Automatic State Machine Induction for String Recognition

Boontee Kruatrachue, Nattachat Pantrakarn, and Kritawan Siriboon

Abstract—**One problem of generating a model to recognize any string is how to generate one that is generalized enough to accept strings with similar patterns and, at the same time, is specific enough to reject the non-target strings. This research focus on generating a model in the form of a state machine to recognize strings derived from the direction information of character's images. The state machine induction process has two steps. The first step is to generate the machine from the strings of each target character (Positive Training), and the second step is to adjust the machine to reject any other string (Negative Training). This automatic state machine induction method can also be applied with any string sequence recognition in other applications.**

Index Terms—**State Machine Induction, DFA Learning, Character Recognition.**

I. Introduction

This paper provides a novel state machine induction method for string recognition. The constraint in constructing model is to generate a model that is generalized enough to accept similar strings in the same class and is specific enough to reject any strings of other classes. The model proposed in this research is a state machine generated automatically from positive and negative training string samples.

The main advantage of our approach is that the generated models generalization can be controlled to suit the training strings. These usages of both positive and negative samples in generalization control helps in improving recognition.

Many previous researches use state machine or grammar induction to generate models from samples. Some DFA induction methods build the prefix tree acceptor from labeled sample and merge systematically equivalent states to construct the smallest DFA. These algorithms, such as Trakhtenbrot-Barzdin [1], EDSM [2], SAGE [3], or ALGERIA [4], etc., may not be suitable for recognizing strings in some specific domains. Usually, the generated models can either be over or under generalization. This is due to the algorithm trying to reduce number of states of the generated machine by preserving training pattern labels.

Our approach is more suitable for generating state machine that is not over or under generalized. In this case, the string contains direction alphabets derived from contour following of a character image. This string sequence has specific patterns that we can force in generating the state model. Hence the model generalization can be controlled.

For the directional chain code strings, there are 2 main patterns [5]. The first one, called *loop,* is the repeating of the same direction in a stroke, *d**. The second one, called *loop pair*, is the direction repetition between two adjacent directions in the chain code, *(d1*d2*)**.

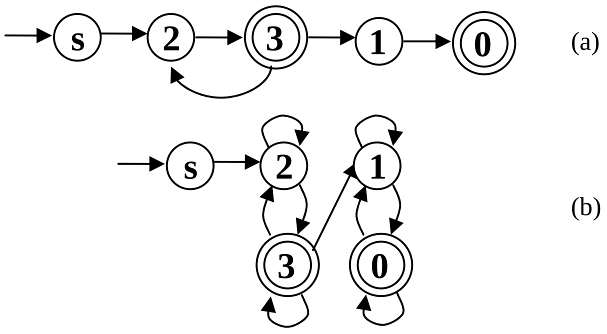


Figure 1 The models generated by other DFA induction and our algorithm.

The difference between model from other approaches and from ours is shown in Fig.1. Both models are generated from training string '2310', and '2323'. Fig.1a shows the model generated by other DFA learning algorithm that is under generalized. Fig.1b shows the suitable model from our approach. If the unknown string '23310' is tested with both models, the first one rejects, but the second one accepts. Actually, the string derived from direction is not concerned about the exact number of inputs. One input and two inputs of the same value are not different. In other DFA induction, only one repeating input can cause that string to be rejected, instead to be accepted.

In order to improve generalized control of the induction process, this research also uses position information of a stroke to decide whether to reuse existing state or to add a new state. This position information along with their length is also used in unknown string recognition.

Some preceding works use only samples in the targeted classes to generate models (positive training), such as CFG [5]

Manuscript received March 22, 2007.

Boontee Kruatrachue is Assoc. Prof. Dr. in the Department of Computer Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520 Thailand (e-mail: booontee@yahoo.com).

Nattachat Pantrakarn is a master student in the Department of Computer Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520 Thailand (e-mail: nattachatp@hotmail.com).

