Speech Recognition
Lecture 12: Lattice Algorithms.

Mehryar Mohri
Courant Institute of Mathematical Sciences
mohri@cims.nyu.edu

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

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

* 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 $\rightarrow$ ε, substitution: a $\rightarrow$ b (a $\neq$ b).
- Example: 2 insertions, 3 deletions, 1 substitution

$$\begin{array}{cccccccc} c & t & t & g & \epsilon & \epsilon & a & c \\ \epsilon & t & a & \epsilon & g & t & \epsilon & c \end{array}$$

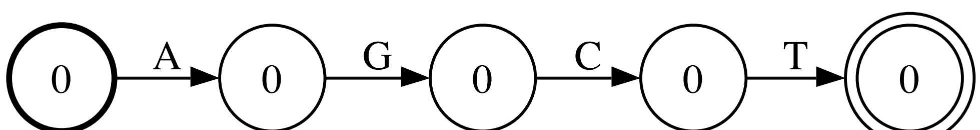
- This is called an alignment.

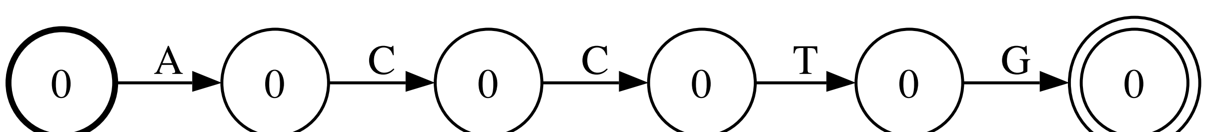
Edit-Distance Computation

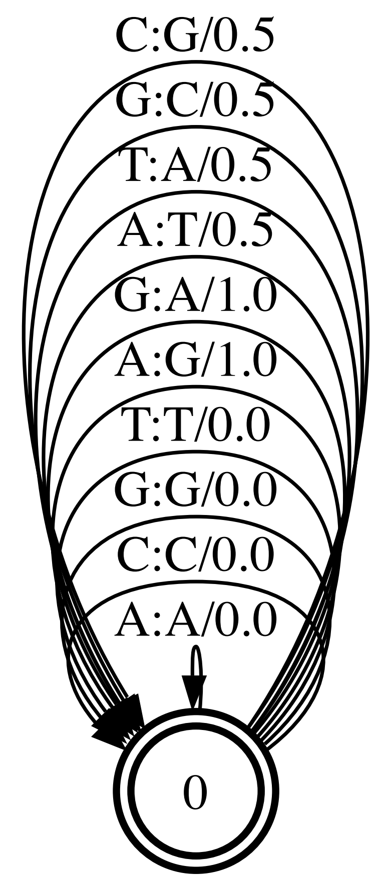
- Standard case: textbook recursive algorithm (Cormen, Leiserson, Rivest, 1992), quadratic complexity, $O(|x||y|)$ for two strings x and y .
- General case: (MM, Pereira, and Riley, 2000; MM, 2003)
  - construct tropical semiring edit-distance transducer T_e with arbitrary edit costs.
  - represent x 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.
- **Representation:**



