

6. Transliteration with WFSTs

Transliteration

- **Transliteration** is the task of mapping strings from one character set to strings in another character set. Replacing characters with their approximate phonetic equivalents.
- **Transduction** is the process of mapping (transducing) one sequence into another.
- A **transducer** is some program that performs such a mapping.



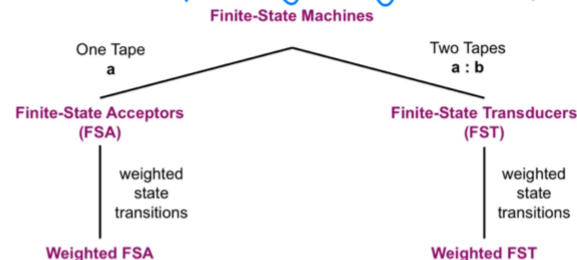
Finite State Machines

Finite-State machines (FSM) or finite-state automata are theoretical models of computation that involve transitions between a finite number of states on regular languages.

* A regular language is a language that can be defined by a regular expression.

FSA v.s. FST

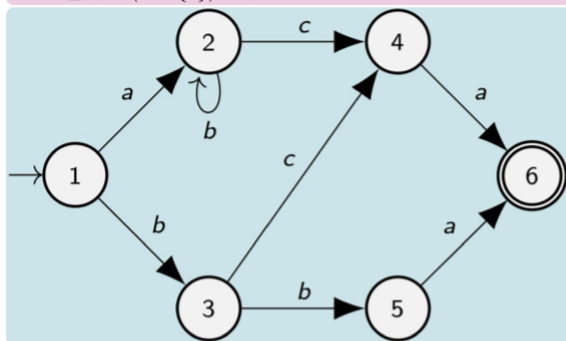
- FSAs are one-tape FSMs, i.e., they only accept one sequence/string at a time.
- FSTs model the mapping/transduction of two sequences/strings.



FSA & WFSA Definition

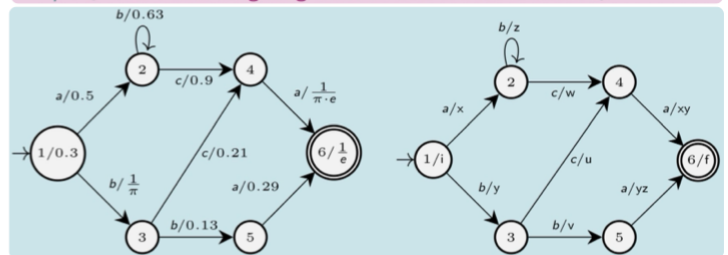
A **finite-state automaton (FSA)** $\mathcal{A}$ is a 5-tuple $(\Sigma, Q, I, F, \delta)$ where

- Σ is an **alphabet**;
- Q is a finite set of **states**;
- $I \subseteq Q$ is the set of **initial states**;
- $F \subseteq Q$ is the set of **final or accepting states**;
- $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$ is a finite multi-set.



A **weighted finite-state automaton** $\mathcal{A}$ over a semiring $\mathcal{W} = (\mathbb{K}, \oplus, \otimes, 0, 1)$ is a 7-tuple $(\Sigma, Q, I, F, \delta, \lambda, \rho)$ where

- Σ is a finite **alphabet**;
- Q is a finite set of **states**;
- $I \subseteq Q$ is a set of **initial states**;
- $F \subseteq Q$ is a set of **final states**;
- $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \mathbb{K} \times Q$ is a finite multi-set of **transitions**;
- $\lambda : Q \rightarrow \mathbb{K}$ is an **initial weighting function** over the set of states Q ;
- $\rho : Q \rightarrow \mathbb{K}$ is a **final weighting function** over the set of states Q .



* WFSA are generalizations of FSA where transitions are weighted with weights from a semiring.

* Any unweighted FSA can be represented as a WFSA over Boolean semiring.

* A language is accepted by a WFSA if it is accepted by an unweighted version of the automaton.

Code & Path

A **path** π is an element of δ^* with **consecutive transitions** $(q_1 \xrightarrow{\bullet/\bullet} q_2, q_2 \xrightarrow{\bullet/\bullet} q_3 \dots q_{n-1} \xrightarrow{\bullet/\bullet} q_n)$, where $\bullet$ is a placeholder.

We use $p(\pi) \stackrel{\text{def}}{=} q_1$ and $n(\pi) \stackrel{\text{def}}{=} q_n$ to denote the origin and the destination of a path, respectively.

The **length** of a path is the number of transition in it; we denote it as $|\pi|$.

