 0.001507
Coefficients t value: Intercept, x	64.97, -0.93	67.41, -3.33
Pr(>|t|)	< 2e-16, 0.355	< 2e-16, 0.00122

Table 12: Test case1 R Summary Table

Analysis of Variance Table

Attribute	ALERGIA	Improved ALERGIA
Degree of freedom of x	1	1
Sum of the square	0.2159	2.1595
Sum of the mean square	0.21586	2.15949
F value	0.865	11.092
Pr(>F)	0.3546	0.001219

Table 13: Test case1 R Analysis of Variance Table

Summary:

From the summary table, we get the estimate coefficient of the linear regression is y=5.964772-0.005018x. The p-value from the T test for both parameters are under the 0.01, so the two parameters are both significant at 0.01 significant level, which means parameters exist. Also from the ANOVA table, it shows that the p-value from the F test for the whole equation is 0.001219, which means it is significant at 0.01 significant level, the linear equation exists. But compare to the result from ALERGIA, the F test for the whole-equation is 0.3546, it is not significant at 0.01 significant level, so we can conclude that ALERGIA is not follow the exponential distribution. This is sufficient to conclude that ALERGIA algorithm performs better with improvement using finite automaton minimization in this case.

Test Case 2:
Regular language : (11+0)*(00+1)*

Date result in curve graphic as shown in Figure 24:

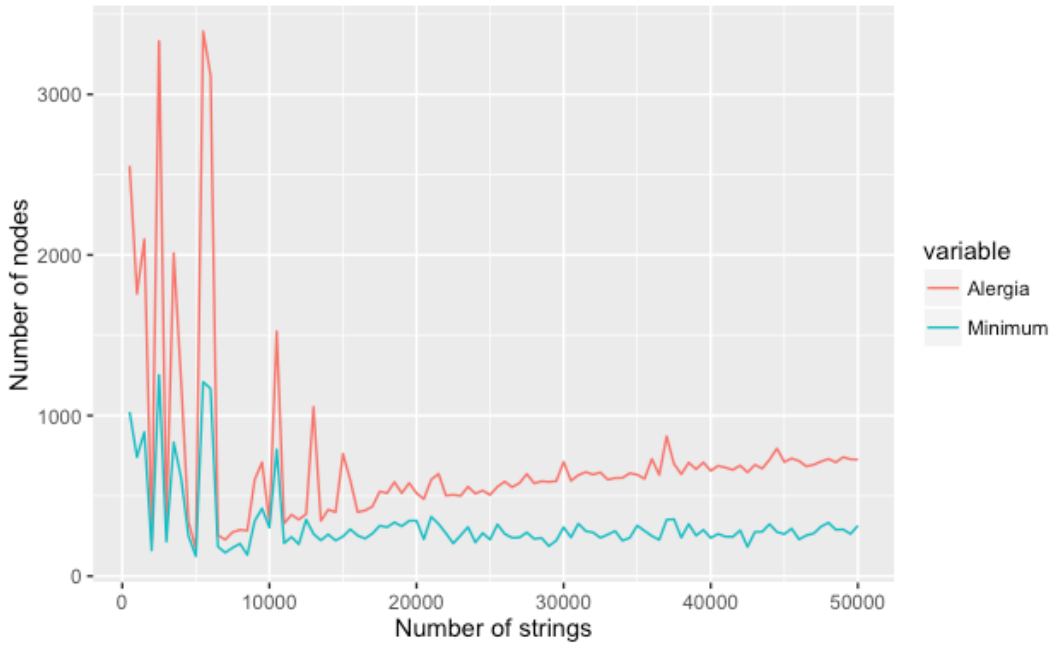

Figure 24: Test case2 R program result

Analysis results in R as shown in Table 14 and Table 15:

Summary Table

Attribute	ALERGIA	Improved ALERGIA
Formula	$log(y) = log(\lambda) - \lambda x$	$log(y) = log(\lambda) - \lambda x$
Coefficients Estimate: Intercept, x	6.382e+00, 1.858e-06	5.842e+00, -6.471e-06
Coefficients Std Error: Intercept, x	1.055e-01, 3.628e-06	8.549e-02, 2.911e-06
Coefficients t value: Intercept, x	60.479, 0.512	68.337, -2.223
Pr(>|t|)	< 2e-16, 0.61	< 2e-16, 0.0285

Table 14: Test case2 R Summary Table

Analysis of Variance Table

Attribute	ALERGIA	Improved ALERGIA
Degree of freedom of x	1	1
Sum of the square	0.0719	0.8977
Sum of the mean square	0.071889	0.89771
F value	0.2622	4.9404
Pr(>F)	0.6098	0.02851

