Search and Decoding (Part II)

Lecture 17

CS 753

Instructor: Preethi Jyothi

Recap: Viterbi beam search decoder

- Time-synchronous search algorithm:
    - For time t, each state is updated by the best score from all states in time t-1
- Beam search prunes unpromising states at every time step.
- At each time-step t, only retain those nodes in the time-state trellis that are within a fixed threshold δ (beam width) of the score of the best hypothesis.

Recap: What are lattices?

- "**Lattices**" are useful when more than one hypothesis is desired from a recognition pass
- A lattice is a weighted, directed acyclic graph which encodes a large number of ASR hypotheses weighted by acoustic model +language model scores specific to a given utterance

Lattice construction using lattice-beam

- Produce a state-level lattice, prune it using “lattice-beam” width (s.t. only arcs or states on paths that are within cutoff cost = best_path_cost + lattice-beam will be retained) and then determinize s.t. there’s a single path for every word sequence
- Naive algorithm
  - Maintain a list of active tokens and links during decoding
  - Turn this structure into an FST, L.
  - When we reach the end of the utterance, prune L using lattice-beam.

A* stack decoder

- So far, we considered a time-synchronous search algorithm that moves through the observation sequence step-by-step
- A* stack decoding is a time-asynchronous algorithm that proceeds by extending one or more hypotheses word by word (i.e. no constraint on hypotheses ending at the same time)
- Running hypotheses are handled using a priority queue sorted on scores. Two problems to be addressed:

  1. Which hypotheses should be extended? (Use A*)
  2. How to choose the next word used in the extensions? (fast-match)

Recall A* algorithm

- To find the best path from a node to a goal node within a weighted graph,
  - A* maintains a tree of paths until one of them terminates in a goal node
  - A* expands a path that minimises `f(n) = g(n) + h(n)` where `n` is the final node on the path, `g(n)` is the cost from the start node to `n` and `h(n)` is a heuristic determining the cost from `n` to the goal node
  - `h(n)`must be admissible i.e. it shouldn't overestimate the true cost to the nearest goal node

---

Nice animations: http://www.redblobgames.com/pathfinding/a-star/introduction.html

A* stack decoder

- So far, we considered a time-synchronous search algorithm that moves through the observation sequence step-by-step
- A* stack decoding is a time-asynchronous algorithm that proceeds by extending one or more hypotheses word by word (i.e. no constraint on hypotheses ending at the same time)
- Running hypotheses are handled using a priority queue sorted on scores. Two problems to be addressed:

  1. Which hypotheses should be extended? (Use A*)
  2. How to choose the next word used in the extensions? (fast-match)

Which hypotheses should be extended?

- A* maintains a priority queue of partial paths and chooses the one with the highest score to be extended
- Score should be related to probability: For a word sequence W given an acoustic sequence O, score $\propto$ Pr(O|W)Pr(W)
- But not exactly this score because this will be biased towards shorter paths
- A* evaluation function based on f(p) = g(p) + h(p) for a partial path p where
    - g(p) = score from the beginning of the utterance to the end of p
    - h(p) = estimate of best scoring extension from p to end of the utterance
- An example of h(p): Compute some average probability prob per frame (over a training corpus). Then h(p) = prob $\times$ (T-t) where t is the end time of the hypothesis and T is the length of the utterance

A* stack decoder

- So far, we considered a time-synchronous search algorithm that moves through the observation sequence step-by-step
- A* stack decoding is a time-asynchronous algorithm that proceeds by extending one or more hypotheses word by word (i.e. no constraint on hypotheses ending at the same time)
- Running hypotheses are handled using a stack which is a priority queue sorted on scores. Two problems to be addressed:

    1. Which hypotheses should be extended? (Use A*)
    2. How to choose the next word used in the extensions? (fast-match)

Fast-match

- Fast-match: Algorithm to quickly find words in the lexicon that are a good match to a portion of the acoustic input

- Acoustics are split into a front part, A, (accounted by the word string so far, W) and the remaining part A'. Fast-match is to find a small subset of words that best match the beginning of A'.

    - Many techniques exist: 1) Rapidly find Pr(A'|w) for all w in the vocabulary and choose words that exceed a threshold
      2) Vocabulary is pre-clustered into subsets of acoustically similar words. Each cluster is associated with a centroid. Match A' against the centroids and choose subsets having centroids whose match exceeds a threshold

---

[B et al.]: Bahl et al., Fast match for continuous speech recognition using allophonic models, 1992

