T-61.184
Automatic Speech Recognition: From Theory to Practice

http://www.cis.hut.fi/Opinnot/T-61.184/
October 25, 2004

Prof. Bryan Pellom
Department of Computer Science
Center for Spoken Language Research
University of Colorado

pellom@cslr.colorado.edu

References for Today's Material

- S. J. Young, N. H. Russel, J.H.S. Thornton, "Token Passing: a Simple Conceptual Model for Connected Speech Recognition Systems", Technical Report TR-38, Cambridge University Engineering Dept., July 1989.
- X. Huang, A. Acero, H. Hon, Spoken Language Processing, Prentice Hall, 2001 (chapters 12 and 13)
- L.R. Rabiner & B. W. Juang, Fundamentals of Speech Recognition, Prentice-Hall, ISBN 0-13 015157-2, 1993 (see chapters 7 and 8)

Search

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

$$\hat{W} = \arg\max_{W} P(\mathbf{O} \mid W) P(W)$$

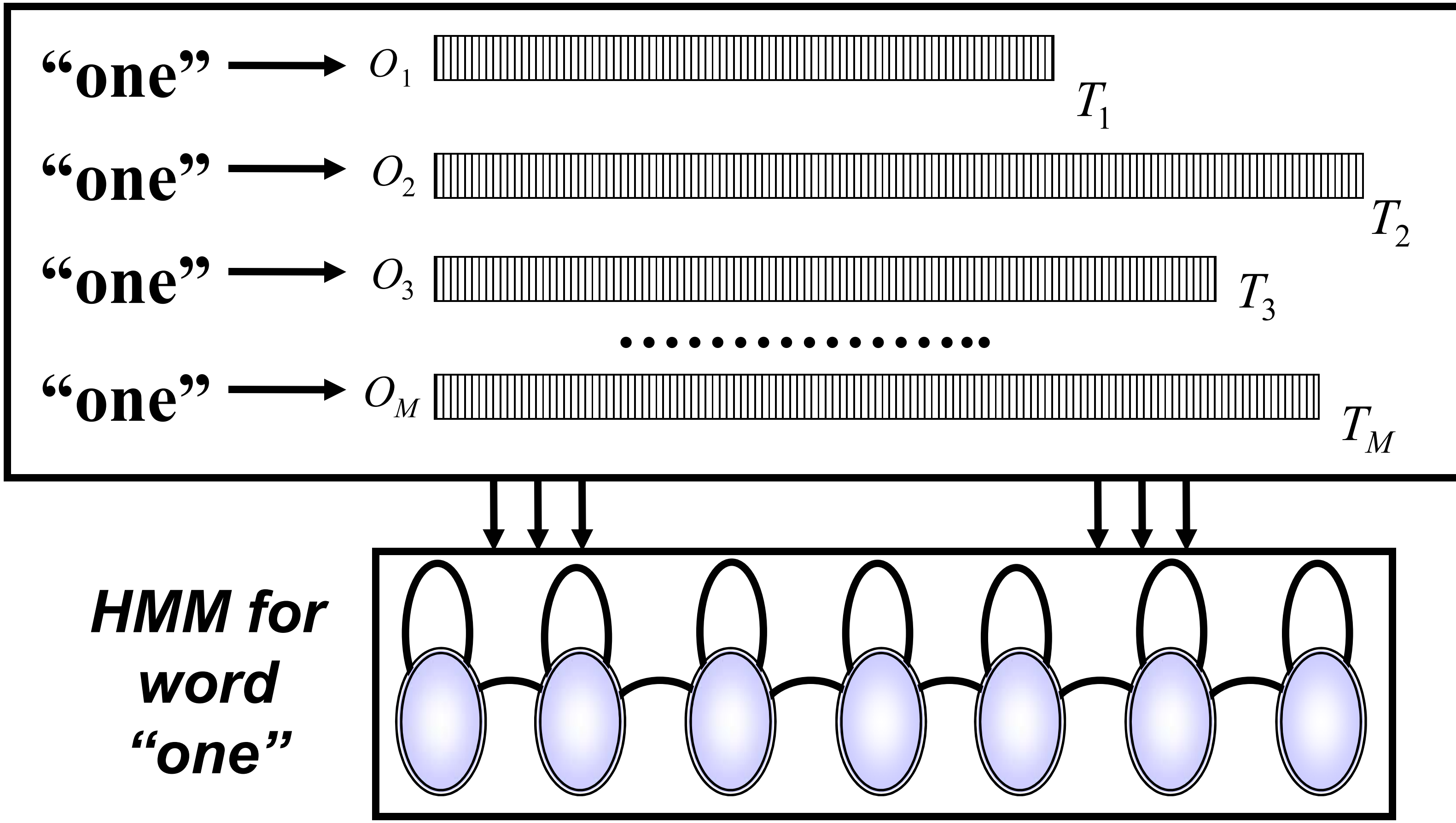
- **Types of input:**
  - Isolated Words
  - Connected Words

