CSE6328.3
Speech & Language Processing

No.9

Hypothesis Search in Large Vocabulary ASR

Prof. Hui Jiang
Department of Computer Science and Engineering
York University

Automatic Speech Recognition (III): Search for LVCSR

Prof. Hui Jiang
Department of Computer Science and Engineering
York University, Toronto, Canada
hj@cse.yorku.ca

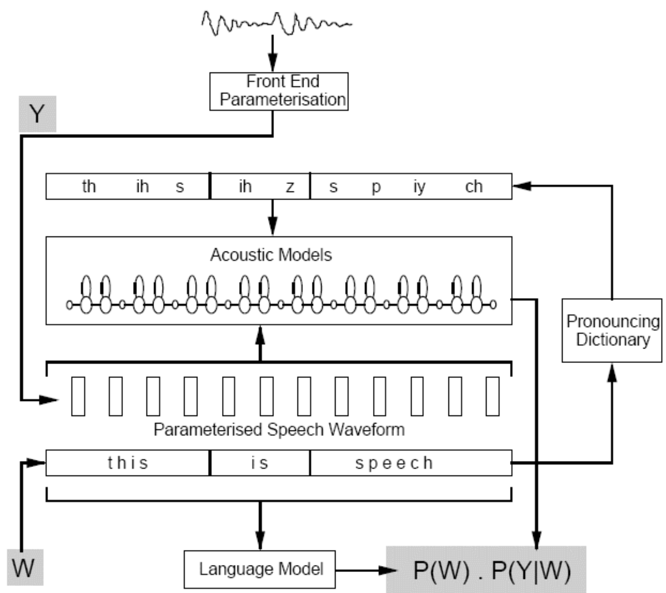
ASR Solution

$$\hat{W} = \arg\max_{W \in \Gamma} p(W \mid X) = \arg\max_{W \in \Gamma} P(W) \cdot p(X \mid W)$$

$$= \arg\max_{W \in \Gamma} \overline{P}_{\Gamma}(W) \cdot \overline{p}_{\Lambda}(X \mid W)$$

- $\overline{p}_{\Lambda}(X \mid W)$ — ***Acoustic Model (AM)***: **gives the probability of generating feature *X* when *W* is uttered.**
- $\overline{P}_{\Gamma}(W)$ — ***Language Model (LM)***: **gives the probability of *W* (word, phrase, sentence) is chosen to say.**

Overview of Statistical ASR



How to postulate word sequence?

- First thought: enumerate all possible word sequences one by one
  - Expand into a large composite HMM
  - Calculate the score and look for the best sequence
  - Impossible even for small vocabulary task, e.g., digit string.
- Solution: build an overall recognition network accommodating all possible word sequence → search for the best path
  - Consider the task grammar and the language modeling constraints (FSG, n-gram, context-free)
  - Build search network based on the task grammar
  - Expand into a single huge composite HMM
  - Given a speech feature sequence, use the Viterbi algorithm to search for the best alignment path through the network.
  - The alignment path → the most likely word sequence (output)
  - Each alignment path corresponds to one word sequence; but each word sequence has many possible alignment paths.
    - Viterbi Approximation → easy implementation

Search Space Representation

- Postulating word sequences is a typical search problem in CS.
- First of all, how to specify search space in ASR?
- Obviously, the search space depends on the underlying grammar.
- In ASR, language grammar is given in the following forms:
  - Finite State Grammar (FSG):

    Applications: voice dialing, digit string recognition, etc.

  - N-gram: uni-gram, bi-gram, tri-gram, 4-gram

    Applications: Dictation system, broadcast news transcription, etc.

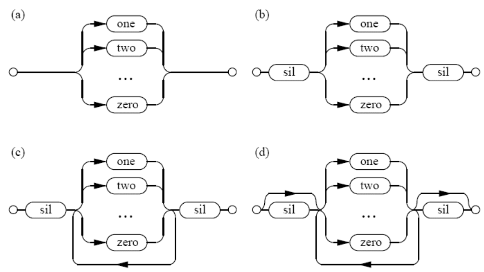
  - Context-free Grammar (CFG) → recursive transition network

    CFG is convenient to refer to high-level task-specific concepts, such as dates, names, inquiry patterns, etc.

    Useful in speech understanding

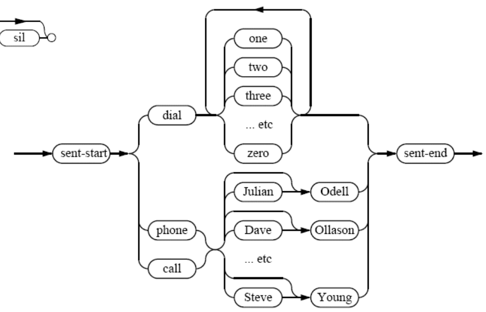
Search Space(1): FSG

- FSG itself is a search network; directly expand into composite HMM based on lexicon and acoustic models.



