Markov models essentially build upon finite-state automata (FSAs)

So, we’re going to review:

▶ Where FSAs fit on the Chomsky Hierarchy
▶ Finite-State Automata (FSAs)
▶ Finite-State Transducers (FSTs)

The Chomsky Hierarchy

▶ Type 0: unrestricted grammar
e.g. NP was V,pp by NP → NP V NP
▶ Type 1: context-sensitive grammar: only one of the lefthand side symbols is replaced on the righthand side, all the others are “the context”
e.g. V NP → V DET dobj N dobj
▶ Type 1: monotonic grammar: the lefthand side contains the same number or less symbols than the righthand side

Finite-State Automata

Definition FSA: A finite-state automaton A is a 5-tuple (Σ, Q, i, F, E) where

Σ is a finite set called the alphabet
Q is a finite set of states
i ∈ Q is the initial state
F ⊆ Q is the set of final states, and
E ⊆ Q × (Σ U e) × Q is the set of transitions

Finite-State Automata (2)

S → aA
S → aB
A → bC
A → bB
B → cA
B → cC
C → aD
Finite-State Automata (2)

Deterministic Finite-State Automata

Definition DFSA:
A deterministic finite-state automaton is a 5-tuple $(\Sigma, Q, i, F, d)$ where

- $\Sigma$ is a finite set called the alphabet
- $Q$ is a finite set of states
- $i \in Q$ is the initial state
- $F \subseteq Q$ is the set of final states, and
- $d$ is the transition function that maps $Q \times \Sigma$ to $Q$

Important Properties of FSAs

- **determinization**: for every non-deterministic finite-state automaton there exists an equivalent deterministic FSA.
- **minimization**: for every non-deterministic finite-state automaton, there exists an equivalent deterministic automaton with a minimal number of states.

What Is in a State

Definition (state):
Given a deterministic FSA $M = (\Sigma, Q, i, F, d)$,
a state of $M$ is a triple $(x, q, y)$
where $q \in Q$ and $x, y \in \Sigma^*$

Example: $x = aaaaa$, $q = S$, $y = bbbbbbbbbb$ for language $a^*b^*$
The "directly derives" Relation

**Definition (directly derives):**
Given a deterministic FSA $M = (\Sigma, Q, i, F, d)$,
a state $(x, q, y)$ directly derives state $(x', q', y')$: $(x, q, y) \vdash (x', q', y')$ iff
1. there is $\sigma \in \Sigma$ such that $y = \sigma y'$ and $x' = x \sigma$ (i.e. the reading head moves right one symbol $\sigma$)
2. $d(q, \sigma) = q'$

Acceptance

**Definition (acceptance):**
Given a deterministic FSA $M = (\Sigma, Q, i, F, d)$ and a string $x \in \Sigma^*$
$M$ accepts $x$ iff there is a $q \in F$ such that $(0, i, x) \vdash^* (x, q, 0)$.

Important Properties of FSAs

Given the FSAs $A$, $A_1$, and $A_2$ and the string $w$, the following properties are decidable:
- membership: $w \in (A)$?
- emptiness: $L(A) = \emptyset$?
- totality: $L(A) = \Sigma^*$?
- subset: $L(A_1) \subseteq L(A_2)$?
- equality: $L(A_1) = L(A_2)$?

The "derives" Relation

**Definition (derives):**
Given a deterministic FSA $M = (\Sigma, Q, i, F, d)$,
a state $A$ derives state $B$: $(x, q, y) \vdash^* (x', q', y')$ iff there is a sequence $S_0 \vdash S_1 \ldots S_k$ such that $A = S_0$ and $B = S_k$.

Language Accepted by $M$

**Definition (language accepted by $M$):**
Given a deterministic FSA $M = (\Sigma, Q, i, F, d)$,
the language $L(M)$ accepted by $M$ is the set of all strings accepted by $M$.

Encoding FSAs as Matrices

- basic idea: encode alphabet symbols that appear on transitions in a given FSA by a state transition matrix
- the transition matrix will have a 1 in a given cell in case its row and column match the from- and to-states of a transition for the alphabet symbol in the FSA; all other cells are filled with 0
- by matrix multiplication, one can determine the number of successful paths through an automaton
**FSA Example**

A non-deterministic automaton:

```
0 1
a b
a
b
a
b
a
0
1
2
```

What is its language?

**FSA Matrix (2)**

```
c = s0 s1 s2
  0 0 0
  0 0 1
  0 0 0
```

```
init = s0 s1 s2
  1 0 0
```

```
final = s0 s1
  0 0
```

**Finite-State Transducers**

**Definition (finite-state transducer):**

A finite-state transducer $T$ is a tuple $(Q, \Sigma, \Gamma, i, F, d)$, where

- $Q$ is a finite set of states
- $\Sigma$ is a finite set called the input alphabet
- $\Gamma$ is a finite set called the output alphabet
- $i \in Q$ is the initial state
- $F \subseteq Q$ is the set of final states, and
- $d$ is the transition relation that maps $q \times \Sigma \times \Gamma$ to $Q$

**Important Properties of FSTs**

- It is decidable whether the relation $[T]$ of a transducer $T$ is empty.
- It is decidable whether there exists a string $y$ such that $x[T]y$ for a given string $x$.
- It is undecidable whether two transducers are equivalent.