Designing an Isolated-Word HMM

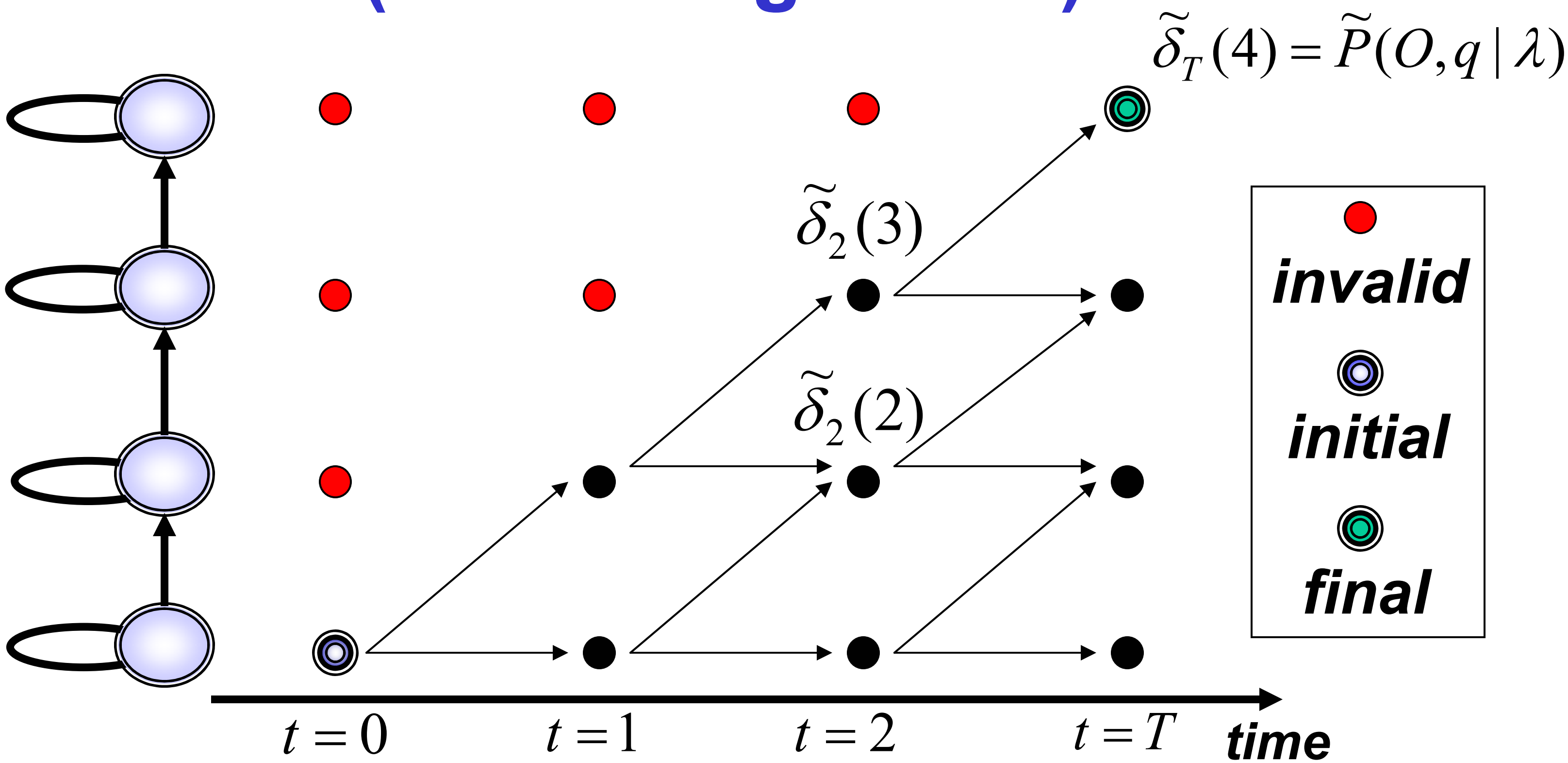
- **Whole-Word Model**
  - ❑ Collect many examples of word spoken in isolation
  - ❑ Assign number of HMM states based on word duration
  - ❑ Estimate HMM model parameters using iterative Forward-Backward algorithm