Kritawan Siriboon is Asst. Prof. in the Department of Computer Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520 Thailand (e-mail: kritawan@diamond.ce.kmitl.ac.th).

or HMM [6], [7]. These models can accept all strings in the classes, but may not distinguish strings from any other class. In this research, we use information from negative samples to create some transitions to trap state. This method prevents the negative strings to reach the final states, in the other word; the models can reject strings from non-targeted classes.

We begin by presenting principle knowledge and some definitions of state machine induction in Section 2. Section 3 defines the previous researches corresponded to this work. We provide an overview of the system in Section 4, and the detail of the state machine induction algorithm in Section 5. Section 6 shows how to setup the experiment and the results for Thai character recognition. Finally, the conclusion and analysis is performed in Section 7.

II. Preliminaries

In this section, we describe some definitions of state machine induction problem that is used throughout this paper.

A. Chain Code

Each printed Thai character is scanned as an image. Each image is encoded as a string, called *chain code*, by using directional information of image contour. Chain codes used in this experiment are based on an eight-way directional system as shown in Fig.2

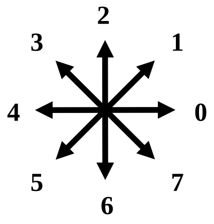
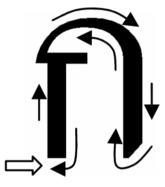


Figure 2 Chain code in 8-way directional system.

Chain code is a sequence of inputs generated from the contour following in 8-way direction of character image. Fig.3 shows the chain code derived from character ‘ก’. The string starts from the left-bottom of image, and follows the contour



```
2222222222222224442121211110
0101007007707676776676666666
6666666666555542222222222222
2223323234334344545545655565
6000006644444666666666666444
```

until meets the initial point.

Figure 3 Chain code example of character ‘ก’.

B. Deterministic Finite Automata (DFA)

The Deterministic Finite Automata or Deterministic Finite State Machine is a finite state machine which has one and only one transition to a next state for each pair of state and input symbol. DFA recognizes the set of regular languages. DFA is defined as quintuple, $M = \{\Sigma, Q, \delta, q_0, F\}$, where

- Σ is a finite input alphabets,
- Q is a finite non-empty set of states,
- $\delta : Q \times \Sigma \rightarrow Q$ is the transition function,
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is the set of final state.

A state $d_0 \in Q$ such that $\forall a \in \Sigma$, $\delta(d_0, a) = d_0$ is called *trap state*. Trap state has no transition out to other states.

C. Positive and Negative Training Set

Let the disjoint set S_+ and S_-, called *positive* and *negative training set* respectively [8], are subsets of Σ^*. S_+ is the set of any string over the alphabet Σ which is accepted by DFA M, and S_- is the set of any string over the alphabet Σ which is rejected by DFA M. Therefore, DFA M recognizes the set of strings $S = S_+ \cup S_-$ when it accepts positive training set and rejects negative training set.

D. Generalization

A machine is generalized when it accepts strings that have similar patterns to the samples. A machine is *over generalized* if it accepts too many strings that may not be in the class. If a machine is too restrict and loses the ability to accept strings that should be considered as the strings in the class, the machine is *under generalized*.

III. Related Works

There are many researches that applied automata induction approach with string recognition. Trakntenbrot and Barzdin introduced an algorithm for constructing the minimum DFA from complete labeled sample in polynomial time [1]. Lang and Pearlmutter [2] organized the Abbadingo One DFA Learning Competition to encourage work on DFA induction from sparse training data. Two algorithms from Price [2] and Juillé [3] are the winners. These algorithms are effective methods to minimize DFA from both positive and negative samples, but some transitions are too generalized or too specific for some strings because it was designed for any string in non-specific domain problem. There are other reviews in DFA learning algorithm in [8].

Hidden Markov Model (HMM) is another state model induction by using probabilistic information [6], [7]. The number of all states in model must be defined before the training stage begins. The structure of states in model cannot be controlled, while our approach can set the pattern of states to suit the strings. Moreover, HMM consumes much more time in the training step.

