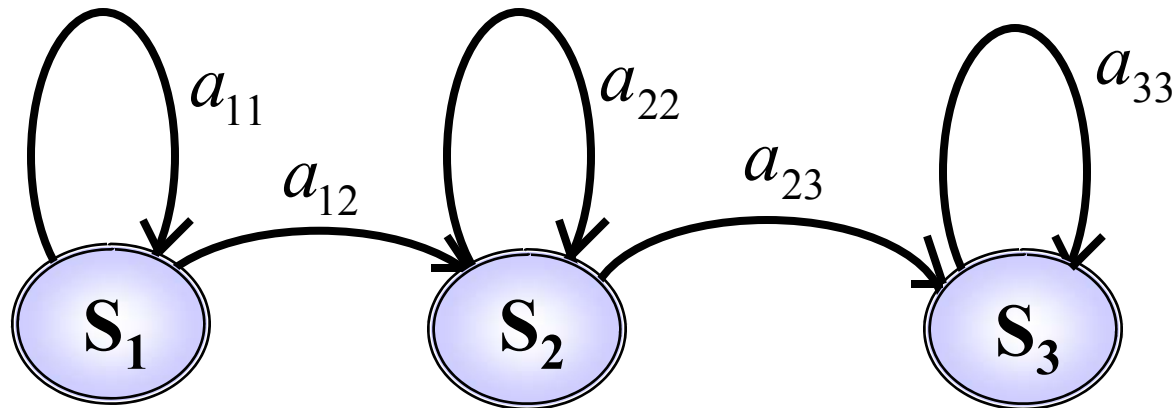


# Practical HMM Training (for Speech Recognition)

- **Our goal is to “assign” extracted feature vectors to HMM states**
- **Two popular methods for training,**
  - Forward-Backward training assigns a probability that each vector was emitted from each HMM state (fuzzy labeling)
  - Viterbi training just assigns a feature vector to a particular state (most likely state from the best path).

# Phoneme HMM

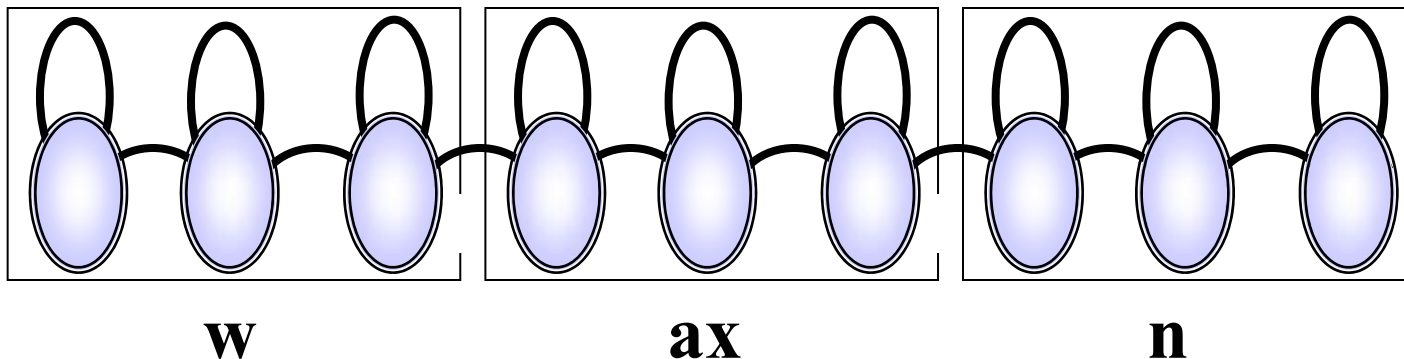
- Let's assume each phoneme is represented by 3 HMM states connected with forward transitions,



- S1 models the beginning part of the sound, S2 the middle, and S3 the end-part of the sound unit.

# Word & Sentence HMMs

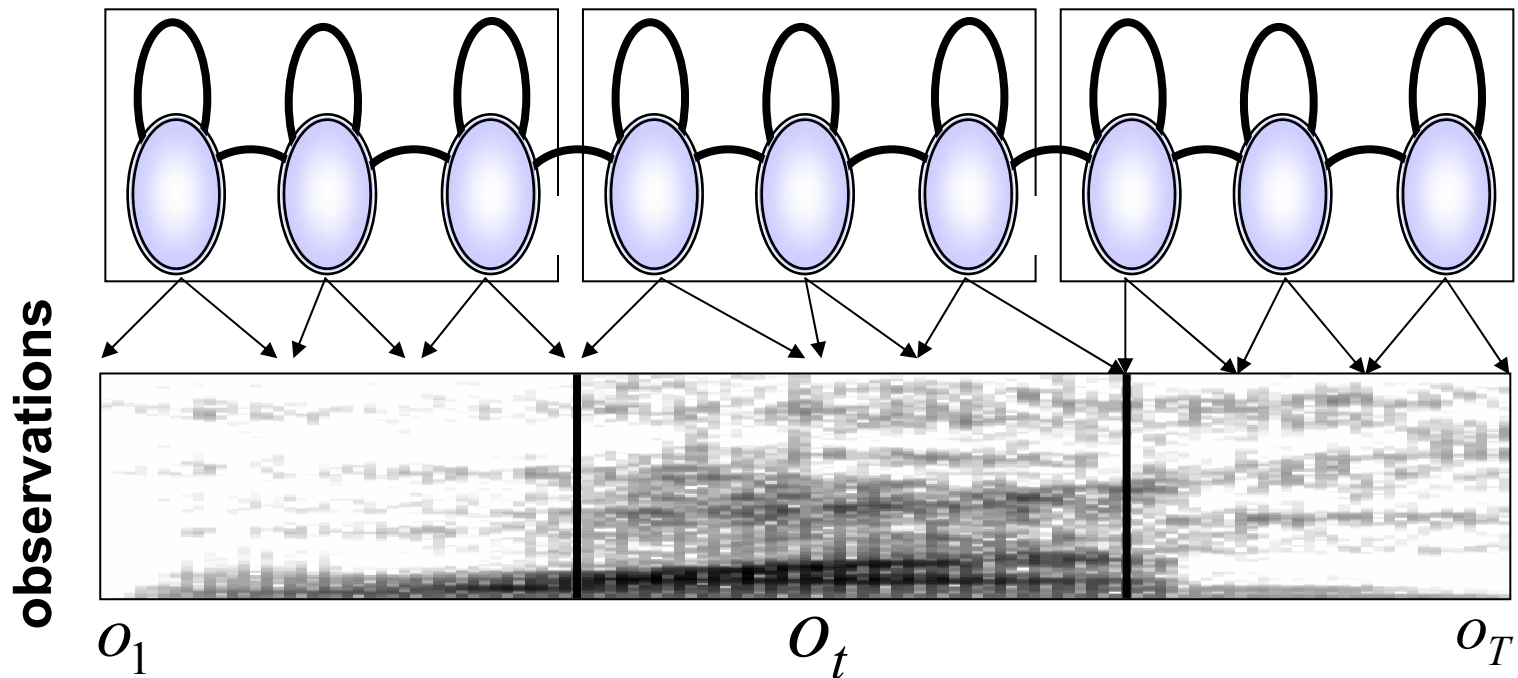
- Construct word & sentence-level HMMs from the phoneme-level units. For example, “ONE” with pronunciation “W AX N”:



- For simplification, let's assume each state is a Gaussian Mixture Model (GMM). We also have transition probabilities between states.

# Viterbi Training

- Given an utterance, we can construct the composite HMM from the phone units and use the Viterbi algorithm to find the best state-sequence (assignment of feature-vectors to HMM states):



# Viterbi Algorithm in Log-Domain

1. **Initialization**  $\tilde{\delta}_1(i) = \tilde{\pi}_i + \tilde{b}_i(\mathbf{o}_1) \quad \psi_1(i) = 0$

2. **Recursion** 
$$\tilde{\delta}_t(j) = \max_{1 \leq i \leq N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}] + \tilde{b}_j(\mathbf{o}_t)$$
$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}]$$

3. **Termination**  $\tilde{P}^* = \max_{1 \leq i \leq N} [\tilde{\delta}_T(i)] \quad q_T^* = \arg \max_{1 \leq i \leq N} [\tilde{\delta}_T(i)]$

4. **Path Back trace**  $q_t^* = \psi_{t+1}(q_{t+1}^*)$

# Viterbi Algorithm Illustration for Feed-Forward HMM Topology

$S_9$												
$S_8$												
$S_7$												
$S_6$												
$S_5$						$\tilde{\delta}_6(5)$						
$S_4$												
$S_3$												
$S_2$												
$S_1$												
	$o_1$	$o_2$	$o_3$	$o_4$	$o_5$	$o_6$	$o_7$	$o_8$	$o_9$	$o_{10}$	$o_{11}$	$o_{12}$

$$\tilde{\delta}_{t=6}(s=5) = \max \left\{ \underbrace{\left[ \tilde{\delta}_{t=5}(s=5) + \tilde{a}_{55} \right]}_{\text{self-loop}}, \underbrace{\left[ \tilde{\delta}_{t=5}(s=4) + \tilde{a}_{45} \right]}_{\text{forward-transition}} \right\} + \tilde{b}_{s=5}(t=6)$$

$$\psi_{t=6}(s=5) = \arg \max \left\{ \underbrace{\left[ \tilde{\delta}_{t=5}(s=5) + a_{55} \right]}_{\text{self-loop}}, \underbrace{\left[ \tilde{\delta}_{t=5}(s=4) + a_{45} \right]}_{\text{forward-transition}} \right\}$$