The **yield** of a path is the concatenation of the input symbols on the edges along the path, which we will mark with $s(\pi)$.

We say a path π is a **cycle** if the starting and finishing states are the same, and we say that a path **contains** a cycle if the same state appears more than once in the path. A WFSM for which some $\pi \in \Pi(\mathcal{A})$ contains a cycle is called **cyclic**.

Example of cycle: $(1 \xrightarrow{b/1} 3, 3 \xrightarrow{b/0.13} 5, 5 \xrightarrow{a/0.29} 1)$

We denote sets of paths with capital Π .

- $\Pi(\mathcal{A})$ as the set of all paths in automaton $\mathcal{A}$;
- $\Pi(s; \mathcal{A})$ as the set of all paths with yield $s \in \Sigma^*$;
- $\Pi(q; \mathcal{A})$ as the set of all paths starting at state $q \in Q$;
- $\Pi(q, q'; \mathcal{A})$ as the set of all paths from state q to q' ;
- $\Pi(q, s, q'; \mathcal{A})$ the set of all paths from state q to q' with the yield s
- $\Pi(Q; \mathcal{A}) = \bigcup_{q \in Q} \Pi(q)$;
- $\Pi(Q, Q'; \mathcal{A}) = \bigcup_{q \in Q, q' \in Q'} \Pi(q, q')$;
- $\Pi(Q, s, Q'; \mathcal{A}) = \bigcup_{q \in Q, q' \in Q'} \Pi(q, s, q')$;

The indexing with $\mathcal{A}$ will be dropped if the WFSM in question is clear from the context.

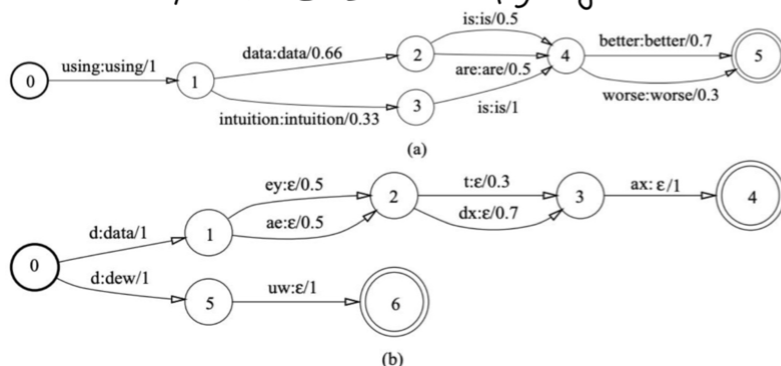
Ambiguity

A FSA $(\Sigma, Q, I, F, \delta, \lambda, \rho)$ is **unambiguous** if, for every string $s \in \Sigma^*$, there is at most one accepting path for that s . Otherwise, we say the WFSM is **ambiguous**.

Weighted Finite-State Transducer (WFST)

A WFST is a tuple $T = \langle Q, \Sigma, \Omega, \lambda, \rho, \delta \rangle$ where:

- Q defines the finite set of states, including initial and ending state(s).
- Σ and Ω are the input and output vocabularies, respectively.
- $\lambda: Q \rightarrow \mathbb{R}$ is a function that maps states to initial scores
- $\rho: Q \rightarrow \mathbb{R}$ maps states to final scores.
- The transition function $\delta: Q \times (\Sigma \cup \epsilon) \times (\Omega \cup \epsilon) \times Q \rightarrow \mathbb{R}$ maps transitions to scores, where ϵ is the empty symbol.



Scoring path

- $\pi = q_{\text{start}} - a:b \rightarrow q_1 - c:d \rightarrow \dots q_{\text{end}}$
- $\text{yield}_{\text{top}}(\pi) \in \Sigma^*, \text{yield}_{\text{bottom}}(\pi) \in \Omega^*$
- $\text{Score}(\pi) = \lambda(q_{\text{start}}) + \delta(q_{\text{start}} - a:b \rightarrow q_1) + \dots + \delta(q_n - c:d \rightarrow \dots q_{\text{end}}) + \rho(q_{\text{end}})$
- * sum of weights $\Rightarrow$ score function
- $\text{score}(\pi) = \sum_{n=1}^{|\pi|} \text{score}(T_n)$

WFSTs as Log-Linear Models

$$P(y|x) = \frac{1}{Z} \exp \{ \text{score}(y, x) \}$$

$$= \frac{1}{Z} \sum_{\pi \in \Pi(x, y)} \exp \left\{ \sum_{n=1}^{|\pi|} \text{score}(T_n) \right\}$$

$$Z = \sum_{y \in \Sigma^*} \exp \{ \text{score}(y, x) \}$$

* Σ^* may be infinitely large. Assumption & Algorithm:

Floyd-Warshall Algorithm

Algorithm 9.1

def FLOYD-WARSHALL ALGORITHM(G):

let W be a $N \times N$ adjacency matrix of a graph G

let dist be a $N \times N$ array of minimum distances initialized to infinity.

for each edge (u, v) :

$\text{dist}[u][v] \leftarrow W[u][v]$

end for

for each vertex v :

$\text{dist}[v][v] \leftarrow 0$

end for

for k from 1 to N :

for i from 1 to N :

for j from 1 to N :

if $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$:

$\text{dist}[i][j] \leftarrow \text{dist}[i][k] + \text{dist}[k][j]$

end if

end for

end for

end for

return dist

▷ "pre-processing" $O(N)$

▷ Assign the weight of the edge (u, v)

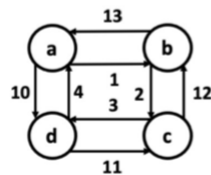
▷ $O(N)$

▷ $O(N^3)$

* Complexity is $O(N^3)$

Example:

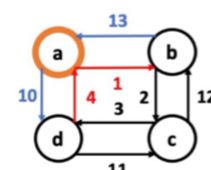
	a	b	c	d
a	0	1	∞	10
b	13	0	2	∞
c	∞	12	0	3
d	4	∞	11	0



$|N| = 4$ vertices

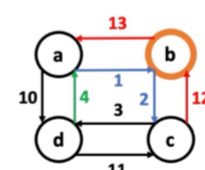
k = a

	a	b	c	d
a	0	1	∞	10
b	13	0	2	23
c	∞	12	0	3
d	4	5	11	0



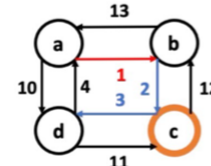
k = b

	a	b	c	d
a	0	1	3	10
b	13	0	2	23
c	25	12	0	3
d	4	5	7	0



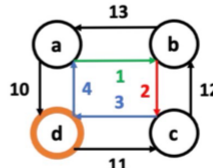
k = c

	a	b	c	d
a	0	1	3	6
b	13	0	2	5
c	25	12	0	3
d	4	5	7	0



k = d

	a	b	c	d
a	0	1	3	6
b	9	0	2	5
c	7	8	0	3
d	4	5	7	0



Matrix Multiplication

$$C_{np} = \sum_{m=1}^n A_{nm} \times B_{mp}$$

* Complexity is $O(N^3)$

Algorithm 9.2

def SEMIRINGMATRIXMULTIPLICATION(A, B):

let A and B be square matrices of size $N \times N$

let C be an empty $N \times N$ matrix

for n from 1 to N :

for p from 1 to N :

sum $\leftarrow 0$

for m from 1 to N :

sum $\leftarrow \text{sum} \oplus A[n][m] \otimes B[m][p]$

end for

$C[n][p] \leftarrow \text{sum}$

```

    end for
  end for
  return C

```



Algorithm 9.3

```

def MATRIX MULTIPLICATION FLOYD-WARSHALL(W):
  let  $W^1$  be the initial adjacency matrix featuring paths of length 1
  for each vertex  $k$  in  $N$ :
     $W^k = 0$  ▷  $= W^0$  (zero-path length)
  end for
  for each vertex  $k$  in  $N$ : ▷  $O(N)$ 
     $W^k = W^k \oplus (W^{k-1} \otimes W^1)$  ▷ matrix multiplication  $O(N^3)$ 
  end for
  return  $W^k$ 

```



Lehmann's algorithm

Algorithm 9.4

```

def LEHMANN'S ALGORITHM(W):
   $W$  be a  $N \times N$  array of minimum distances initialized to  $\bar{0}$ 
  for each edge  $(u, v)$ :
     $W[u][v] \leftarrow W[u][v]$  ▷ this corresponds to  $W^1$  (one step path length)
  end for
  for each vertex  $v$ :
     $W[v][v] \leftarrow W[v][v]$  ▷ this corresponds to  $W^0$ 
  end for
  for  $k$  from 1 to  $N$ :
    for  $i$  from 1 to  $N$ :
      for  $j$  from 1 to  $N$ :
         $W[i][j] \leftarrow W[i][j] \oplus (W[i][k] \otimes W[k][k]^* \otimes W[k][j])$ 
      end for
    end for
  end for
  return  $W$ 

```