Table 15: Test case2 R Analysis of Variance Table

Summary:

From the summary table, we get the estimate coefficient of the linear regression is y=5.842-6.471e-06x. The p-value from the T test for both parameters are under the 0.05, so the two parameters are both significant at 0.05 significant level, which means parameters exist. Also from the ANOVA table, it shows that the p-value from the F test for the whole equation is 0.0285, which means it is significant at 0.05 significant level, the linear equation exists. But compare to the result from ALERGIA, the F test for the whole-equation is 0.6098, it is not significant at 0.01 significant level, so we can conclude that ALERGIA is not follow the exponential distribution. This is sufficient to conclude that ALERGIA algorithm performs better with improvement using finite automaton minimization in this case.

Test Case 3:

Regular language : 10*(0+11)*01*

Date result in curve graphic as shown in Figure 25:

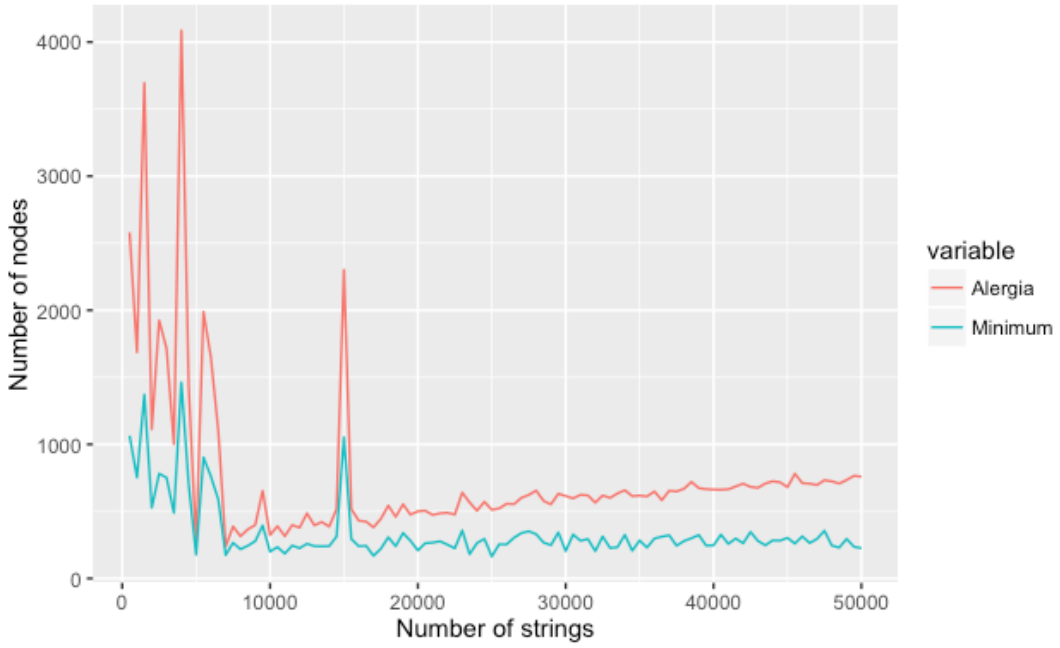

Figure 25: Test case3 R program result

Analysis results in R as shown in Table 16 and Table 17:

Summary Table

Attribute	ALERGIA	Improved ALERGIA
Formula	$log(y) = log(\lambda) - \lambda x$	$log(y) = log(\lambda) - \lambda x$
Coefficients Estimate: Intercept, x	6.565e+00, -3.818e-06	5.995e+00, -1.052e-05
Coefficients Std Error: Intercept, x	1.000e-01, 3.439e-06	8.394e-02, 2.859e-06
Coefficients t value: Intercept, x	65.64, -1.11	71.423, -3.678
Pr(>|t|)	< 2e-16, 0.27	< 2e-16, 0.000382

Table 16: Test case3 R Summary Table

Analysis of Variance Table

Attribute	ALERGIA	Improved ALERGIA
Degree of freedom of x	1	1
Sum of the square	0.3037	2.3704
Sum of the mean square	0.30367	2.37036
F value	1.2327	13.531
Pr(>F)	0.2696	0.0003818

Table 17: Test case3 R Analysis of Variance Table

Summary:

From the summary table, we get the estimate coefficient of the linear regression is y=5.995-1.052e-05x. The p-value from the T test for both parameters are under the 0.01, so the two parameters are both significant at 0.01 significant level, which means parameters exist. Also from the ANOVA table, it shows that the p-value from the F test for the whole equation is 0.0003818, which means it is significant at 0.01 significant level, the linear equation exists. But compare to the result from ALERGIA, the F test for the whole-equation is 0.2696, it is not significant at 0.01 significant level, so we can conclude that ALERGIA is not follow the exponential distribution. This is sufficient to conclude that ALERGIA algorithm performs better with improvement using finite automaton minimization in this case.