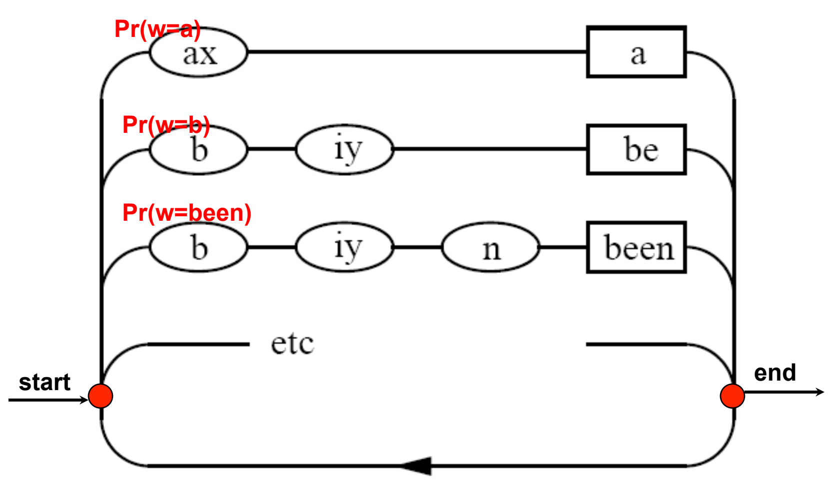
(a) single digit
(b) single digit with start/end silence
(c) Digit string with start/end silence
(d) Digit string with optional silence

Voice Dialing



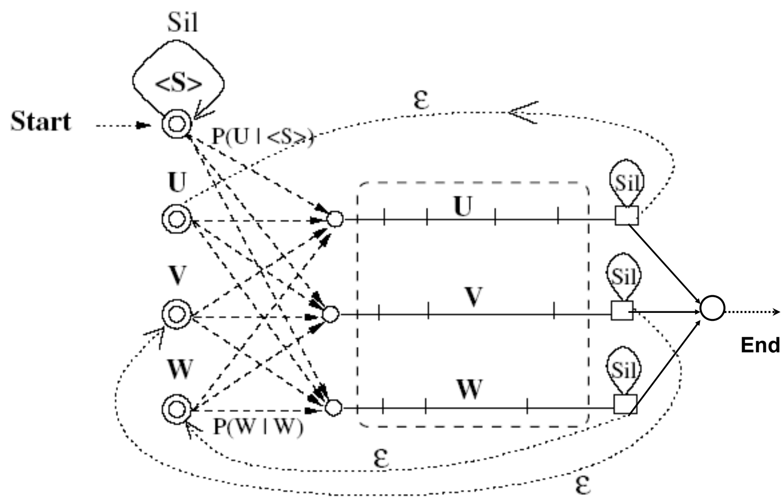
Search Space(2): Unigram

- Word-loop network is sufficient for unigram LM.



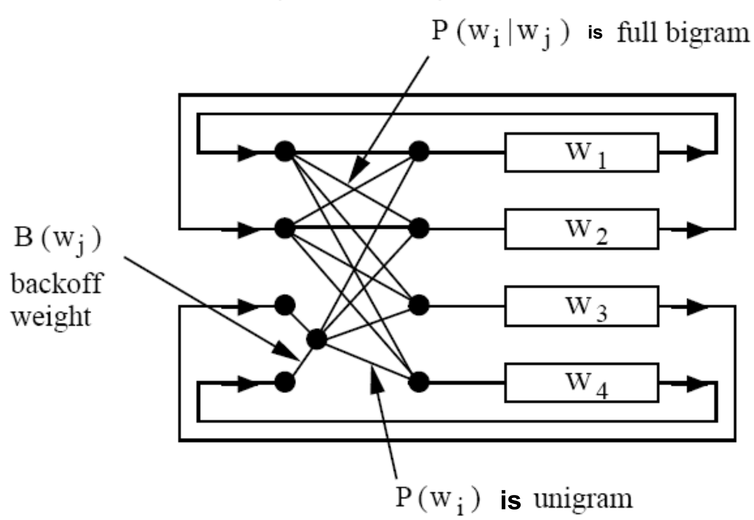
Search Space(3): Bigram

- Network for bi-gram is a bit complex; need more glue nodes.



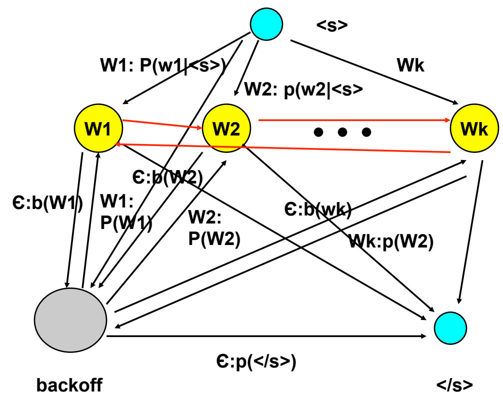
Search Space(4): Back-off Bi-gram

- If back-off bi-gram is used, glue nodes can be merged for back-off contexts to reduce links. (used in HTK)



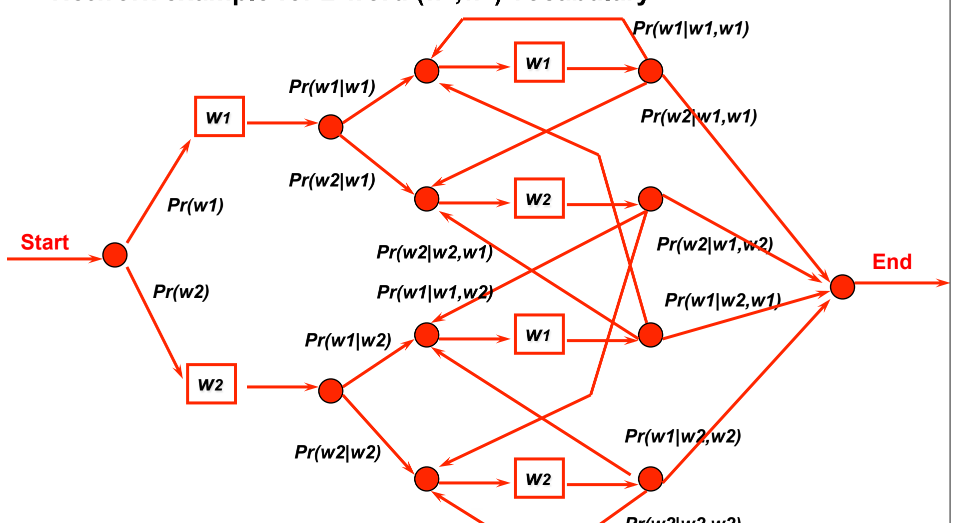
Search Space(5): Back-off Bigram LM with WFST