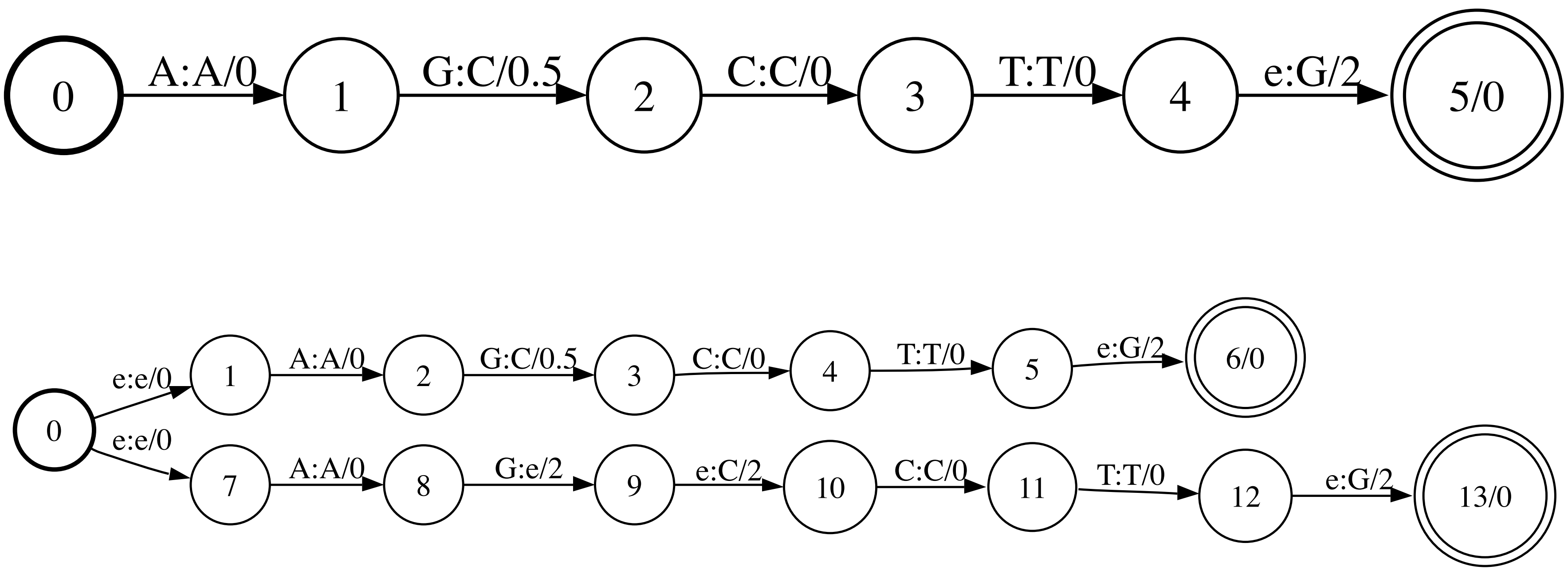





```
echo "A G C T" | farcompilestrings >X.fsm
```

Global Alignment - Example

- **Program**: `fsmcompose X.fsm Te.fsm Y.fsm | fsmbestpath -n 1 >A.fsm`
- **Graphical representation**:



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 and grammar 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): $O(|E| + N \ \log(|E|/|Q|))$.
  - (MM, 2002): shortest-distance algorithm, *N*-tropical semiring.
  - (Eppstein, 2002): $O(|E| + |Q| \log |Q| + N)$.

+ explicit representation of *N* best paths: $O(|Q| \ N^2)$.

N-Shortest Strings $\neq$ *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.

Thresh	Non-Unique	Unique
1.5	8	2
2.0	24	4
2.5	54	4
3.0	1536	48

N-Shortest Paths

- Program: fsmprune -c1.5 lat.fsm | farprintstrings -c -iNAB.wordlist

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 to run -2037.42
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: fsmprune -c1.5 lat.fsm |
  farprintstrings -c -u -iNAB.wordlist

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

Algorithms Based on *N*-Best Paths

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

- Idea: use *K*-best paths algorithm to generate $K \gg N$ distincts paths.
- Problems:
  - K not known in advance.
  - in practive, K may be sometimes quite large, that is $K \sim 2^N$, which affects both time and space complexity.