T-61.184

# Viterbi Training

- **For each training example, use current HMM models to assign (align) feature vectors to HMM states.**
  - Assignment is made by using the Viterbi algorithm.
  - Assignment is based on most-likely path through composite HMM model
  - We refer to this as “Viterbi forced-alignment”
- **Group feature vectors assigned to each HMM state and estimate new HMM state parameters (e.g., using GMM update equations).**
- **Repeat alignment / retraining process**

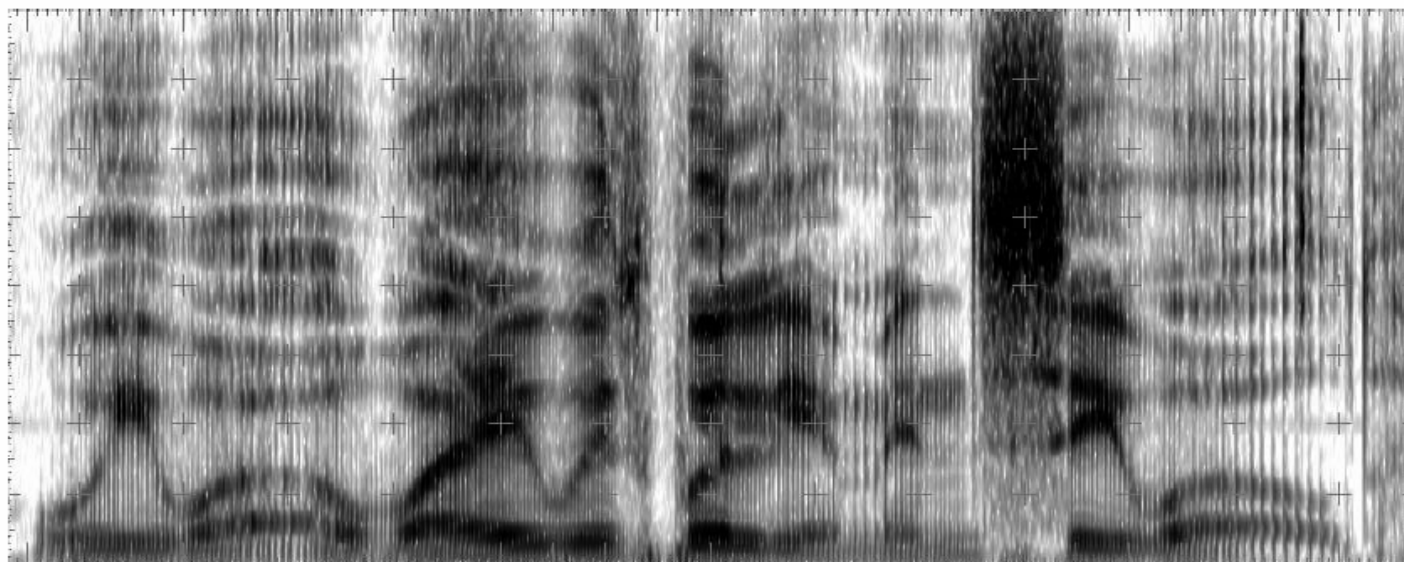
# Forward-Backward Training

- Rather than assigning each feature vector to a particular HMM state, we compute a “fuzzy-assignment”.
- “Fuzzy-assignment” is based on the probability of being in state  $i$  at time  $t$ ,
  - Requires computing the forward and backward variables.
  - FB training is more expensive than Viterbi training

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^N \alpha_t(j)\beta_t(j)}$$

## Acoustic Modeling Issues

- How to take into account variabilities in the acoustic signal? For example, the context-dependency of “w” in this example,



WE WERE AWAY WITH WILLIAM IN SEA WORLD

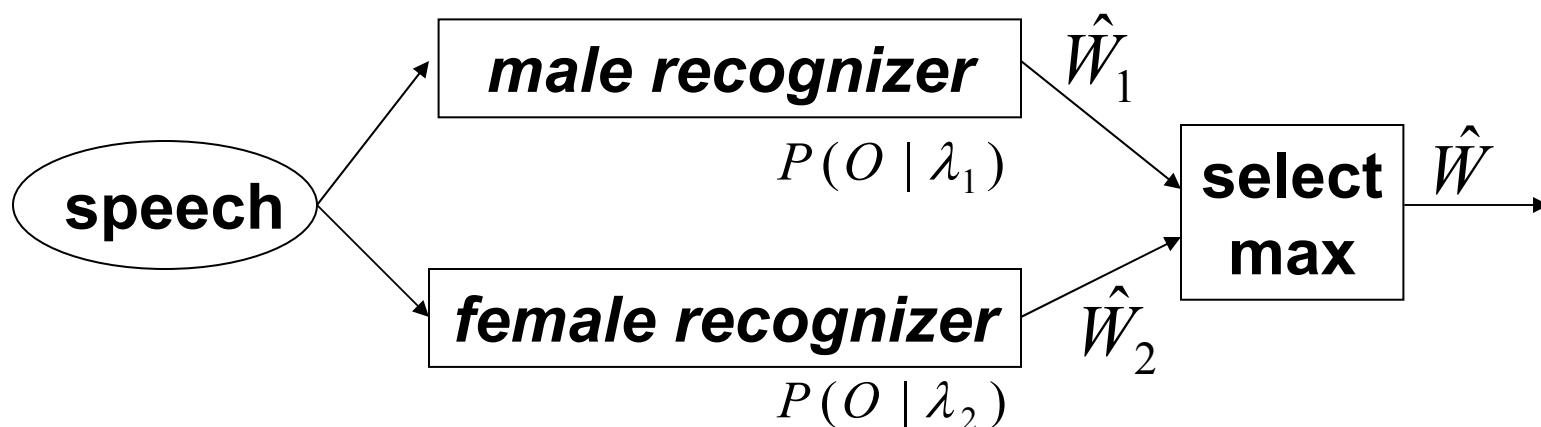
T-61.184

# Types of Acoustic Variability

- **Environmental Variability**
- **Between-Speaker Variability**
  - Gender, Age
  - Dialect
  - Speaking Style
    - formal vs. informal
    - Planned vs. spontaneous
- **Within-Speaker Variability**
  - Variations within an utterance (could be due to prosody)
  - Speaker-specific co-articulation

# Variability-Dependent Recognition

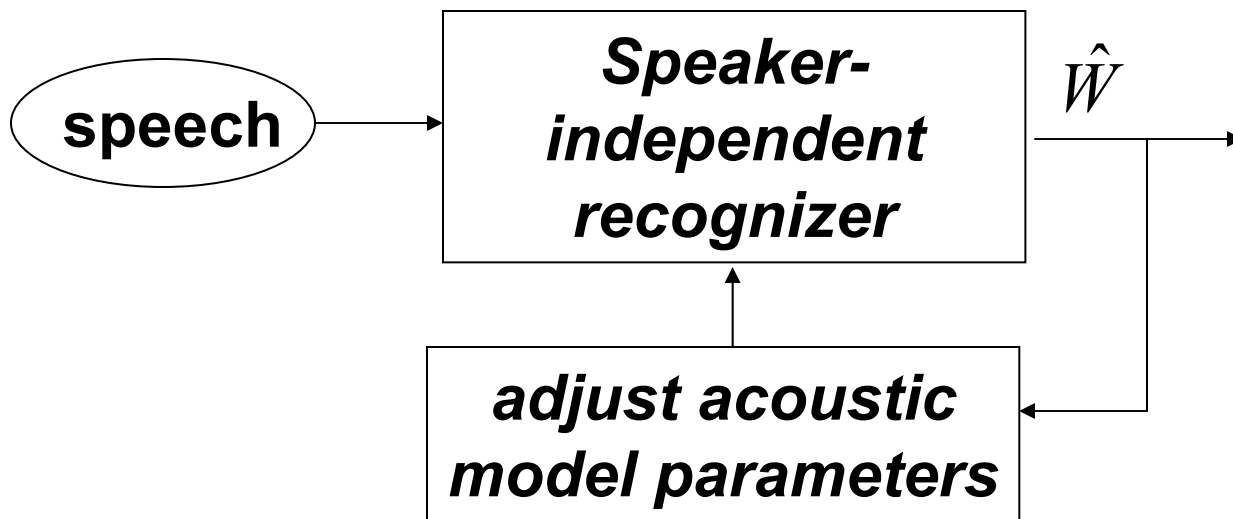
- One simple method might be to estimate model parameters under each condition,



- Can not account for all factors (data-sparse).  
Also very inefficient.