- No full context in back-off n-gram LM.
  - Observed context: use n-gram condition probabilities.
  - Unobserved context: back-off to lower level n-1 gram.
- WFST for back-off bi-gram LM:



Search Space(6): Trigram

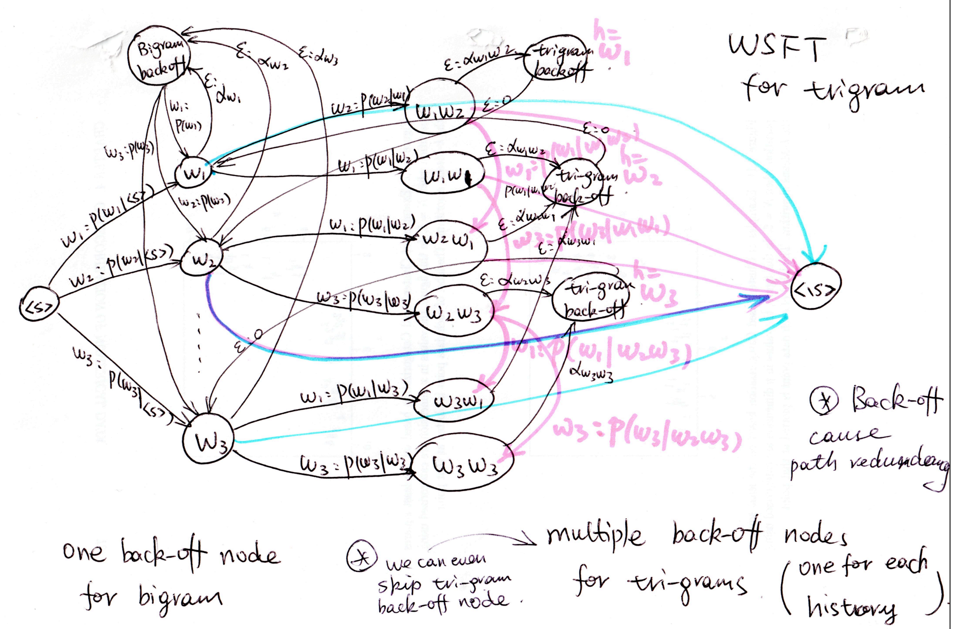
- Network for tri-gram becomes significantly complicated.
- Network example for 2-word (*w1,w2*) vocabulary



Search Space(7): Back-off Trigram

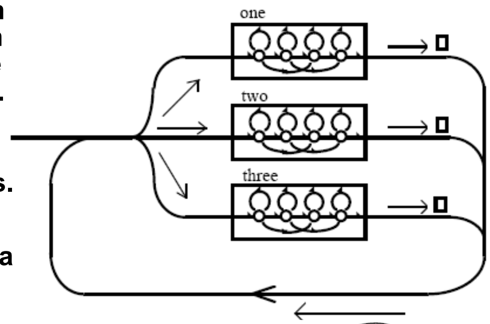
- Representation of a full trigram LM for large vocabulary is prohibitive.
- It is possible to represent a back-off trigram LM even for very large vocabulary.
- WFST example …

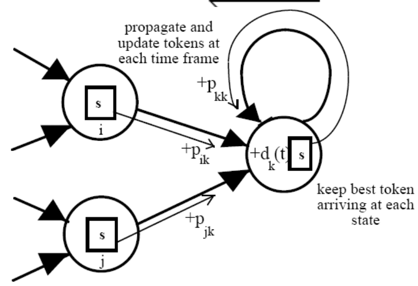
Search Space(8): Back-off Trigram



Token Passing (1): simple implementation model for Viterbi decoding

- For a large or even medium size HMM, hard to maintain 2-D trellis to implement the Viterbi decoding algorithm.
- Token passing paradigm: equivalent; easy to implement for large HMM's.
- Token passing:
  - Each HMM state holds a movable token which contains all info about its partial travel from a HMM start state up to the current state, e.g. partial log prob δ(.) and the partial path.
  - Viterbi search becomes a token propagation process.





Token Passing Algorithm

- Initialization: each HMM initial state holds a token with value 0;
- Propagation:
  - For each observation feature vector o_t, $t=1,2,\ldots, T$,
    - For each HMM state *i* do

      (1) Pass a copy of the token in state i to all connecting states j by following HMM state transition; updating value of the new tokens by $a_{ij}+b_j(o_t)$;

      (2) Discard the original tokens;

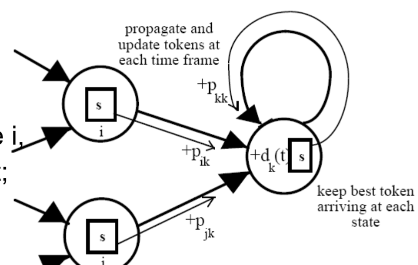
      End
    - For each HMM state *i* do

      if more than one tokens enter state i,

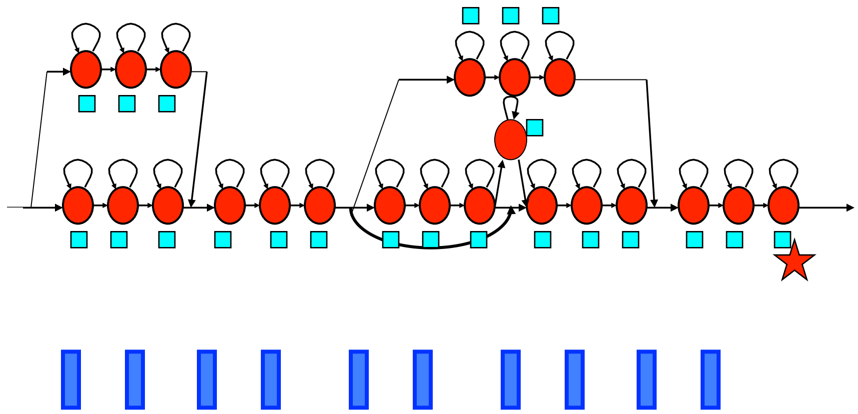
      keep the best one, discard the rest;

