Natural Language Processing

Paper Code: CS-821/6
UG 8th Sem

Dr. Samit Biswas, *Assistant Professor*

Department of Computer Sc. and Technology

Indian Institute of Engineering Science and Technology, Shibpur

Email: samit@cs.iiests.ac.in

Plan for Today

- Finite State Transducers (FST)

Finite State Transducers (FST) - An Overview

- A finite-state acceptor can only output two responses:
  - ACCEPT or REJECT (→ useful for e.g. *spell checking*)
- Return more interesting information with a finite state transducer.
- “Mapping” between *upper* language and *lower* language
- Analysis process of a finite state transducer
  - Start at the start state/beginning of the input string
  - Match the input symbols against the lower-side symbols on the arcs, consume all input symbols and find a path to a final state.
  - If successful:
    - return string of upper-side symbols on the path as result.
    - If not successful: return nothing (reject)

Finite State Transducers (FST) - An Overview

- FSAutomata have Input Labels.
- FSTransducers have input:output pairs on labels.

Q	a finite set of N states $q_0, q_1, \ldots, q_{N-1}$
Σ	a finite set corresponding to the input alphabet
$q_0 \in Q$	the start state
$F \subseteq Q$	the set of final states
$\delta(q, w)$	$Q \times \Sigma^* \rightarrow 2^Q$

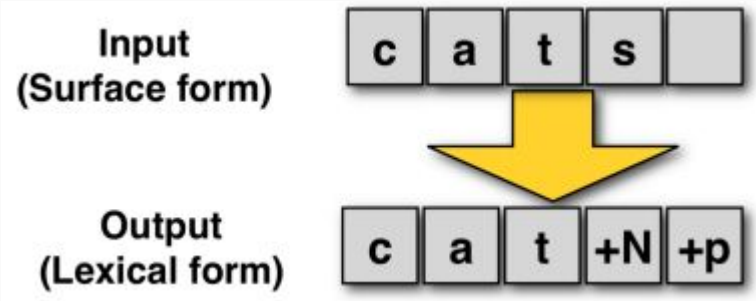
Finite State Transducers (FST) - An Overview

- FSAutomata have Input Labels.
- FSTransducers have input:output pairs on labels.

Q	a finite set of N states $q_0, q_1, \ldots, q_{N\_1}$
Σ	a finite set corresponding to the input alphabet
Δ	a finite set corresponding to the output alphabet
$q_0 \in Q$	the start state
$F \subseteq Q$	the set of final states
$\delta(q, w)$	$Q \times \Sigma^* \rightarrow 2^Q$
$\sigma(q, w)$	$Q \times \Sigma^* \rightarrow 2^{\Delta^*}$

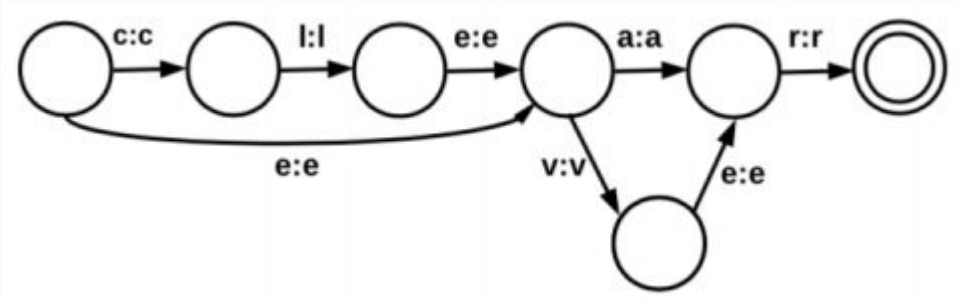
Finite State Transducers (FST) - Morphological Generation

- **FSAs** can recognize (accept) a string, but they don't tell us its internal structure.
- We need a machine that maps (transduces) the input string into an output string that encodes its structure:



Finite State Transducers (FST)

