Large vocabulary ASR

Peter Bell

Automatic Speech Recognition – ASR Lecture 9
10 February 2020
HMM Speech Recognition

Recorded Speech

Signal Analysis

Acoustic Model

Language Model

Search Space

Decoded Text (Transcription)

Training Data
The Search Problem in ASR

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

$$\hat{W} = \arg \max_W P(W|X)$$

$$= \arg \max_W p(X|W) P(W)$$

- Use pronunciation knowledge to construct HMMs for all possible words
- Finding the most probable state sequence allows us to recover the most probable word sequence
- *Viterbi decoding* is an efficient way of finding the most probable state sequence, but even this is infeasible as the vocabulary gets very large or when a stronger language model is used
Recap: the word HMM

HMM naturally generates an alignment between hidden states and observation sequence

"right"
Viterbi algorithm finds the best path through the trellis – giving the highest $p(X, Q)$. 

Simplified version with one state per phone

\[
\begin{array}{ccccccc}
& & & & & & \text{time} \\
\text{state} & & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
\end{array}
\]

\[
\begin{array}{ccccccc}
& & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
\end{array}
\]
Isolated word recognition

“right”

“ten”

“pin”
Viterbi algorithm: isolated word recognition
Even worse when recognising connected words...
- The number of words in the utterance is not known
- Word boundaries are not known: any of the $V$ words may potentially start at each frame.
Connected word recognition

![Diagram of connected word recognition](image)
Viterbi algorithm: connected word recognition

Add transitions between all word-final and word-initial states
Connected word recognition

Viterbi decoding finds the best word sequence

BUT: have to consider $|V|^2$ inter-word transitions at every time step
Connected word recognition

state

\( x_1 \) \( x_2 \) \( x_3 \) \( x_4 \) \( x_5 \) \( x_6 \) \( x_7 \)

time

ASR Lecture 9

Large vocabulary ASR
Integrating the language model

So far we’ve estimated HMM transition probabilities from audio data, as part of the acoustic model

Transitions *between words rightarrow* use a language model

*n*-gram language model:

\[ p(w_i|h_i) = p(w_i|w_{i-n}, \ldots w_{i-1}) \]

Integrate the language model directly in the Viterbi search
Incorporating a bigram language model

\[ P(ten | pin) \]
\[ P(pin | pin) \]
\[ P(pin | right) \]
Incorporating a bigram language model
Incorporating a trigram language model

Need to duplicate HMM states to incorporate extended word history
Computational Issues

- Viterbi decoding performs an exact search in an efficient manner
- But exact search is not possible for large vocabulary tasks
  - Long-span language models and the use of cross-word triphones greatly increase the size of the search space
- Solutions:
  - Beam search (prune low probability hypotheses)
  - Tree structured lexicons
  - Language model look-ahead
  - Dynamic search structures
  - Multipass search (→ two-stage decoding)
  - Best-first search (→ stack decoding / A* search)

An alternative approach: Weighted Finite State Transducers (WFST)
Computational Issues

- Viterbi decoding performs an exact search in an efficient manner
- But exact search is not possible for large vocabulary tasks
  - Long-span language models and the use of cross-word triphones greatly increase the size of the search space
- Solutions:
  - Beam search (prune low probability hypotheses)
  - Tree structured lexicons
  - Language model look-ahead
  - Dynamic search structures
  - Multipass search (→ two-stage decoding)
  - Best-first search (→ stack decoding / A* search)
- An alternative approach: Weighted Finite State Transducers (WFST)
During Viterbi decoding, don’t propagate tokens whose probability falls a certain amount below the current best path.

Result is only an approximation to the best path.
Tree-structured lexicon

Figure adapted from Ortmans & Ney, “The time-conditioned approach in dynamic programming search for LVCSR”
Tree-structured lexicon

Reduces the number of state transition computations

For clarity, not all the connections are shown
Language model look-ahead

- Aim to make pruning more efficient
- In tree-structured decoding, look ahead to find out the best LM score for any words further down the tree
- This information can be pre-computed and stored at each node in the tree
- States in the tree are pruned early if we know that none of the possibilities will receive good enough probabilities from the LM.
Weighted finite state automaton that transduces an input sequence to an output sequence (Mohri et al 2008)