This research adopts from previous work that used context-free grammar (CFG) in Thai character recognition [5]. The previous research used only positive training strings to generate models using loop, loop pair, and trap loop pair policy. We represent all models as DFA instead of CFG, and adapt models to reject non-targeted strings by using negative samples. In addition, we use both the position and length information of inputs to determine appropriate state for each input.

Our approach is DFA induction method designed for string recognition derived from Thai characters. There are many advantages in this method: (1) models can both accept positive strings and reject negative strings, (2) models can be controlled to accept similar strings by using loop and loop pair from [5], (3) models use the position and length information to distinguish each input in string, and (4) models have no loop back transition which results in over generalization.

IV. System Overview

To design a system to recognize any string pattern, the goal is to generate one model for each class of patterns such that the model has to accept any strings in the targeted class, and reject any strings in the non-targeted class.

In an automatic state machine induction for Thai character recognition, there are 3 main phases (Fig.4) as follows:-

1) *Preprocessing phase*: encoding each sample image of a character to string, called chain code, using directional information.
2) *State Machine Induction phase*: generating model to accept strings derived from each character class by using chain code strings as training set.
   a. *Positive Training*: generating model to accept any string derived from target character.
   b. *Negative Training*: adjusting model to reject any string derived from non-target character.
3) *Competition phase*: passing unknown string to each model and compare result to find the most possible character.

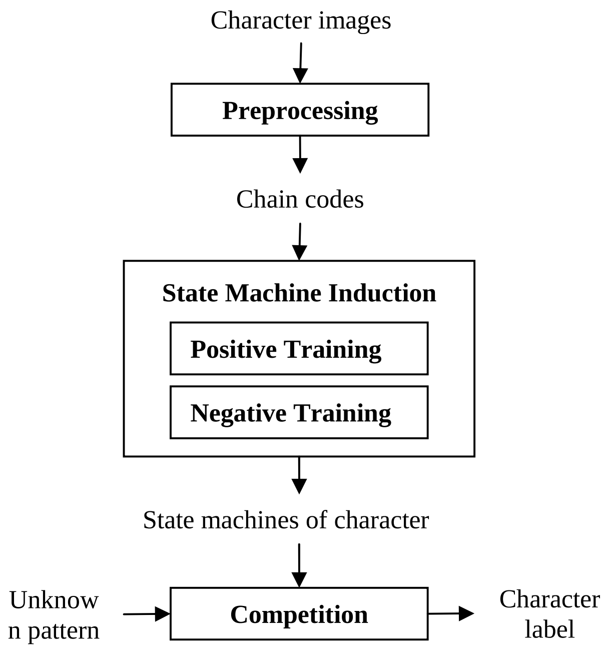


Figure 4 System overview.

V. Automatic State Machine Induction

State machines represented in this work are DFAs created from sample strings. We firstly begin with generating DFA from strings of targeted character class. Secondly, DFA is adjusted by creating new state, called trap state, and setting some transitions from the existing states to trap state. Each character class has one DFA model. After that, the unknown pattern is passed into each DFA to find which label it is.

A. Representation

All of the state machines represented in this research are Deterministic Finite Automata (DFA) with the following constraints. 1) Every state in DFA must have only one loop back to itself for the same input. This input is defined as its value. If there is a new input label at the same state, this input causes transition from this state to the next. The new input does not belong to the current state, since the state already has a loop. 2) The input that causes the transition will be the value of the next state (the next state will loop into that next state with the same input that cause the transition).

The representation of the state in DFA is demonstrated in Fig.5. After the second input '0', new '1' is coming. The transition from the previous state for input '1' is added to the new state valued '1'. Input '1' will repeat at the same state until the next value of input appears. After repeating '1' for five times, the next input '2' comes out and moves to the next state.

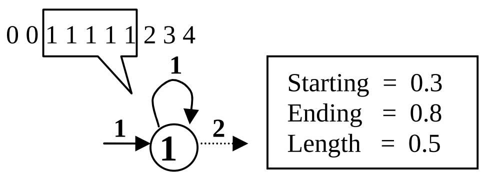


Figure 5 The representation in state machine.