N-Best String Algorithm

(MM 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

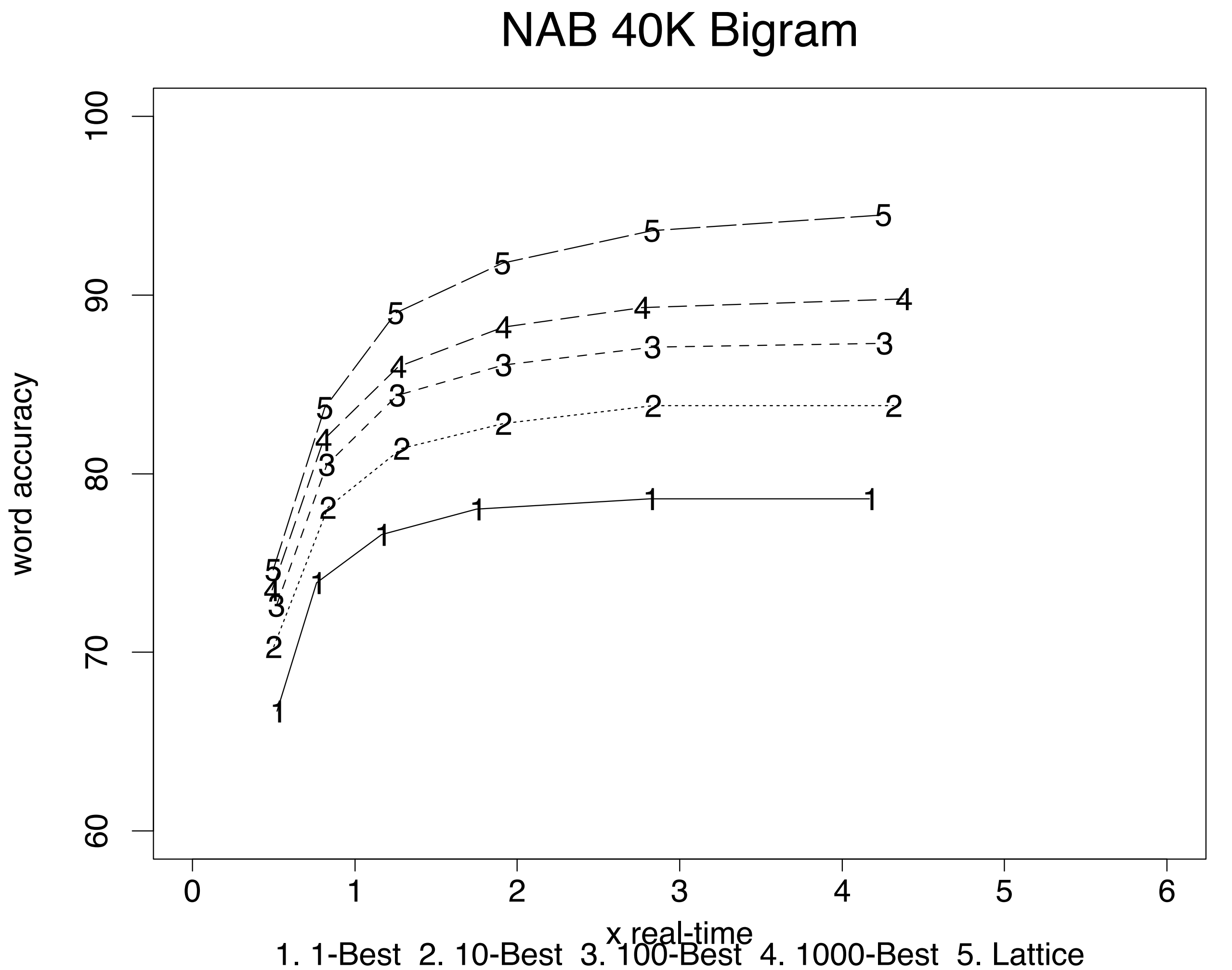
- Definition: let $d(q, F)$ denote the shortest distance from q to the set of final states F in input (non-deterministic) automaton A, and let $d'(q', F')$ be defined in the same way in the resulting (deterministic) automaton B.
- Theorem: for any state $q' = \{(q_1, w_1), \ldots, (q_n, w_n)\}$ in B, the following holds:

$$d'(q', F') = \min_{i=1,\ldots,n} \{w_i + d(q_i, F)\}.$$

Simple *N*-Shortest-Paths Algorithm

for $p \leftarrow 1$ **to** $|Q'|$ **do** $r[p] \leftarrow 0$
$\pi[(i', 0)] \leftarrow \text{NIL}$
$S \leftarrow \{(i', 0)\}$
while $S \neq \emptyset$
  do $(p, c) \leftarrow head(S)$; $\text{DEQUEUE}(S)$
    $r[p] \leftarrow r[p] + 1$
    if $(r[p] = N$ **and** $p \in F)$ **then** **exit**
    if $r[p] \leq N$
      then **for** each $e \in E[p]$
        do $c' \leftarrow c + w[e]$
          $\pi[(n[e], c')] \leftarrow (p, c)$
          $\text{ENQUEUE}(S, (n[e], c'))$

