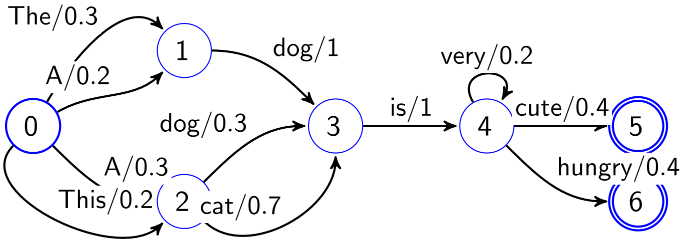
Lecture 16: Weighted Finite State Transducers (WFST)

Mark Hasegawa-Johnson

ECE 417: Multimedia Signal Processing, Fall 2020

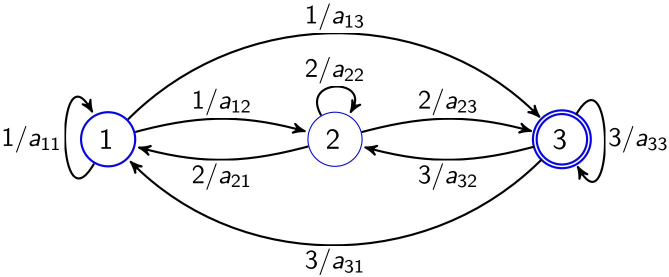
Outline

Weighted Finite State Acceptors



- An **FSA** specifies a set of strings. A string is in the set if it corresponds to a valid path from start to end, and not otherwise.
- A **WFSA** also specifies a probability mass function over the set.

Every Markov Model is a WFSA



A Markov Model (but not an HMM!) may be interpreted as a WFSA: just assign a label to each edge. The label might just be the state number, or it might be something more useful.

Best-Path Algorithm for a WFSA

Given:

- Input string, $S = [s_1, \ldots, s_T]$. For example, the string "A dog is very very hungry" has $T = 5$ words.
- Edges, e, each have predecessor state $p[e] \in Q$, next state $n[e] \in Q$, weight $w[e] \in \overline{\mathbb{R}}$ and label $\ell[e] \in \Sigma$.
- **Initialize:**

$$\delta_0(i) = \begin{cases} \bar{1} & i = \text{initial state} \\ \bar{0} & \text{otherwise} \end{cases}$$

- **Iterate:**

$$\delta_t(j) = \underset{e:n[e]=j,\ell[e]=s_t}{\text{best}} \delta_{t-1}(p[e]) \otimes w[e]$$

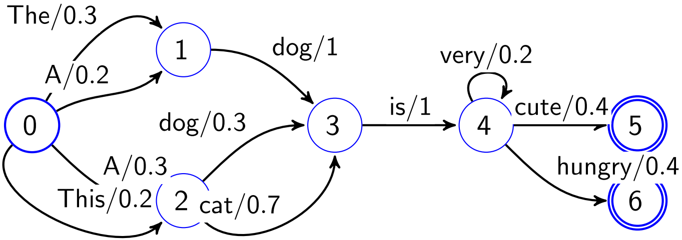
$$\psi_t(j) = \underset{e:n[e]=j,\ell[e]=s_t}{\text{argbest}} \delta_{t-1}(p[e]) \otimes w[e]$$

- **Backtrace:**

$$e_t^* = \psi(q_{t+1}^*), \qquad q_t^* = p[e_t^*]$$

Determinization

A WFSA is said to be **deterministic** if, for any given (predecessor state $p[e]$, label $\ell[e]$), there is at most one such edge. For example, this WFSA is not deterministic.



How to Determinize a WFSA

The only general algorithm for **determinizing** a WFSA is the following exponential-time algorithm:

- For every state in A, for every set of edges $e_1, \ldots, e_K$ that all have the same label:
  - Create a new edge, e, with weight $w[e] = w[e_1] \oplus \cdots \oplus w[e_K]$.
  - Create a brand new successor state $n[e]$.
  - For every edge leaving any of the original successor states $n[e_k]$, $1 \leq k \leq K$, whose label is unique:
    - Copy it to $n[e]$, $\otimes$ its weight by $w[e_k]/w[e]$
  - For every set of edges leaving $n[e_k]$ that all have the same label:
    - Recurse!