Normally, the state has just the transition information but has no position or length information of each input that causes a transition in a state. In this research, we add position and length of input that cause the loop transition in the state. All positions are normalized in range [0, 1]. As shown in Fig.5, the additional information is used in each state as follows:-

- *Starting point* is the minimum position of the first input reaching that state,
- *Ending point* is the maximum position of the last input before moving to the next state,
- *Length* is the maximum interval of inputs that stay at that state.

Each state generated in DFA has two formats, called *loop* and *loop pair*. Loop state represents the sequence of identical direction inputs, *d**. The same input can repeat at the same state by looping into itself. Loop pair stands for the sequence of inputs that have adjacent direction, *(d1*d2*)**. If a new state represents input symbol that is adjacent to former input, forward and backward transitions are added for both states. Fig.6 shows the example of loop and loop pair states. Fig.6a shows a single state represented sequences of '1', such as 1, 111, 11111111, ... Fig.6b displays a pair of states that has adjacent value '1' and '2' represented any combination of '1' and '2', such as 12, 111212, 12122211, …

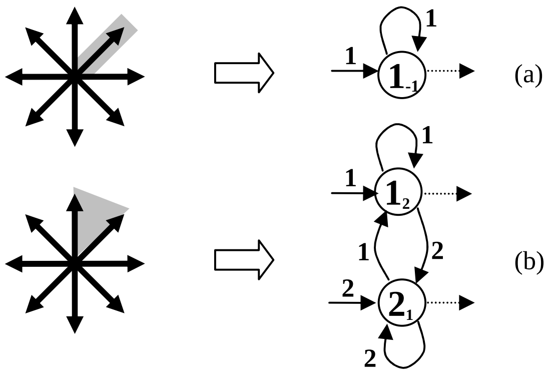


Figure 6 Loop and loop pair state.

The subscript number in each state represents the appropriated value of its pair state, for example, state value '1' can have pair state that has value '0' or '2'. The pair state value '-1' means that there is no expected pair state. This pair state value helps in adding new loop pair state easier. Any state with the subscript value '-1' or undefined loop pair can add any adjacent state as its loop pair. But any state with defined subscript can add a loop pair state that has a value equal to the defined subscript. Any state that already has a loop pair state can not add another loop pair state.

B. Induction Phase

In the induction phase, each DFA is generated by two steps. First, the strings from targeted class are used to generate a DFA that accepts strings in the targeted class. Second, all non-targeted strings are trained and a trap state is created to prevent negative samples reaching the final states of DFA. The former step is called *positive training*, and the latter is called *negative training*.

Initially, a starting state is created for every machine at the beginning. The first training string is used to create states and transitions of an initial model. Each input alphabet of the string is read. If there is no transition for that input, a new transition is added from the present state to the next proper state. If there is no proper state, a new state is generated and the transition for that specific input is added. These steps are repeated until the last input of the string. The current state at the end of the string is set as the final state. The next pattern is read, and all the processes are restarted at the starting state.

All additional transitions must be in the forwarding position, i.e., they cannot add the transition from current state to the previous state. Before creating a new state, all states in DFA are explored to find the state with the same value as the current input. The input position must also be inside that state's position. If that state exists and the transition is forwarded, the transition of that input is added to that state. Otherwise, a new state is created. This forward transition restriction is to avoid generating loop among many states. This causes the acceptance of extra sequences that does not occur in the trained pattern.

Fig.7 shows each step in generating DFA from the first training '2210123244' string pattern.

