The Application of Finite State Automata

Ruirui Zhang
201800301072
April 11, 2020

Abstract

Finite State automata is an important cornerstone of computer science. It has applications in many fields, such as natural language processing, pedestrian navigation, and Web applications. This paper briefly discusses several applications of Finite State automata.

Keywords: Finite State Automata, Application, OCR, web

Introduction on Automata Theory

Automata Theory is an exciting, theoretical branch of computer science. It established its roots during the 20th Century, as mathematicians began developing - both theoretically and literally - machines which imitated certain features of man, completing calculations more quickly and reliably. The word **automaton** itself, closely related to the word "automation", denotes automatic processes carrying out the production of specific processes. Simply stated, automata theory deals with the logic of computation with respect to simple machines, referred to as **automata**. Through automata, computer scientists are able to understand how machines compute functions and solve problems. **A finite-state machine** is formally defined as a 5-tuple $(Q, I, Z, , W)$ such that:

- Q = finite set of states
- I = finite set of input symbols
- Z = finite set of output symbols
- = mapping of IxQ into Q called the state transition function, i.e. $IxQ \quad Q$
- W = mapping W of IxQ onto Z, called the output function

- A = set of accept states where F is a subset of Q

1 Application on OCR

In western language OCR systems ,the post-processing of selecting the best result from some candidates is absolutely necessary.Spell-check can provide reliable information for this task.However,there are some limitations in previous methods in different extents.In this part [2],the finite automaton model is applied to the post -processing procedure.It combines the spell -check with the character recognition results.

1.1 About OCR

OCR is short for Optical Character Recognition.
The first research into OCR technology began in Europe, where German Taushek received a patent for OCR as early as 1929. Because of the characteristics of western languages, the post-processing operation of selecting the best results from the candidate results is indispensable in western languages OCR, and the post-processing using word spelling check is completely feasible.

1.2 Finite State Automata in OCR

At present, finite automata model has been widely used in pattern matching. There are two main applications in the whole OCR system:

1. **Recognition post-processing**: in the picture of western document, the adhesion between characters is a common phenomenon. Therefore, after character segmentation, the recognition results and spelling check are often used to verify the character segmentation to achieve the optimal goal.
2. **System post-processing**: after all words have been sharding and identified, the words that have not passed spell check need to be further processed to select the best recognition result from the candidates before the output result.

For the same word, recognition post-processing will be carried out once or more, and the system post-processing at most once. The spelling check in the recognition post processing requires a full match, while that in the system post processing requires an approximate match.

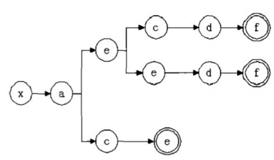

Figure 1: The finite automata model in recognition post-processing

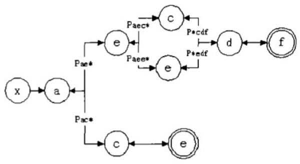

Figure 2: The finite automata model in system post-processing

Because the spelling check in the post-recognition process requires full matching, its finite automaton model is relatively simple. It only needs to input the characters in the word from left to right, and the last word that can be accepted by the final state is the legal word.
However, since the finite automaton model cannot process the words containing the rejection characters and the ambiguous recognition results, the finite automaton model shown in Figure 2 is used for the post-processing of the system. Where, the mark on the bidirectional arc represents the probability of the branch, such as $P(aec*)$ represents the probability of c after the string ae, and $P(*cdf)$ represents the probability of c before the string df. Obviously, for the finite automata model shown in figure 2, $P(ae*) + P(ac*) = 1$.
The operation process of finite automata model in system post-processing is as follows:

1. Input the recognition result from the head of finite automaton.
2. When the rejection character is encountered, the current branch is intercepted to generate a new finite automaton, and the tail of the branch is added with an empty state x , which is used as the head of the newly generated finite automaton, and the recognition result is input into the head of the new finite automaton. For example, if $aecdf$ is identified as $(a, e)(a, e) \sim (b, d)(f)$, the result is input to the head of the finite automaton shown in figure 2. When the recognition character is encountered, the recognition result becomes $ae \sim (b, d)(f)$ and reaches the branch shown in figure 3. The new finite automaton generated by intercepting this branch is shown in figure 4. The recognition result is inverted into $(f)(b, d) \sim ea$, which is input into the head of the finite automaton as shown in figure 4. Finally, the recognition result can be determined as $ae \sim df$.
3. When encountering ambiguous recognition results, the branch with high probability is preferred.

Figure 3: Finite automaton branch

In this way, the recognition results can be correctly selected from the candidate results through the recognition post-processing and system post-processing.

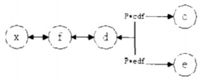

Figure 4: Finite automata generated by the branch

2 Application on public key encryption algorithm

- The development and application of cryptography plays an inestimable role in solving the problem of information security.
  Public key encryption has two keys, one is the public encryption key, the other is the secret decryption key.
  It works like this: any user A wants to communicate secretly with user B. First, A **uses** B's public key to encrypt the plaintext to be transmitted to obtain the ciphertext, and then transmits it to user b. when user B receives the sent ciphertext, it decrypts the ciphertext with its own decryption key to obtain the plaintext sent by A.

- An encryption algorithm that realizes the process of encryption and decryption and the function of digital signature uses the principle of limited automaton reversibility.This encryption system has the characteristics of high security, short key and fast speed.