Test Case 4:

Regular language : 0+1(10)*0(01)*

Date result in curve graphic as shown in Figure 26:

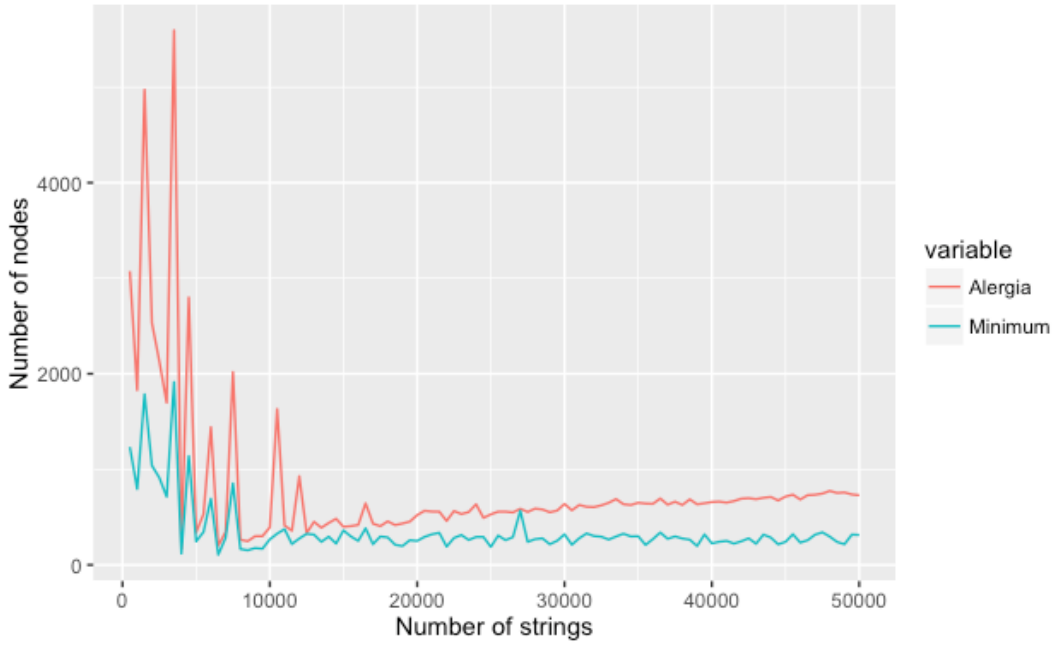

Figure 26: Test case4 R program result

Analysis results in R as shown in Table 18 and Table 19:

Summary Table

Attribute	ALERGIA	Improved ALERGIA
Formula	$log(y) = log(\lambda) - \lambda x$	$log(y) = log(\lambda) - \lambda x$
Coefficients Estimate: Intercept, x	6.541e+00, -3.279e-06	5.987e+00, -1.048e-05
Coefficients Std Error: Intercept, x	1.162e-01, 3.994e-06	9.423e-02 , 3.209e-06
Coefficients t value: Intercept, x	56.303, -0.821	63.530, -3.267
Pr(>|t|)	< 2e-16, 0.414	< 2e-16, 0.00149

Table 18: Test case4 R Summary Table

Analysis of Variance Table

Attribute	ALERGIA	Improved ALERGIA
Degree of freedom of x	1	1
Sum of the square	0.224	2.3565
Sum of the mean square	0.22398	2.35646
F value	0.6739	10.673
Pr(>F)	0.4137	0.001494

Table 19: Test case4 R Analysis of Variance Table

Summary:

From the summary table, we get the estimate coefficient of the linear regression is y=5.987-1.048e-05x. The p-value from the T test for both parameters are under the 0.01, so the two parameters are both significant at 0.01 significant level, which means parameters exist. Also from the ANOVA table, it shows that the p-value from the F test for the whole equation is 0.001494, which means it is significant at 0.01 significant level, the linear equation exists. But compare to the result from ALERGIA, the F test for the whole-equation is 0.4137, it is not significant at 0.01 significant level, so we can conclude that ALERGIA is not follow the exponential distribution. This is sufficient to conclude that ALERGIA algorithm performs better with improvement using finite automaton minimization in this case.