Outline

Semirings

A **semiring** is a set of numbers, over which it's possible to define a operators $\otimes$ and $\oplus$, and identity elements $\bar{1}$ and $\bar{0}$.

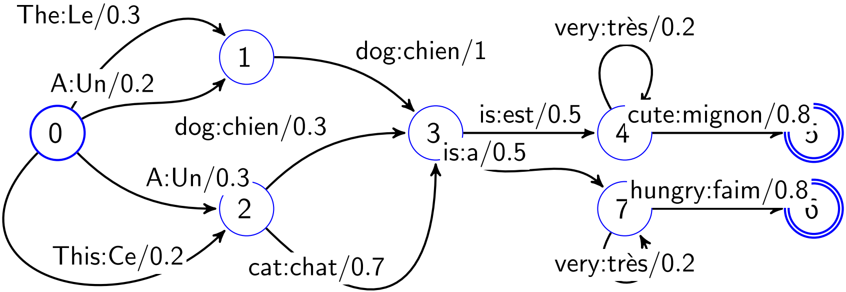
- The **Probability Semiring** is the set of non-negative real numbers $\mathbb{R}_+$, with $\otimes = \cdot$, $\oplus = +$, $\bar{1} = 1$, and $\bar{0} = 0$.
- The **Log Semiring** is the extended reals $\mathbb{R} \cup \{\infty\}$, with $\otimes = +$, $\oplus = -\text{logsumexp}(-,-)$, $\bar{1} = 0$, and $\bar{0} = \infty$.
- The **Tropical Semiring** is just the log semiring, but with $\oplus = \min$. In other words, instead of adding the probabilities of two paths, we choose the best path:

$$a \oplus b = \min(a, b)$$

Mohri et al. (2001) formalize it like this: a **semiring** is $K = \{\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1}\}$ where $\mathbb{K}$ is a set of numbers.

Outline

Weighted Finite State Transducers



A **(Weighted) Finite State Transducer (WFST)** is a (W)FSA with two labels on every edge:

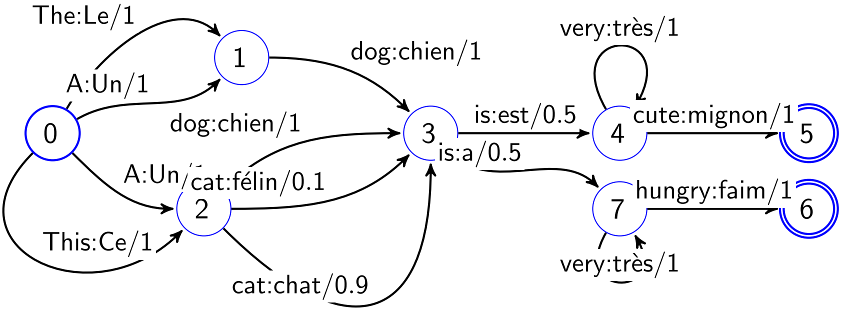
- An input label, $i \in \Sigma$, and
- An output label, $o \in \Omega$.

What it's for

- An **FST** specifies a mapping between two sets of strings.
  - The input set is $\mathcal{I} \subset \Sigma^*$, where Σ^* is the set of all strings containing zero or more letters from the alphabet Σ.
  - The output set is $\mathcal{O} \subset \Omega^*$.
  - For every $\vec{i} = [i_1, \ldots, i_T] \in \mathcal{I}$, the FST specifies one or more possible translations $\vec{o} = [o_1, \ldots, o_T] \in \mathcal{O}$.
- A **WFST** also specifies a probability mass function over the translations. The example on the previous slide was normalized to compute a joint pmf $p(\vec{i}, \vec{o})$, but other WFSAs might be normalized to compute a conditional pmf $p(\vec{o}|\vec{i})$, or something else.