A* stack decoder

function STACK-DECODING() **returns** *min-distance*

- Initialize the priority queue with a null sentence.
- Pop the best (highest score) sentence s off the queue.
- If (s is marked end-of-sentence (EOS)) output s and terminate.
- Get list of candidate next words by doing fast matches.
- For each candidate next word w:
  - Create a new candidate sentence $s+w$.
  - Use forward algorithm to compute acoustic likelihood L of $s+w$
  - Compute language model probability P of extended sentence $s+w$
  - Compute "score" for $s+w$ (a function of L, P, and ???)
  - if (end-of-sentence) set EOS flag for $s+w$.
  - Insert $s+w$ into the queue together with its score and EOS flag

Image from [JM]: Jurafsky & Martin, SLP 2nd edition, Chapter 10

Example (1)

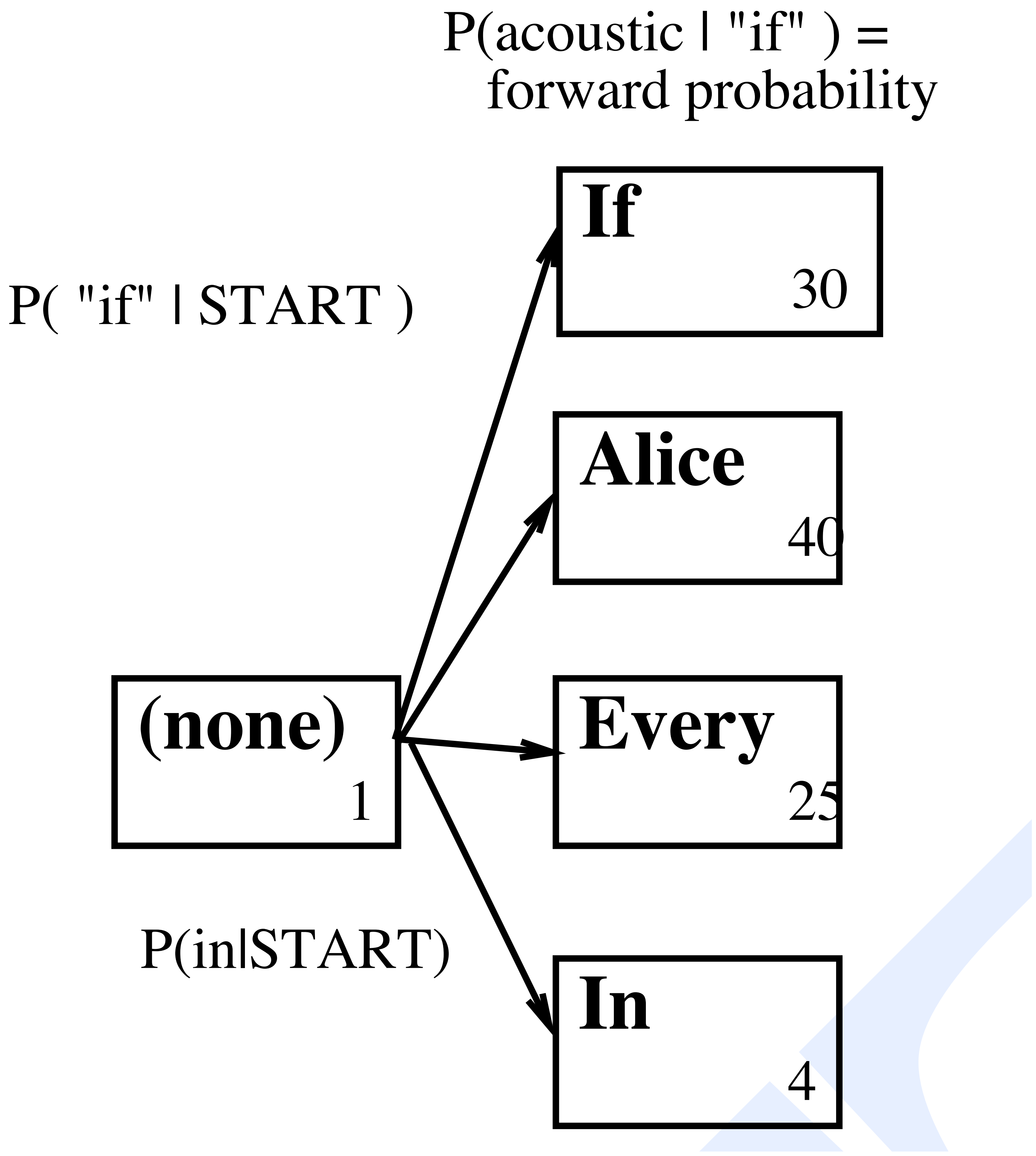


Image from [JM]: Jurafsky & Martin, SLP 2nd edition, Chapter 10

Example (2)