- **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

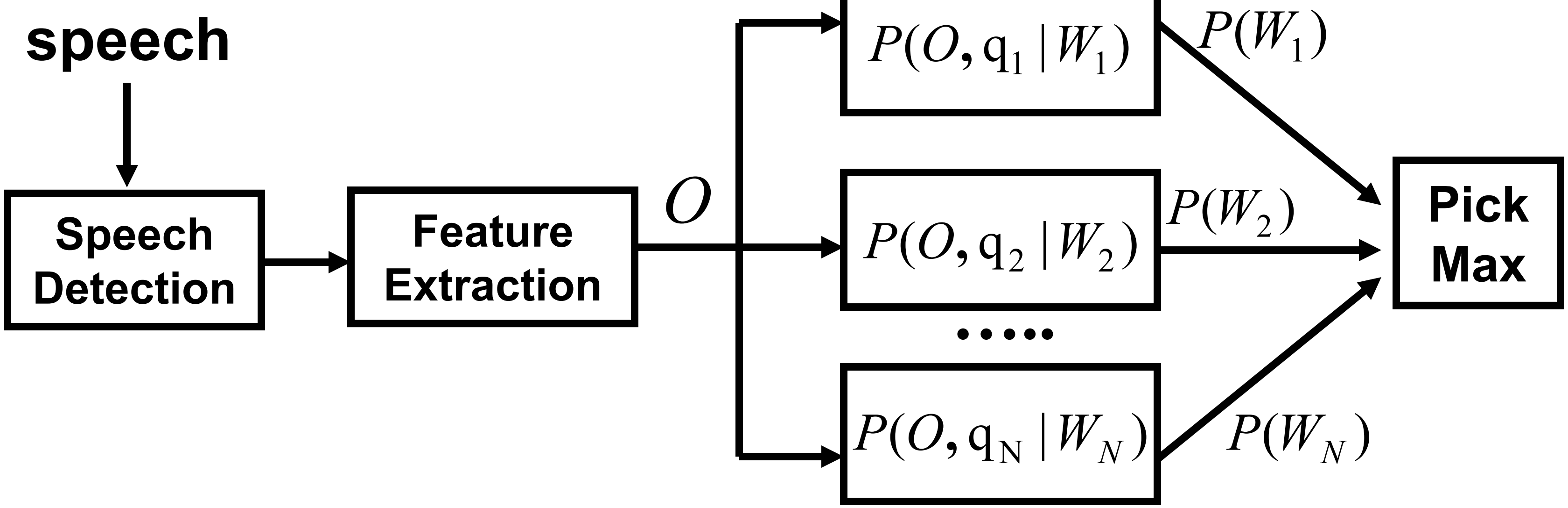


Computing Log-Probability of Model (Viterbi Algorithm)