Normalizing for Conditional Probability

Here is a WFST whose weights are normalized to compute $p(\vec{o}|\vec{i})$:



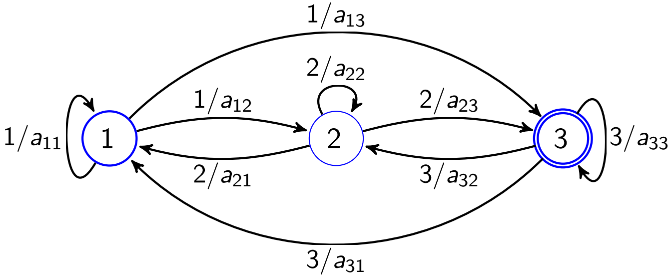
Normalizing for Conditional Probability

Normalizing for **conditional probability** allows us to separately represent the two parts of a hidden Markov model.

1. The transition probabilities, a_{ij}, are the weights on a WFSA.
2. The observation probabilities, $b_j(\vec{x}_t)$, are the weights on a WFST.

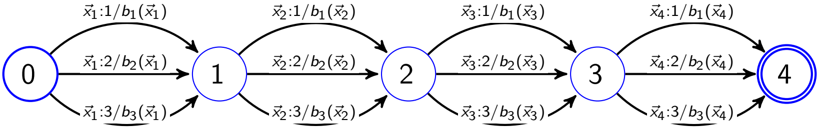
WFSA: Symbols on the edges are called PDFIDs

It is no longer useful to say that "the labels on the edges are the state numbers." Instead, let's call them **pdfids**.



Observation Probabilities as Conditional Edge Weights

Now we can create a new WFST whose **output symbols are pdfids** j, whose **input symbols are observations**, $\vec{x}_t$, and whose **weights are the observation probabilities,** $b_j(\vec{x}_t)$.



Hooray! We've almost re-created the HMM!

So far we have:

- You can create a WFSA whose weights are the transition probabilities.
- You can create a WFST whose weights are the observation probabilities.

Here are the problems:

1. How can we combine them?
2. Even if we could combine them, can this do anything that an HMM couldn't already do?

Outline

Composition

The main reason to use WFSTs is an operator called "composition." Suppose you have

1. A WFST, R, that translates strings $a \in \mathcal{A}$ into strings $b \in \mathcal{B}$ with joint probability $p(a, b)$.
2. Another WFST, S, that translates strings $b \in \mathcal{B}$ into strings $c \in \mathcal{C}$ with conditional probability $p(c|b)$.

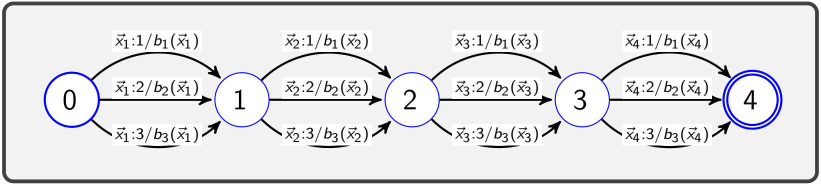
The operation $T = R \circ S$ gives you a WFST, T, that translates strings $a \in \mathcal{A}$ into strings $c \in \mathcal{C}$ with joint probability

