Improved Lattice-based Decoding in SRI’s STT System

Jing Zheng
Andreas Stolcke
Ramana Gadde

SRI International

Overview

• Motivation
• Lattice generation
• Lattice expansion
• Lattice rescoring
• Sausage decoding
• Things to do
• Questions for other teams
Motivation (1)

- SRI system uses lattices to constrain search, thereby enabling use of cross-word models and higher-order LMs in decoding (with large vocabularies).
- We use N-best lists for final rescoring and knowledge source combination.
- Typical processing stages:
  1. Word graph lattice (PFSG) generation with bigram LM & within-word models
  2. Lattice expansion using higher-order LM
  3. N-best generation from expanded lattices using crossword models
  4. N-best rescoring with best LM(s), phone-duration model, pronunciation and pause model, prosody, etc.
  5. Sausage construction (with optional system combination)
  6. Word-posterior decoding

Motivation (2)

- Problems with current approach:
  - N-best rescoring constrains search space
  - Effective hypothesis density of N-best list depends on waveform segmentation (longer segments ⇒ fewer hypotheses per unit time)
  - Sausage quality suffers (both WER and confidence measures)
  - N-best lists containing phone alignments are redundant and huge (~1GB gzipped for CTS eval data), processing is dominated by I/O.
- Solution approach:
  - Output lattices containing time and acoustic information (HTK-style)
  - Do all rescoring on lattices
- Caveat: Implemented most of the tools needed, but some still missing. Results from complete system run at next meeting!
PFSGs versus HTK Lattices

- Probabilistic Finite State Grammar format:
  - Word associated with nodes
  - Probabilities associated with arcs
  - Light weight to represent grammar constraint
  - Hard to encode other information
  - Used by SRI decoder (Decipher™)

- HTK standard lattice format
  - Times associated with nodes
  - Word, AM, LM & pronunciation scores associated with arcs
  - Optionally encode phone alignments for each word
  - Easy to extend to encode other information
  - Widely used across research sites
  -⇒ potential for exchanging tools and data with other sites

Lattice Generation

- Changes in Decipher™ recognizer:
  - Support both PFSG and HTK lattice generation
  - Support PFSG to HTK lattice conversion
  - Three algorithms:
    - Word-pair algorithm
      - Use word-pair for word boundary optimization
      - Accurate acoustic scores, some overhead in search
    - Direct algorithm
      - Record word transitions directly, no word-pair optimization
      - Minimum search overhead, used for fast lattice generation (e.g., MMIE training)
    - Lexical-tree algorithm
      - Most flexible, can use high-order LM, crossword AM
      - Accurate word boundaries without extra overhead
      - Still need improvement for efficiency
Lattice Generation, Cont’d

• Word arc attributes obtained in decoding (for HTK lattices)
  – AM, LM scores
  – Phone labels and durations (optional)
  – State durations (optional)
  – Word arc posterior probability (optional)

• Usage in training
  – State alignment can be easily obtained from HTK lattice with state duration info
  – Can be easily used for MMIE denominator counts generation.
  – Can be used for enhanced Viterbi training, which considers different word alignments.
    • We did not observe win from enhanced Viterbi-training over standard Viterbi training!

Lattice Expansion

• Originally planned to use HTK tools for lattice LM rescoring, but found incompatibilities
  – Need to handle pause hypotheses
  – Undocumented assumptions about lattice structure

• Modified SRILM lattice-tool
  – Reads & writes HTK format
  – Converts arc-based lattices to node-based PFSG representation (and back)

• Improved LM expansion algorithm
  – Handles null and pause nodes directly -- no longer need to remove them prior to expansion
  – Expands multiword lattices using non-multiword LM
  – Performs “compact” expansion, using LM backoffs (Weng et al. ICSLP 98) for arbitrary backoff LMs, not just trigrams
    • Standard expansion: 29MB/speaker, 1.5xRT for expansion
    • Compact expansion: 5MB/speaker, 0.4xRT
Lattice Rescoring