- The public key cryptosystem of finite automata (FAPKC) was first proposed by Tao and Chen in 1985 by using the reversibility theory of finite automata. [4] At the same time, a specific finite automaton public key cryptosystem FAPKC0 is proposed. After that, they successively proposed FAPKC1, 2, 3 and 4, which constituted an algorithm system of finite automaton encryption.
  What these algorithms have in common is that there is a combined automaton C (M1,M0) in the public key.

- In FAPKC3, the composite finite automaton in the public key is composed of a storage finite automaton and an input storage finite automaton. The key consists of two stored finite automata and their initial state. The finite automata in each public key and its corresponding finite automata in the key are weakly reversible with a delay of a certain number of steps.

3 Application on pedestrian navigation

Automatic recognition of pedestrian navigation state is a difficult problem in pedestrian navigation research. There is an algorithm for pedestrian navigation state matching based on finite state automata [1] . Its core idea is to match the current navigation state of pedestrian on the basis of recognizing pedestrian actions.
The finite state automata theory is used to establish the state transition model,

and a pedestrian navigation state matching algorithm based on the model is designed.

Tab.4 State transition table

输入动作	前一状态		
	S_1	S_2	S_3
A_1	S_2	S_2	S_2
A_2	S_1	S_1	—
A_3	S_2	S_2	S_2
A_4	—	S_3	S_3
A_5	—	—	S_2

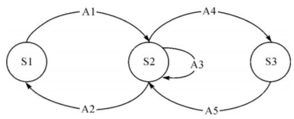

Figure 5: State transition table and diagram used in this algorithm

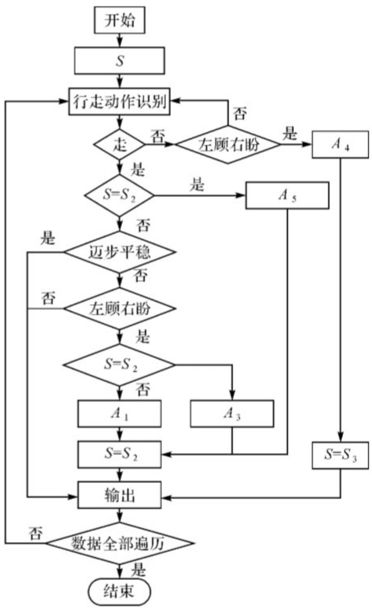

Figure 6: The flow diagram of the matching algorithm.

In the constructed finite state machine model, the movement represented by A1 in the action set is walking, with an unstable step frequency accompanied by left and right head movement.
A2 represents walking with a steady gait, which can be ignored in practice.
A3 represents the reconfirmation of unfamiliar state, and the behavior is the same as A1.

A4 represents the pedestrian's movement before entering the lost state from the unfamiliar state, which is shown as stopping walking and moving the head from side to side.
A5 means that the pedestrian is out of the lost state and has established a certain cognition of the current path environment. The action is to start walking again.
That paper also designs a pedestrian navigation state matching algorithm

based on this model. The picture is shown in the above.

4 Application on Web application system

Finite state automata can describe the behavior of software, so the Web application system can be transformed into a finite state automata model, which can be used as a basis for designing test cases and determining whether the output is correct or not. [3]
Because many factors are considered in the modeling of Web application system, which leads to the complexity of the later model structure, this technique USES the minimization method of finite state automata to simplify the automata model.
The modeling method is firstly to establish a flow chart for the test Web application system, then to abstract the finite state automata model according to the flow chart, and finally to simplify the automata model by the finite state automata minimization method.
The function test of Web application system is to test item by item according to the function description in the product specification to check whether the Web application system realizes the expected function.
After the Web application system is transformed into an automaton model, the test problem becomes the verification of the state transition process with corresponding automata.
For example, the test of a simple module flow of an e-commerce website login can be converted into an NFA model.
Let M be an NFA, and the five states of $'login'$, $'home\ page'$, $'shopping\ cart'$,

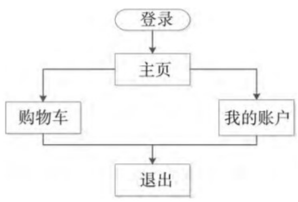

Figure 7: Simple flow chart for the electric business login.

$'my\ account'$ and $'exit'$ as shown in the figure are abstracted as the states of automaton M, A, B, C, D, E, respectively.
Enter user information when login to the main page, abstract is a, other input information is click, abstract is b; The initial state q_0 is login, which is A in the state set; The state in the terminating state set F is exit, and there is only E in the terminating state set. The NFA is shown in the following figure.

Conclusion

Finite automaton has been widely used in various fields of computer, but it also has great limitations. For example, in the recognition of some languages, it usually can only deal with artificial language, while for the natural language

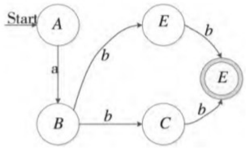

Figure 8: NFA's transition diagram for the electric business login.

with hidden rules, it is often difficult to construct accurate finite automata. In my opinion, although the model of finite state automata is a highly theoretical model, we can see its influence in every aspect of life. Learning the theory of finite state automata is helpful for our future study and research.

References

1. Zhixiang Fang, Hao Luo, and Ling Li. Finite-state automaton assisted pedestrian navigation state matching algorithm. 2017.

2. Kai Wang, Jianming Qi, and Qingren Wang. Finite automata model in ocr post-processing. 2004.

3. Linlin Wang, Yaojun Liu, et al. Web application system testing based on finite state automata minimization. *Journal of taiyuan university of science and technology*, (2017 01):11–15, 2017.

4. Hao Yan, Xiaoyong Yan, and Weiyan Shen. Research and improvement of public key encryption algorithm based on finite automata theory. *Computer security*, 9:12–14, 2008.