$$p(a, c) = \sum_{b \in \mathcal{B}} p(a, b) p(c|b)$$

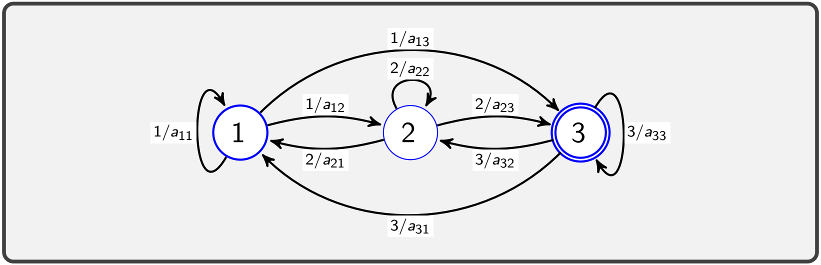
The WFST Composition Algorithm

1. **Initialize:** The initial state of T is a pair, $i_T = (i_R, i_S)$, encoding the initial states of both R and S.
2. **Iterate:** While there is any state $q_T = (q_R, q_S)$ with edges $(e_R = a : b, e_S = b : c)$ that have not yet been copied to e_T,
   1. Create a new edge e_T with next state $n[e_T] = (n[e_R], n[e_S])$ and labels $i[e_T] : o[e_T] = i[e_R] : o[e_S] = a : c$.
   2. If an edge with the same $n[e_T]$, $i[e_T]$, and $o[e_T]$ already exists, then update its weight:
   $$w[e_T] = w[e_T] \oplus (w[e_R] \otimes w[e_S])$$
   3. If not, create a new edge with
   $$w[e_T] = w[e_R] \otimes w[e_S]$$
3. **Terminate:** A state $q_T = (q_R, q_S)$ is a final state if both q_R and q_S are final states.

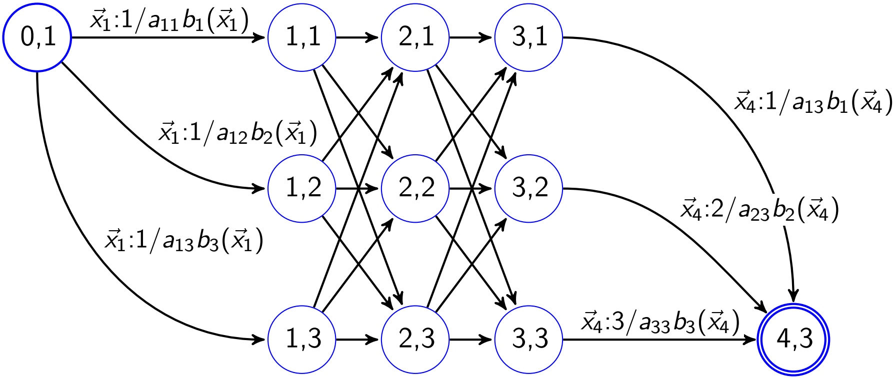
Composition Example: HMM



$\circ$



Composition Example: HMM



Outline

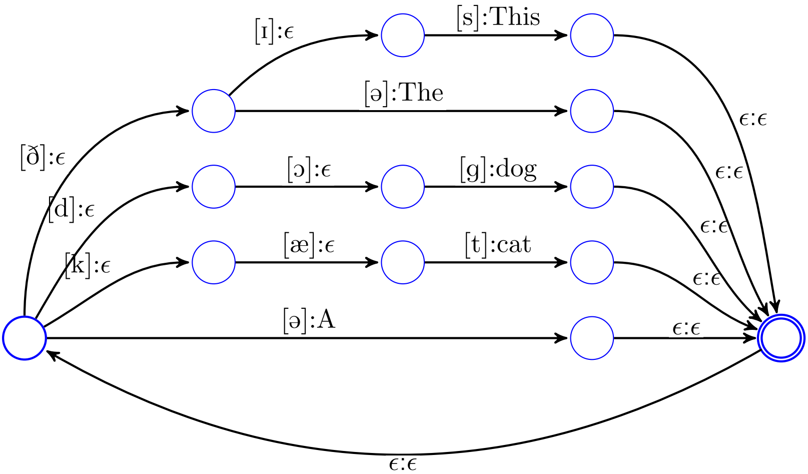
Doing Useful Stuff: The Epsilon Transition

- There's only one more thing you need to do useful stuff: nothing.
- To be more precise: we can use the label ϵ (pronounced "epsilon") to mean "nothing at all."