- More lattice-tool enhancements (similar to HTK)
  - Prune lattices to maximum density -- effectively limits computation for “bad” utterances
  - Pronunciation scoring using probabilities from a dictionary
  - Viterbi decoding with weighted combination of AM, LM, pron, duration scores
- Duration scoring of HTK lattices
  - Reads phone alignment information for each word hypothesis
  - Uses posterior probabilities to estimate speaking rate (for normalization purposes)
  - Outputs new lattices with added duration scores
- Some knowledge sources not implemented yet
  - Pause language model \( P(\text{pause} | \text{prev word, next word}) \)
  - Pause-conditioned duration model
  - Need to perform lattice expansion at pause hypotheses

Sausage Decoding

- Goal: obtain linearized lattices (confusion networks) to
  - Pick words with highest posteriors / minimum expected error
  - Compute posterior probabilities for confidence estimation
- Problems with existing algorithm (Mangu et al. 1999) and implementation:
  - Can be slow without significant lattice pruning -- runtime \( O(|\text{lattice nodes}|^3) \)
  - Lattice alignment not guided by minimization of expected word error
  - Lattice pruning required to prevent bad alignments
  - Reads HTK format, but makes assumptions about structure
- Implemented new sausage algorithm
  - Runtime \( O(|\text{lattice nodes}| \times |\text{sausage length}|) \)
  - Minimizes expected word error in performing alignments
  - Not sensitive to bad, low probability hypotheses
  - Differs only in alignment of lattice nodes & arcs; posterior probability computation is unchanged.
Lattice Alignment Algorithm

- **Algorithm**
  1. Pick highest probability path as initial sausage $S$
  2. Repeat while unaligned nodes remain
     a. Choose unaligned node $n$ with highest posterior probability
     b. Compute highest probability path $p$ through $n$
     c. Align unaligned (middle) portion of $p$ to corresponding portion of current sausage $S$, minimizing expected word error between $p$ and $S$

- **Observations:**
  - Step 2c takes time proportional to the unaligned portion of $p$ and as a result aligns all nodes on $p$. Hence overall efficiency.
  - Skip step 2c if path to align violates topological order of lattices (only happens with very low probability arcs)
  - Algorithm is generalization of N-best sausage algorithm currently used in SRILM

Results with New Sausages

- **WER on dev2001 test set, first rescoring pass**

<table>
<thead>
<tr>
<th></th>
<th>N-best</th>
<th>Sausages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Old</td>
</tr>
<tr>
<td>2-gram LM</td>
<td>34.3</td>
<td>33.4</td>
</tr>
<tr>
<td>3-gram LM</td>
<td>32.6</td>
<td>32.4</td>
</tr>
<tr>
<td>4-gram LM</td>
<td>31.4</td>
<td>31.3</td>
</tr>
<tr>
<td>4-gram LM, optimized weights</td>
<td>30.9</td>
<td>n/a</td>
</tr>
<tr>
<td>4-gram LM + duration model</td>
<td>30.3</td>
<td>n/a</td>
</tr>
</tbody>
</table>

  *Note: weights optimized for N-best, not lattice decoding*

- **Runtimes (using same pruning threshold):**
  - Old sausages: 0.32xRT
  - New sausages: 0.08xRT
Things To Do

- Verify old lattice algorithm and test new one by exchanging data with Lidia @ IBM.
- Implement pause LM and pause-conditioned duration scoring for lattices (worth \( \approx 0.3\% \text{ abs.} \))
- Score weight optimization for lattice decoding
  - N-best rescoring weights are not necessarily optimal for larger search space.
  - Grid search not feasible because of large number of dimensions (\( \geq 5 \))
  - Exact word error computation on lattices is computationally expensive because of large number of distinct hypotheses.
  - Will use approximate alignments and simplex search (cf. Dimitra’s thesis)
- Lattice algorithms will be part of next SRILM release.

Summary

- Completely lattice-based rescoring is important for efficiency and best results in posterior-based decoding.
- SRI decoder modified to generate HTK lattices
- SRILM modified to support HTK lattices expansion and rescoring.
- Duration scoring of phone alignments in lattices.
- New sausage (lattice alignment) algorithm
- Some missing pieces
Questions for EARS Teams

- Should we try to exchange lattices for diagnostic experiments (e.g., rescore best acoustic system with new knowledge sources)?
- Should we try to standardize lattices for system combination effort?
- We observed diminishing returns from posterior-based decoding as systems get better. Experiences at other sites?