N-Best String Alg. - Experiments



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

N-Best String Alg. - Properties

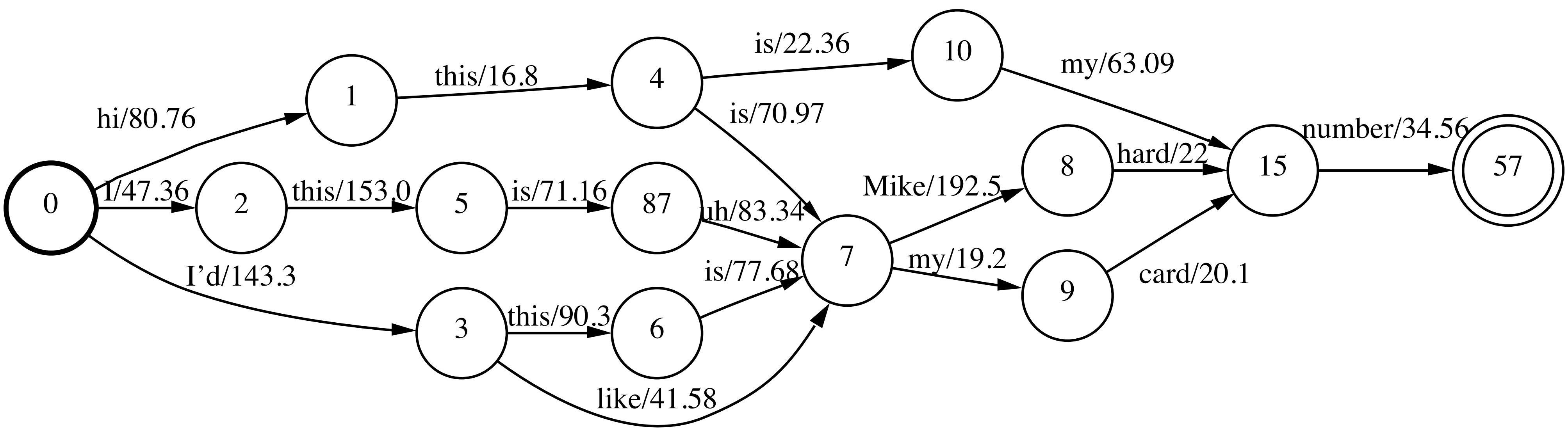
- Simplicity and efficiency:
  - easy to implement: combine two general algorithms.
  - works with any *N*-best paths algorithm.
  - empirically efficient.
- Generality:
  - arbitrary input automaton (not nec. acyclic).
  - incorporated in FSM Library (`fsmbestpath`).

