Search and Weighted Finite-State Transducers

Dr. John McDonough

Spoken Language Systems
Saarland University

June 15, 2009
Introduction

- Last lecture, we considered the search process, whereby an automatic speech recognition system finds the most likely word hypothesis.
- We also considered the generation of word lattices, which are an efficient way of encoding multiple word hypotheses.
- All of this development was based on the formulation of statistical ASR as the problem of finding the shortest path through a graph.
- In this next lecture, we will consider how such a search graph can be constructed.

**Coverage:** Wölfel and McDonough (2009) Sections 7.2–7.3.
Definition: A semiring \( K = (\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1}) \) consists of a set \( \mathbb{K} \), an associative and commutative operation \( \oplus \), an associative operation \( \otimes \), the identity \( \bar{0} \) under \( \oplus \), and the identity \( \bar{1} \) under \( \otimes \). By definition, \( \otimes \) distributes over \( \oplus \) and

\[
\bar{0} \otimes a = a \otimes \bar{0} = \bar{0}.
\]

- A semiring is a ring that may lack negation. While this definition may seem excessively formal, it will prove useful in that operations on FSA can be defined in terms of the operations on an abstract semiring.
- Thereafter, the definitions of the several algorithms need not be modified when the semiring is changed.
- A simple example is the semiring of natural numbers \( (\mathbb{N}, +, \cdot, 0, 1) \).
Examples of Semirings

- In ASR we typically use one of two semirings, depending on the operation.
- The *tropical semiring* \((\mathbb{R}^+, \min, +, 0, 1)\), where \(\mathbb{R}^+\) denotes the set of non-negative real numbers, is useful for finding the shortest path through a search graph.
- The set \(\mathbb{R}^+\) is used in the tropical semiring because the hypothesis scores represent negative log-likelihoods.
- The two operations on weights correspond to the multiplication of two probabilities, which is equivalent to addition in the negative log-likelihood domain, and discarding all but the lowest weight, such as is done by the Viterbi algorithm.
Examples of Semirings (cont’d.)

- The log-probability semiring \((\mathbb{R}^+, \oplus_{\log}, +, 0, 1)\) differs from the tropical semiring only inasmuch as the min operation has been replaced with the log-add operation \(\oplus_{\log}\), which is defined as
  \[
  a \oplus_{\log} b \triangleq -\log(e^{-a} + e^{-b}).
  \]

- The log-probability semiring is typically used for the weight pushing equivalence transformation discussed later.

- In addition to the tropical and log-probability semiring which clearly operate on real numbers, it is also possible to define the string semiring wherein the weights are in fact strings, and the operation \(\oplus = \wedge\) corresponds to taking the longest common substring, while \(\odot = \cdot\) corresponds to concatenation of two strings.

- Hence, the string semiring can be expressed as
  \[
  K_{\text{string}} = (\Sigma^* \cup \infty, \wedge, \cdot, \infty, \epsilon).
  \]
We now define our first automaton, the **weighted finite-state acceptor** (WFSA) $A = (\Sigma, Q, E, i, F, \lambda, \rho)$ on the semiring $K = (\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ consists of

- an alphabet $\Sigma$,
- a finite set of states $Q$,
- a finite set of transitions $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \mathbb{K} \times Q$,
- a initial state $i \in Q$ with weight $\lambda$,
- a set of end states $F \subseteq Q$,
- and a function $\rho$ mapping from $F$ to $\mathbb{R}^+$.

A transition or edge $e = (p[e], l[e], w[e], n[e]) \in E$ consists of

- a previous state $p[e]$,
- a next state $n[e]$,
- a label $l[e] \in \Sigma$, and
- a weight $w[e] \in \mathbb{K}$.

A final state $n \in F$ may have an associated weight $\rho(n)$. 

**Weighted Finite-State Acceptors**
A simple WFSA is shown in Figure 1.

This acceptor would assign the input string “red white blue” a weight of $0.5 + 0.3 + 0.2 + 0.8 = 1.8$. 
Successful Path

- As already explained, speech recognition will be posed as the problem of finding the shortest path through a WFSA, where the length of a path will be determined by a combined AM and LM score.

- Hence, we will require a formal definition of a path: A path $\pi$ through an acceptor $A$ is a sequence of transitions $e_1 \cdots e_K$, such that

$$n[e_k] = p[e_{k+1}] \quad \forall \ k = 1, \ldots, K - 1.$$ 

- A successful path $\pi = e_1 \cdots e_K$ is a path from the initial state $i$ to an end state $f \in F$. 