      End

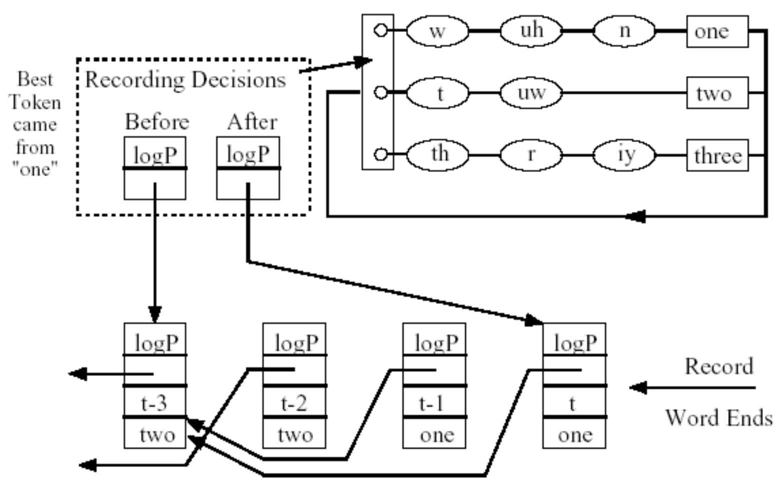
    End
- Termination:
  - Examine all final states, the token with the best value passed the best path; its value → Viterbi score; recover path.



Token Passing Example



Token Passing: record boundaries



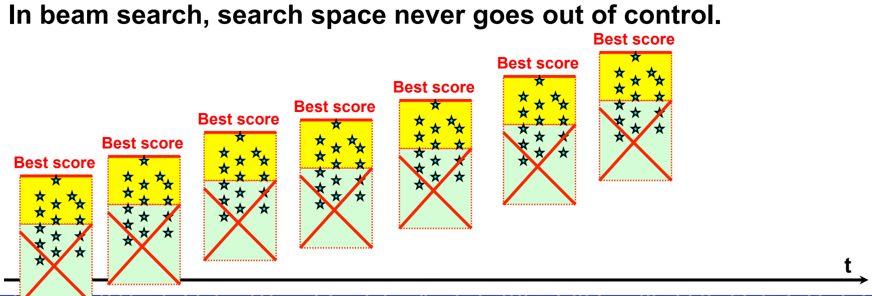
Recording Word Boundary Decisions

Techniques to Accelerate Search in ASR

- Beam search
  - Prune unlikely candidates at the earliest stage.
- Fast-match
- Tree-organized pronunciation lexicon
  - For data sharing and better pruning strategy.
  - How to construct search space for tree lexicon?
  - Language Model Look-Ahead: how to apply LM earlier?
- One-pass search vs. Multi-pass search
  - Integrated one-pass search: integrate all available knowledge sources and explore the whole search space once; slow.
  - Multi-pass search: use partial knowledge (e.g., simpler models) to reduce search space; explore the reduced search space by more complicated models; fast.
- Dynamical network expansion
- Static decoding based on minimized WFST
- Alternative outputs:
  - N-Best list: how to generate?
  - Word-graph: compact representation of more candidates.

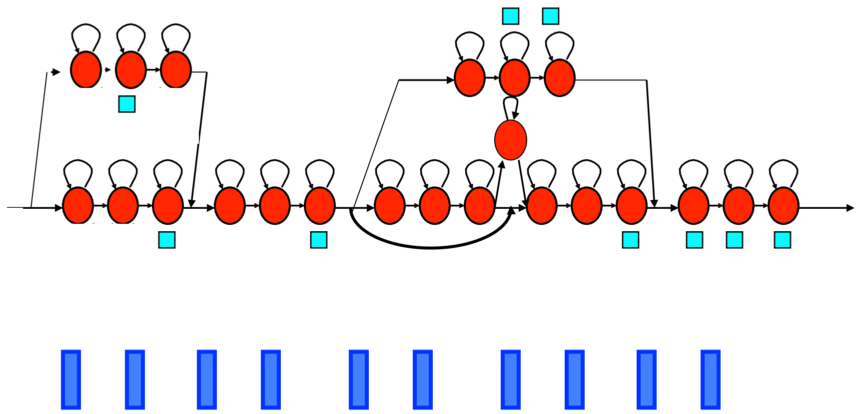
Beam Search (I)

- Beam Search: every time frame, the best score in all partial paths (tokens in token passing) is noted and any partial paths (tokens) whose score lies more than a beam-width below this best score is pruned from further consideration.
- Instead of searching for the entire dark room, just follow the beam of your flashlight.
- Beam-width is a pre-set constant to control the degree of pruning.
- Beam search makes the prohibitive search problem feasible.
- In beam search, search space never goes out of control.



Beam Search (II)

- Beam search is THE most important pruning strategy to accelerate search in speech recognition.
- Beam search is not admissible: it may miss the best path; but this seldom happens in practice if the beam-width is set properly.



