Finite State Machines for NLP

Ye Kyaw Thu[1,2,3]

[1]National Electronics and Computer Technology Center (NECTEC), Thailand
[2]Language Understanding Lab., Myanmar
[3]Language and Speech Science Research Lab., Waseda University, Japan

NLP Class, UTYCC, Pyin Oo Lwin, Myanmar

email: ka2pluskha2@gmail.com

November 8, 2019

Lecture Outline

Motivation

- ကျွန်တော် သီအိုရီအနေနဲ့ စိတ်ဝင်စားတယ်
- နောက်တချက်က ကျောင်းသားအများစုက FSAသီအိုရီကို သင်ဖူးကြပေမဲ့ လက်တွေ့မသုံးတတ်ကြဘူး
- မြန်မာစာ NLP R&D အတွက် finite state machine တွေကိုလည်း သုံးကြရအောင်

Finite State Automata (FSA)

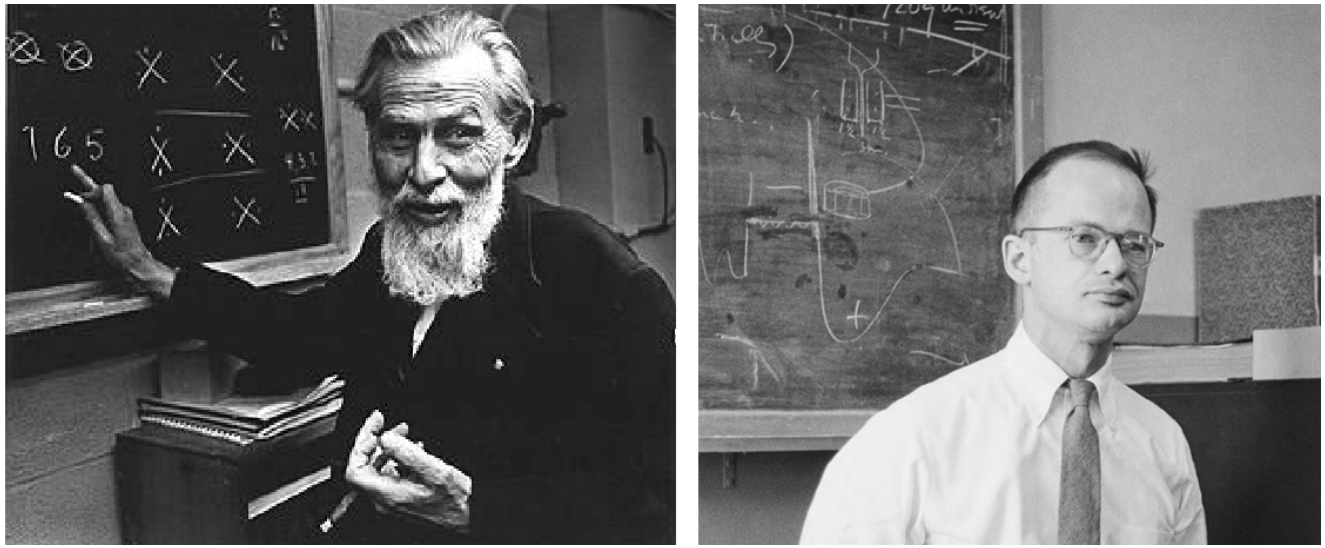

Figure: Left: Warren S. McCullough, Right: Walter Pitts

- ၁၉၄၃ မှာ neuro-psychologists တွေဖြစ်ကြတဲ့ Warren S. McCullough Walter Pitts က ပထမဦးဆုံး Finite State Automata သီအိုရီကို a model for human brain ဆိုပြီးတော့ မိတ်ဆက်ခဲ့တယ်
- Finite automata နဲ့ microprocessor တွေကိုလည်း မော်ဒယ်ဆောက်လို့ရတယ်
- Regular set of sequences တွေဖြစ်တဲ့ logic, algebra, regular expression တွေနဲ့ ဆက်စပ်တယ်

Finite State Automata (FSA)

A finite state automaton is a quintuple $(S, \Sigma, \delta, s_0, F)$, where:

- S is a finite set called the states;
- Σ is a finite input alphabet;
- $\delta : S \times \Sigma \rightarrow S$ is the transition function;
- $s_0 \in S$ is the start state; and
- $F \subseteq S$ is the set of accept states.