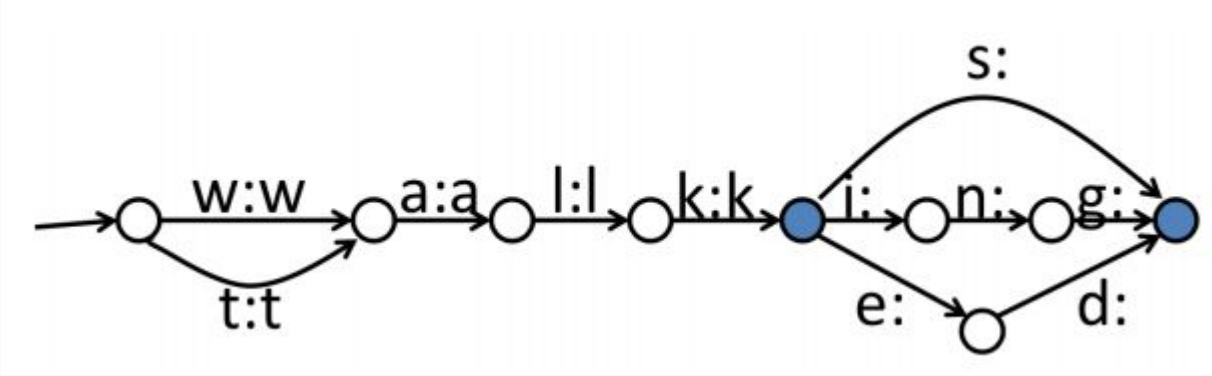
Example1:



- input: *clear*, output: *clear*
- input: *clever*, output: *clever*, . . .
- Alphabet of pairs of symbols u:l
- upper language: lexical language
- lower language: surface language
- An acceptor can be viewed as an identity transducer

Finite State Transducers (FST)

Example2:



- This can map strings to (sets of) other strings
- can map talk, talks, talked, talking to talk. (in generation mode)
- can also map walk to walk, walks, walked, walking. (in analysis mode).

Finite State Transducers - Some Operations

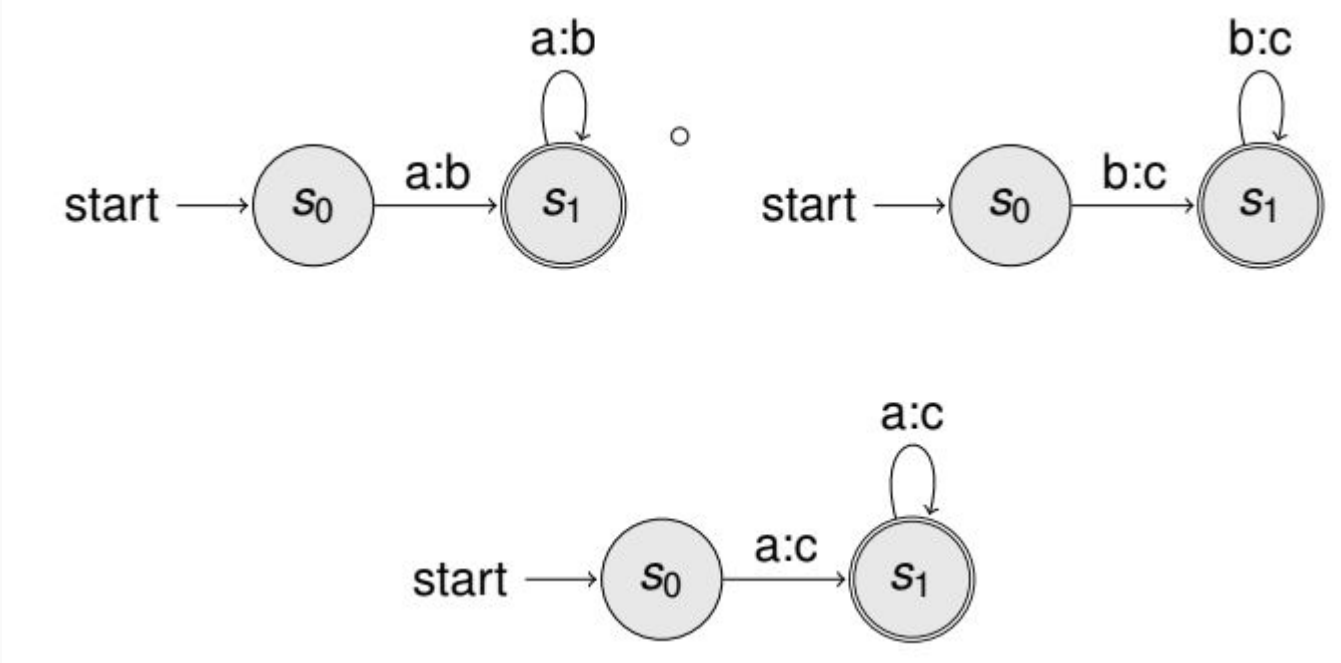
- **Inversion:** The inversion of a transducer $T(T^{-1})$ simply switches the input and output labels. Thus if T maps from the input alphabet I to the output alphabet O, T^{-1} maps from O to I.