Beam Search (III)

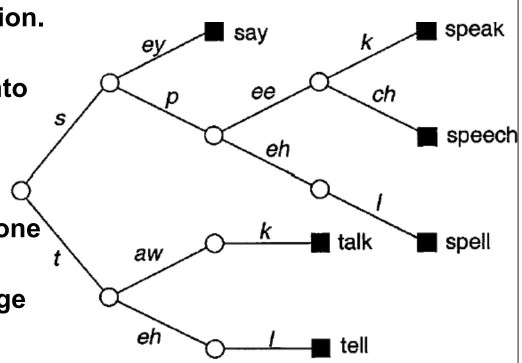
- Acoustic pruning: retain only hypotheses with a score close to the best hypothesis for further consideration.
  - Regular beam search for all in-HMM partial candidates.
  - Acoustic beam-width P_a.
- Language model pruning (word ending pruning):
  - The optimal path seems more stable at the word-ending points during the search especially after applying LM scores.
  - More aggressive pruning is possible at word-end.
  - Word-ending (LM) beam-width P_{LM}. (P_{LM} can be chosen to smaller than P_a to ward off more unlikely candidates)
- Histogram Pruning:
  - Each time, instead of setting a beam width, survive only the best N candidates.
  - Sorting is prohibitive; usually implement by histogram.

ASR Search Algorithms

- Dynamic search network expansion
  - Tree lexicon
  - Language model look-ahead
  - Dynamic expansion
- Static optimized network
  - Static back-off LM network
  - Expansion using WFST composition
  - Optimization using WFST determinization and minimization

Tree Lexicon Organization

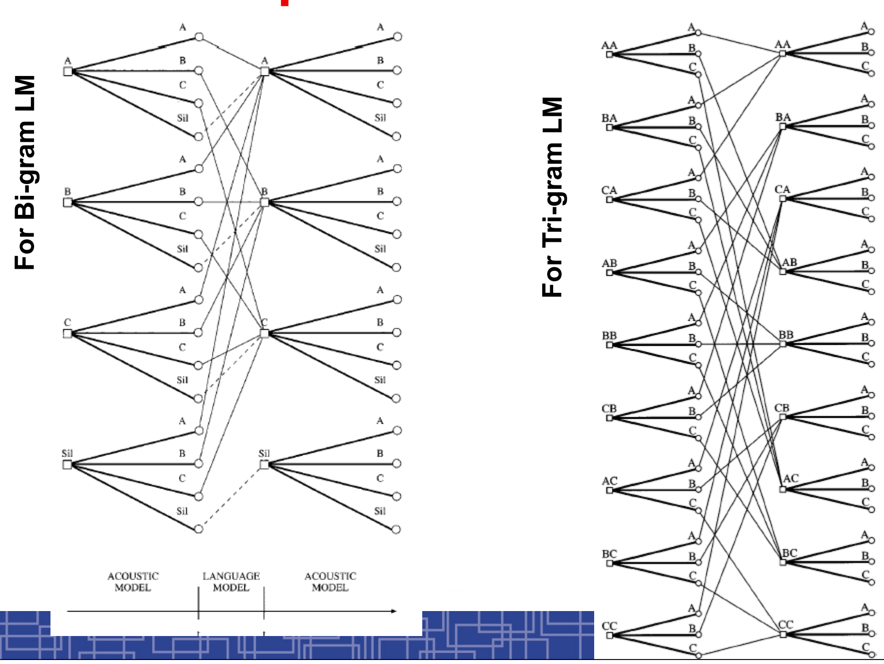
- Linear lexicon: each word in vocabulary is modeled separately:
  - Essentially, it is a linear sequence of phonemes according to its pronunciation.
- Tree lexicon: all words in vocabulary can be organized into a prefix tree:
  - Better data sharing; more effective pruning.
  - Each leaf node represents one word.
  - Extremely important for large vocabulary cases.



Tree Lexicon: problems

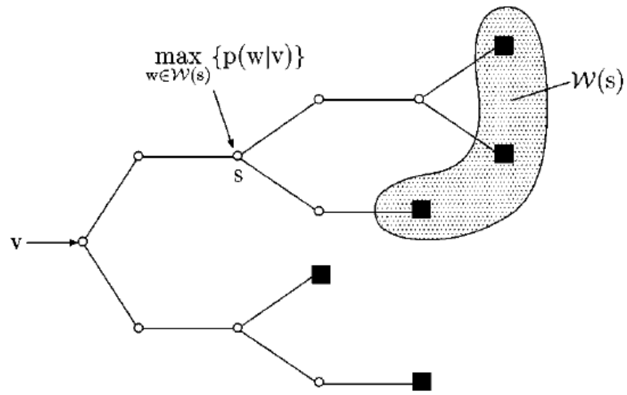
- Problems with a tree lexicon:
  - The identity of the hypothesized word is unknown until reaching a leaf node.
    - Language model (LM) scores can't be applied until at the end of tree → ineffective pruning in beam search
  - Search space is hard to formulate unless making lots of tree copies.
- Conceptual example:
  - Three words in vocabulary
  - A network for only 2-word sentences
  - For bi-gram: introducing merging nodes for previous word
  - For tri-gram: introducing merging nodes for previous two words

Search Space for Tree Lexicon:

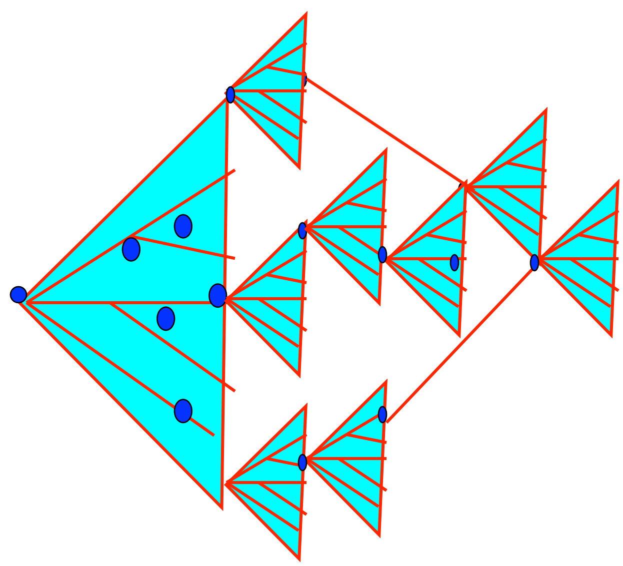


Language Model Look-ahead

- In tree lexicon, can't apply LM score due to unknown id of current word.
- Better to incorporate LM knowledge as soon as possible to prune those unlikely candidates in grammar.
- LM look-ahead: apply maximum LM scores of all words which can be reached from the current node.