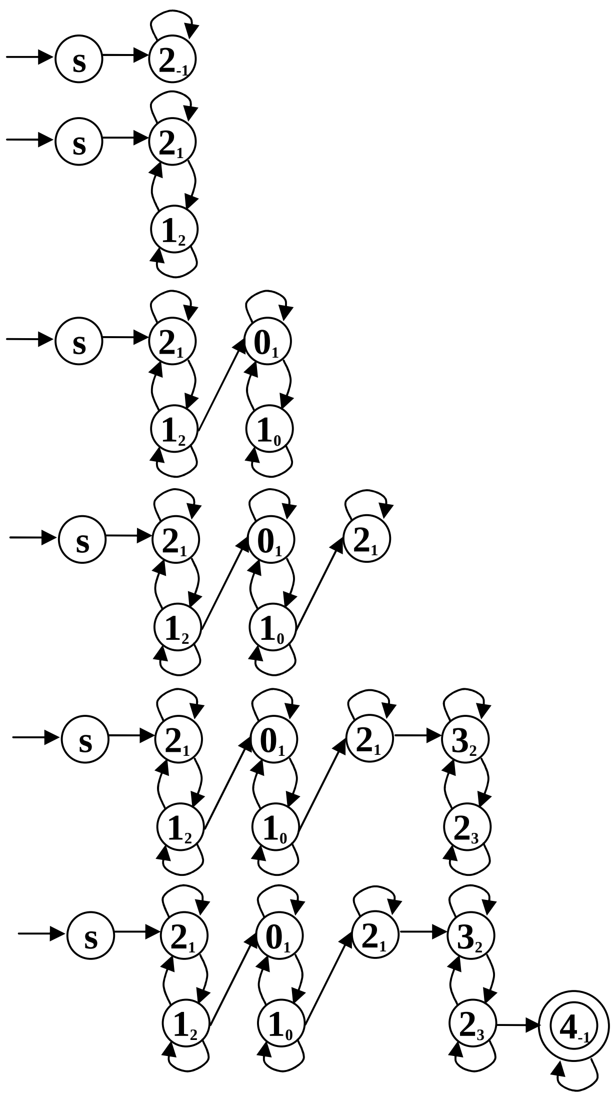


Figure 7 An example of DFA generated from positive training.

After DFA model is generated from positive samples, all patterns from any other character class, called negative samples, are passed into DFA to collect error information. If negative pattern can be accepted by DFA or reach the final state, some error information of that pattern is recorded to use as information to create trap state. The error can occur in any state where there is no transition defined for that input alphabet. Hence, if there is no defined next state for that alphabet, the next state remains in the same current state. The number of errors for each alphabet in each state is kept.

When all negative patterns have already passed, the new "*trap state*" are generated. Trap state is used to prevent negative patterns to reach the final states by adding transitions from error state to trap state. This forces DFA to reject negative strings.

An example of an adapted DFA after generating transitions to trap state is shown in Fig.8. From the final DFA generated in Fig.7, some negative patterns are passed into model. The negative sequence strings, 32130223243 and 2102143245, can reach the final state. Some transitions are added from the states that have error to the trap state.

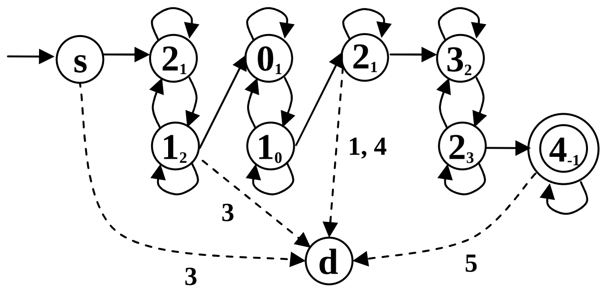


Figure 8 An example of DFA adapted from negative training.

C. Competition Phase

After all DFA models are generated from training samples, the unknown pattern can be identified in a competition phase. begin at the initial state, and then read each input one by one. If there is a transition for current input, then move to the next state. This process continues until all inputs have passed. If the last state is a final state, this means that DFA accepts this string. The unknown string is tested with every DFA.

There may be more than one DFA that accepts the input string. In order to choose the right one, the following parsing information is needed.

- *Transition error*. The total number of inputs in any state that have no transition.
- *Position error*. The total number of inputs that are not between starting and ending point of a current state. This includes all the inputs before starting and after ending position.
- *Length error*. The total number of inputs exceeding the length of a current state.

If there is no transition for current input, the current state does not change until there is transition out to the next state. To find the most aligned DFA, the minimum summation of all three errors is selected.