# Variability-Adapted Recognition

- Another solution adapts the parameters of the recognizer to better match the input,



- Adaptation can be supervised or unsupervised

## An Ideal Acoustic Model also...

- **Accounts for context-dependency**
  - A phoneme produced in one phonetic context may be similar to the same phoneme produced in another phonetic context. (the converse is also true!)
- **Provides a compact & trainable representation**
  - Which is trainable from finite amounts of data
- **Provides a general representation**
  - Allows new words to be modeled which may not have been seen in the training data

# Whole-Word HMMs

- **Assign a number of HMM states to model a word as a whole.**
- **Passes the test?**
  - Accurate – Yes, if you have enough data and your environment consists of a small vocabulary. No, if you are trying to model context changes between words.
  - Compact – No, need too many states as vocabulary increases. Probably not enough training data to model \*every\* word. What about infrequent words???
  - General – No, can't build new words using this representation.

# Context-Independent Phoneme HMMs

- **Context-independent models consist of a single  $M$ -state HMM (e.g.,  $M=3$ ), one for each phoneme unit**
- **Also referred to as “monophone” models**
- **Passes the test?**
  - Accurate – No, does not accurately model coarticulation
  - Compact – Yes,  $M$  states x  $N$  phonemes leads to only a few parameters which need to be estimated.
  - General – Yes, you can construct new words by stringing together the units.

# Context-Dependent Triphone HMMs

- **Context-dependent models which consist of a single 3-state HMM, one for each phoneme unit modeled with the immediate left-and-right phonetic context**
- **Passes the test?**
  - Accurate – Yes, takes coarticulation into account
  - Compact – Yes, Trainable – No: For N phonemes, there exists  $N \times N \times N$  triphone models. Too many parameters to estimate!
  - General – Yes, you can construct new words by stringing together the units.

# Describing Context-Dependent Phonetic Models

- **Monophone:**
  - ❑ A single model used to represent phoneme in all contexts.
  
- **Biphone:**
  - ❑ Each model represents a particular left or right context.
  - ❑ Left-context biphone notation: (a-b)
  - ❑ Right-context biphone notation: (b+c)
  
- **Triphone:**
  - ❑ Each model represents a particular left & right context.
  - ❑ (a-b+c) refers to phoneme “b” with “a” preceding and “c” immediately following.

# Context-Dependent Model Examples

- **Monophone:**

- BRYAN → B R **AY** **AX** N

- **Biphone**

- Left-Context: → SIL-**B** B-**R** R-**AY** **AY-AX** AX-**N**

- Right-Context: → **B+R** **R+AY** **AY+AX** **AX+N** **N+SIL**

- **Triphone**

- → SIL-**B+R** B-**R+AY** R-**AY+AX** **AY-AX+N** AX-**N+SIL**

# Word-Boundary Modeling

- ***Word-internal Context-Dependent Model Sequence***  
(backs off to left and right biphone models at word boundaries):

BRYAN PELLOM → SIL **B**+R B-**R**-AY R-**AY**+AX  
AY-**AX**+N AX-**N** **P**+EH P-**EH**+L EH-**L**+AX L-**AX**+M  
AX-**M** SIL

- ***Cross-Word Context-Dependent Triphone Sequence***

BRYAN PELLOM → SIL-**B**+R B-**R**+AY R-**AY**+AX AY-**AX**+N  
AX-**N**+P N-**P**+EH P-**EH**+L EH-**L**+AX L-**AX**+M AX-**M**+SIL

# Triphone Acoustic Models

- **Provide nice trade-off**
  - ❑ Compact, General, Accurate
  - ❑ Assumes dependency on just previous and following phoneme.
- **Modeling and Estimation Issues:**
  - ❑ Not all triphone contexts appear in training data.  
We call these “unseen” triphones.
  - ❑ Many triphone contexts occur infrequently in the training data  
(data-sparse modeling problem)
- **Solution**
  - ❑ Cluster HMM states which share similar statistical distributions
  - ❑ Estimate HMM parameters using resulting pooled data
  - ❑ How to cluster the data?????

# Trainability of Acoustic Models

- **Tradeoff exists between the level of detail of the acoustic model and our ability to adequately estimate the parameters of the model**
- **Methods for improving trainability,**
  - ❑ Backing-off : triphones → biphones → monophones
  - ❑ Smoothing : interpolate parameters of more specific models with those of less specific (better trained) models
  - ❑ Sharing : cluster similar contexts

## Recall the Bayes Rule Formulation

- Using Bayes Rule,

$$P(W | O) = \frac{P(O | W)P(W)}{P(O)}$$

- Since  $P(O)$  does not impact optimization,

$$\begin{aligned}\hat{W} &= \arg \max_W P(W | O) \\ &= \arg \max_W P(O | W)P(W)\end{aligned}$$

# Practical Speech Recognition

- In practice, we work with log-probabilities,

$$\hat{W} = \arg \max_W \{ \log(P(O | W)P(W)) \}$$

- Common to scale LM probabilities by a grammar scale factor (“s”) and also include a word-transition penalty (“p”):

$$\hat{W} = \arg \max_W \left\{ \underbrace{\log(P(O | W))}_{\text{acoustic model}} + s \cdot \underbrace{\log(P(W))}_{\text{language model}} + p \right\}$$

# Language Models

- Assign probabilities to word sequences  $P(W)$
- Aids in reducing search space and ambiguity
- Resolves most homonyms:

Write a letter to Mr. Wright right away

- Constraint / Flexibility tradeoff

# Statistical Language Models

- **Want to estimate,**

$$P(W) = P(w_1 w_2 \cdots w_N)$$

- **Can decompose probability left-to-right**

$$\begin{aligned} P(W) &= P(w_1, w_2, \dots, w_N) \\ &= P(w_1)P(w_2 | w_1) \cdots P(w_N | w_1, w_2 \cdots w_{N-1}) \\ &= \prod_{n=1}^N P(w_n | w_1, w_2 \cdots w_{n-1}) \end{aligned}$$

## Statistical Language Model Example

$$\begin{aligned} P(W) &= P(\text{center for spoken language research}) \\ &= P(\text{center})P(\text{for} \mid \text{center})P(\text{spoken} \mid \text{center for})\cdots \\ &\quad \cdots P(\text{research} \mid \text{center for spoken language}) \end{aligned}$$

- **Impossible to model the entire word sequence... never enough training data!**
- **Need to consider restricting the word-history used in computation of the probability estimate.**

## “Markov Model” of Language

- Cluster histories ending in same last N-1 words.  
“Markov Model” of Language.

- **N=1**  $P(w_n | w_1, w_2 \cdots w_{n-1}) = P(w_n)$

- **N=2**  $P(w_n | w_1, w_2 \cdots w_{n-1}) = P(w_n | w_{n-1})$

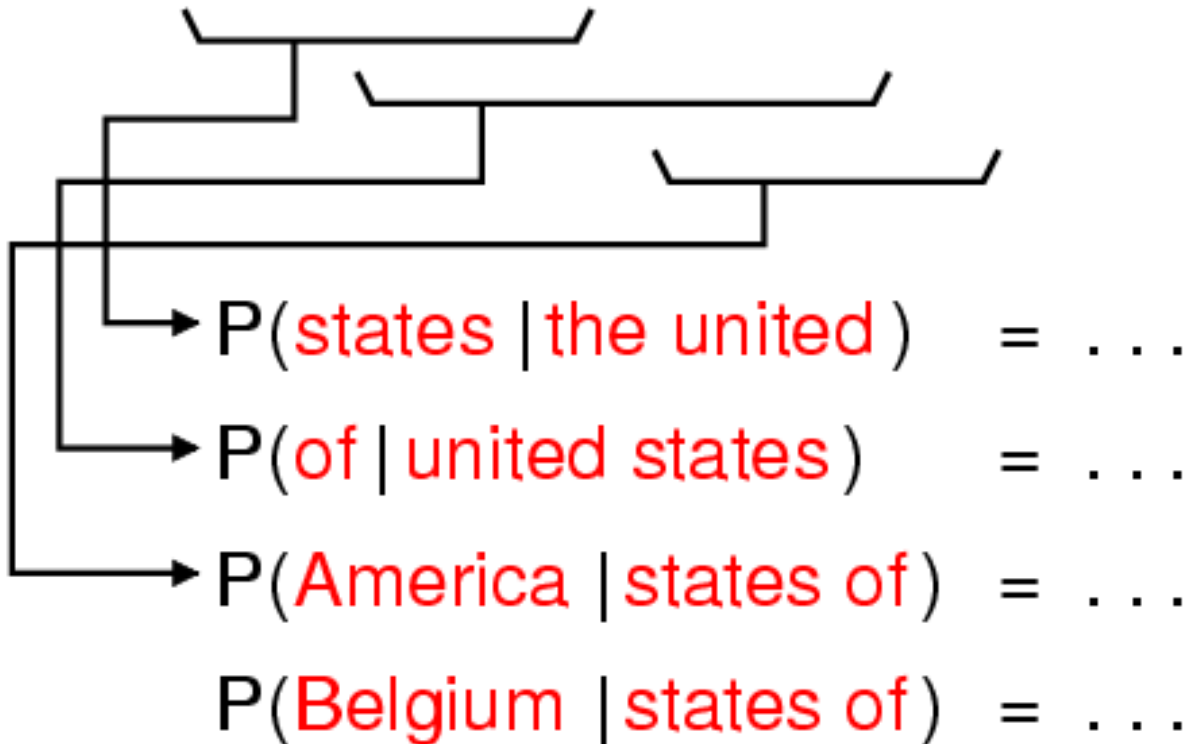
- **N=3**  $P(w_n | w_1, w_2 \cdots w_{n-1}) = P(w_n | w_{n-1}, w_{n-2})$

# N-gram Language Model

- **N-gram models compute the probability of a word based on previous N-1 words:**
  - ❑ N=1 (Unigram)
  - ❑ N=2 (Bigram)
  - ❑ N=3 (Trigram)
- **Probabilities are estimated from a corpus of training data (text data).**
- **Once model is known, new sentences can be randomly generated by the model!**
- **Syntax roughly encoded by model, but ungrammatical and semantically “strange” sentences can be produced**

## 3-gram Example

... the united states of ???