Dynamic Network Expansion



How to handle huge search space in large vocabulary

- Fast Match: phoneme look-ahead
  - Look-ahead some feature vectors to determine a small set of most likely phoneme from the current time point.
- Multiple-pass search strategy:
  - 1st pass: use simple language model (unigram, bi-gram) to reduce search space.
  - 2nd pass: use more complicated model (such as tri-gram) to search for the result only in the above reduced space.
- Single-pass search strategy:
  - Dynamic network expansion:
    - No a whole static network is built beforehand (too big).
    - Expand the net dynamically during the search process.

Static Optimization Network using WFST

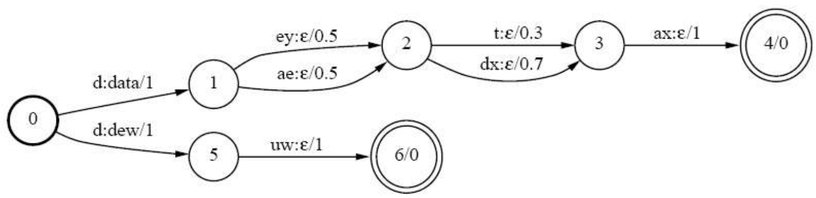
- Build a huge static search network from LM: Composition
  - LM-based Grammar WFST (G)
  - Pronunciation Lexicon (L)
  - Context-Dependency Transducer (C)
  - Sub-word HMM (H)

$$F = H \circ C \circ L \circ G$$

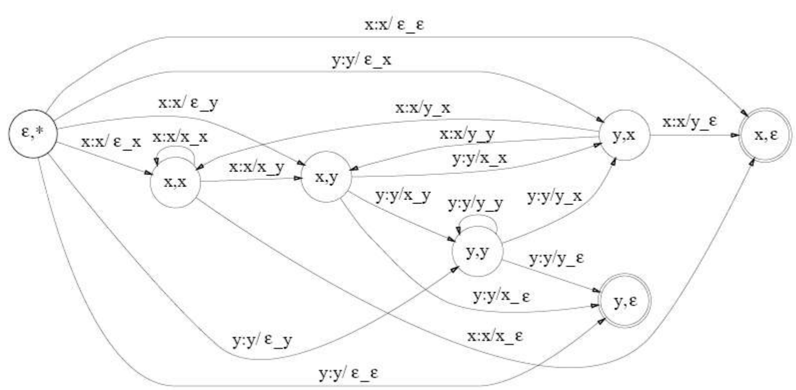
- Compact the network using graph algorithms.
  - Determinization
  - Minimization