This Lecture

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

Speech Recognition Lattices

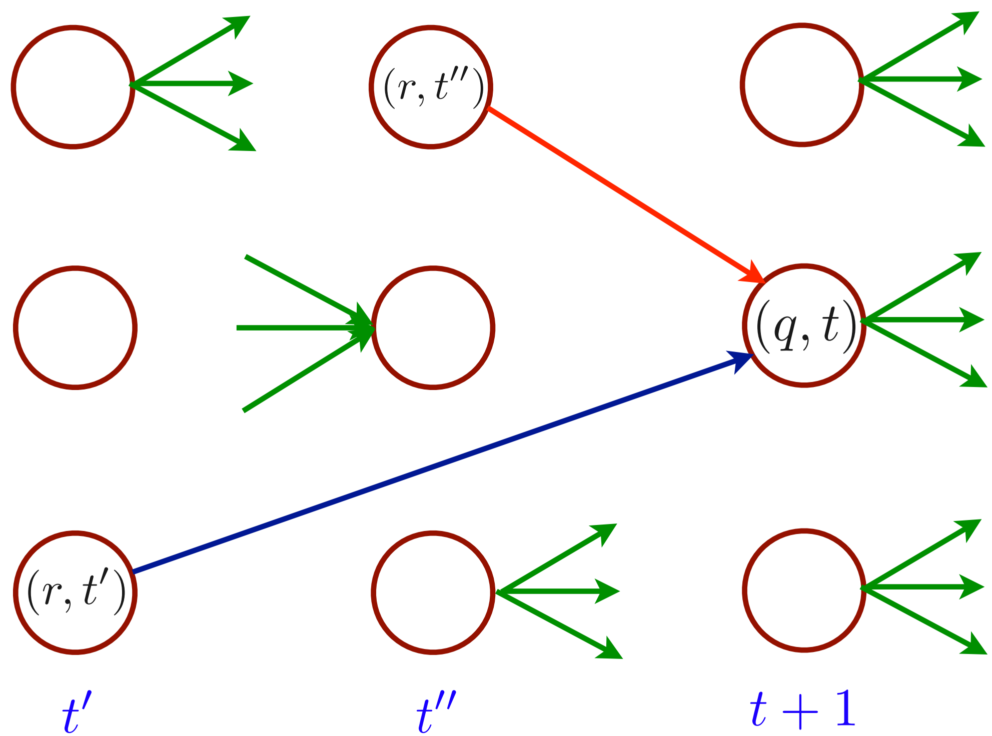
- Definition: weighted automaton representing speech recognizer’s alternative hypotheses.



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.