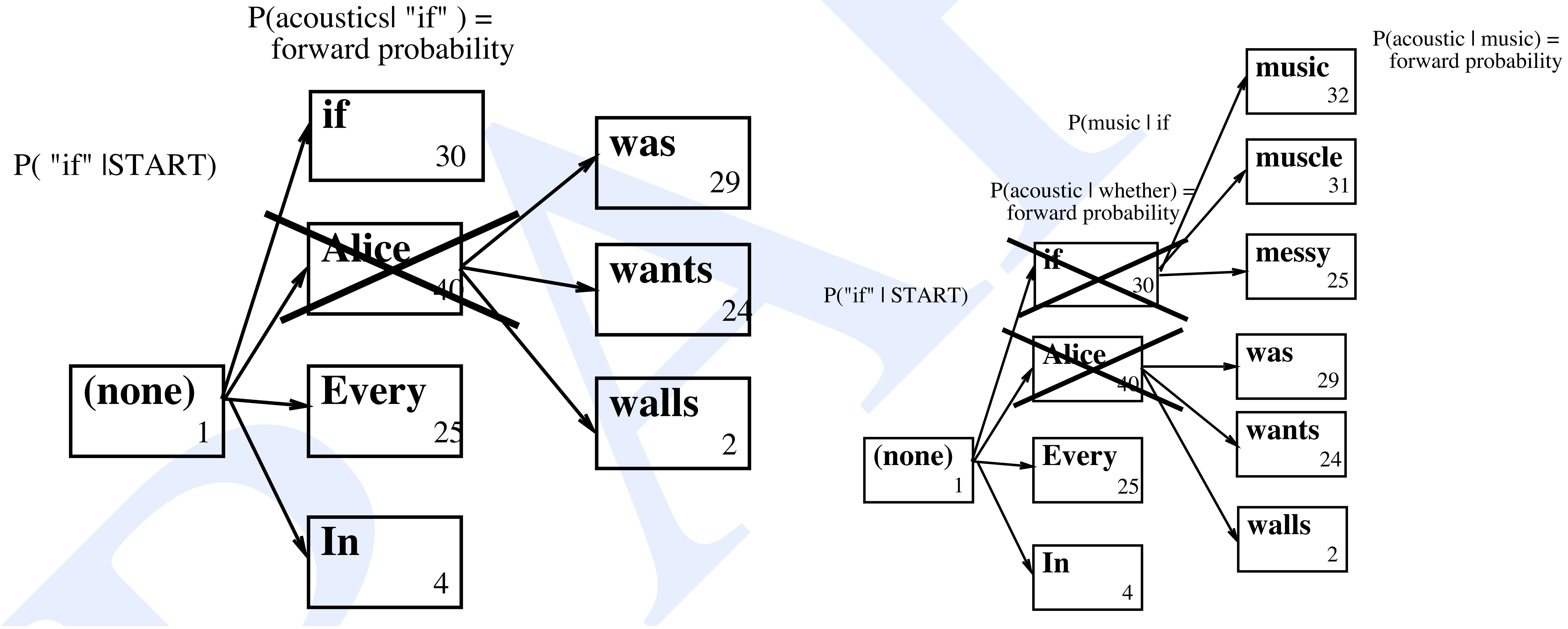


Image from [JM]: Jurafsky & Martin, SLP 2nd edition, Chapter 10

Moving on to multi-pass decoding

- We learned about two algorithms (beam search & A*) with the help of which one can search through the decoding graph in a first-pass decoding

- However, some models are too expensive to implement in first-pass decoding (e.g. RNN-based LMs)

- Multi-pass decoding:
  - First, use simpler model (e.g. Ngram LMs) to find most probable word sequences and represent as a word lattice or N-best list
  - Rescore first-pass hypotheses using complex model to find the best word sequence

Multi-pass decoding with N-best lists

- Simple algorithm: Modify the Viterbi algorithm to return the N-best word sequences for a given speech input

Rank	Path	AM logprob	LM logprob
1.	it’s an area that’s naturally sort of mysterious	-7193.53	-20.25
2.	that’s an area that’s naturally sort of mysterious	-7192.28	-21.11
3.	it’s an area that’s not really sort of mysterious	-7221.68	-18.91
4.	that scenario that’s naturally sort of mysterious	-7189.19	-22.08
5.	there’s an area that’s naturally sort of mysterious	-7198.35	-21.34
6.	that’s an area that’s not really sort of mysterious	-7220.44	-19.77
7.	the scenario that’s naturally sort of mysterious	-7205.42	-21.50
8.	so it’s an area that’s naturally sort of mysterious	-7195.92	-21.71
9.	that scenario that’s not really sort of mysterious	-7217.34	-20.70
10.	there’s an area that’s not really sort of mysterious	-7226.51	-20.01

- Problem: N-best lists aren’t as diverse as we’d like. And, not enough information in N-best lists to effectively use other knowledge sources

