Notes on Weighted Finite-State Automata

Chris Dyer and Noah A. Smith

September 24, 2015

1 Definitions

A **weighted finite state automaton** (WFSA) is a tuple consisting of:

- Q, a finite set of states;
- Σ, a finite vocabulary of symbols;
- $I \subseteq Q$, a set of initial states;
- $F \subseteq Q$, a set of final states;
- $E \subseteq (Q \times \Sigma \times Q)$, a set of of tuples (current state, symbol, next state) each representing a transition (compare δ in the DFSA definition we gave before);
- λ, a function from initial states to weights ($\lambda : I \to \mathbb{R}_{\geq 0}$);
- w, a function from transitions to weights ($w : E \to \mathbb{R}_{\geq 0}$); and
- ρ, a function of final states to weights ($\rho : F \to \mathbb{R}_{\geq 0}$).

If the score of a transition, starting state, or final state is zero, that means it is "not allowed." Thus, there is a certain amount of redundancy in this definition (e.g., all states could be I, but the "real" non-initial states have a weight of 0). Other definitions that reduce redundancy are sometimes given, but this one is relatively standard and simplifies the algorithm analysis we want to do. However, it is important to pay close attention to how WFSAs are defined each time you see them used in a paper (or homework or exam).

For a transition $e \in E$, $w(e)$ denotes its transition weight, $p(e)$ denotes the **previous state**, and $n(e)$ denote the **next state**. A **path** $\pi = e_1 e_2 \cdots e_\ell$ is a sequence of transitions such that $n(e_i) = p(e_{i+1})$ for $i \in [1, \ell)$. By abuse of notation, we let $n(\pi) = n(e_\ell)$ and $e(\pi) = p(e_1)$. Similarly, we define the **transition weight** of a path π to be

$$w(\pi) = w(e_1) \times w(e_2) \times \cdots \times w(e_\ell) \tag{1}$$

$$= \prod_{i=1}^{\ell} w(e_i). \tag{2}$$

And we define the **output weight** (score) of a path to be

$$\mathit{score}(\pi) = \lambda(p(\pi)) \times w(\pi) \times \rho(n(\pi)).$$

2 Determinism and ambiguity

A **deterministic** WFSA is a WFSA such that (1) there is exactly one $q \in I$ and $\rho(q) \neq 0$ and (2) for every $(q, \sigma) \in Q \times \Sigma$ there is at most one $r \in Q$ such that $e = (q, \sigma, r) \in E$ and $w(e) \neq 0$. An **unambiguous** WFSA is a WFSA such that there is at most one accepting path per string. This is a weaker property than being determinizeable.

3 Not all WFSAs can be determinized.

Intuitively, to determinize a WFSA F, we must create a new automaton such that every sequence $\boldsymbol{s} \in \Sigma^*$ has at most one path. To do this, we need to define what how the determinization algorithm should score such a path when F has multiple paths for $\boldsymbol{s}$. One standard answer to this is to say that the scores should be aggregated by **summation**. That is, for any ambiguous string $\boldsymbol{s} \in L(F)$, the sum of scores of all paths recognizing $\boldsymbol{s}$ should be the score of the single path in the determinized automaton. Alternatively, one may stipulate that the scores should be aggregated by **maximization**.

In contrast to nondeterministic FSAs (but similar to FSTs), not all WFSAs can be determinized! An example of WFSA that can not be determinized is shown in Figure 1. Assume scores are aggregated by summation. This machine is a union of two WFSAs, each accepting the language a^*. For any string a^n, the score of the single path in the desired determinized WFSA will be $(0.5)^n + (0.3)^n$. Why is this a problem? It is not hard to show that any deterministic WFSA that accepts a^* will have a "lollipop" typology. Consider paths accepting large-n strings a^n, such that $n = j + k\ell$ (for some values of j, k, ℓ). Such paths will first proceed along a pre-cyclic "stem" sequence of length j (call the score of this part p), then proceed to go around a length-ℓ cycle (the "head" of the lollipop) some k times (call the score of this part q), and finally stop in the same stopping state (with stopping weight r). So the scores of strings with length $a^{j+k\ell}$, for $k \in \{0, 1, \ldots\}$, will be $pq^k r$—that is, the determinized WFSA produces weights that decay *geometrically* with k. However, $(0.5)^n + (0.3)^n$ is not a geometric progression (proof by contradiction), therefore the given WFSA cannot be determinized.[1]

4 Not all unambiguous WFSAs can be determinized.

An example, building off the same line of reasoning, is shown in Figure 1 (Right).