  $T = \{(a, a1), (a, a2), (b, b1), (c, c1), (c, a)\}$

  $T^{-1} = \{(a1, a), (a2, a), (b1, b), (c1, c), (a, c)\}$

Finite State Transducers - Some Operations

- **Composition:** T If T_1 is a transducer from I_1 to O_1 and T_2 a transducer from O_1 to O_2, then $T_1 \circ T_2$ maps from I_1 to O_2.
- Transducer Function
  $T_1 \circ T_2(x) = T_1(T_2(x))$
- Example:


a:b
start
s_0
a:b
s_1
∘
b:c
start
s_0
b:c
s_1
a:c
start
s_0
a:c
s_1

Spelling Rules

Name	Description of Rule	Example
Consonant doubling	1-letter consonant doubled before *-ing/-ed*	beg/begging
E deletion	Silent e dropped before -ing and –ed	make/making
E insertion	e added after –*s, -z, -x, -ch, -sh* before *-s*	watch/watches
Y replacement	*-y* changes to –*ie* before *-s*, and to *-i* before *-ed*	try/tries
K insertion	verbs ending with *vowel + -c* add *-k*	panic/panicked

Porter Stemmer

- Lexicon free stemmer; heuristic rules for deriving the stem.
- Rules to rewrite the suffix of a word.
  - “ational” → “ate” (eg. relational → relate)
  - “-ing” → e (eg. motoring → motor)
- Purported to improve recall in IR engines.

- Errors occur:
  - organization → organ, doing → doe, university → universe
-

Role of Morphology in Machine Translation

- Every MT system contains a bilingual lexicon
- Bilingual lexicon: a table mapping the source language token to target language token(s).
- Two options:
  - Full-form lexicon
    - every word form of the source token is paired with the target token large table if the vocabulary is large for morphologically rich languages
  - Root-form lexicon
    - pairing of stems from the two languages
    - Reduces the size of the lexicon
    - requires morphological analysis for source language
    - bats → (bat, V, 3sg) (bat, N, pl)
    - morphological generation for target language
    - (bat, V, 3sg) → bats
- Unknown words: words not covered in the bilingual lexicon
  - with morphology, one can guess the syntactic function

References

References

- Daniel Jurafsky & James H. Martin, “*Speech and Language Processing*”, Prentice Hall
- Diana Maynard, Kalina Bontcheva, Isabelle Augenstein, “*Natural Language Processing for the Semantic Web*”, A Publication in the Morgan & Claypool Publishers series.
- Steven Bird, Ewan Klein and Edward Loper, “*Natural Language Processing with Python*”, By O‘Reilly.
- Christopher Manning and Hinrich Schtze, “*Foundations of Statistical Natural Language Processing*”, The MIT Press.

Thank You

Contacts:

Dr. Samit Biswas
Department of CST,
IIEST, Shibpur

samit@cs.iiests.ac.in
https://www.iiests.ac.in/IIEST/Faculty/cs-samit