Finite State Automata (FSA)

FSA ကို စာနဲ့ အလွယ်ရှင်းပြရရင်

- သူက nodes တွေ၊ မြှားတွေ နဲ့ ဆွဲထားတဲ့ graph ပုံပါပဲ။
- ဝင်လာတဲ့ input ကို လက်ခံတာ၊ ပယ်တာကို လုပ်ပါတယ်။
- input အားလုံးကို လက်ခံနိုင်တယ်၊ ဖတ်လို့ ပြီးတယ်ဆိုရင်၊ initial node ကနေ final node အထိ အရောက်သွားနိုင်တယ်၊ တနည်းအားဖြင့်ပြောပြရရင် automaton ရဲ့ final state ကိုရောက်သွားတယ်။
- node တစ်ခုကနေ နောက်ထပ် node တစ်ခု၊ state တစ်ခုကနေ နောက်ထပ် state တစ်ခုစီကို input value ကိုကြည့်ပြီး သွားမယ့်လမ်းကြောင်းကို ရွေးသွားတဲ့ ပုံစံပါ။
- တချို့ node တွေက ϵ (epsilon) သို့မဟုတ် empty string ကို pass လုပ်ပေးပါလိမ့်မယ်။
- final state ကိုတော့ double wall (double circle) နဲ့ ကိုယ်စားပြုပုံဆွဲတယ်။

Finite State Automata (FSA)

- ဒီ lecture မှာ OpenFST ကိုသုံးပါမယ်။
- OpenFST က Google နဲ့ Courant Institute of Mathematical Sciences, New York University တို့က ပူးပေါင်းပြီးတော့ develop လုပ်ထားတဲ့ Open source tool ဖြစ်ပါတယ်။
- Finite state automata ကို အခြေခံပြီးတော့ ဖြစ်လာတဲ့ Finite state transducers တွေကို လွယ်လွယ်ကူကူ မော်ဒယ်ဆောက်ပြီးတော့ operation တွေကို run နိုင်ဖို့ build လုပ်ထားတာပါ
- Link: http://www.openfst.org

Finite State Automata (FSA)

- ဒီနေရာမှာ အသေးစိတ် မရှင်းနိုင်ပေမဲ့ FSA, FST တွေကို ထဲထဲဝင်ဝင် နားလည်ဖို့က ကျောင်းသားတွေအနေနဲ့က regular expression (RE) ကို သိထားသင့်ပါတယ်။
- ဥပမာ က*ခ+ဂ? ဆိုတဲ့ RE ကို ပြန်စဉ်းစားကြည့်ရအောင်
- A regular expression followed by an asterisk (*) matches zero or more occurrences of the regular expression
- A regular expression followed by a plus sign (+) matches one or more occurrences of the one-character regular expression ရှေးစရာရှိရင် ပထမဆုံး matched ဖြစ်တဲ့ string ကိုပဲ ယူလိမ့်မယ်
- A regular expression followed by a question mark (?) matches zero or one occurrence of the one-character regular expression

Finite State Automata (FSA)

Figure: testing RE က*ခ+ဂ?

- online RE tool တစ်ခုဖြစ်တဲ့ https://regexr.com/ ကို သုံးပြီး စမ်းကြည့်နိုင်တယ်
- RE ရေးနေကြအတိုင်း ကိုယ် စမ်းချင်တဲ့ RE ကို / (forward slash) နှစ်ခုရဲ့ ကြားထဲမှာ ရေးတယ်
- g က global ကို ဆိုလိုတယ်
- g REF flag ထည့်ထားမှ စာကြောင်း တစ်ကြောင်းလုံးမှာ match ဖြစ်သမျှ RE patterm တွေကို ဆွဲထုတ်ပေးနိုင်တယ်