Image from [JM]: Jurafsky & Martin, SLP 2nd edition, Chapter 10

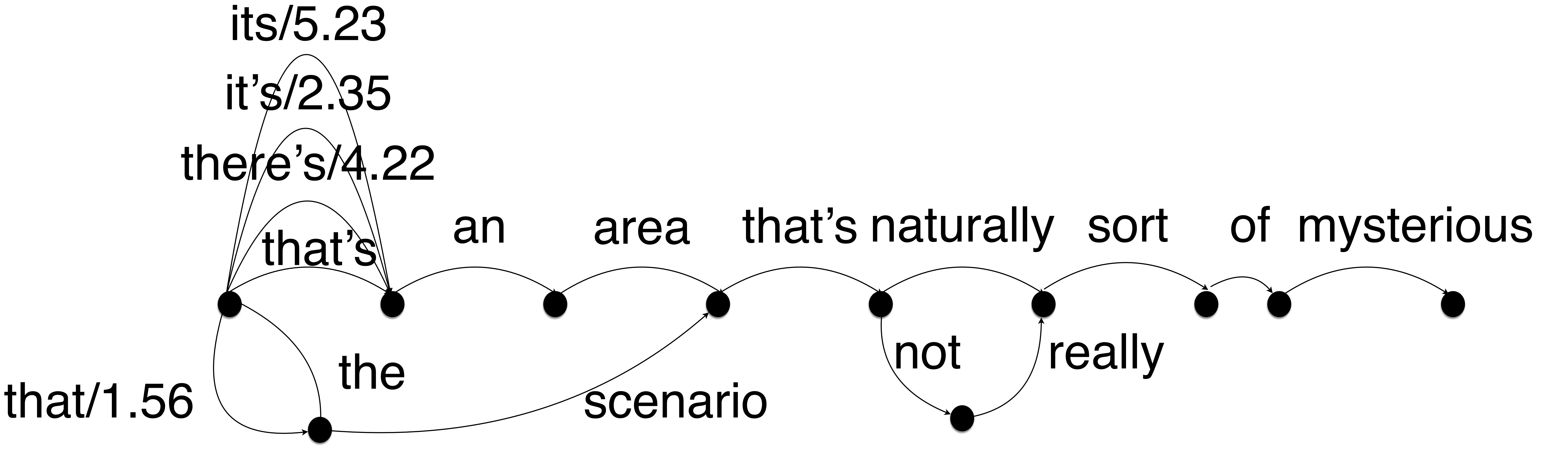
Multi-pass decoding with N-best lists

- Simple algorithm: Modify the Viterbi algorithm to return the N-best word sequences for a given speech input

Image from [JM]: Jurafsky & Martin, SLP 2nd edition, Chapter 10

Multi-pass decoding with lattices

ASR lattice: Weighted automata/directed graph representing alternate ASR hypotheses