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

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**

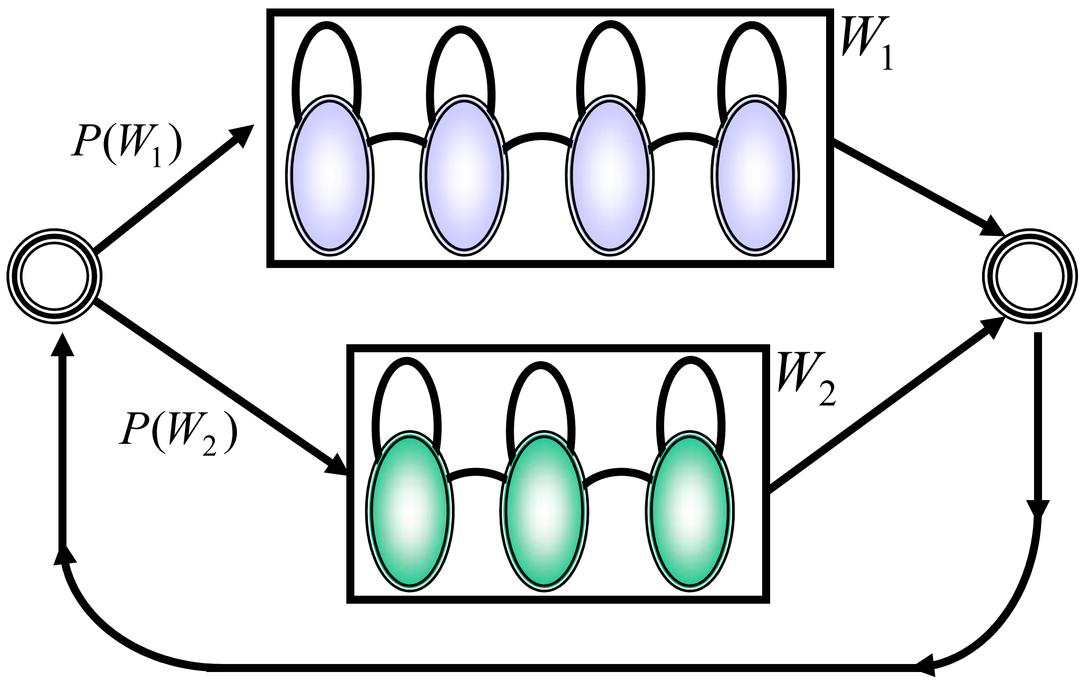
Connected-Word (Continuous) Speech Recognition

- **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 → M^V possible word sequences).**

T-61.184

Simple Connected-Word Example

- **Consider this hypothetical network consisting of 2 words,**



Connected-Word Log-Viterbi Search

- **Remember at each node, we must compute,**

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

- **Where β_{ij} is the (log) language model score,**

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

- **Recall "s" is the grammar-scale factor and "p" is a log-scale word transition penalty**