T-61.184

# Estimating N-gram Probabilities

- Given a text corpus, define the number of occurrences [count] of word (n) by,

$$C(w_n)$$

- Count of occurrences of word (n-1) followed by word (n),

$$C(w_{n-1}, w_n)$$

- And for 3 words,

$$C(w_{n-2}, w_{n-1}, w_n)$$

# Obtaining N-gram Probabilities

- **Maximum likelihood estimates of word probabilities are based on counting frequency of occurrence of word sequences from a training set of text data:**

$$P(w_n | w_{n-1}) = \frac{C(w_{n-1}, w_n)}{C(w_{n-1})}$$

$$P(w_n | w_{n-2}, w_{n-1}) = \frac{C(w_{n-2}, w_{n-1}, w_n)}{C(w_{n-2}, w_{n-1})}$$

# Search

- **Goal of ASR search is to find the most likely string of symbols (e.g., words) to account for the observed speech waveform:**

$$\hat{W} = \arg \max_W P(\mathbf{O} | W)P(W)$$

- **Types of input:**
  - Isolated Words
  - Connected Words

# Designing an Isolated-Word HMM

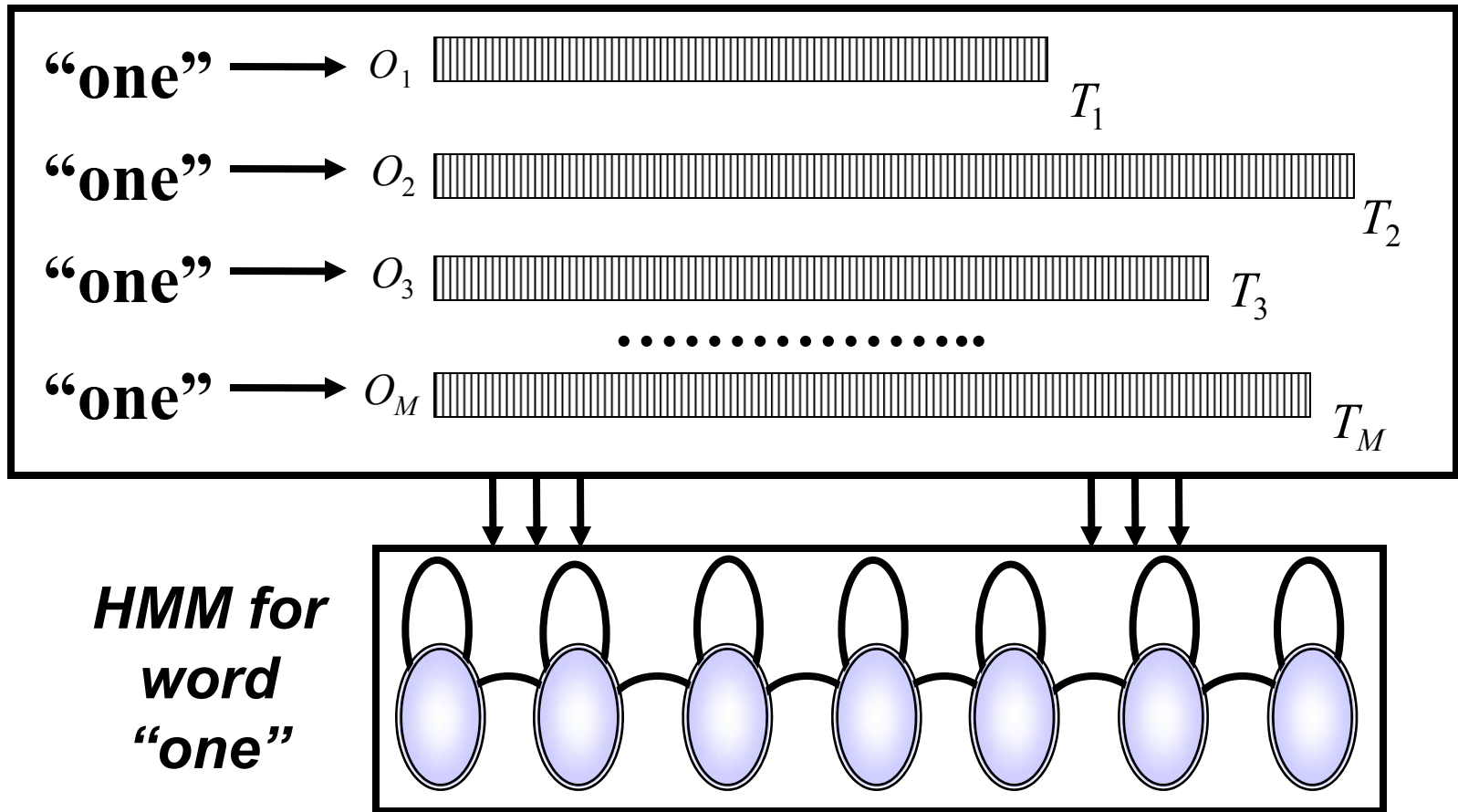
## ■ Whole-Word Model

- Collect many examples of word spoken in isolation
- Assign number of HMM states based on word duration
- Estimate HMM model parameters using iterative Forward-Backward algorithm

## ■ Subword-Unit Model

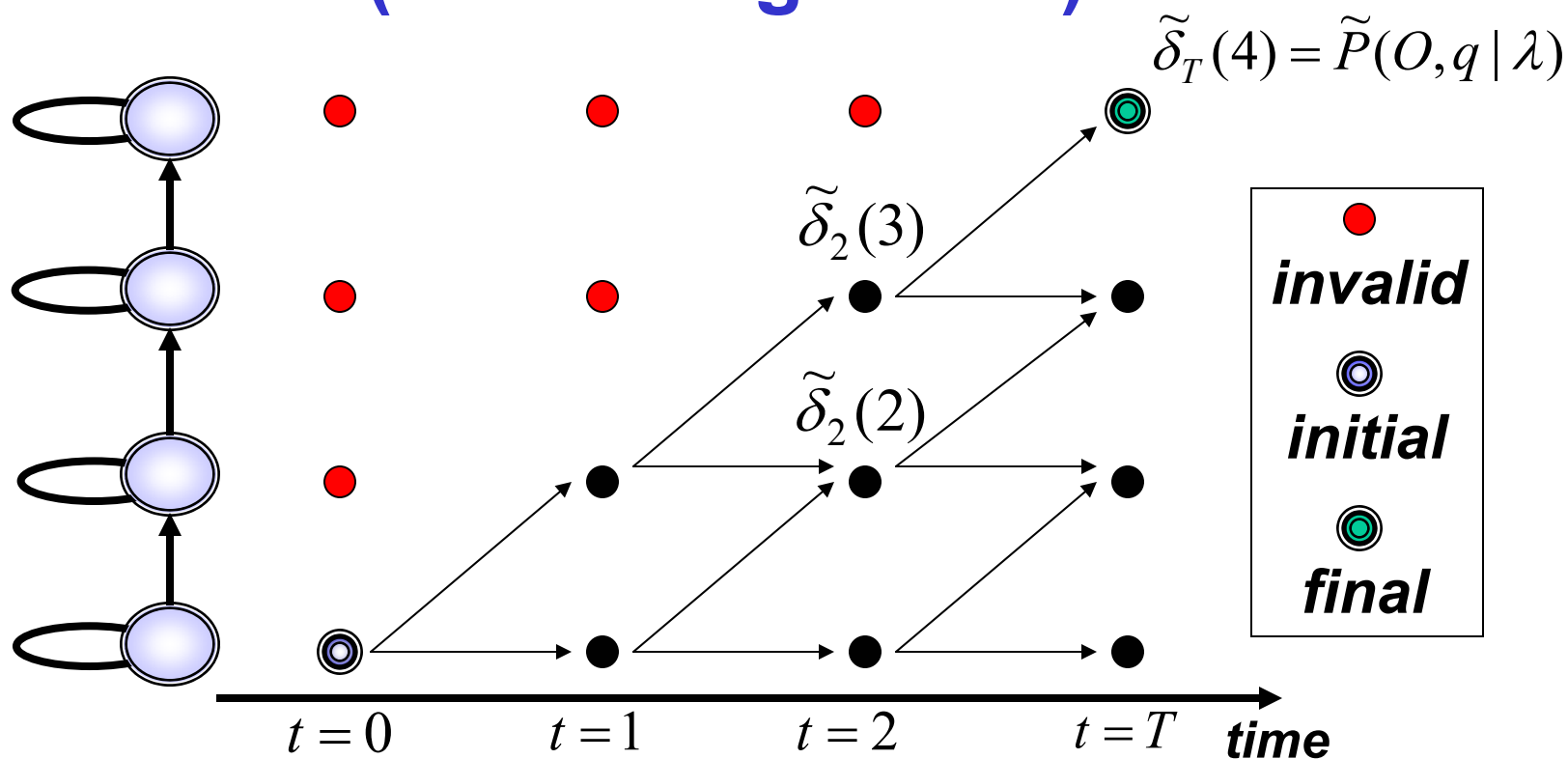
- Collect “large” corpus of speech and estimate phonetic-unit HMMs (e.g., decision-tree state clustered triphones)
- Construct word-level HMM from phoneme-level HMMs
- More general than “whole-word” approach