A weighted finite-state acceptor is so-named because it accepts strings from \( \Sigma^* \), the Kleene closure of the alphabet \( \Sigma \), and assigns a weight to each accepted string.

A string \( s \) is accepted by \( A \) iff there is a successful path \( \pi \) labeled with \( s \) through \( A \).

The label \( l[\pi] \) for an entire path \( \pi = e_1 \cdots e_K \) can be formed through the concatenation of all labels on the individual transitions:

\[
l[\pi] \triangleq l[e_1] \cdots l[e_K].
\]

The weight \( w[\pi] \) of a path \( \pi \) can be represented as

\[
w[\pi] \triangleq \lambda \otimes w[e_1] \otimes \cdots \otimes w[e_K] \otimes \rho(n[e_K]),
\]

where \( \rho(n[e_K]) \) is the final weight.

Typically, \( \Sigma \) contains \( \epsilon \), which, as stated before, denotes the null symbol.
We now generalize our notion of a WFSA in order to consider machines that translate one string of symbols into a second string of symbols from a different alphabet along with a weight.

**Weighted finite-state transducer:** A WFST $T = (\Sigma, \Omega, Q, E, i, F, \lambda, \rho)$ on the semiring $\mathbb{K}$ consists

- of an *input alphabet* $\Sigma$,
- an *output alphabet* $\Omega$,
- a set of states $Q$,
- a set of transitions $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Omega \cup \{\epsilon\}) \times \mathbb{K} \times Q$
- a initial state $i \in Q$ with weight $\lambda$,
- a set of final states $F \subseteq Q$,
- and a function $\rho$ mapping from $F$ to $\mathbb{R}^+$. 
A transition $e = (p[e], l_i[e], l_o[e], w[e], n[e]) \in E$ consists of

- a previous state $p[e]$,
- a next state $n[e]$,
- an input symbol $l_i[e]$,
- an output symbol $l_o[e]$, and
- a weight $w[e]$.
A WFST, such as that shown in Figure 2 maps an input string to an output string and a weight.

For example, such a transducer would map the input string “red white blue” to the output string “yellow blue red” with a weight of $0.5 + 0.3 + 0.2 + 0.8 = 1.8$.

It differs from the WFSA only in that the edges of the WFST have two labels, an input and an output, rather than one.

A string $s$ is accepted by a WFST $T$ iff there is a successful path $\pi$ labeled with $l[\pi] = s$.

The weight of this path is $w[\pi]$, and its output string is

$$l_0[\pi] \triangleq l_0[e_1] \cdot \cdots \cdot l_0[e_K].$$

Any $\epsilon$–symbols appearing in $l_0[\pi]$ can be ignored.
Figure: A simple weighted finite-state transducer.
Weighted Composition

Consider a transducer $S$ which maps an input string $u$ to an output string $v$ with a weight of $w_1$, and a transducer $T$ which maps input string $v$ to output string $y$ with weight $w_2$.

The *composition* $R = S \circ T$ of $S$ and $T$ maps string $u$ directly to $y$ with weight $w = w_1 \otimes w_2$.

We will adopt the convention that the components of particular transducer are denoted by subscripts; e.g., $Q_R$ denotes the set of states of the transducer $R$.

In the absence then of $\epsilon$–transitions, the construction of such a transducer $R$ is straightforward.

It entails simply pairing the output symbols on the transitions of a node $n_S \in Q_S$ with the input symbols on the transitions of a node $n_T \in Q_T$, beginning with the initial nodes $i_S$ and $i_T$. 
From the figure, it is clear that the transition from State 0 labeled with $a:b/0.5$ in $S$ has been paired with the transition from State 0 labeled with $b:c/0.2$ in $T$, resulting in the transition labeled $a:c/0.7$ in $R$. After each successful pairing, the new node $n_R = (n_S, n_T)$ is placed on a queue to eventually have its adjacency list expanded.

**Figure:** Weighted composition of two simple transducers.
Composition is not Local

- The pairing of the transitions of $n_S$ with those of $n_T$ is \textit{local}, inasmuch as it only entails the consideration of the adjacency lists of two nodes at a time.
- This fact provides for the so-called lazy implementation of weighted composition.
- As $R$ is constructed it can so happen that nodes are created that do not lie on a successful path; i.e., from such a node, there is no path to an end state.
- Such nodes are typically removed or \textit{purged} from the graph as a final step.
- It is worth noting, however, that this purge step is \textit{not} a local operation as it is necessary to consider the entire transducer $R$ in order to determine if any given node is on a successful path.
Composition Filter