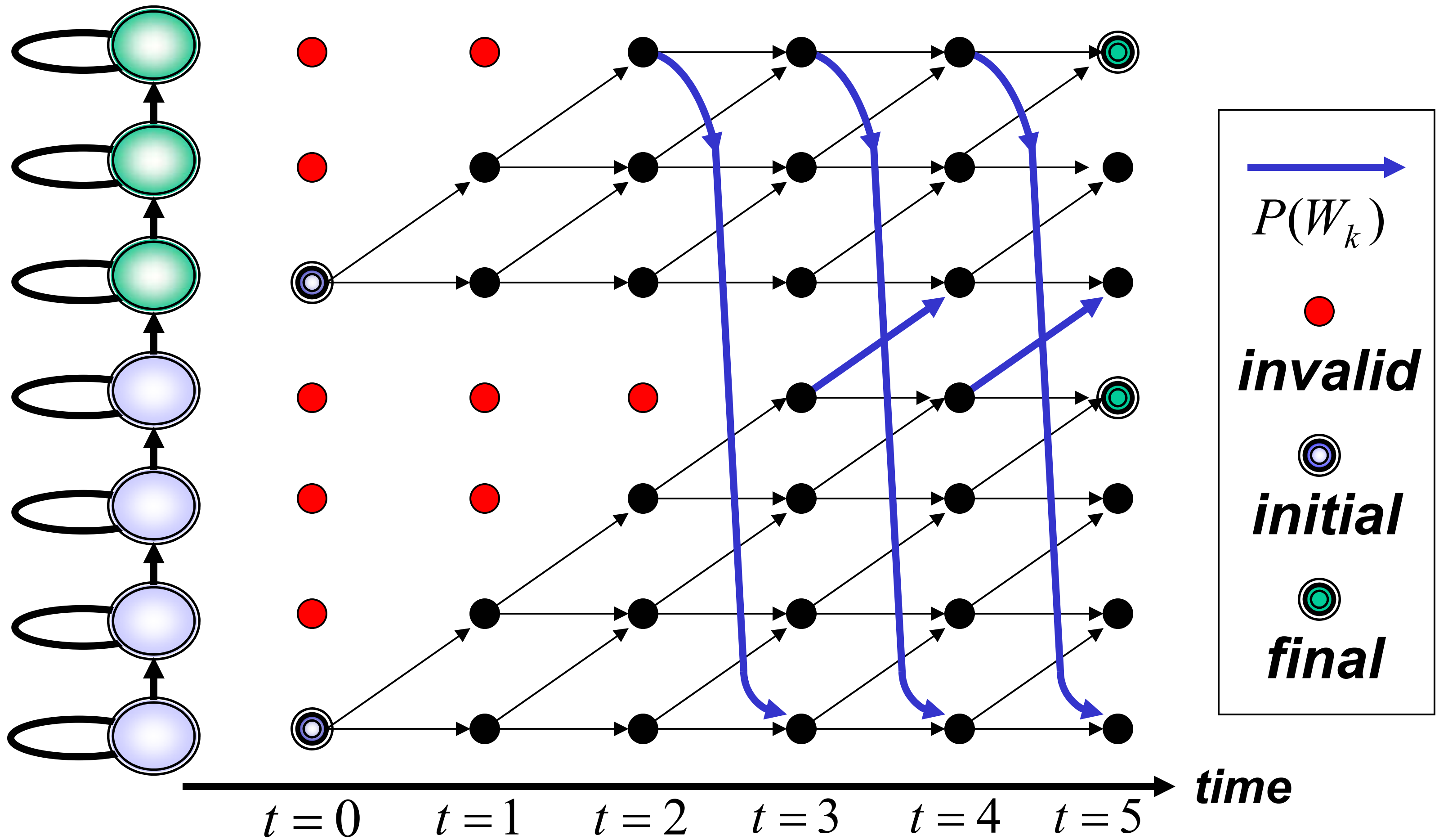
Connected-Word Log-Viterbi Search

- **Remember at each node, we must also compute,**

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

- **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,

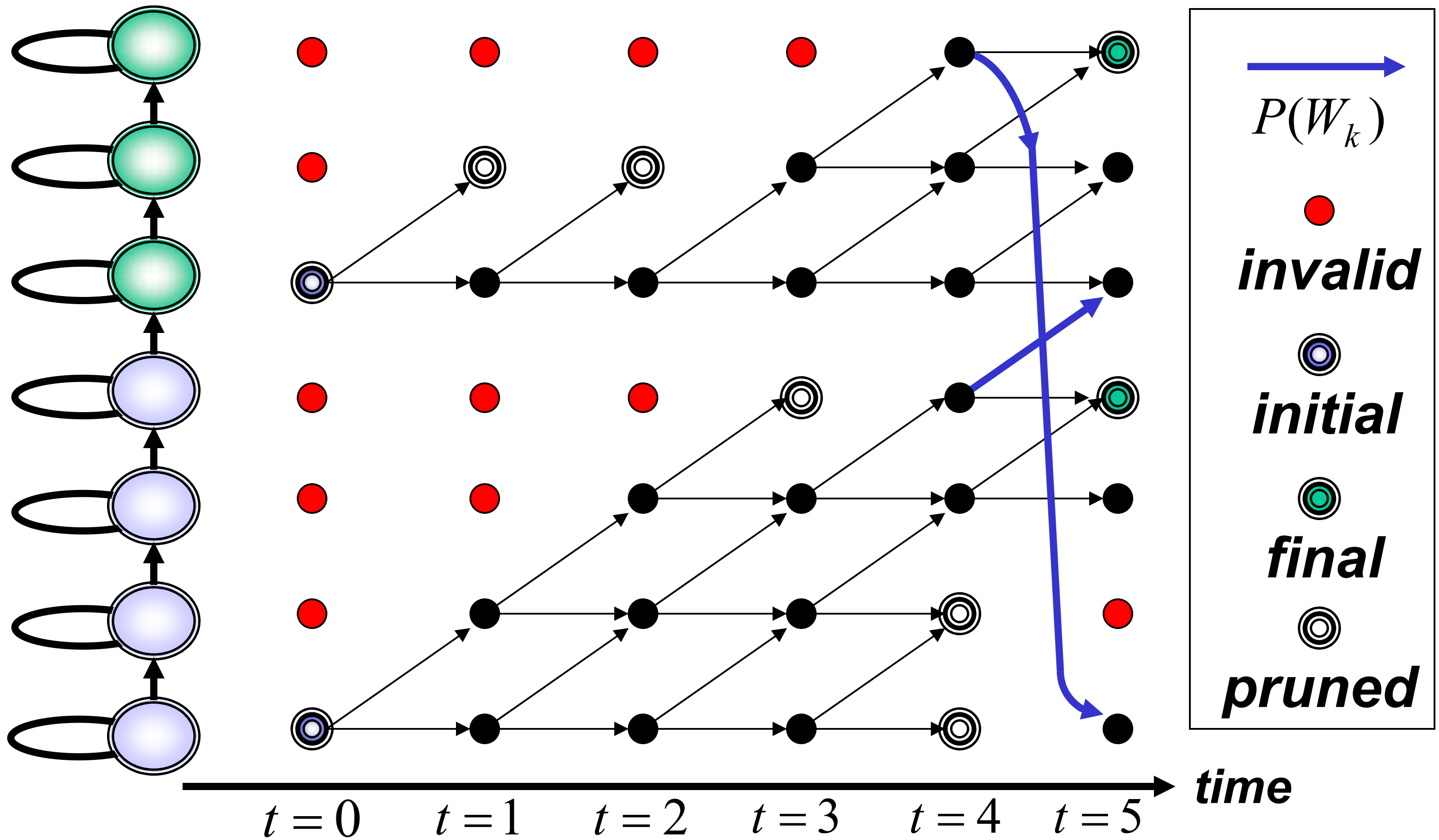
$$\tilde{\delta}_t^{MAX} = \max_{1 \le i \le N} \left[\tilde{\delta}_t(i) \right]$$