Multi-pass decoding with lattices

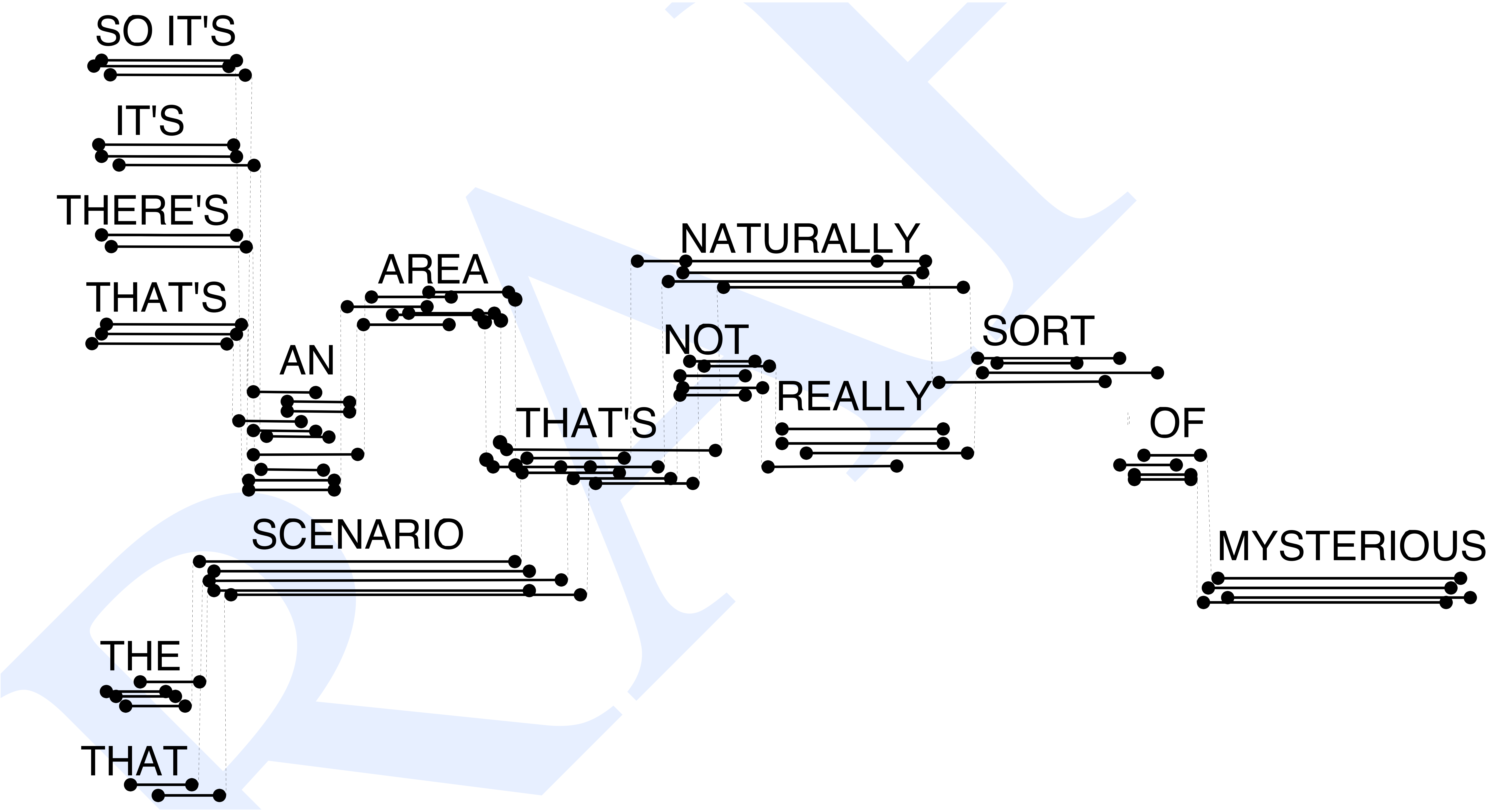
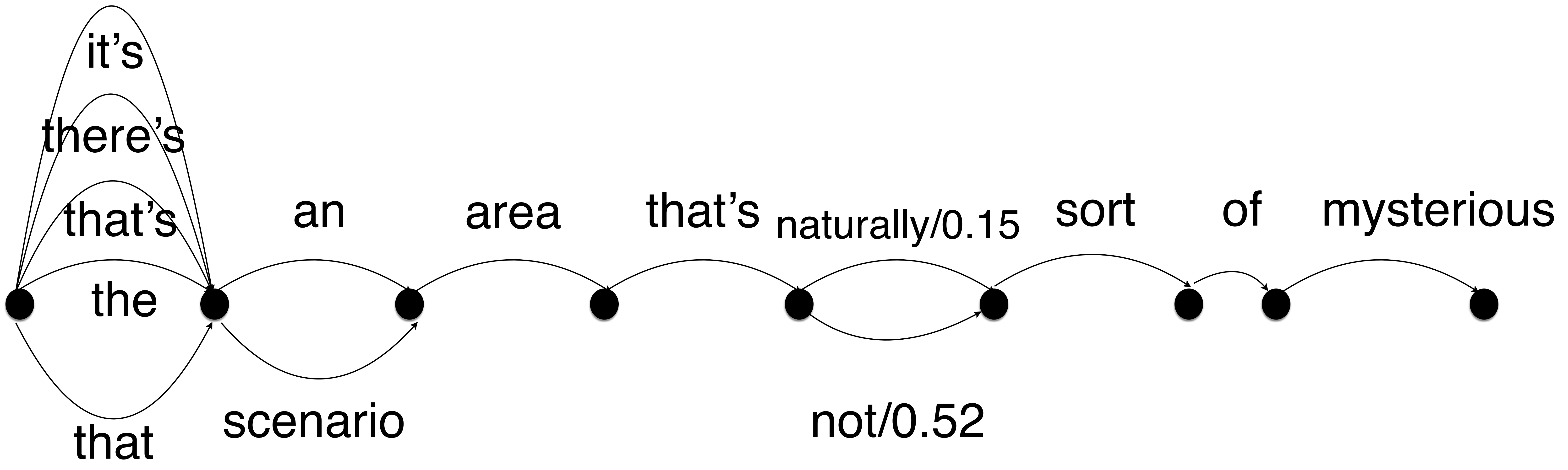