- When $\epsilon$–symbols are introduced, composition becomes more complicated, as it is necessary to specify when an $\epsilon$–symbol on the output of a transition in $n_S$ can be combined with an $\epsilon$–symbol on the input of $n_T$.
- In order to avoid the creation of redundant paths through $R$, it is necessary to replace the composition $S \circ T$ with $S \circ V \circ T$, where $V$ is a filter.
- Hence, a node $n_R \in Q_R$ is specified by a triple $(n_S, n_T, f)$, where $f \in \{0, 1, 2\}$ is an index indicating the state of $V$.
- In effect, the filter specifies that after a lone $\epsilon$-transition on either the input or output side is taken, placing the filter in State 1 or State 2 respectively, an $\epsilon$-transition on the other side may not be taken until a non-$\epsilon$ match between input and output occurs, thereby returning the filter to State 0 over one of the edges labeled with $x:x$. 
Composition Filter Schematic

Figure: Filter used during composition with $\epsilon$-symbols.
We now present the first of a series of equivalence transformations.

We begin with a pair of definitions.

Equivalent: Two WFSAs are equivalent if for any accepted input sequence, they produce the same weight. Two WFSTs are equivalent if for any accepted input sequence they produce the same output sequence and the same weight.

Deterministic: A WFST is deterministic if at most one transition from any node is labeled with any given input symbol.
Advantages of Deterministic Transducers

- It is typically advantageous to work with deterministic WFSTs, because there is at most one path through the transducer labeled with a given input string.
- This implies that the effort required to learn if a given string is accepted by a transducer, and to calculate the associated weight and output string, is linear with the length of the string, and does not depend on the size of the transducer.
- More to the point, it implies that the acoustic likelihood that must be calculated when taking a transition during decoding need only be calculated once.
- This has a decisive impact on the efficiency of the search process inherent in ASR.
Thus we are led to consider our first equivalence operation, *determinization*, which produces a deterministic transducer $\tau_2$ that is equivalent to some given transducer $\tau_1$.

**String-to-weight subsequential transducer:** A *string-to-weight subsequential transducer* on the semiring $\mathbb{K}$ is an 8–tuple $\tau = (\Sigma, Q, i, F, \delta, \sigma, \lambda, \rho)$ consisting of

- of an *input alphabet* $\Sigma$,
- a set of states $Q$,
- an initial state $i \in Q$ with weight $\lambda \in \mathbb{R}^*$,
- a set of final states $F \subseteq Q$,
- a transition function $\delta$ mapping $Q \times \Sigma$ to $Q$,
- a output function $\sigma$ mapping $Q \times \Sigma$ to $\mathbb{R}^+$,
- and a final weight function $\rho$ mapping from $F$ to $\mathbb{R}^+$.

The determinization algorithm for weighted automata is similar to the classical powerset construction for the determinization of conventional automata.
Details of Weighted Determinization

- The states in the determinized transducer correspond to *subsets* of states in the original transducer, together with a residual weight.
- The initial state $i_2$ in $\tau_2$ corresponds only to the initial state $i_1$.
- The subset of states together with their residual weights that can be reached from $i_1$ through a transition with the input label $a$ then form a state in $\tau_2$.
- As there may be several transitions with input label $a$ having different weights, the output of the transition from $i_2$ labeled with $a$ can only have the minimum weight of all transitions from $i_1$ labeled with $a$.
- The *residual weight* above this minimum must then be carried along in the definition of the subset to be applied later.
- Each time a new state in $\tau_2$, consisting of a subset of the states of $\tau_1$ together with their residual weights, is defined, it is added to a queue $Q$, so that it will eventually have its adjacency list.
In order to clearly describe such an algorithm, let us define the following sets:

- $\Gamma(q_2, a) = \{ (q, x) \in q_2 : \exists e = (q, a, \sigma(e), n_1(e)) \in E_1 \}$ denotes the set of pairs $(q, x)$ which are elements of $q_2$ where $q$ has at least one edge labeled with $a$;

- $\gamma(q_2, a) = \{ (q, x, e) \in q_2 \times E_1 : e = (q, a, \sigma_1(e), n_1(e)) \in E_1 \}$ denotes the set of triples $(q, x, e)$ where $(q, x)$ is a pair in $q_2$ such that $q$ admits a transition with input label $a$;

- $\nu(q_2, a) = \{ q' \in Q_1 : \exists (q, x) \in q_2, \exists e = (q, a, \sigma_1[e], q') \in E_1 \}$ is the set of states $q'$ in $Q_1$ that can be reached by transitions labeled with $a$ from the states of subset $q_2$. 
Pseudocode for Weighted Determinization