$$\min(\det(F))$$

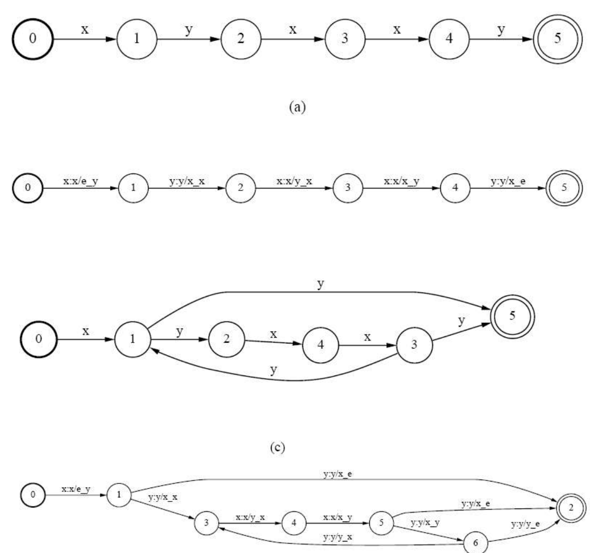
Pronunciation Lexicon WFST



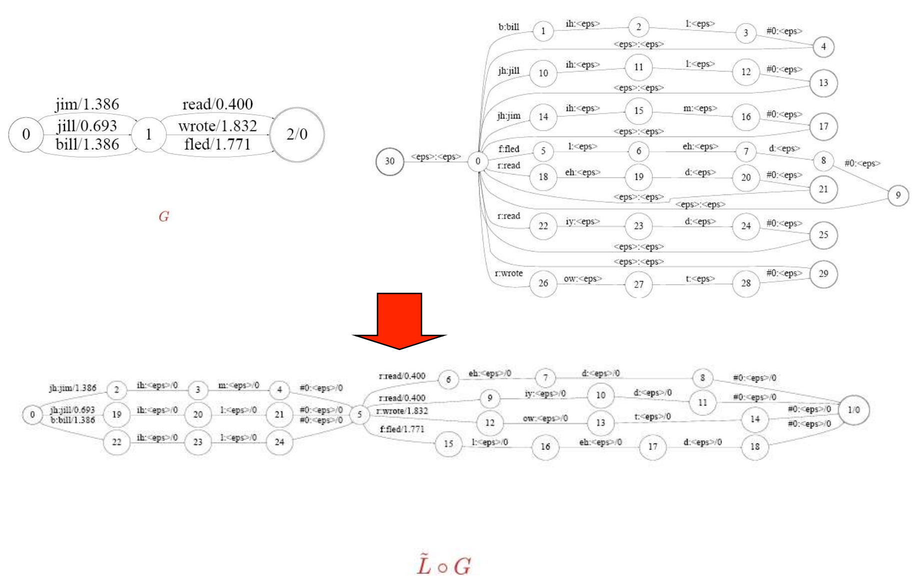
Context Dependency WFST



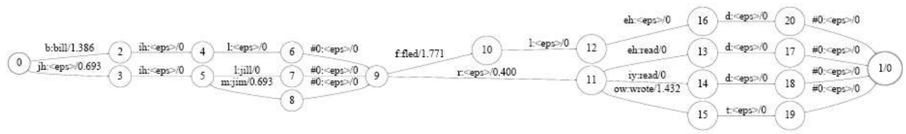
Context Dependency



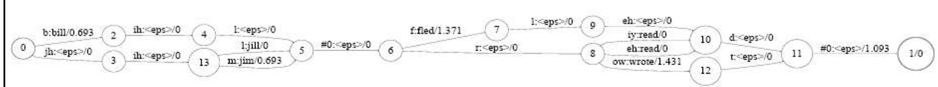
WFST for ASR



WFST for ASR

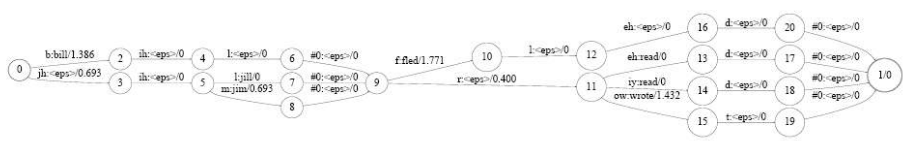


$det(\tilde{L} \circ G)$

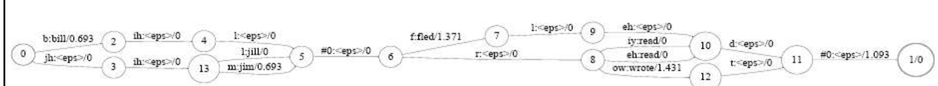


$min(det(\tilde{L} \circ G))$

WFST for ASR

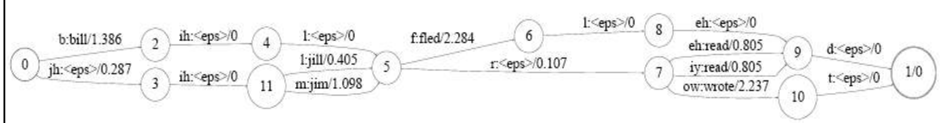


$det(\tilde{L} \circ G)$



$min(det(\tilde{L} \circ G))$

WFST for ASR



$$push(\pi_\epsilon(min(det(\tilde{L} \circ G))))$$

WFST for Speech Recognition

network	states	transitions
G	1,339,664	3,926,010
$L \circ G$	8,606,729	11,406,721
$det(L \circ G)$	7,082,404	9,836,629
$C \circ det(L \circ G))$	7,273,035	10,201,269
$det(H \circ C \circ L \circ G)$	18,317,359	21,237,992
F	3,188,274	6,108,907
$min(F)$	2,616,948	5,497,952

comparable size

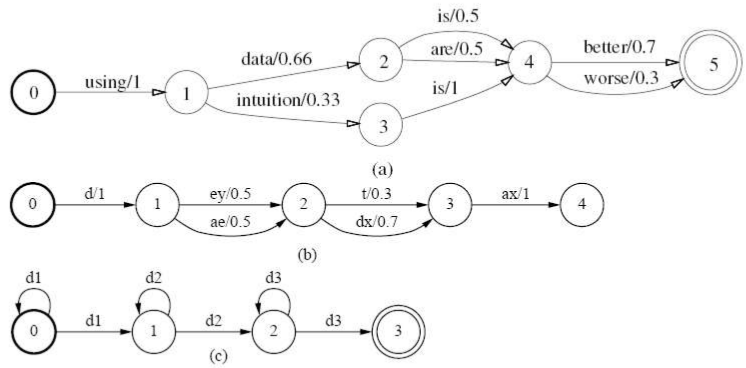
Table 1: Size of the first-passs recognition networks in the NAB 40, 000-word vocabulary task.

network	x real-time
$C \circ L \circ G$	12.5
$C \circ det(L \circ G)$	1.2
$det(H \circ C \circ L \circ G)$	1.0
$min(F)$	0.7

Table 2: Recognition speed of the first-pass networks in the NAB 40, 000-word vocabulary task at 83% word accuracy

Weighted Finite State Transducer (WFST)

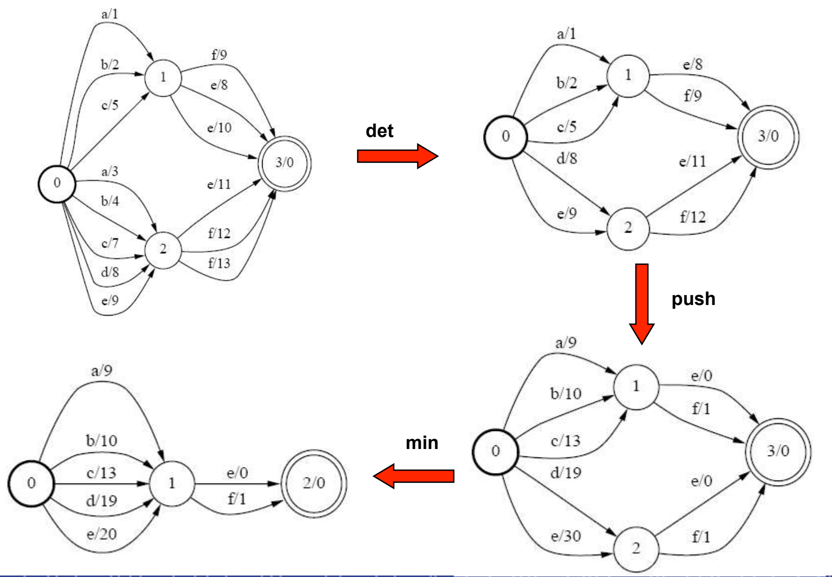
- **WFST: weighted finite state transducer (or acceptor):**