Finite State Automata (FSA)

symbol file ဆောက်ဖို့ လိုအပ်တယ်
Finite state acceptor မောဒယ်အတွက်ဆိုရင်တော့ --isymbols option အတွက်ပဲ လိုအပ်လိမ့်မယ်။

filename: my.syms

ϵ<TAB>0
က<TAB>1
ခ<TAB>2
ဂ<TAB>3
ဃ<TAB>4
င<TAB>5

Finite State Automata (FSA)

regex file လည်း ပြင်ဆင်ဖို့ လိုအပ်တယ်
filename က ကြိုက်သလိုပေးလို့ရပါတယ်။

filename: regex.fsa.txt

0<TAB>1<TAB>ϵ
0<TAB>2<TAB>ϵ
2<TAB>2<TAB>ဣ
2<TAB>1<TAB>ϵ
1<TAB>3<TAB>ခ
3<TAB>3<TAB>ခ
3<TAB>4<TAB>ϵ
3<TAB>4<TAB>ဂ
4

Finite State Automata (FSA)

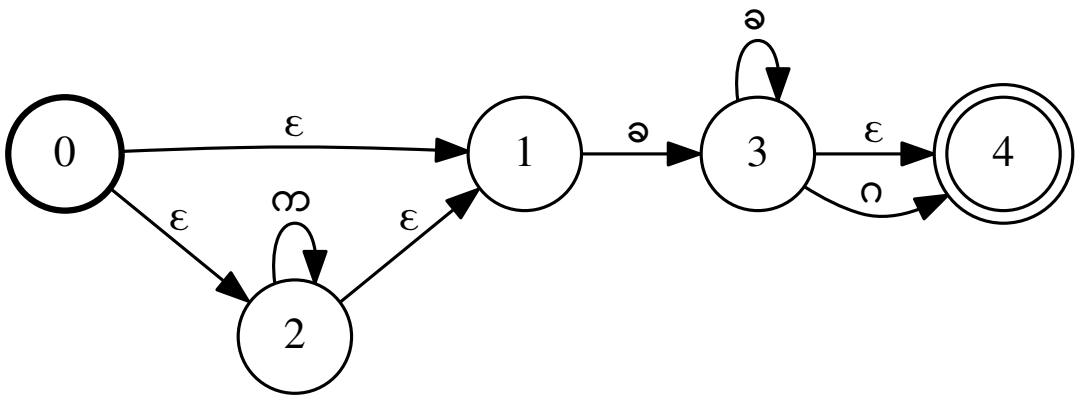

Figure: Finite state automata of ɔ*ə+ɔ?

- fstcompile --acceptor --isymbols=my.syms regex.fsa.txt > regex.fsa
- fstdraw --portrait --acceptor --isymbols=my.syms regex.fsa | dot -Tpdf > regex.pdf

Finite State Automata (FSA)

testing လုပ်ကြည့်ဖို့အတွက် input ဖိုင်ကို ပြင်ဆင်မယ်

filename: input.fsa.txt

0<TAB>1<TAB>က
1<TAB>2<TAB>က
2<TAB>3<TAB>ခ
3<TAB>4<TAB>ခ
4<TAB>5<TAB>ဂ
5

Finite State Automata (FSA)

- fstcompile --acceptor --isymbols=my.syms ./input.fsa.txt > input.fsa
- fstprint --isymbols=my.syms ./input.fsa

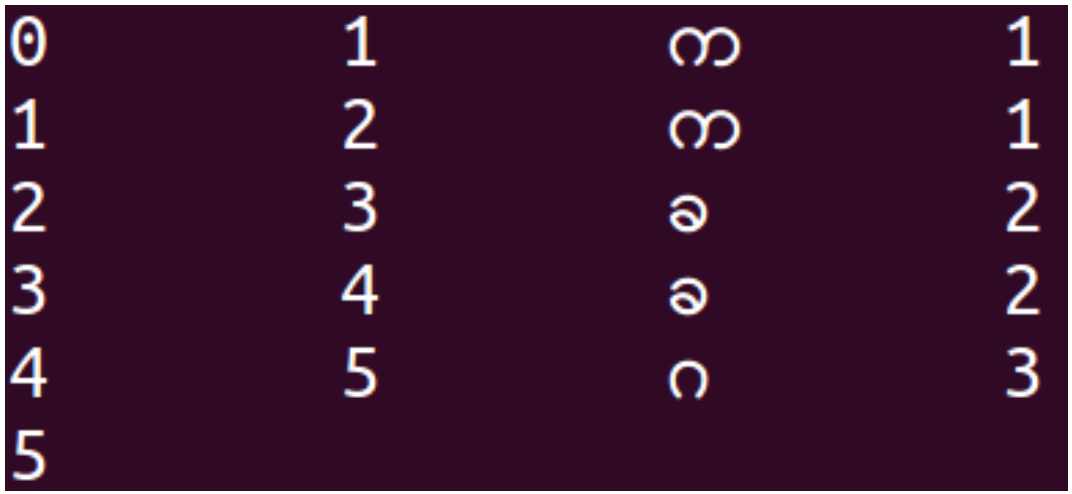

Figure: fstprint command output screen

Finite State Automata (FSA)