Pseudocode for the complete algorithm is provided in the listing below.

```python
00 def determinize(τ₁, τ₂):
01     F₂ ← ∅
02     i₂ ← i₁
03     λ₂ ← λ₁
04     Q ← { i₂ }
05     while |Q| > 0:
06         pop q₂ from Q
07         if ∃ (q, x) ∈ q₂ such that q ∈ F₁:
08             F₂ ← F₂ ∪ {q₂}
09             ρ₂(q₂) ← ⊕ x ⊙ ρ₁(q)
10                 q∈F₁,(q,x)∈ q₂
11         for a such that Γ(q₂, a) ≠ ∅:
12             σ₂(q₂, a) ← (q, x)∈Γ(q₂, a) [ x ⊕ ⊕ e=(q,a,σ₁[e],n₁[e])∈ E₁ ]
13             δ₂(q₂, a) ← (q, x, t)∈ γ(q₂, a),n₁[e]=q
14         if δ₂(q₂, a) ∉ Q₂:
15             Q₂ ← Q₂ ∪ { δ₂(q₂, a) }
16             push δ₂(q₂, a) on Q
```
Pseudocode for Weighted Determinization

- The weighted determinization algorithm is perhaps most easily understood by specializing all operations for the tropical semiring.
- This implies $\oplus$ is replaced by min and $\odot$ is replaced by $\oplus$.
- The algorithm begins by initializing the set $F_2$ of final states of $\tau_2$ to $\emptyset$ in Line 01, and equating the initial state and weight $i_2$ and $\lambda_2$ respectively to their counterparts in $\tau_1$ in Lines 02–03.
- The initial state $i_2$ is pushed onto the queue $Q$ (Line 04).
- In Line 05, the next subset $q_2$ to have its adjacency list expanded is popped from $Q$.
- If $q_2$ contains one or more pairs $(q, x)$ comprised of a state $q \in Q_1$ and residual weight $x$ whereby $q \in F_1$, then $q_2$ is added to the set of final states $F_2$ in Line 08 and assigned a final weight $\rho_2(q_2)$ equivalent to the minimum of all $x \odot \rho_1(q)$ where $(q, x) \in q_2$ and $q \in F_1$ in Line 09.
The next step is to begin expanding the adjacency list of $q_2$ in Line 10, which specifies that the input symbols on the edges of the adjacency list of $q_2$ is obtained from the union of the input symbols on the adjacency lists of all $q$ such that there exists $(q, x) \in q_2$.

In Line 11, the weight assigned the edge labeled with $a$ on the adjacency list of $q_2$ is obtained by considering each $(q, x) \in \Gamma(q_2, a)$ and finding the edge with the minimum weight on the adjacency list of $q$ that is labeled with $a$ and multiplying this minimum weight with the residual weight $x$.

Thereafter, the minimum of all the weights $x$ is taken for all pairs $(q, x)$ in $\Gamma(q_2, a)$.

In Line 12, the identity of the new subset of $(q, x) \in Q_2$ is determined and assigned to $\delta(q_2, a)$.

If this new subset is previously unseen, it is added to the set of states of $\tau_2$ in Line 14 and pushed onto the queue $Q$. 
A simple example of weighted determinization is shown in Figure 5.

The two WFSTs in the figure are equivalent over the tropical semiring in that they both accept the same input strings, and for any given input string, produce the same output string and the same weight.

For example, the original transducer will accept the input string \(aba\) along either of two successful paths, namely, using the state sequence \(0 \rightarrow 1 \rightarrow 3 \rightarrow 3\) or the state sequence \(0 \rightarrow 1 \rightarrow 4 \rightarrow 3\).

Both sequences produce the string \(ab\) as output, but the former yields a weight of \(0.1 + 0.4 + 0.6 = 1.1\), while the latter assigns a weight of \(0.1 + 0.3 + 0.5 = 0.9\).

Hence, given that these WFSTs are defined over the tropical semiring, the final weight assigned to the input \(aba\) is 0.9, the minimum of the weights along the two successful paths.
Before Determinization

After Determinization

There is, however, but a single sequence labeled with this input, namely, that with the state sequence $0 \rightarrow 1 \rightarrow 4 \rightarrow 5$, which produces a weight of $0.1 + 0.3 + 0.5 = 0.9$. 

Diagram of Weighted Determinization
In this next lecture, we considered how such a search graph can be constructed using the theory of weighted finite-state transducers. This yields a very efficient, but can be costly in terms of main memory (RAM). Recently, hybrid decoding techniques have been developed that yield nearly the efficiency of WFST search without the tremendous RAM requirements.