- States connected by transitions. Each transition has
  - input label
  - output label
  - weight

- Weights use the log semi-ring or tropical semi-ring – with operations that correspond to multiplication and addition of probabilities

- There is a single start state. Any state can optionally be a final state (with a weight)

- Used by Kaldi
Weighted Finite State Acceptors

![Diagram of Weighted Finite State Acceptors]

Figure 1: Weighted finite-state acceptor examples. By convention, we often treat acceptors and transducers uniformly. The examples in Figure 2 encode a portion of the information in the WFSAs of Figure 1(a)-(b) as WFSTs. Figure 2(a) represents a toy pronunciation lexicon, in this example as a mapping from phone strings to words in the lexicon, with probabilities representing the likelihoods of alternative pronunciations. It can be read along a path from the start state to a final state to a word string with a particular weight. The word corresponding to a pronunciation is output by the transition that consumes the first phone, and context-independent phones. More precisely, a transducer specifies a binary relation between strings: two strings are in the relation when there is a path from an initial to a final state in the transducer that produces no output. In general, this is a relation rather than a function since the same input might be transduced to different strings along two distinct paths. For a weighted transducer, each string might be transduced to different strings along different paths, and the second string as the sequence of output labels along the path (with weight). When losing word identity. Similarly, HMM structures of large vocabulary ASR...

- Transition 1: using/1, intuition/0.33, data/0.66
- Transition 2: are/0.5, is/1
- Transition 3: better/0.7, worse/0.3
- Transition 4: d/1, ey/0.5, ae/0.5, t/0.3, dx/0.7, ax/1
Weighted Finite State Transducers

Acceptor

![Diagram of an acceptor transducer with weighted states and transitions.]

Transducer

![Diagram of a transducer transducer with weighted states and transitions.]

The label marked with their unique number. The initial state of a weighted finite-state acceptor is the initial state of a weighted finite-state transducer.

Probabilities representing the likelihoods of alternative words are encoded by the lexicon, in this example, as a mapping from phone strings to words in the same language model as Figure 1(a) by giving each pronunciation an associated weight.

The key transducer operations to combine, optimize, search and produce a phone-to-word alignment of a pronunciation are similar to the weighted acceptors in Figure 1.

Many of those operations are the weighted transducer generalizations of classical algorithms for unweighted finite-state transducers for more than one word without losing word identity.

Similarly, HMM structures of different levels of representation can be combined in a transducer transducer generalization.

The examples in Figure 2 are similar to a weighted acceptor except that it has additional information about the transition identical input and output labels. This adds a more complex structure to the acceptor transducer.

Each operation in a transition identical input and output labels adds more information about the transition: the transducer can represent a relation between phones and words or between HMMs over an acceptor: the transducer can represent a relation between phones and words or between HMMs over a finite set of symbols.
Weighted Finite State Transducers

**Acceptor**

0 \[ \xrightarrow{d/1} \] 1 \[ \xrightarrow{\text{ey/0.5}} \] 2 \[ \xrightarrow{\text{t/0.3}} \] 3 \[ \xrightarrow{\text{ax/1}} \] 4

\[ \text{ae/0.5} \]

\[ \text{dx/0.7} \]

**Transducer**

0 \[ \xrightarrow{d:\text{data/1}} \] 1 \[ \xrightarrow{\text{ey:}\varepsilon/0.5} \] 2 \[ \xrightarrow{t:}\varepsilon/0.3} \] 3 \[ \xrightarrow{\text{ax: } \varepsilon/1} \] 4

\[ d:\text{dew/1} \]

\[ \text{ae:}\varepsilon/0.5 \]

\[ \text{dx:}\varepsilon/0.7 \]

\[ \text{uw:}\varepsilon/1 \]
The HMM as a WFST
Composition Combine transducers $T_1$ and $T_2$ into a single transducer acting as if the output of $T_1$ was passed into $T_2$.

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>$G$ word-level grammar</td>
<td>words</td>
<td>words</td>
</tr>
<tr>
<td>$L$ pronunciation lexicon</td>
<td>phones</td>
<td>words</td>
</tr>
<tr>
<td>$C$ context-dependency</td>
<td>CD phones</td>
<td>phones</td>
</tr>
<tr>
<td>$H$ 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