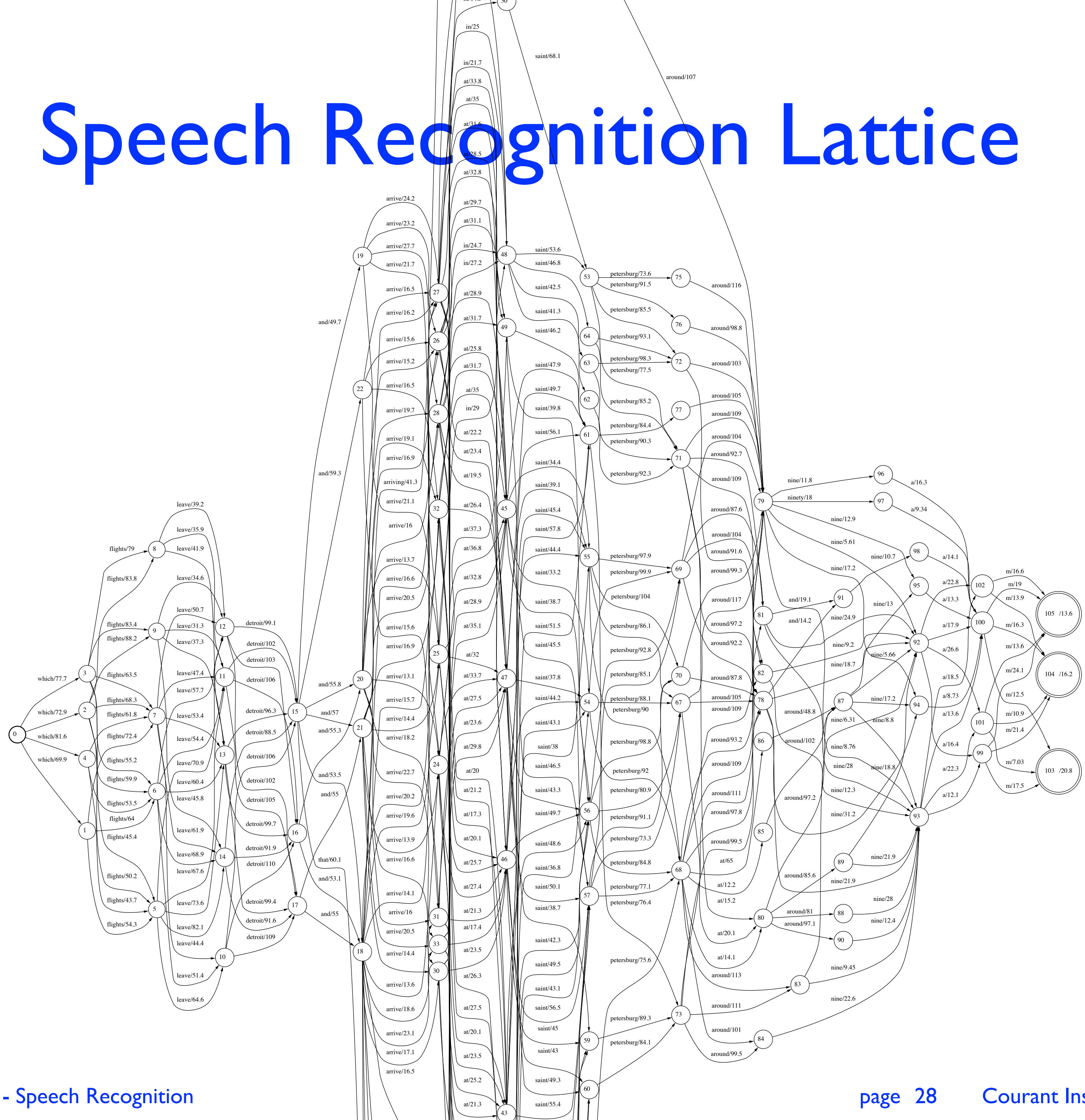
Lattice Generation

- 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

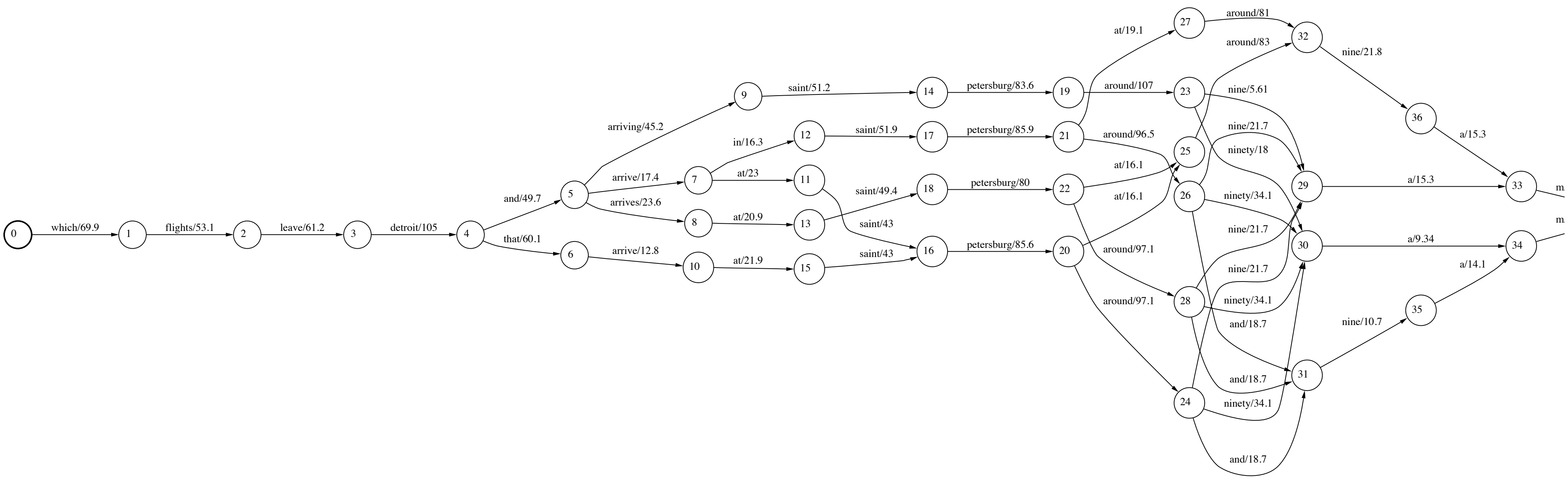
- Heuristics: not all paths within beam are kept in lattice.
- Lattice quality: oracle accuracy, that is best accuracy achieved by any path in lattice.
- Optimizations: weighted determinization and minimization.
  - in general, dramatic reduction of redundancy and size.
  - bad for some lattices, typically uncertain cases.