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

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

T-61.184

Hypothetical Beam Search



Issues with the "Trellis" Search

- **Important note : language model is applied at the point that we transition <u>into</u> 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 <u>object</u> 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: s

Token Passing Idea

- **At each time, "t", we examine the tokens that are assigned to nodes in the network**
- **Tokens are <u>propagated</u> 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 <u>merged</u> 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)

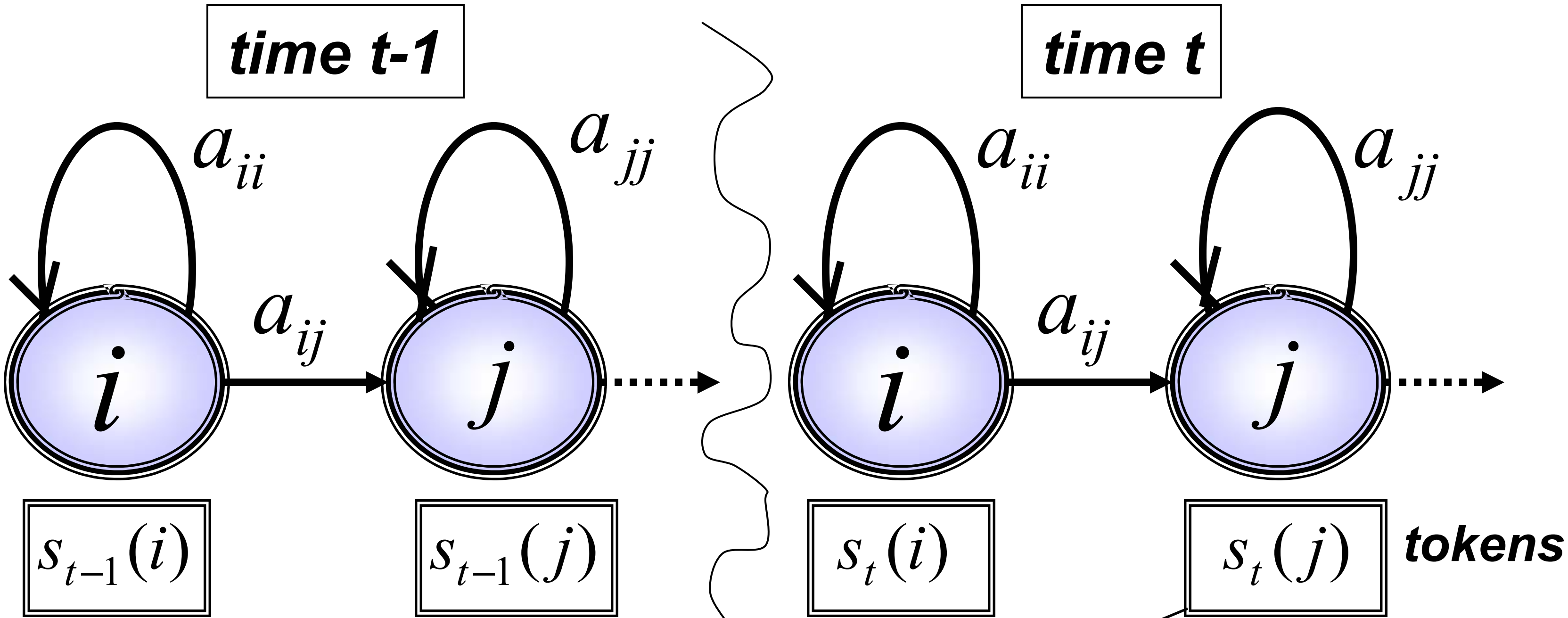
```
for t := 1 to T
    foreach state i do
```
<u>Pass token copy</u> in state i to all connecting states j, increment,