WFST Operations

- **Composition:** $C = A \circ B$
- **Determinization:** $D = det(C)$
  - ***deterministic automaton: every state has at most one out-going transition with any given label.***
- **Re-weighting (Weight pushing):** $E = push(D)$
- **Minimization:** $F = min(E)$

WFST Operations: Examples



Multiple Outputs

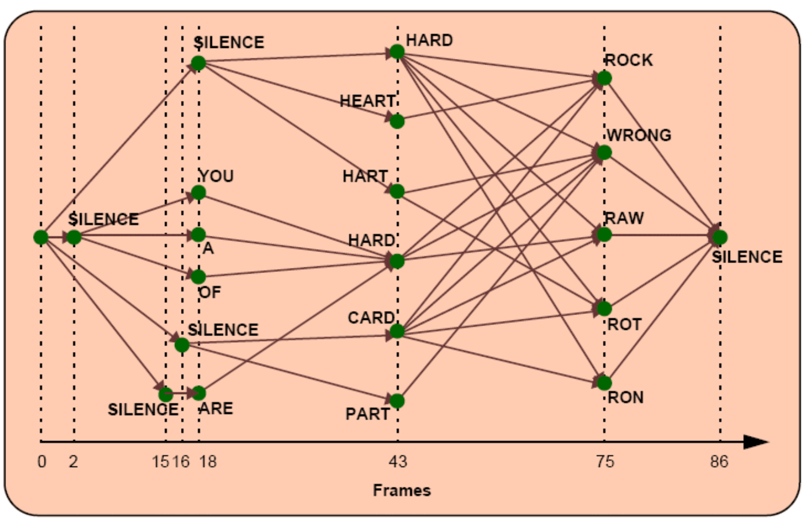
- How to generate a short list of multiple outputs instead of a single best?
  - To apply more knowledge to pick up one.
- N-Best List:
  - A list of top N best candidates
- Word graph:
  - A compact representation of a large number of candidates.
- How to generate N-best list or word graph from search process?
  - Standard Viterbi search can find the best one.
  - Modify the Viterbi somewhat for this feature.

N-Best List: example

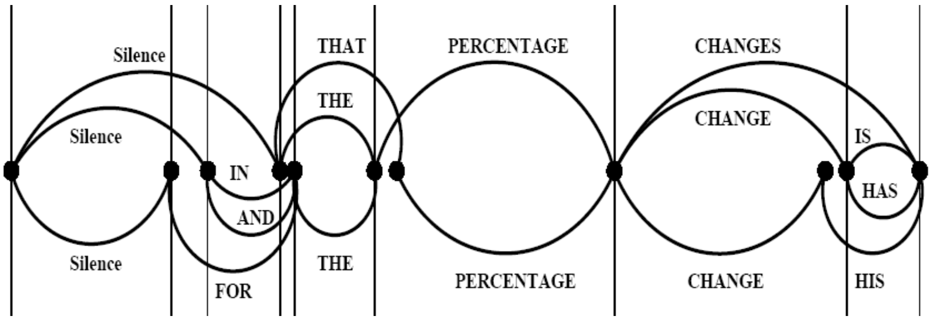
Rank	Hypotheses	Likelihood
1	SILENCE HARD ROCK SILENCE	-5880.11
2	SILENCE HARD WRONG SILENCE	-5905.17
3	SILENCE HARD RAW SILENCE	-5906.32
4	SILENCE A HARD ROCK SILENCE	-5920.68
5	SILENCE HARD ROT SILENCE	-5922.05
6	SILENCE HARD RON SILENCE	-5923.69
7	SILENCE CARD WRONG SILENCE	-5924.51
8	SILENCE CARD RAW SILENCE	-5925.66
9	SILENCE YOU HARD ROCK SILENCE	-5928.95
10	SILENCE HART WRONG SILENCE	-5929.97
11	SILENCE HEART WRONG SILENCE	-5930.42
12	SILENCE ARE HARD ROCK SILENCE	-5936.11
13	SILENCE CARD ROCK SILENCE	-5936.86
14	SILENCE OF HARD ROCK SILENCE	-5937.56
15	SILENCE CARD ROT SILENCE	-5941.39
16	SILENCE CARD RON SILENCE	-5943.03
17	SILENCE A HARD WRONG SILENCE	-5945.74
18	SILENCE PART WRONG SILENCE	-5946.36
19	SILENCE HART ROT SILENCE	-5946.85
20	SILENCE A HARD RAW SILENCE	-5946.89

True Transcription: *hard rock*

Word Graph (Lattice): example (1)



Word Graph: example (2)



Other search strategies:

- Viterbi algorithm: time-synchronous breadth-first search
- Depth-first: A* search (or stack decoding)
  - Time-asynchronous search
  - Expend and evaluate partial hypothesis from a stack.
  - Widely used in AI search.
  - Admissible: the best path is guaranteed as long as the heuristics are not over-estimated.
  - Not popular anymore in speech recognition.
  - NO TIME to cover.