Example: Epsilon Transitions in the Pronlex

- A "pronlex" (pronunciation lexicon) is a WFST that maps from phoneme strings to words.
- A "phoneme string" is a sequence of many labels. A word is just one label. The extra labels in the output side of the WFST all use ϵ, to mean that they don't generate any extra output string.

Example Pronlex



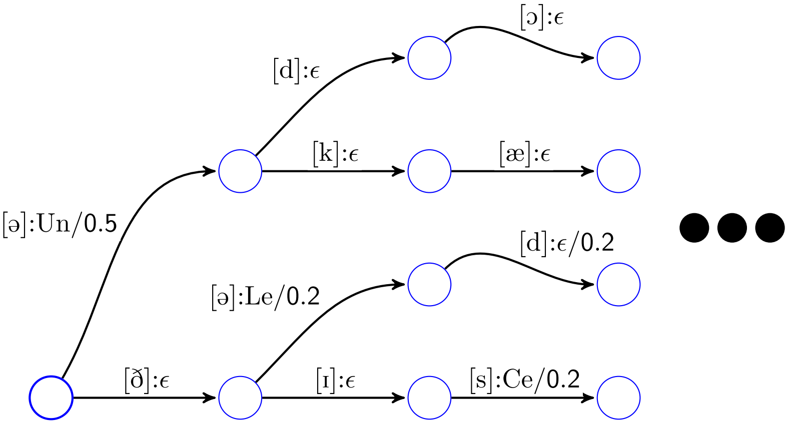
Example: Speech-to-Text Translation

- For example, suppose you have some English speech. You'd like to convert it to French text.
- Suppose you have an English pronlex, L, that maps English phonemes to words.
- You also have a translator, G, that maps English words to French words.
- Then

$$T = L \circ G$$

maps from English phonemes to French words.

Example: Speech-to-Text Translation



Example: Speech-to-Text Translation

Suppose you have:

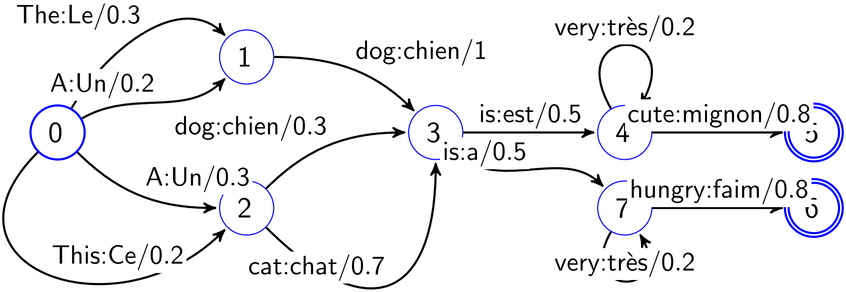
- Observer, B, maps from $\vec{x}_t$ to j, with weights $b_j(\vec{x}_t)$.
- HMM, H, maps from i and j to phonemes, with weights a_{ij}.
- Pronlex, L, maps from phonemes to English words.
- Grammar, G, maps from English words to French words.

Then the translation of audio frames into French words is given by

$$B \circ H \circ L \circ G$$

Outline

Weighted Finite State Transducers



A **(Weighted) Finite State Transducer (WFST)** is a (W)FSA with two labels on every edge:

- An input label, $i \in \Sigma$, and
- An output label, $o \in \Omega$.

The WFST Composition Algorithm

$$T = R \circ S$$

1. **Initialize:** The initial state of T is a pair, $i_T = (i_R, i_S)$, encoding the initial states of both R and S.
2. **Iterate:** Each edge $e_T = (e_R, e_S)$:
   - Starts at $p[e_T] = (p[e_R], p[e_S])$
   - Has the edge label $i[e_R] : o[e_S]$.
   - Ends at $n[e_T] = (n[e_R], n[e_S])$.
   - Has the weight $w[e_T] = w[e_R] \otimes w[e_S]$, possibly summed $(\oplus)$ over nondeterministic (e_R, e_S) pairs.
3. **Terminate:** A state $q_T = (q_R, q_S)$ is a final state if both q_R and q_S are final states.