fsa မော်ဒယ် နှစ်ခုကို compose လုပ်ပြီးတော့ output ကို print ထုတ်ကြည့်ရအောင်

- fstcompose ./input.fsa ./regex.fsa | fstprint --isymbols=my.syms --osymbols=my.syms

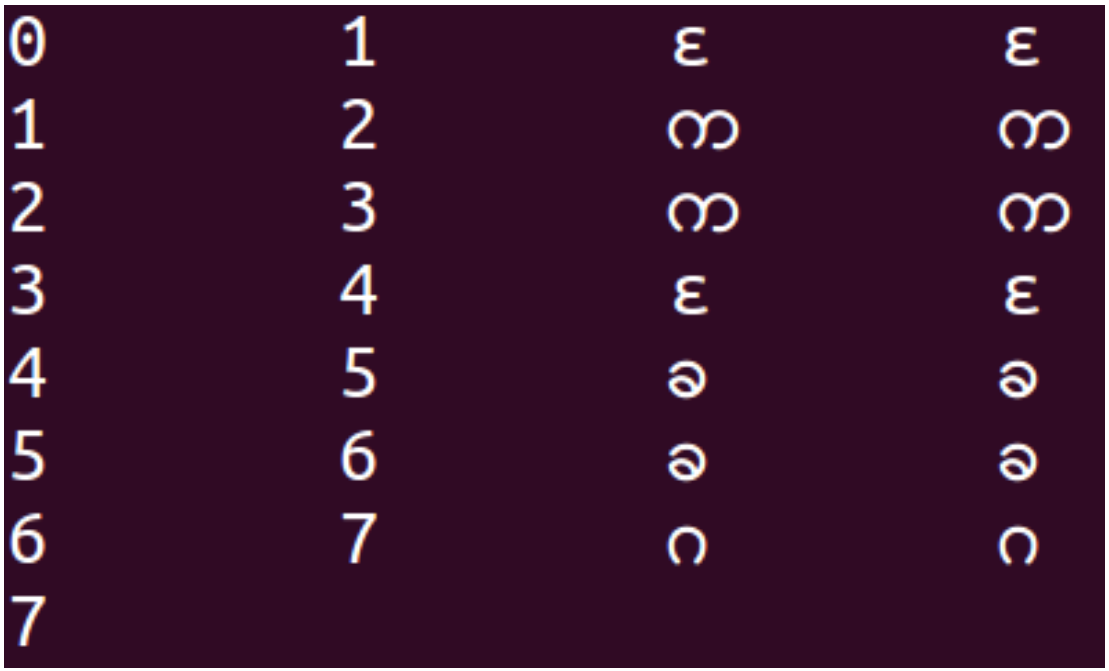

Figure: fstprint command output screen

Finite State Automata (FSA)

symbols file ကို မပေးရင် ဂဏန်းနဲ့ပဲ ရိုက်ထုတ်ပြလိမ့်မယ်။

- fstcompose ./input.fsa ./regex.fsa | fstprint

```
0	1	0	0
1	2	1	1
2	3	1	1
3	4	0	0
4	5	2	2
5	6	2	2
6	7	3	3
7
```

Figure: fstprint command output screen

Finite State Transducer (FST)

Let's do above steps on your computer
1st install OpenFST

Finite State Transducer (FST)

A finite state automaton is a sextuple $(\Sigma, \Gamma, S, s_0, \delta, \omega)$, where:

- S is a finite, non-empty set of states
- Σ is the input alphabet (a finite non-empty set of symbols)
- Γ is the output alphabet (a finite, non-empty set of symbols)
- s_0 is the initial state, an element of S. In a nondeterministic finite automaton, s_0 is a set of initial states.
- δ is the state-transition function: $\delta : S \times \Sigma \rightarrow S$.
- ω is the output function.

to be continue ... :)

Thank you!