5 Best path algorithm for nondeterministic WFSAs.

Given an ε-free nondeterministic WFSA $\mathcal{M} = (Q, \Sigma, I, F, E, \lambda, w, \rho)$, a sequence of symbols $\boldsymbol{s} = s_1 s_2 \cdots s_\ell \in \Sigma^\ell$, we want to find $\hat{\pi} = \hat{e}_1 \hat{e}_2 \cdots \hat{e}_\ell$, the best scoring path through the WFSA that produces $\boldsymbol{s}$. The first step is to calculate the table of α values ($\alpha : Q \times [0, \ell] \to \mathbb{R}_{\geq 0}$). Let $\alpha(q, t)$ be

[1] Thanks to Jason Eisner (personal communication) for this example.

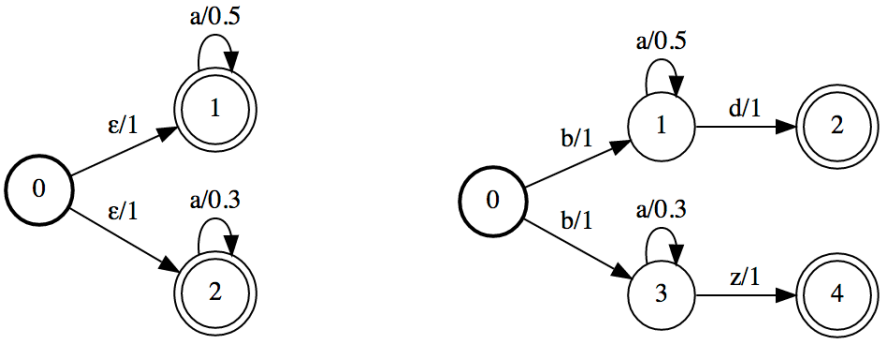

Figure 1: (Left) A WFSA that cannot be determinized. In this example, $\lambda(0) = \rho(1) = \rho(2) = 1$ and all other starting and stopping weights are 0. (Right) An **unambiguous** WFSA that cannot be determinized ($\rho(0) = \lambda(2) = \lambda(4) = 1$).

the weight of the best state sequence prefix ending in state q and emitting $s_1 s_2 \cdots s_t$. For $t = 0$,

$$\alpha(q, 0) = \begin{cases} \lambda(q) & \text{if } q \in I \\ 0 & \text{if } q \in Q - I \end{cases}$$

for $t > 0$,

$$\begin{aligned} \alpha(q,t) &= \max_{e_1 e_2 \cdots e_t : n(e_t) = q} \lambda(p(e_1)) \times \prod_{i=1}^{t} w(e_i) \quad \forall q \in Q, \forall t \in [1, \ell) \\ &= \max_{e_1 e_2 \cdots e_t : n(e_t) = q} \underbrace{\lambda(p(e_1)) \times \left(\prod_{i=1}^{t-1} w(e_i) \right)}_{= \alpha(q_{t-1}, t-1)} \times w(\langle n(e_{t-1}), s_t, q \rangle) \\ &= \max_{q_{t-1} \in Q} \alpha(q_{t-1}, t-1) \times w(\langle q_{t-1}, s_t, q \rangle) \end{aligned}$$

The second step is to **backtrack** from the end of the sequence to the beginning. Thus,

$$\hat{q}_\ell = \arg\max_{q \in F} \alpha(q, \ell) \times \rho(q), \tag{3}$$

and then for $t = \ell - 1$ to 0,

$$\hat{q}_t = \arg\max_{q \in Q} \alpha(q, t) \times w(q, s_{t+1}, \hat{q}_{t+1}).$$

This step can be simplified by keeping **backpointers** in step 1. Every time a "max" is computed, store the "arg max" state, which is basically remembering which state from timestep $t-1$ leads to the best path into a state at timestep i. Letting $\beta : Q \times \{1, \ldots, \ell\} \to Q$ be the backpointer function, we start the backtrace with Equation 3 and proceed for $t = \ell - 1$ to 0:

$$\hat{q}_t = \beta(\hat{q}_{t+1}, t+1)$$

We can then write the best path in terms of the transitions $\hat{e}_i = \langle \hat{q}_{i-1}, s_i, \hat{q}_i \rangle$.