Speech Recognition Lattice



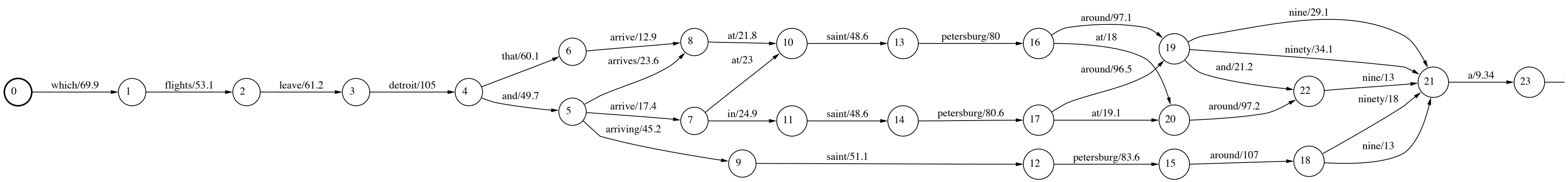
Lattice after Determinization

(MM, 1997)



Lattice after Minimization

(MM, 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

- Maximum likelihood (joint):

$$F = \operatorname*{argmax}_{\theta} \sum_{i=1}^{m} \log p_\theta(\mathbf{o}_i, \mathbf{w}_i).$$

- Conditional maximum likelihood (CML):

$$F = \operatorname*{argmax}_{\theta} \sum_{i=1}^{m} \log p_\theta(\mathbf{o}_i | \mathbf{w}_i) = \operatorname*{argmax}_{\theta} \sum_{i=1}^{m} \log \frac{p_\theta(\mathbf{o}_i, \mathbf{w}_i)}{p_\theta(\mathbf{o}_i)}.$$

- Maximum mutual information (MMI/MMIE)

$$F = \operatorname*{argmax}_{\theta} \sum_{i=1}^{m} \log \frac{p_\theta(\mathbf{o}_i, \mathbf{w}_i)}{p_\theta(\mathbf{o}_i) p_\theta(\mathbf{w}_i)}.$$

Equivalenty to CML when independent of theta.

References

- Y. Chow and R. Schwartz, The N-Best Algorithm: An Efficient Procedure for Finding top N Sentence Hypotheses. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing* (*ICASSP* '90), Albuquerque, New Mexico, April 1990, pp. 81–84.
- S. E. Dreyfus. An appraisal of some shortest path algorithms. *Operations Research*, 17:395-412, 1969.
- David Eppstein, Finding the shortest paths, *SIAM Journal of Computing*, vol.28, no. 2, pp. 652–673, 1998.
- Andrej Ljolje and Fernando Pereira and Michael Riley, Efficient general lattice generation and rescoring. In Proceedings of the European Conference on Speech Communication and Technology (Eurospeech '99), Budapest, Hungary, 1999.
- Mehryar Mohri. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*, 23:2, 1997.
- Mehryar Mohri. Statistical Natural Language Processing. In M. Lothaire, editor, *Applied Combinatorics on Words*. Cambridge University Press, 2005.

References

- Mehryar Mohri. Edit-Distance of Weighted Automata: General Definitions and Algorithms. *International Journal of Foundations of Computer Science*, 14(6):957-982, 2003.

- Mehryar Mohri and Michael Riley. An Efficient Algorithm for the *N*-Best-Strings Problem. In *Proceedings of the International Conference on Spoken Language Processing 2002* (*ICSLP* '02), Denver, Colorado, September 2002.

- Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. The Design Principles of a Weighted Finite-State Transducer Library. *Theoretical Computer Science*, 231:17-32, January 2000.

- Julian Odell. *The Use of Context in Large Vocabulary Speech Recognition*. Ph.D. thesis, 1995. Cambridge University, UK.

- Frank Soong and Eng-Fong Huang, A Tree-Trellis Based Fast Search for Finding the N Best Sentence Hypotheses in Continuous Speech Recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing* (*ICASSP* '91), Toronto, Canada, November 1991, pp. 705–708.