Test Case 5:

Regular language : 1(0+1)*0+1(01+10)*0

Date result in curve graphic as shown in Figure 27:

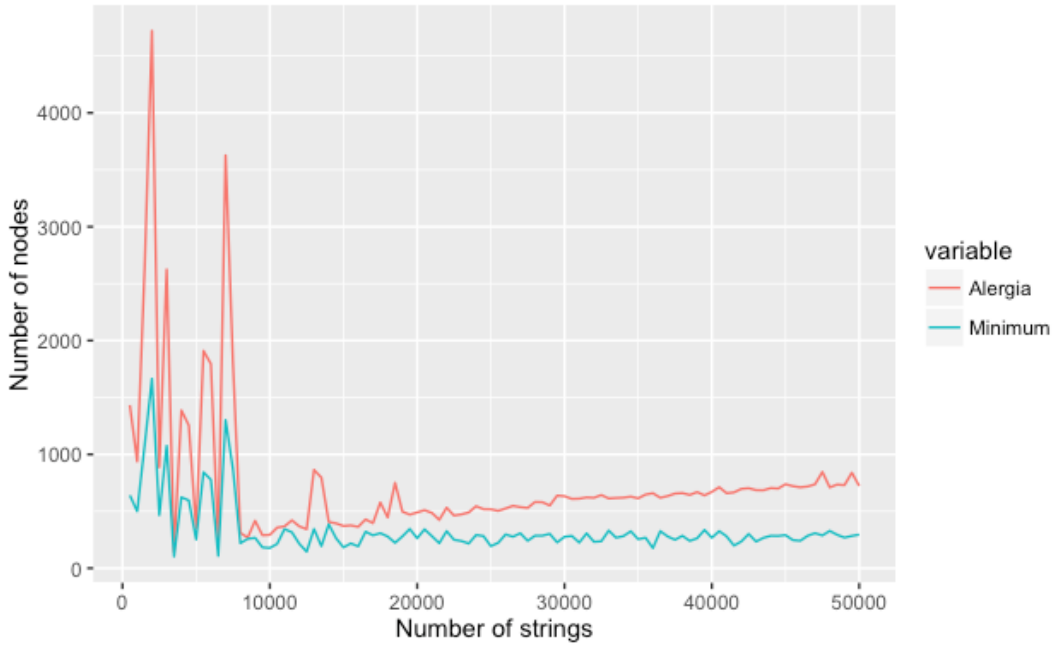

Figure 27: Test case5 R program result

Analysis results in R as shown in Table 20 and Table 21:

Summary Table

Attribute	ALERGIA	Improved ALERGIA
Formula	$log(y) = log(\lambda) - \lambda x$	$log(y) = log(\lambda) - \lambda x$
Coefficients Estimate: Intercept, x	6.412e+00, 4.395e-07	5.916e+00, -8.486e-06
Coefficients Std Error: Intercept, x	1.062e-01, 3.651e-06	8.829e-02, 3.007e-06
Coefficients t value: Intercept, x	60.39, 0.12	67.002, -2.822
Pr(>|t|)	< 2e-16, 0.904	< 2e-16, 0.00576

Table 20: Test case5 R Summary Table

Analysis of Variance Table

Attribute	ALERGIA	Improved ALERGIA
Degree of freedom of x	1	1
Sum of the square	0.004	1.5437
Sum of the mean square	0.004024	1.54371
F value	0.0145	7.965
Pr(>F)	0.9044	0.005765

Table 21: Test case5 R Analysis of Variance Table

Summary:

From the summary table, we get the estimate coefficient of the linear regression is y=5.916-8.486e-06x. The p-value from the T test for both parameters are under the 0.01, so the two parameters are both significant at 0.01 significant level, which means parameters exist. Also from the ANOVA table, it shows that the p-value from the F test for the whole equation is 0.005765, which means it is significant at 0.01 significant level, the linear equation exists. But compare to the result from ALERLGIA, the F test for the whole-equation is 0.9044, it is not significant at 0.01 significant level, so we can conclude that ALERGIA is not follow the exponential distribution. This is sufficient to conclude that ALERGIA algorithm performs better with improvement using finite automaton minimization in this case.

7. Future Work

Because only string frequencies are used in Stochastic DFA minimization improvement based on results of the ALERGIA algorithm, more research and proof on finding minimized equivalent Stochastic DFA based on probabilities will be needed.

Since the program is running on a single computer with limited memory and storage, the maximum sample size could only reach 50000 strings. In order to further the pattern recognition of an infinite regular language, a better program including secondary memory implementation, better choice of data structure, and more efficient algorithms will be needed to run on a larger scale cluster with high-speed servers.

Tools like Hadoop and Spark will also be in future blue print to compute big data sample within efficient computing model, such as map-reduce. Then test cases, which train much larger sample sizes that have more various α values in ALERGIA algorithm, can be used for a better approximation in pattern recognition with automatons on real data, such as English text, DNA sequences, etc.