# Whole-Word HMM



T-61.184

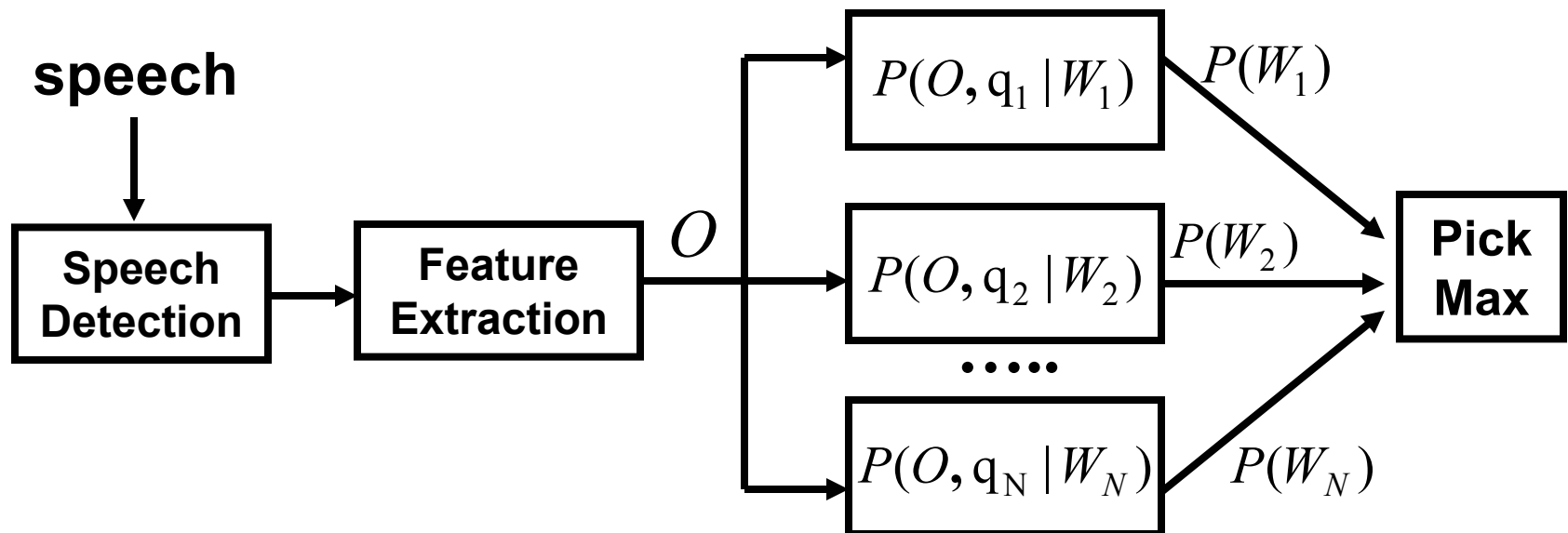
# Computing Log-Probability of Model (Viterbi Algorithm)



$$\tilde{\delta}_t(j) = \max_{1 \leq i \leq N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}] + \tilde{b}_j(\mathbf{o}_t)$$

T-61.184

# Isolated Word Recognition



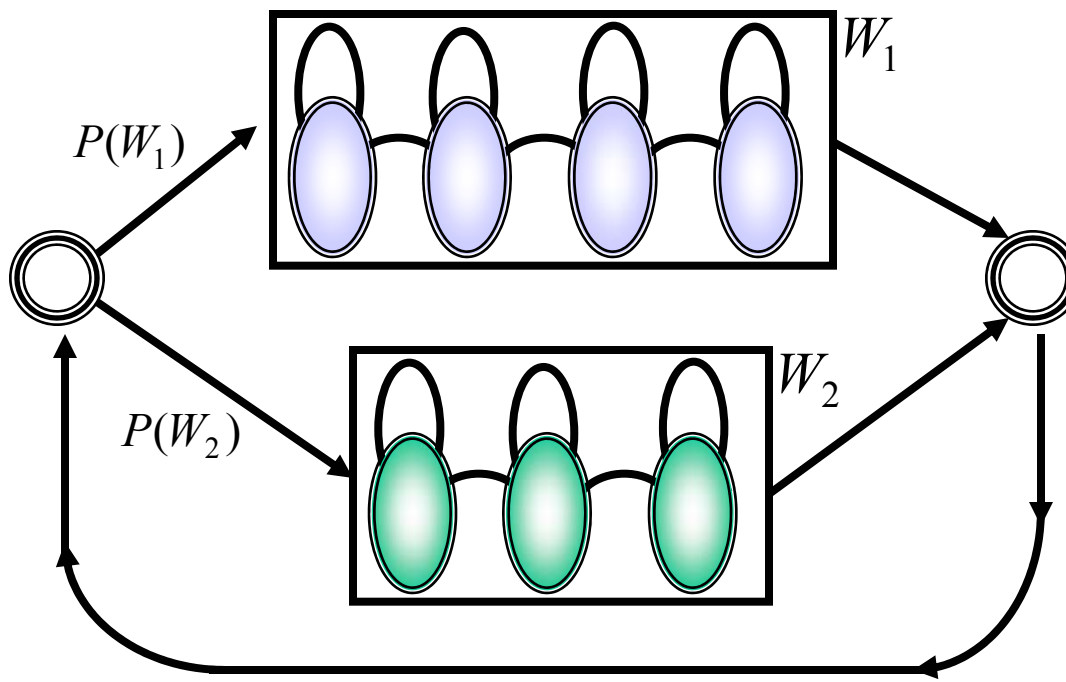
- $P(O|W)$  computed using Viterbi algorithm rather than Forward-Algorithm.
- Viterbi provides probability path represented by most-likely state sequence. Simplifies our recognizer

# Connected-Word (Continuous) Speech Recognition

- Utterance boundaries are unknown
- Number of words spoken in audio is unknown
- Exact position of word-boundaries are often unclear and difficult to determine
- Can not exhaustively search for all possibilities (M= num words, V=length of utterance  $\rightarrow M^V$  possible word sequences).

# Simple Connected-Word Example

- Consider this hypothetical network consisting of 2 words,



T-61.184

# Connected-Word Log-Viterbi Search

- Remember at each node, we must compute,

$$\tilde{\delta}_t(j) = \max_{1 \leq i \leq N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij} + \tilde{\beta}_{ij}] + \tilde{b}_j(\mathbf{o}_t)$$

- Where  $\beta_{ij}$  is the (log) language model score,

$$\tilde{\beta}_{ij} = \left\{ \begin{array}{ll} s\tilde{P}(W_k) + p & : \text{if "i" is the last state of any word} \\ & \text{"j" is the initial state of kth word} \\ 0 & : \text{otherwise} \end{array} \right\}$$

- Recall “s” is the grammar-scale factor and “p” is a log-scale word transition penalty