Image from [JM]: Jurafsky & Martin, SLP 2nd edition, Chapter 10

Multi-pass decoding with confusion networks

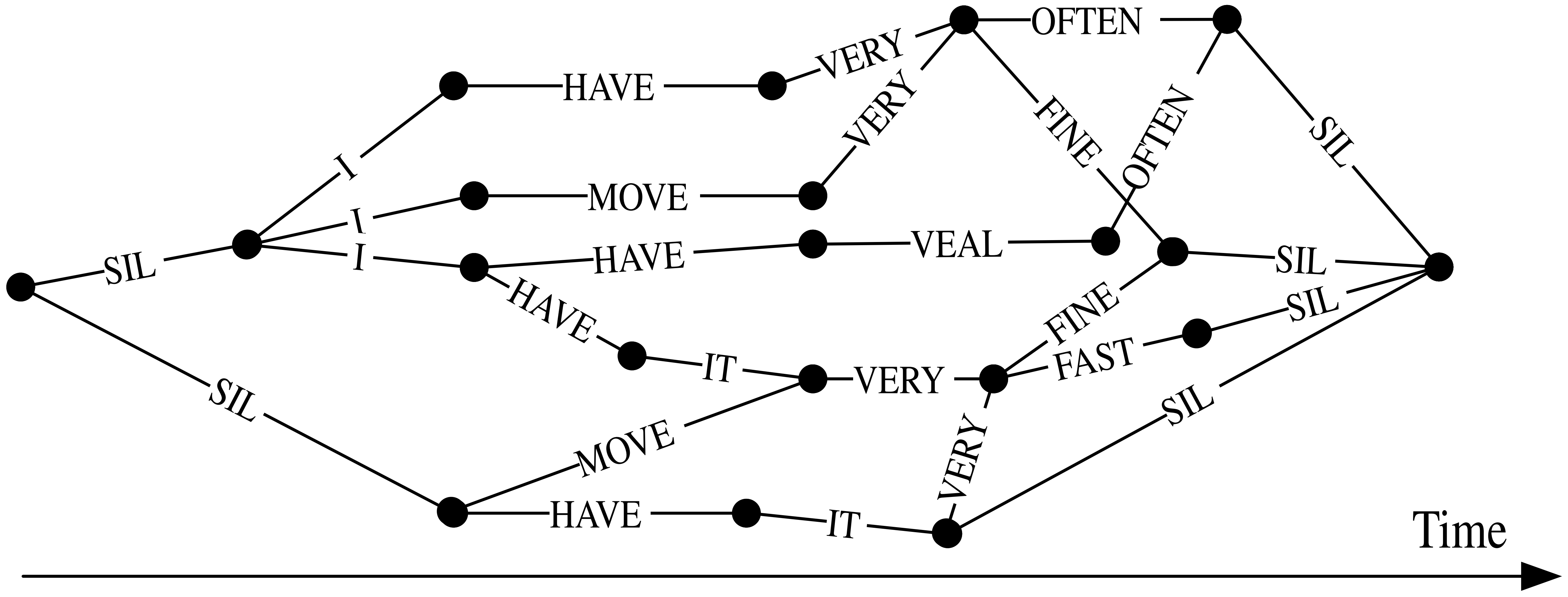
- Confusion networks/sausages: Lattices that show competing/ confusable words and can be used to compute posterior probabilities at the word level



Word Confusion Networks

Word confusion networks are normalised word lattices that provide alignments for a fraction of word sequences in the word lattice

