Speech Recognition
Lecture 11: Lattice Algorithms.

Eugene Weinstein
Google, NYU Courant Institute
eugenew@cs.nyu.edu
Slide Credit: Mehryar Mohri
This Lecture

- Speech recognition evaluation
- $N$-best strings algorithms
- Lattice generation
- Discriminative training
Performance Measure

- **Accuracy**: based on edit-distance of speech recognition transcription and reference transcription.
  - word or phone accuracy.
  - lattice oracle accuracy: edit-distance of lattice and reference transcription.

- **Note**: performance measure does not match the quantity optimized to learn models.
  - word-error rate lattices.
# Word Error Rates

<table>
<thead>
<tr>
<th>CORPUS (DARPA)</th>
<th>TYPE OF SPEECH</th>
<th>VOCABULARY SIZE</th>
<th>WORD ERROR RATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connected Digit Strings</td>
<td>Read Text</td>
<td>10</td>
<td>0.3%</td>
</tr>
<tr>
<td>Airline Travel Information</td>
<td>Spontaneous</td>
<td>2500</td>
<td>2.5%</td>
</tr>
<tr>
<td>Wall Street Journal</td>
<td>Read Text</td>
<td>64,000</td>
<td>6.6%</td>
</tr>
<tr>
<td>Radio (Marketplace)</td>
<td>Mixed</td>
<td>64,000</td>
<td>13%</td>
</tr>
<tr>
<td>Switchboard*</td>
<td>Conversational</td>
<td>28,000</td>
<td>37%</td>
</tr>
<tr>
<td>Call Home*</td>
<td>Conversational</td>
<td>28,000</td>
<td>40%</td>
</tr>
</tbody>
</table>

* Based on 1998 evaluation
Edit-Distance

Definition: minimal cost of a sequence of edit operations transforming one string into another.

Edit operations and costs:

- standard edit-distance definition: insertion, deletions, substitutions, all with same cost one.
- general case: more general operations, arbitrary non-negative costs.

Application: measuring word error rate in speech recognition and other string processing tasks.
Local Edits

- Edit operations: insertion: $\varepsilon \rightarrow a$, deletion: $a \rightarrow \varepsilon$, substitution: $a \rightarrow b$ ($a \neq b$).

- **Example**: 2 insertions, 3 deletions, 1 substitution

```
c t t g e e a c
```

```
\varepsilon \ t \ a \ \varepsilon \ g \ t \ \varepsilon \ c
```

- This is called an alignment.
Edit-Distance Computation

- **Standard case:** textbook recursive algorithm (Cormen, Leiserson, Rivest, 1992), quadratic complexity, $O(|x||y|)$ for two strings $x$ and $y$.

- **General case:** (Mohri, Pereira, and Riley, 2000; Mohri, 2003)
  
  - construct tropical semiring edit-distance transducer $T_e$ with arbitrary edit costs.
  
  - represent $x$ and $y$ by automata $X$ and $Y$.
  
  - compute best path of $X \circ T_e \circ Y$.
  
  - complexity quadratic: $O(|T_e||X||Y|)$. 
Global Alignment - Example

Example: $c(A, G) = 1, c(A, T) = c(G, C) = .5$, no cost for matching symbols. Insertion/deletion cost: 2 (not pictured).

Representation:

```
echo "A G C T" | ngramsymbols > agct.syms;
echo "A G C T" | farcompilestrings --symbols=agct.syms --keep_symbols=1 --generate_keys=1 | farextract
```
Global Alignment - Example

**Program:**
```
fstcompose X.fst Te.fst  |
fstcompose - Y.fst  |  fstshortestpath --
nshortest=1  >  A.fst
```

**Graphical representation:**

![Graphical representation of the alignment](image)
Edit-Distance of Automata

**Definition:** the edit-distance of two automata $A$ and $B$ is the minimum edit-distance of a string accepted by $A$ and a string accepted by $B$.

**Computation:**
- best path of $A \circ T_e \circ B$.
- complexity for acyclic automata: $O(|T_e||A||B|)$.

**Generality:** any weighted transducer in the tropical semiring defines an edit-distance. Learning edit-distance transducer using EM algorithm.
This Lecture

- Speech recognition evaluation
- \(N\)-best strings algorithms
- Lattice generation
- Discriminative training
N-Best Sequences

- **Motivation**: rescoring.
  - first pass using a simple acoustic model and grammar to produce lattice or N-best list.
  - re-evaluate alternatives with a more sophisticated model or use new information.