8. Conclusion

We proposed a method for pattern recognition for symbolic data using automata and ALERGIA algorithm. The stochastic DFA results from ALERGIA algorithm are improved with DFA minimization with considering only string frequencies. The linear regression from exponential distribution is used in statistical inference. Hypothesis test is used to analyze and conclude that the improved ALERGIA algorithm has a better performance than the ALERGIA algorithm in pattern recognition in automatons.

In all five test cases, linear regressions from the improved ALERGIA algorithm have negative slopes, which are extremely close to 0. P values from Student's T-Test and F-Test are at either 0.01 or 0.05 significance level. It is concluded that the improved ALERGIA generates stable approximation in pattern recognition with increased data sample from a specific regular language, and may closely approximate the corresponding DFAs of the regular expressions in all test cases if the sample size is large enough. In this case, the ALERGIA algorithm is proved to have a good usability in pattern recognition for a set of strings from a formal language Σ^* for approximating its Turing machine, and can be taken advantage of in future research.

Dr. Lin [16] [17] has been researching related topic since 2005 with former student A. Yazdhankhah [18] for Master's Thesis at San Jose State University. The results seem to be promising for future research and applications.

9. References

1. Christopher Bishop: Pattern Recognition and Machine Learning (Information Science and Statistics) (2007)

2. Powell, Michael James David. *Approximation theory and methods*. Cambridge university press, 1981.

3. T. Y. Lin, “Rough Patterns in Data-Rough Sets and Foundation of Intrusion Detection Systems,” Journal of Foundation of Computer Science and Decision Support, Vol.18, No. 3-4, 1993. 225-241.

4. Hopcroft, John E. *Introduction to automata theory, languages, and computation*. Pearson Education India, 1979.

5. P. Linz, An Introduction to Formal Languages and Automata, 4th edition, Sudbury: Jones and Bartlett Publishers, 2006, pp. 38-39.

6. Tsau-Young Lin, Asmi H. Shah, "Stochastic Finite Automata for the translation of DNA to protein", *BIG-DATA*, 2014, 2014 IEEE International Conference on Big Data (Big Data), 2014 IEEE International Conference on Big Data (Big Data) 2014, pp. 1060-1067, doi:10.1109/BigData.2014.7004340

7. Carrasco, Rafael C., and José Oncina. "Learning stochastic regular grammars by means of a state merging method." *Grammatical Inference and Applications*. Springer Berlin Heidelberg, 1994. 139-152.

8. Kutner, Michael H. *Applied linear statistical models*. Vol. 4. Chicago: Irwin, 1996.

9. Ihaka, Ross, and Robert Gentleman. "R: a language for data analysis and graphics." *Journal of computational and graphical statistics* 5.3 (1996): 299-314.

10. Chambers, John M. *Programming with data: A guide to the S language*. Springer Science & Business Media, 1998.

11. Balakrishnan, K. *Exponential distribution: theory, methods and applications*. CRC press, 1996.

12. Bera AK, Premaratne G. General hypothesis testing. A companion to theoretical econometrics. 2001.

13. Owen, D. B. "The power of Student's t-test." *Journal of the American Statistical Association* 60.309 (1965): 320-333.

14. Lomax, Richard G., and Debbie L. Hahs-Vaughn. *Statistical concepts: A second course*. Routledge, 2013.

15. Forbes, Catherine, Merran Evans, Nicholas Hastings, and Brian Peacock.*Statistical distributions*. John Wiley & Sons, 2011.

16. Baliga, Priya, and T. Y. Lin. "Kolmogorov complexity based automata modeling for intrusion detection." *Granular Computing, 2005 IEEE International Conference on*. Vol. 2. IEEE, 2005.

17. T. Y. Lin, “Neighborhood Systems and Approximation in Database and Knowledge Base Systems”, Proceedings of the Fourth International Symposium on Methodologies

of Intelligent Systems, Poster Session, October 12-15, 1989, pp. 75-86
18. A. Yazdhankhah, "Discovering Pattern using Automata," M. S. thesis, San Jose State University, San Jose, 2011.

APPENDIX: Development Environment

Software Specifications	
Language	Java 1.8, R
Integrated Development Environment	Eclipse Java Mars, RStudio
Operating System	OS X on BSD kernel

Table 22: Software Specifications

Hardware Specifications	
Model	Macbook Pro (Retina, 15-inch, Mid 2015)
RAM	16 GB
CPU	Intel Core i7 vPro, 2.2 GHz

Table 23: Hardware Specifications