(a) Word Lattice



(b) Confusion Network

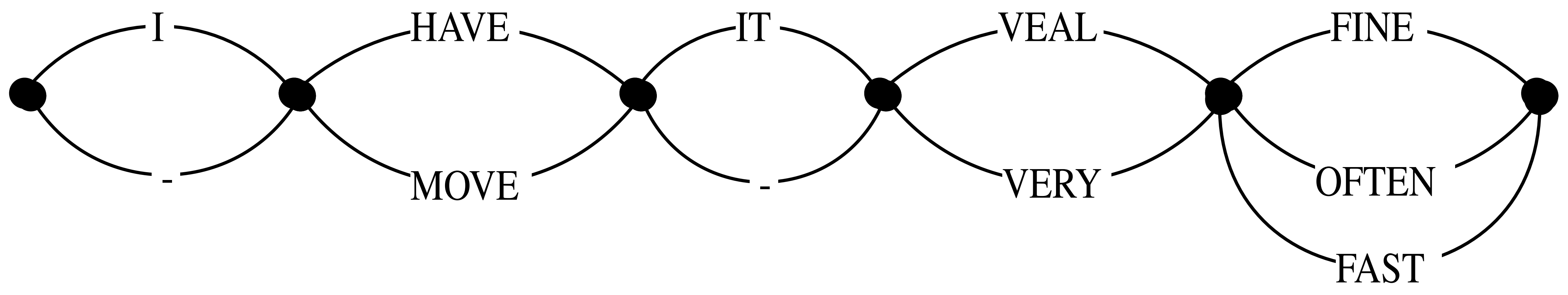
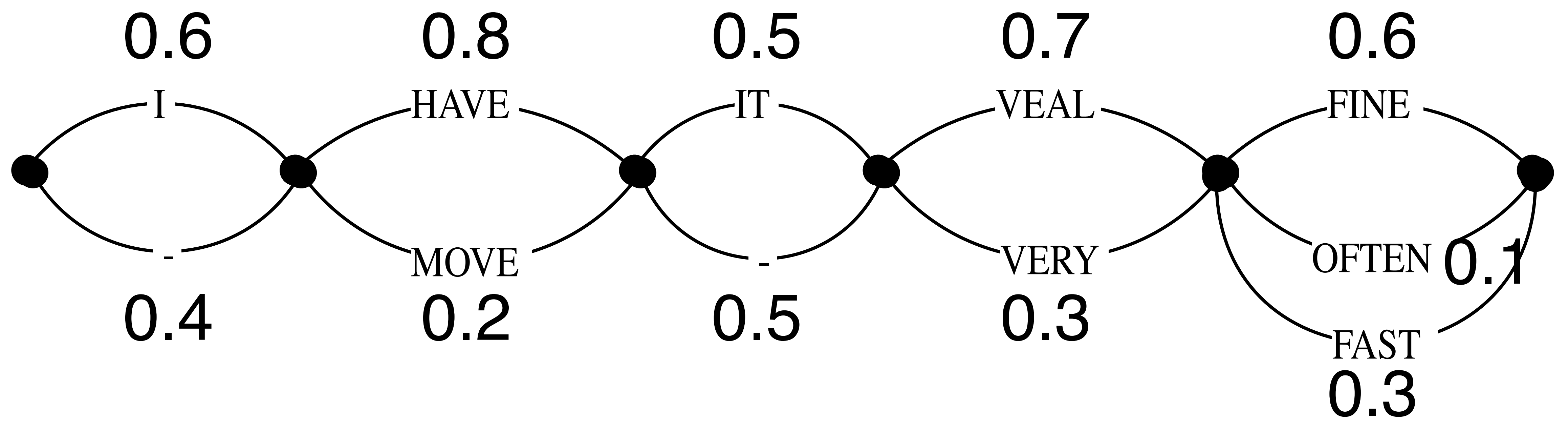


Image from [GY08]: Gales & Young, Application of HMMs in speech recognition, NOW book, 2008

Word posterior probabilities in the word confusion network

- Each arc in the confusion network is marked with the posterior probability of the corresponding word w
- First, find the link probability of w from the word lattice:
  - Joint probability of a path a (corr. to word sequence w) and acoustic observations O: $\mathrm{Pr}(a, O) = \mathrm{Pr}_{\mathrm{AM}}(O|a)\mathrm{Pr}_{\mathrm{LM}}(w)$
  - For each link l, the joint probabilities of all paths through l are summed to find the link probability:

$$\mathrm{Pr}(l|O) = \frac{\sum_{a \in \mathcal{A}} \mathrm{Pr}(a, O)}{\mathrm{Pr}(O)}$$



Constructing word confusion network

- Second step in estimating word posteriors is the clustering of links that correspond to the same word/confusion set
- This clustering is done in two stages:
  1. Links that correspond to the same word and overlap in time are combined
  2. Links corresponding to different words are clustered into confusion sets. Clustering algorithm is based on phonetic similarity, time overlap and word posteriors. More details in [LBS00]

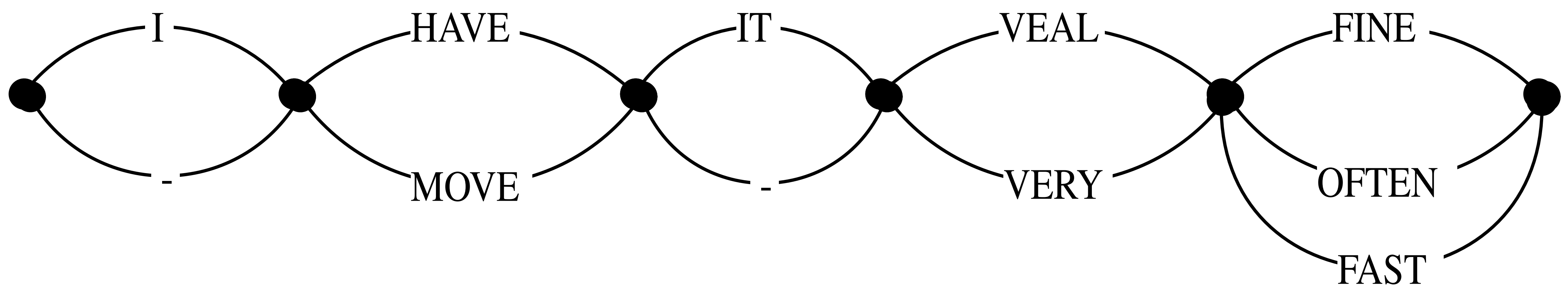


Image from [LBS00]: L. Mangu et al., "Finding consensus in speech recognition", Computer Speech & Lang, 2000

System Combination

- Combining recognition outputs from multiple systems to produce a hypothesis that is more accurate than any of the original systems

- Most widely used technique: ROVER [ROVER].