- **General problem**:
  - speech recognition, handwriting recognition.
  - information extraction, image processing.
**N-Shortest-Paths Problem**

- **Problem**: given a weighted directed graph $G$, a source state $s$ and a set of destination or final states $F$, find the $N$ shortest paths in $G$ from $s$ to $F$.

- **Algorithms**:
  
  - (Dreyfus, 1969): $O(|E| + N \log(|E|/|Q|))$.
  
  
  - (Eppstein, 2002): $O(|E| + |Q| \log |Q| + N)$. 
**N-Shortest Strings ≠ N-Shortest-Paths**

- **Problem**: given a weighted directed graph $G$, a source state $s$ and a set of destination or final states $F$, find the $N$ shortest strings in $G$ from $s$ to $F$.

- **Example**: NAB Eval 95.

<table>
<thead>
<tr>
<th>Thresh</th>
<th>Non-Unique</th>
<th>Unique</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>2.0</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>2.5</td>
<td>54</td>
<td>4</td>
</tr>
<tr>
<td>3.0</td>
<td>1536</td>
<td>48</td>
</tr>
</tbody>
</table>
N-Shortest Paths

Program:  
\[
\text{fstprune --delta=1.5 lat.fst | farprintstrings --print_weight --symbols=nab.syms}
\]

in addition the launch of Microsoft corporation's windows ninety five software will mean more memory will be required to run -2038.46
in addition the launch of Microsoft corporation's windows ninety five software will mean more memory will be required around -2037.8
in addition the launch of Microsoft corporation's windows ninety five software will mean more memory will be required to run -2037.51
in addition the launch of Microsoft corporation's windows ninety five software will mean more memory will be required around -2036.85
in addition the launch of Microsoft corporation's windows ninety five software will mean more memory will be required around -2036.76
in addition the launch of Microsoft corporation's windows ninety five software will mean more memory will be required to run -2036.47
in addition the launch of Microsoft corporation's windows ninety five software will mean more memory will be required around -2035.81
N-Shortest Strings

Program:

\[ \text{fstprune --delta=1.5 lat.fst | farprintstrings --print_weight --symbols=nab.syms --unique} \]

In addition, the launch of Microsoft Corporation's Windows Ninety Five software will mean more memory will be required to run around -2038.46.

In addition, the launch of Microsoft Corporation's Windows Ninety Five software will mean more memory will be required around -2037.8.
Algorithms Based on *N*-Best Paths

(Chow and Schwartz, 1990; Soon and Huang, 1991)

- **Idea:** use *K*-best paths algorithm to generate $K \gg N$ distinct paths.

- **Problems:**
  - $K$ not known in advance.
  - in practice, $K$ may be sometimes quite large, that is $K \sim 2^N$, which affects both time and space complexity.
**N-Best String Algorithm**

(Mohri and Riley, 2002)

- **Idea**: apply N-best paths algorithm to on-the-fly determinization of input automaton. **But**, N-best paths algorithms require shortest distances to F’.

- **Weighted determinization** (partial):
  - eliminates redundancy, no determinizability issue.
  - on-demand computation: only the part needed is computed.
  - on-the-fly computation of the needed shortest-distances to final states.
Shortest-Distances to Final States

**Definition:** let \( d(q, F) \) denote the shortest distance from \( q \) to the set of final states \( F \) in input (non-deterministic) automaton \( A \), and let \( d'(q', F') \) be defined in the same way in the resulting (deterministic) automaton \( B \).

**Theorem:** for any state \( q' = \{(q_1, w_1), \ldots, (q_n, w_n)\} \) in \( B \), the following holds:

\[
d'(q', F') = \min_{i=1,\ldots,n} \{ w_i + d(q_i, F') \}.
\]
Simple $N$-Shortest-Paths Algorithm

1. $\textbf{for } p \leftarrow 1 \textbf{ to } |Q'| \textbf{ do } r[p] \leftarrow 0$
2. $\pi[(i', 0)] \leftarrow \text{NIL}$
3. $S \leftarrow \{(i', 0)\}$
4. $\textbf{while } S \neq \emptyset$
5. \hspace{1em} $\textbf{do } (p, c) \leftarrow \text{head}(S); \text{DEQUEUE}(S)$
6. \hspace{2em} $r[p] \leftarrow r[p] + 1$
7. \hspace{2em} if $(r[p] = N \text{ and } p \in F)$ then exit
8. \hspace{2em} if $r[p] \leq N$
9. \hspace{3em} then for each $e \in E[p]$
10. \hspace{4em} $\textbf{do } c' \leftarrow c + w[e]$
11. \hspace{3em} $\pi[(n[e], c')] \leftarrow (p, c)$
12. \hspace{3em} $\text{ENQUEUE}(S, (n[e], c'))$

