Decoding and WFSTs

Steve Renals

Automatic Speech Recognition – ASR Lecture 13
9 March 2017
The Search Problem in ASR

- **Find the most probable word sequence** \( \hat{W} = w_1, w_2, \ldots, w_M \) given the acoustic observations \( X = x_1, x_2, \ldots, x_n \):

  \[
  \hat{W} = \arg \max_W P(W|X) = \arg \max_W p(X|W) P(W)
  \]

  - acoustic model
  - language model

- Words are composed of state sequences so this problem corresponds to finding the most probable allowable state sequence (given the constraints of pronunciation lexicon and language model) - **Viterbi decoding**

- In a large vocabulary task evaluating all possible word sequences in infeasible (even using an efficient exact algorithm)
  - Reduce the size of the search space through pruning unlikely hypotheses
  - Eliminate repeated computations
Connected Word Recognition

- The number of words in the utterance is not known
- Word boundaries are not known: $V$ words may potentially start at each frame
The number of words in the utterance is not known.

Word boundaries are not known: $V$ words may potentially start at each frame.

speech: “the cat ate the canary”
Time Alignment Path

<table>
<thead>
<tr>
<th>Time</th>
<th>States</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Word1</td>
</tr>
<tr>
<td></td>
<td>Word2</td>
</tr>
<tr>
<td></td>
<td>Word3</td>
</tr>
<tr>
<td></td>
<td>Word4</td>
</tr>
</tbody>
</table>

Time

States

Word1 Word2 Word3 Word4
Backpointer array keeps track of word sequence for a path:
backpointer[word][wordStartFrame] = (prevWord, prevWordStartFrame)

Backtrace through backpointer array to obtain the word sequence for a path
Incorporating a bigram language model

Trigram or longer span models require a word history.
Computational Issues

- Viterbi decoding performs an exact search in an efficient manner
- Exact search is not possible for large vocabulary tasks
  - Cross-word triphones need to be handled carefully since the acoustic score of a word-final phone depends on the initial phone of the next word
  - Long-span language models (e.g., trigrams) greatly increase the size of the search space
- Solutions:
  - Beam search (prune low probability hypotheses)
  - Dynamic search structures
  - Multipass search (→ two-stage decoding)
  - Best-first search (→ stack decoding / A* search)
  - An alternative approach: Weighted Finite State Transducers (WFST)
Weighted Finite State Transducers

- Used by Kaldi
- Weighted finite state automaton that transduces an input sequence to an output sequence (Mohri 2008)
- States connected by transitions. Each transition has
  - input label
  - output label
  - weight
Weighted Finite State Acceptors

![Diagram of Weighted Finite State Acceptors](image_url)
Weighted Finite State Transducers

Acceptor

Transducer
Weighted Finite State Transducers

**Acceptor**

```
0  d/1  1  ey/0.5  ae/0.5  2  t/0.3  dx/0.7  3  ax/1  4
```

**Transducer**

```
0  d: data/1  1  ey:ε/0.5  ae:ε/0.5  2  t:ε/0.3  dx:ε/0.7  3  ax: ε/1  4
```

---

The diagrams illustrate weighted finite-state acceptors and transducers. The acceptor diagram shows transitions labeled with symbols and weights, while the transducer diagram includes additional transitions indicating weight assignments for specific operations. The acceptor and transducer are related through the operations implemented by these automata, which can be used in speech processing tasks such as pronunciation lexicon composition and decoding.
**Composition** Combine transducers at different levels. For example if $G$ is a finite state grammar and $L$ is a pronunciation dictionary then $L \circ G$ transduces a phone string to word strings allowed by the grammar.

**Determinisation** Ensure that each state has no more than a single output transition for a given input label.

**Minimisation** transforms a transducer to an equivalent transducer with the fewest possible states and transitions.
Applying WFSTs to speech recognition

- Represent the following components as WFSTs:

<table>
<thead>
<tr>
<th>transducer</th>
<th>input sequence</th>
<th>output sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>G  \text{word-level grammar}</td>
<td>words</td>
<td>words</td>
</tr>
<tr>
<td>L  \text{pronunciation lexicon}</td>
<td>phones</td>
<td>words</td>
</tr>
<tr>
<td>C  \text{context-dependency}</td>
<td>CD phones</td>
<td>phones</td>
</tr>
<tr>
<td>H  \text{HMM}</td>
<td>HMM states</td>
<td>CD phones</td>
</tr>
</tbody>
</table>

- Composing $L$ and $G$ results in a transducer $L \circ G$ that maps a phone sequence to a word sequence.

- $H \circ C \circ L \circ G$ results in a transducer that maps from HMM states to a word sequence.
Figure 17: Recognition transducer construction: (a) gramm

Springer Handbook on Speech Processing and Speech Communication 24

0 ... e x i c o n ˜L ,( c ) ˜L ◦ G ,( d ) det( ˜L ◦ G ) ,( e )
min tropical (det( ˜L ◦ G )) ,( f ) min log (det( ˜L ◦ G )) .
$L \circ G$, $\det(L \circ G)$, $\min(\det(L \circ G))$
Context dependency transducer $C$

Context-independent “string”

```
0 1 2 3 4 5
x  y  x  x  y
```

Context-dependency transducer (weights not shown)

```
0 1 2 3 4 5
x:x/e_y  y:y/x_x  x:x/y_x  x:x/x_y  y:y/x_e
```

(x/e_y – x with left context e (start/end) and right context y)
Decoding using WFSTs

- We can represent the HMM acoustic model, pronunciation lexicon and n-gram language model as four transducers: H, C, L, G
- Combining the transducers gives an overall “decoding graph” for our ASR system – but minimisation and determination means it is much smaller than naively combining the transducers
- But it is important in which order the algorithms are combined otherwise the transducers may “blow-up” – basically after each composition, first determinise then minimise
- In Kaldi, ignoring one or two details

\[ HCLG = \min(\det(H \circ \min(\det(C \circ \min(\det(L \circ G)))))) \]

Decoding and WFSTs in Kaldi –