  - 1-best word sequences from each system are aligned using a greedy dynamic programming algorithm

  - Voting-based decision made for words aligned together

  - Can we do better than just looking at 1-best sequences?

bbn1.ctm	there's	a	lot	of	@	like	societies	@	@	ruin	engineers	and	lakes
cmu-isl1.ctm	there's	the	labs	@	@	like	societies	@	for	women	engineers	i	think
cu-htk2.ctm	there's	the	last	@	@	like	societies	@	true	of	engineers	and	like
dragon1.ctm	was	@	alive	@	the	legal	society	is	for	women	engineers	and	like
sri1.ctm	there's	a	lot	of	@	like	society's	@	@	through	engineers	@	like

Image from [ROVER]: Fiscus, Post-processing method to yield reduced word error rates, 1997

System Combination

- Combining recognition outputs from multiple systems to produce a hypothesis that is more accurate than any of the original systems

- Most widely used technique: ROVER [ROVER].

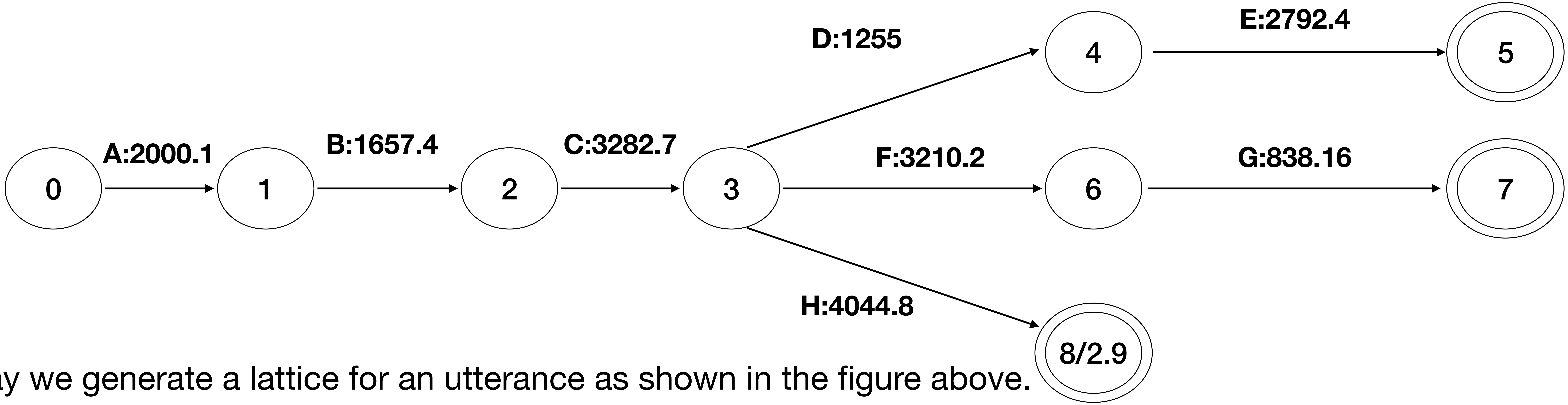
  - 1-best word sequences from each system are aligned using a greedy dynamic programming algorithm

  - Voting-based decision made for words aligned together

  - Could align confusion networks instead of 1-best sequences

bbn1.ctm	there's	a	lot	of	@	like	societies	@	@	ruin	engineers	and	lakes
cmu-isl1.ctm	there's	the	labs	@	@	like	societies	@	for	women	engineers	i	think
cu-htk2.ctm	there's	the	last	@	@	like	societies	@	true	of	engineers	and	like
dragon1.ctm	was	@	alive	@	the	legal	society	is	for	women	engineers	and	like
sri1.ctm	there's	a	lot	of	@	like	society's	@	@	through	engineers	@	like

Image from [ROVER]: Fiscus, Post-processing method to yield reduced word error rates, 1997



Say we generate a lattice for an utterance as shown in the figure above. Tick the correct answers for how the graph will change if this lattice is pruned with different values of beam size, B.

1. B = 2
   a) Graph will stay the same
   b) States 4 and 5 and arcs labeled with D and E will be pruned
   c) States 6 and 7 and arcs labeled with F and G will be pruned
   d) State 8 and the arc labeled with H will be pruned

2. B = 0.4
   a) Graph will stay the same
   b) States 4 and 5 and arcs labeled with D and E will be pruned
   c) States 6 and 7 and arcs labeled with F and G will be pruned
   d) State 8 and the arc labeled with H will be pruned