Queue ordering based on:

$(p, c) < (p', c') \iff (c + d[p, F] < c' + d(p', F))$
N-Best String Alg. - Experiments

Additional time to pay for N-best very small even for large $N$. 

NAB 40K Bigram

word accuracy

x real-time

1. 1-Best 2. 10-Best 3. 100-Best 4. 1000-Best 5. Lattice

Eugene Weinstein - Speech Recognition
N-Best String Alg. - Properties

- **Simplicity and efficiency:**
  - easy to implement: combine two general algorithms.
  - works with any $N$-best paths algorithm.
  - empirically efficient.

- **Generality:**
  - arbitrary input automaton (not nec. acyclic).
  - incorporated in OpenFST Library (`fstshortestpath`).
This Lecture

- Speech recognition evaluation
- $N$-best strings algorithms
- Lattice generation
- Discriminative training
**Speech Recognition Lattices**

**Definition:** weighted automaton representing speech recognizer’s alternative hypotheses.
Lattice Generation

(Odell, 1995; Ljolje et al., 1999)

Procedure: given transition $e$ in $N$, keep in lattice transition $((p[e], t'), i[e], o[e], (n[e], t))$ with best start time $(p[e], t')$ during Viterbi decoding.
Lattice Generation

- **Computation time**: little extra computation over one-best.

- **Optimization**:
  - projection on output (words or phonemes).
  - epsilon-removal.
  - pruning: keeps transitions and states lying on paths whose total weight is within a threshold of the best path.
  - garbage-collection (use same pruning).
Notes

- **Heuristics**: not all paths within beam are kept in lattice.

- **Lattice quality**: oracle accuracy, that is best accuracy achieved by any path in lattice.

- **Optimizations**: weighted determinization and minimization.
  - in general, dramatic reduction of redundancy and size.
  - bad for some lattices, typically uncertain cases.
Speech Recognition Lattice
Lattice after Determinization

(Mohri, 1997)
Lattice after Minimization

(Mohri, 1997)
This Lecture

- Speech recognition evaluation
- $N$-best strings algorithms
- Lattice generation
- Discriminative training
Discriminative Techniques

- **Maximum-likelihood**: parameters adjusted to increase joint likelihood of acoustic and CD phone or word sequences, irrespective of the probability of other word hypotheses.

- **Discriminative techniques**: takes into account competing word hypotheses and attempts to reduce the probability of incorrect ones.

  - Main problems: computationally expensive, generalization.
Objective Functions

(Woodland and Povey, 2002)

- **Maximum likelihood (joint):**

  \[ F = \arg\max_{\theta} \sum_{i=1}^{m} \log p_{\theta}(o_i, w_i). \]

- **Conditional maximum likelihood (CML):**

  \[ F = \arg\max_{\theta} \sum_{i=1}^{m} \log p_{\theta}(w_i|o_i) = \arg\max_{\theta} \sum_{i=1}^{m} \log \frac{p_{\theta}(o_i, w_i)}{p_{\theta}(o_i)} \]

- **Maximum mutual information (MMI/MMIE)**

  \[ F = \arg\max_{\theta} \sum_{i=1}^{m} \log \frac{p_{\theta}(o_i|w_i)p_{\theta}(w_i)}{\sum_{\hat{w}} p_{\theta}(o_i|\hat{w})p_{\theta}(\hat{w})} \]

  Equivalent to CML when independent of theta.
Discriminative Training Algorithm

- Produce numerator lattice by decoding with only the reference transcript allowed (similar to alignment for MLE training).
- Produce denominator lattice by doing open-ended decode with a simple language model (unigram model trained on transcripts of training data set).
- Gather Gaussian component occupancy counts, first- and second-order statistics of data points.
- Perform model updates using modified Baum-Welch update equations.


References


References

- Phil Woodland and Daniel Povey, Large Scale Discriminative Training for Speech Recognition. *Computer Speech and Language*, 16, pp. 25-47, 2002.