Generalized Path Algorithms and Semirings

Algorithms for NLP

October 2, 2014
Outline

• HMMs with Higher Markov Order
• Generalizations of “Best Paths”
• Features on WFSAs [time permitting]
• Other shortest paths algorithms [time permitting]
HMMs with Higher Markov Order
Back to Part of Speech Tagging

After paying the medical bills, Frances was nearly broke.

```
RB  VBG  DT  JJ  NNS,  NNP  VBZ  RB  JJ
```

- If the HMM's states are POS tags, then:
  - we have weights for these word/tag pairs:
    
    ```
    After-RB paying-VBG the-DT medical-JJ bills-NNS -, Frances-NNP was-VBZ nearly-RB broke-JJ .-
    ```

  - we have weights for these consecutive pairs:
    
    ```
    RB-VBG  VBG-DT  DT-JJ  JJ-NNS  NNS-,  ,-NNP  NNP-VBZ  VBZ-RB  RB-JJ  JJ-
    ```

- What if we want *triples* of POS tags?
“Trigram HMM”

• Condition each POS tag on the previous *two*.  
  – Like a trigram model on POS tags.

• What does this imply for our state space?

• For the runtime of Viterbi?
Trigram HMM States

• In the solution we saw last week, each state encodes the POS of the current word.
• Transition weight $\gamma(l_{i+1} | l_i)$ looks at current state to consider next state.
• $|Q| = |L| + 1$

• In a trigram (or “second-order”) HMM, states must encode the POS of the previous word, too.
• Transition weight is now $\gamma(l_i, l_{i+1} | l_{i-1}, l_i)$.
• $|Q| = |L|^2 + |L| + 1$, but only $|L| + 1$ valid transitions from each state.
Decoupling Labels and States

• States can “remember” histories of more than one label.
• Trigram HMMs tend to perform better for POS tagging, NER, and chunking.
• If we have \(|L|\) labels, an “\(n\)-gram HMM” is going to have \(O(|L|^{n-1})\) states.
  – Naïve Viterbi: \(O(|L|^{2(n-1)} m)\) for length \(m\) sequence
  – Clever Viterbi: \(O(|L|^n m)\)
Generalizations of “Best Paths”
# Weighted Finite-State Automaton

<table>
<thead>
<tr>
<th>Element</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q$</td>
<td>Finite set of states</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>Finite vocabulary</td>
</tr>
<tr>
<td>$I \subseteq Q$</td>
<td>Set of initial states</td>
</tr>
<tr>
<td>$F \subseteq Q$</td>
<td>Set of final states</td>
</tr>
<tr>
<td>$E \subseteq (Q \times (\Sigma \cup {\varepsilon}) \times Q)$</td>
<td>Set of transitions (edges)</td>
</tr>
<tr>
<td>$\lambda : I \rightarrow \mathbb{R}_{\geq 0}$</td>
<td>Initial weights</td>
</tr>
<tr>
<td>$\rho : F \rightarrow \mathbb{R}_{\geq 0}$</td>
<td>Final weights</td>
</tr>
<tr>
<td>$w : E \rightarrow \mathbb{R}_{\geq 0}$</td>
<td>Transition weights</td>
</tr>
</tbody>
</table>
Preliminaries

• Define the set of paths from state $r \in Q$ to $r' \in Q$ as $P(r, r')$

• This may be generalized to subsets $R, R' \subseteq Q$ as $P(R, R') = \bigcup_{r \in R, r' \in R'} P(r, r')$

• Thus, the set of all possible computation paths in a WFSA is $P(I, F)$
Preliminaries

• Given a sequence $s \in \Sigma^*$, states $r$ and $r'$ in $Q$, let the set of paths from $r$ to $r'$ generating $s$ be designated as $P(r, s, r')$

• Define $P(R, s, R')$ as above.
Viterbi Revisited

• Viterbi is a **shortest path algorithm**
  – Find the shortest path from any start state to any end state that generates a sequence $s$

• Given the above definitions and

\[ s \in \Sigma^*, \quad \mathcal{M} = (Q, \Sigma, I, F, E, \lambda, \rho, w) \]

\[ \text{Viterbi}(s, \mathcal{M}) = \max_{\pi} P(I, s, F) \]
Viterbi: Another View

• Given an unweighted “input” FSA and a WFSA

• Perform intersection (or composition)
  – Use unweighted algorithm and copy the weights from the edges/states in the input weighted machine to the output machine

• What does this compute?

Paths in $P(I', F')$ are in a one-to-one correspondence with paths in $P(I, s, F)$
Lattice
• Given $s \in \Sigma^*$, $\mathcal{M} = (Q, \Sigma, I, F, E, \lambda, \rho, w)$
• Intersect/compose to get
  $\mathcal{M}' = (Q', \Sigma', I', F', E', \lambda', \rho', w')$
• The path/score we want is

\[
\text{Viterbi}(s, \mathcal{M})
= \max_{\pi \in P(I, s, F)} \lambda(p(\pi)) \times w(\pi) \times \rho(n(\pi))
= \max_{\pi' \in P(I', F')} \lambda'(p(\pi')) \times w'(\pi') \times \rho'(n(\pi'))
= \min_{\pi' \in P(I', F')} -\lambda'(p(\pi')) \times w'(\pi') \times \rho'(n(\pi'))
\]

What is this problem?
Shortest Path Problems

• Viterbi is a shortest path problem
• Shortest path problems
  – Shortest path from state \( q \) to \( r \)
  – Shortest path from any starting state in \( I \) to any ending state in \( F \)
  – Shortest path between all pairs of states (related)
• Numerous algorithms exist for solving these (Dijkstra’s, A*, etc.)
Recall

• When we introduced **weighted determinization**, we made a decision about how to aggregate path weights
  – Addition
  – Max
• Goal: develop a fast algorithm to compute **arbitrary aggregations** over all paths
• That is: **generalized best path problems**
Aggregating over Paths

\[ \mathcal{M} = (Q, I, F, E, \lambda, w, \rho) \]

\[ \text{Viterbi}(\mathcal{M}) = \max_{\pi} \lambda(p(\pi)) \times w(\pi) \times \rho(n(\pi)) \]

Aggregation

Multiplication
Chalk Talk