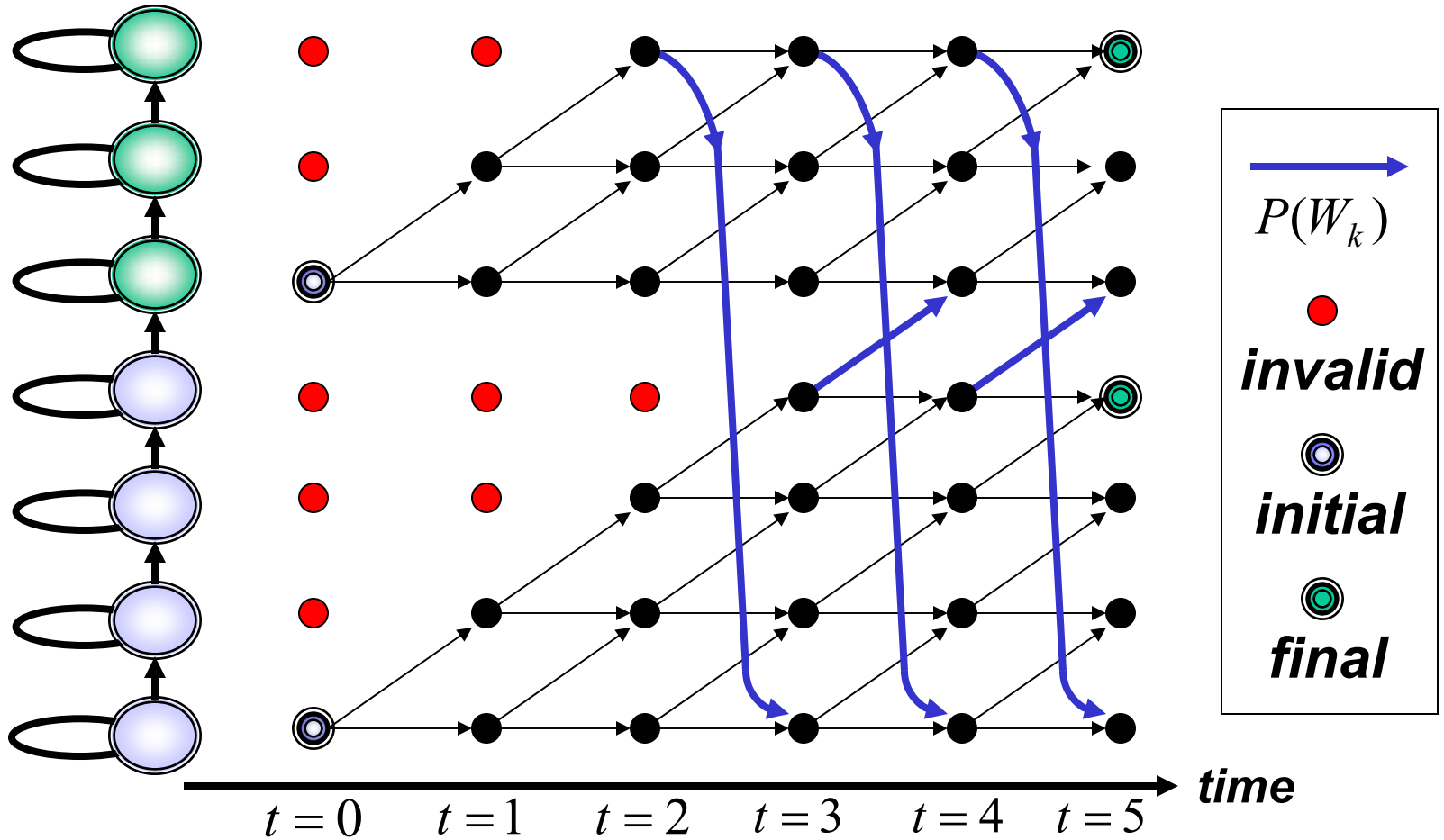
# Connected-Word Log-Viterbi Search

- Remember at each node, we must also compute,

$$\psi_t(j) = \arg \max_{1 \leq i \leq N} \left[ \tilde{\delta}_{t-1}(i) + \tilde{a}_{ij} + \tilde{\beta}_{ij} \right]$$

- This allows us to “back-trace” to discover the most-probable state-sequence.
- Words and word-boundaries are found during “back-trace”. Going backwards we look for state transitions from state 0 into the last state of another word.

# Connected-Word Viterbi Search



T-61.184

# Viterbi with Beam-Pruning

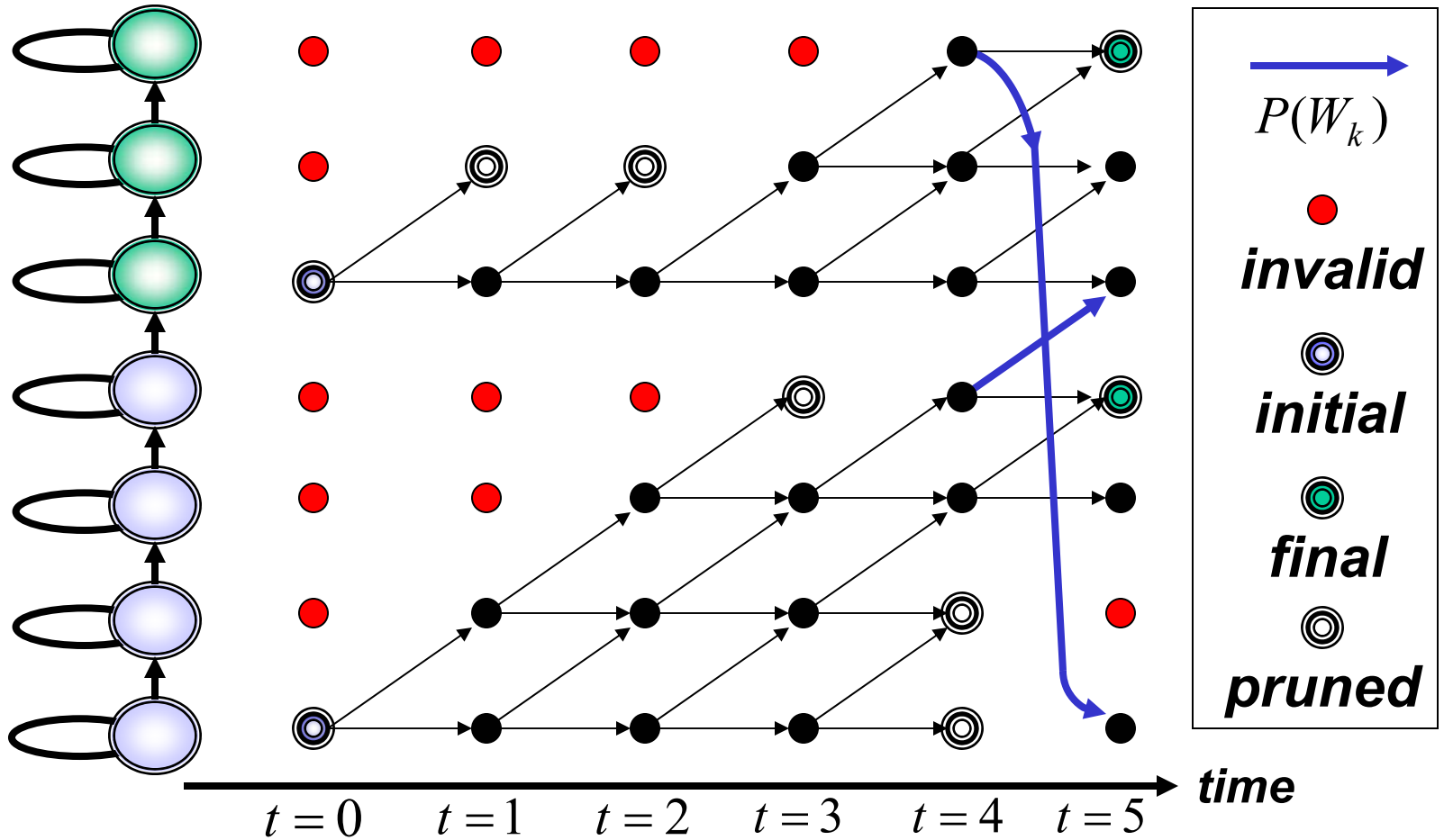
- **Idea : Prune away low-scoring paths,**
  - At each time,  $t$ , determine the log-probability of the absolute best Viterbi path,

$$\tilde{\delta}_t^{MAX} = \max_{1 \leq i \leq N} [\tilde{\delta}_t(i)]$$

- Prune away paths which fall below a pre-determined “beam” (BW) from the maximum probable path.  
“Deactivate” state “ $j$ ” if,

$$\tilde{\delta}_t(j) < \tilde{\delta}_t^{MAX} - BW$$

# Hypothetical Beam Search



T-61.184

## Issues with the “Trellis” Search

- **Important note : language model is applied at the point that we transition into the word.**
- **As the number of words increases, so do the number of states and interconnections**
  - “Beam-Search” Improves efficiency
  - Still difficult to evaluate the entire search space
- **Not easy to incorporate word histories (e.g., n-gram models) into such a framework**
- **Not easy to account for between-word acoustics**

# The Token Passing Model

- **Proposed by Young et al. (1989)**
- **Provides a conceptually appealing framework for connected word speech recognition search**
- **Allows for arbitrarily complex networks to be constructed and searched**
- **Efficiently allows n-gram language models to be applied during search**

# Token Passing Approach

- Let's assume each HMM state can hold (multiple) movable “token(s)”
- Think of a token as an object that can move from state-to-state in our network
- For now, let's assume each token carries with it the (log-scale) Viterbi path cost:  $\mathcal{S}$

# Token Passing Idea

- At each time, “ $t$ ”, we examine the tokens that are assigned to nodes in the network
- Tokens are propagated to reachable network positions at time  $t+1$ ,
  - Make a copy of the token
  - Adjust path score to account for HMM transition and observation probability
- Tokens are merged based on Viterbi algorithm,
  - Select token with best-path by picking the one with the maximum score
  - Discard all other “competing” tokens

# Token Passing Algorithm

## ■ Initialization ( $t=0$ )

- Initialize each initial state to hold a token with,  $s = 0$
- All other states initialized with a token of score,  $s = -\infty$