VI. Experiments

To test its accuracy, state machine induction method was applied with the recognition of Thai character images. All strings used in this experiment are derived from contour following of printed character's images in chain code direction. We use 5,906 patterns of both positive and negative samples from 66 character labels, including numbers, alphabets, vowels, and intonation marks, to generate 66 DFAs. There are 4,920 patterns for training set, and 986 patterns for test set. The average number of states for each generated DFA is 76 and 77 respectively for using only positive training, and using both positive and negative training. The negative training step used only one additional state, trap state, to improve recognition rate.

TABLE I
Competition Result

Approach	*Training set*		*Test set*	
Hidden Markov Model	85.61%		82.56%	
Positive Training	98.70% *	1.30% **	79.41% *	1.62% **
Positive & Negative Training	98.70% *	1.30% **	83.67% *	2.33% **

* Percent of correct patterns
** Percent of unidentified patterns

Table 1 shows the result in competition phase for training and test set. Our approach is compared to the well-known probabilistic state model approach, Hidden Markov Model (HMM). We used Left-Right-Left topology 30 states HMM with final state modification [7] for testing HMM. The HMM used only the positive training. The proposed state machine induction use both positive and negative training. If the winner in competition has the same label as expected label, that pattern is recognized correctly. The unidentified pattern is the pattern which has two or more winners when tested with all DFAs. This pattern cannot be identified which label it should be. The recognition accuracy for the test set of our approach is lower than HMM when positive training is used but about the same with HMM when used with both positive and negative training (1.11% better). In recognition the training set, the proposed state induction is much better than HMM (13.09%).

VII. Conclusion

This paper proposed a new approach to generate DFA automatically from both positive and negative samples by using position and length information. The algorithm was applied with strings taken from character's images in chain code direction. The generated models are simple and do not consume much more time in induction and competition phase. The recognition results are comparable to the HMM.

One problem found in character recognition is that some patterns are very similar, such as 'ข' and 'ฃ'. They cannot be differentiated between significant curve and noise. Noises of string cause many problems in recognition. Some noises cause the generated DFA to be over or under generalized. Some states are created but used only once or twice in training pattern. Some transitions are added but some significant states are skipped. These noises should be cleaned before the strings are used in training.

To improve efficiency, other features can be used in induction or competition phase. For example, in off-line character recognition, statistical usage of each state and transition can be used. Some unused states or transitions can be pruned to decrease restriction of models. Eliminating noise still performs the important role for string recognition in every specific domain.

REFERENCES

[1] B. Trakhtenbrot, and Ya. Barzdin, *Finite Automata: Behavior and Synthesis.* North Holland Publishing Company, Amterdam, 1973.

[2] H. Juillé and J. B. Pollack, "A sampling-based heuristic for tree search applied to grammar induction," in Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98) Tenth Conference on Innovative Applications of Artificial Intelligence (IAAI-98), Madison, Wisconsin, USA, 1998, pp. 26-30.

[3] K. J. Lang, B. A. Pearlmutter, and R. A. Price, "Results of the Abbadingo one DFA learning competition and a new Evidence-Driven State Merging algorithm," Lecture Notes in Computer Science, 1433: 1998, pp. 1-12.

[4] R. C. Carrasco and J. Oncina, "Learning stochastic regular grammars by means of a state merging method," In The 2nd Intl. Collo. on Grammatical Inference and Applications, 1994, pp. 139-152.

[5] B. Kruatrachue, P. Polsuntikul, and K. Siriboon, "Dynamic construction of context free grammar from sample for on-line Thai handwriting recognition," AISTA 2004: International Conference, Luxembourg, Nov. 15 – 18, 2004.

[6] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications inspeech recognition," Proceeding of the IEEE, Vol.77, 1989, pp. 257-286.

[7] K. Siriboon, A. Jirayusakul, and B. Kruatrachue**, "**HMM topology selection for on-line Thai handwritten recognition," in Proceeding of the first International Symposium on Cyber Worlds, 2002.

[8] O. Cicchello and S. C. Kremer, "Inducing grammars from sparse data sets: A survey of algorithms and results," Journal of Machine Learning Research 4, 2003, pp. 603-632.