## ■ Algorithm ( $t>0$ ):

- Propagate tokens to all possible “next” states
- Prune tokens whose path scores fall below a search beam

## ■ Termination ( $t=T$ )

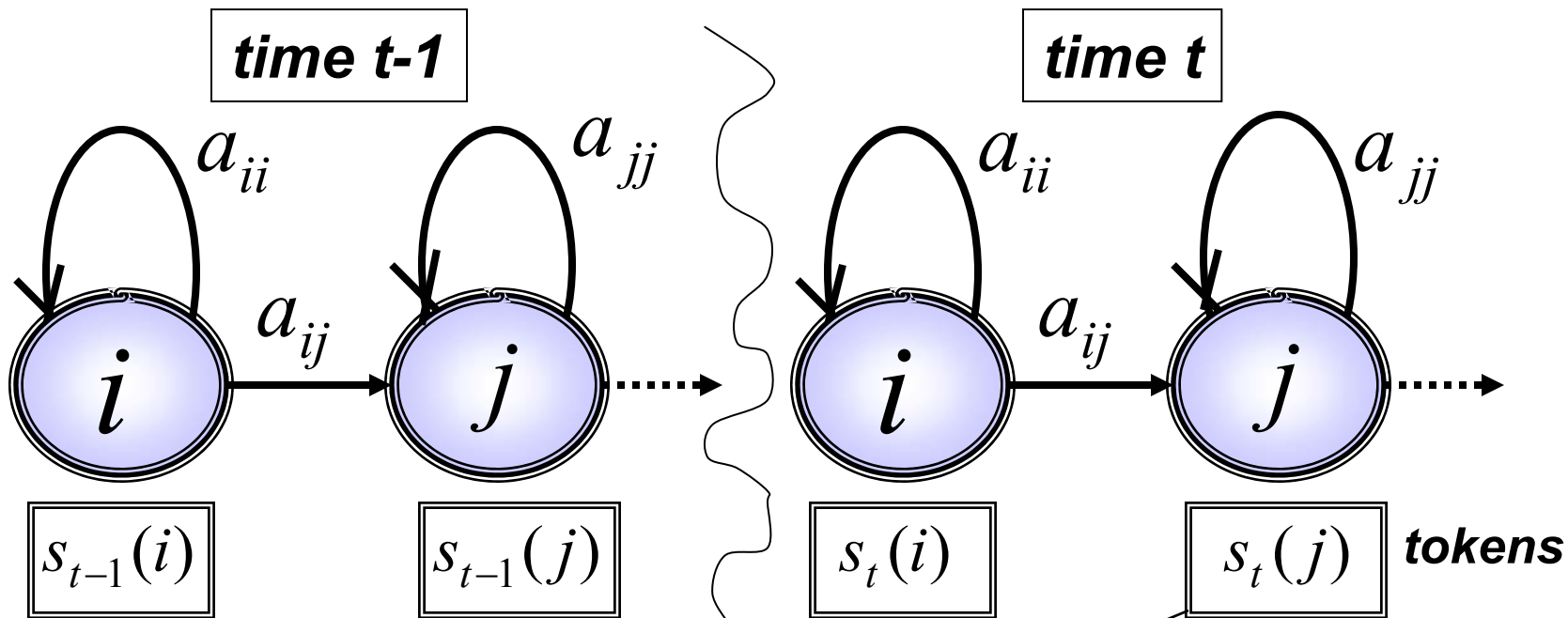
- Examine the tokens in all possible final states
- Find the token with the largest Viterbi path score
- This is the probability of the most likely state alignment

# Token Propagation (Without Language Model)

```
for t := 1 to T
  foreach state i do
    Pass token copy in state i to all connecting states j,
    increment,
    
$$s = s + \tilde{a}_{ij} + \tilde{b}_j(\mathbf{o}_t)$$

  end
  foreach state i do
    Find the token in state i with the largest s and discard
    the rest of the tokens in state i. (Viterbi Search)
  end
end
```

# Token Propagation Example

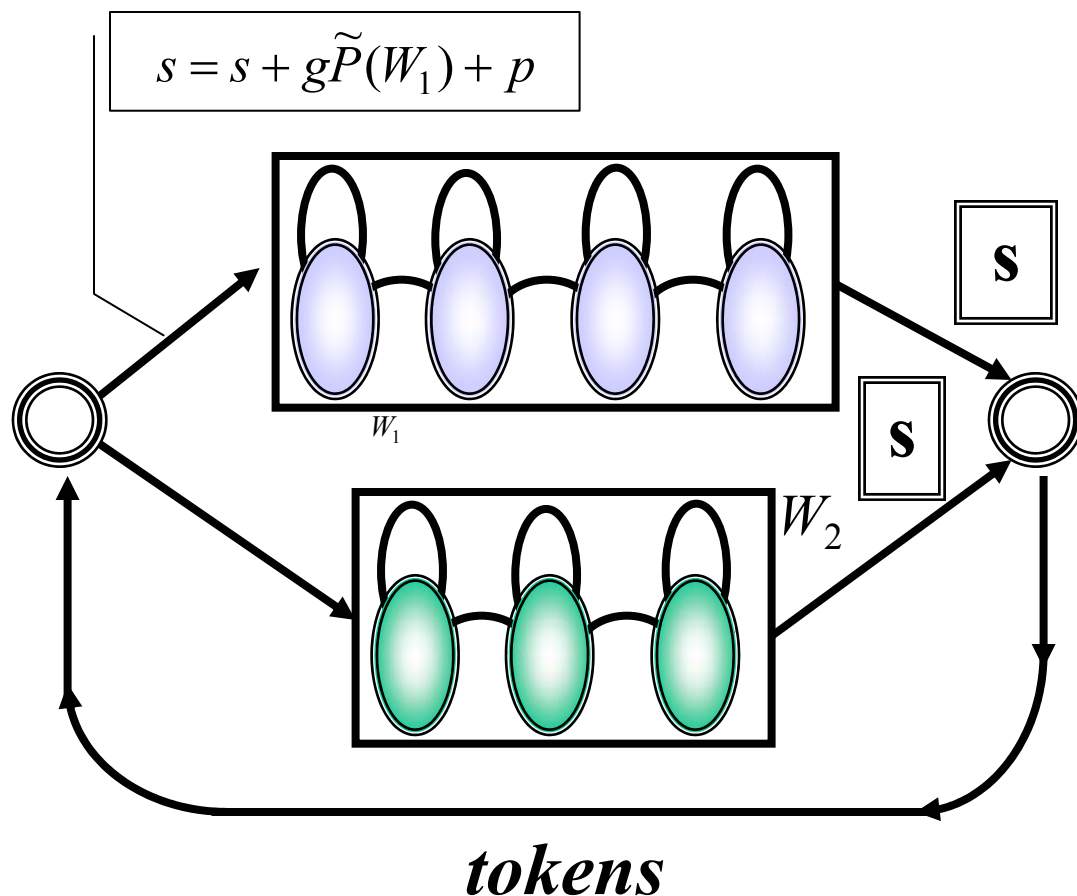


$$s_t(j) = \max \left\{ \underbrace{s_{t-1}(i) + \tilde{a}_{ij} + \tilde{b}_j(o_t)}_{\text{forward transition token}}, \underbrace{s_{t-1}(j) + \tilde{a}_{jj} + \tilde{b}_j(o_t)}_{\text{self-loop transition token}} \right\}$$

# Token Passing Model for Connected Word Recognition

- **Individual word models are connected together into a looped composite model**
  - Can transition from final state of word “i” to initial state of word “j”.
- **Path scores are maintained by tokens**
  - Language model score added to path when transitioning between words.
- **Path through network also maintained by tokens**
  - Allows us to recover best word sequence

# Connected Word Example (with Token Passing)



- Tokens emitted from last state of each word propagate to initial state of each word.
- Language model score added to path score upon word-entry.

T-61.184

# Maintaining Path Information

- The previous example assumes a unigram language model. Knowledge of the previous word is not maintained by the tokens.
  - For connected word recognition, we don't care much about the underlying state sequence within each word model
  - We care about transitions between words and when they occur
- Must augment token structure with a path identifier & path score

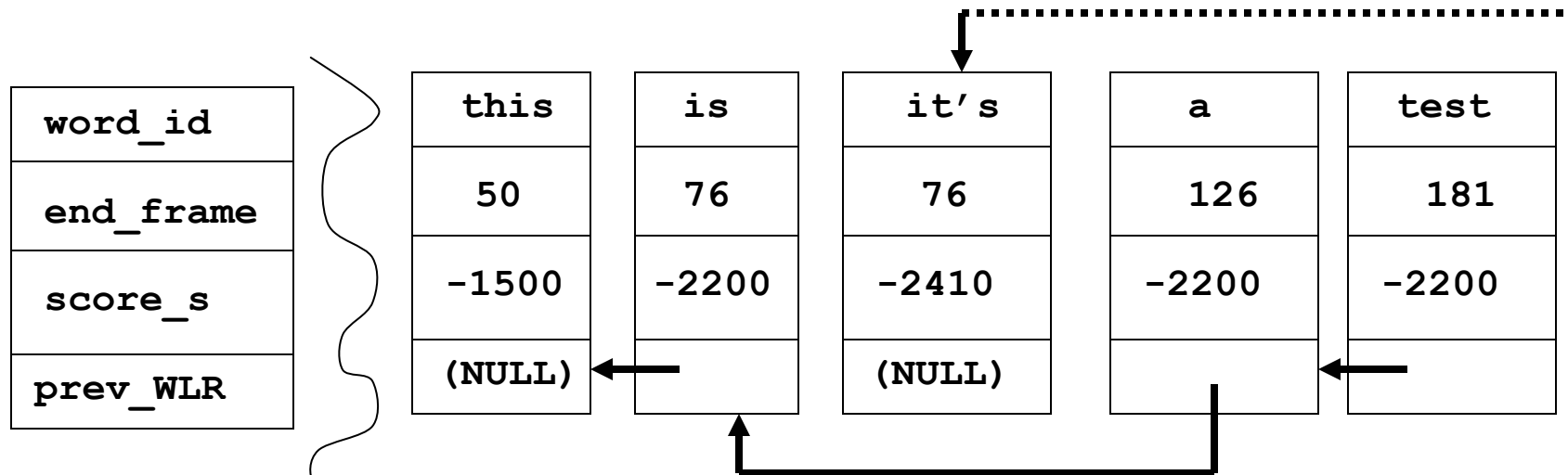
# Word-Link Record

- Path Identifier points to a record (data structure) containing word-boundary information
- Word-Link Record (WLR): data structure created each time a token exits a word. Contains,
  - ❑ Word Identifier (e.g., “hello”)
  - ❑ Word End Frame (e.g., “time=t”)
  - ❑ Viterbi Path Score at time t.
  - ❑ Pointer to previous WLR

<code>word_id</code>
<code>end_frame</code>
<code>path_score_s</code>
<code>previous_WLR</code>

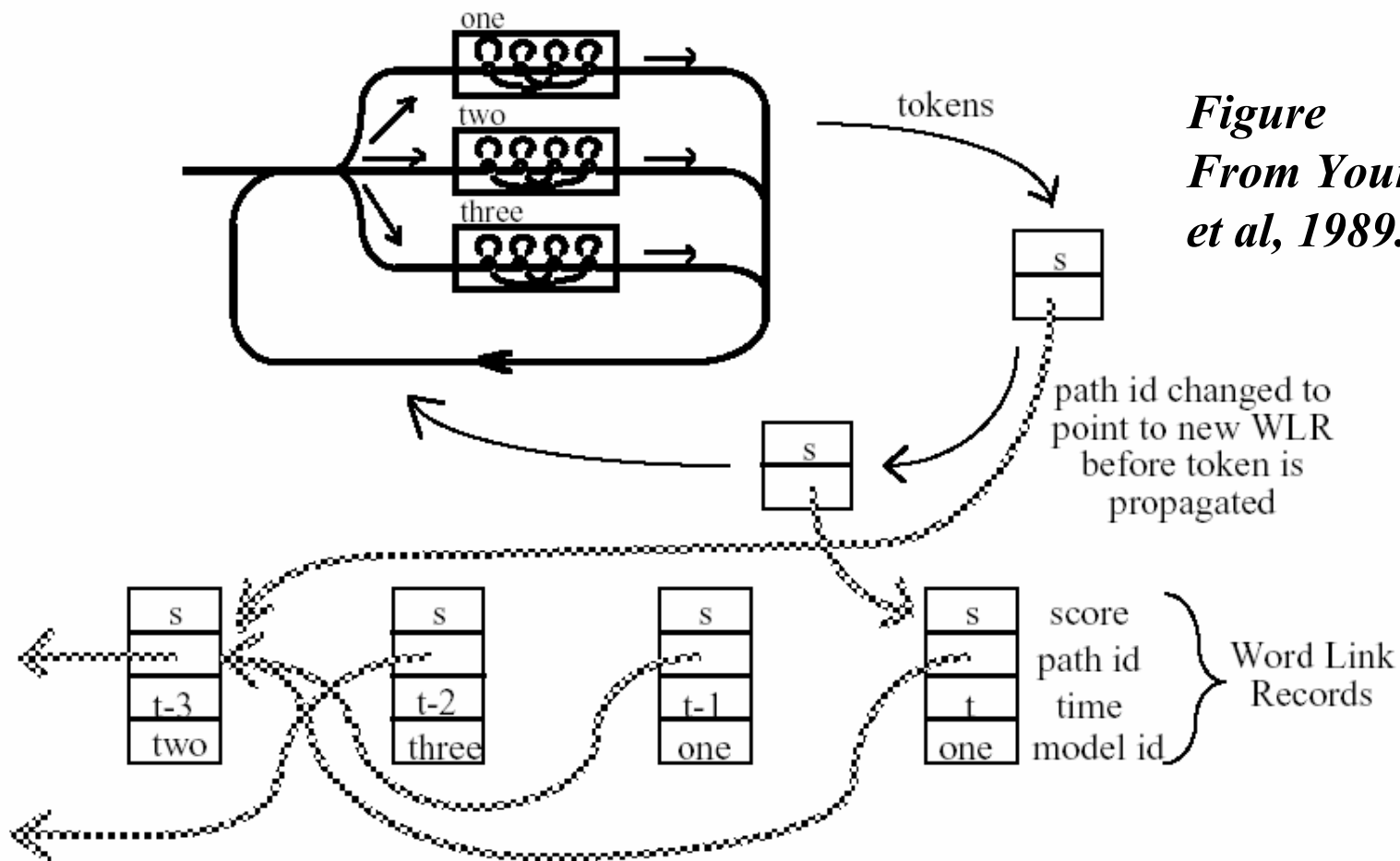
# Word-Link Record

- WLR's link together to provide search outcome:



*“is” begins at frame 50 (.5 sec), ends at frame 76 (0.76 sec). The total path cost for the word is -700. “This” begins at frame 0 and ends at frame 50.*

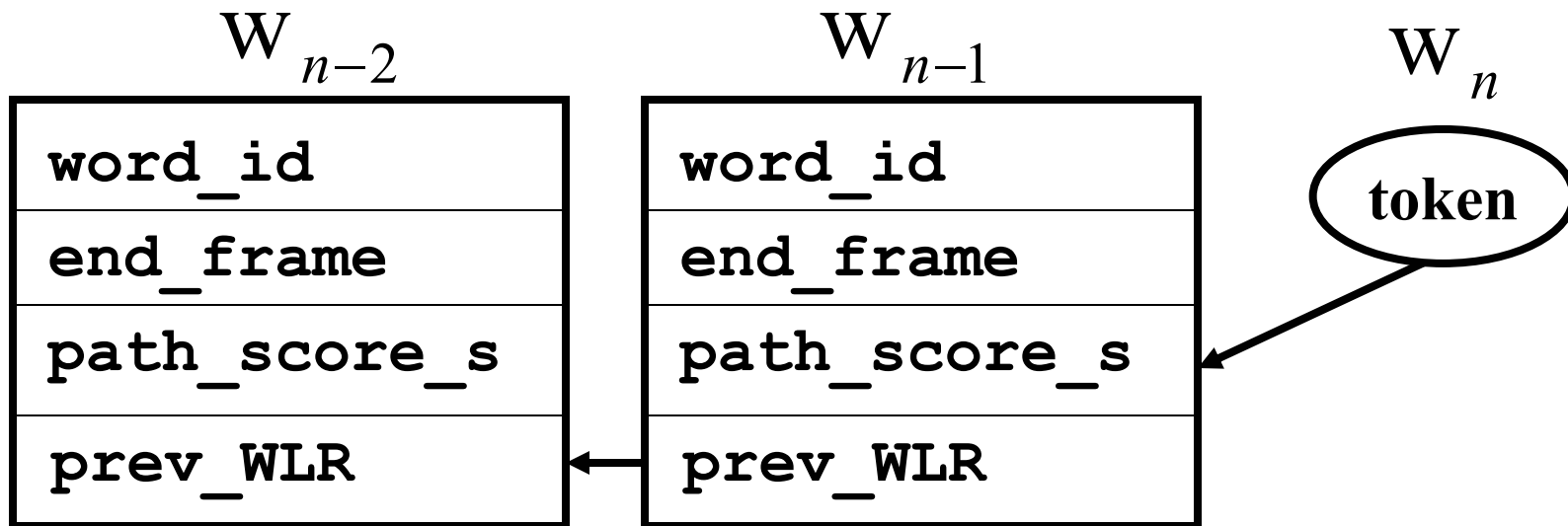
# Illustration of WLR Generation



T-61.184

## WLRs as a Word-History Provider

- Each propagating token contains a pointer to a word link record
- Tracing back provides word-history



T-61.184

## Incorporating N-gram Language Models During Token Passing Search

- When a token exits a word and is about to propagate into a new word, we can augment the token's path cost with the LM score.
- Upon exit, each token contains pointer to a word link record. Can obtain previous word(s) from WLR
- Therefore, update the path with,

$$s = s + g\tilde{P}(W_n | W_{n-1}, W_{n-2}) + p$$