$$s = s + \widetilde{a}_{ij} + \widetilde{b}_j(\mathbf{o}_t)$$

```
    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

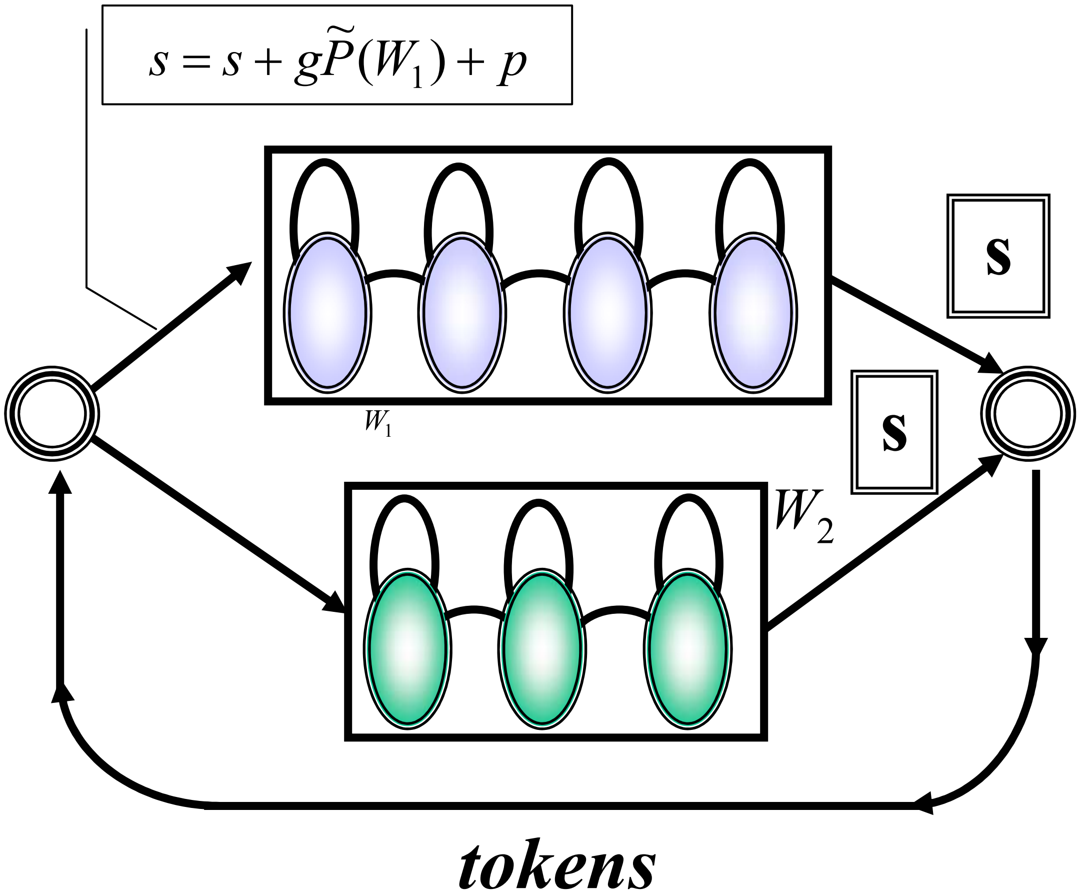


$$s_t(j) = \max\left\{ \underbrace{s_{t-1}(i) + \tilde{a}_{ij} + \tilde{b}_j(o_t)}_{\text{forward transition token}} \ , \ \underbrace{s_{t-1}(j) + \tilde{a}_{jj} + \tilde{b}_j(o_t)}_{\text{self-loop transition token}} \right\}$$

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.**

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 <u>path identifier</u> & <u>path score</u>

Word-Link Record

- **<u>Path Identifier</u> points to a record (data structure) containing word-boundary information**

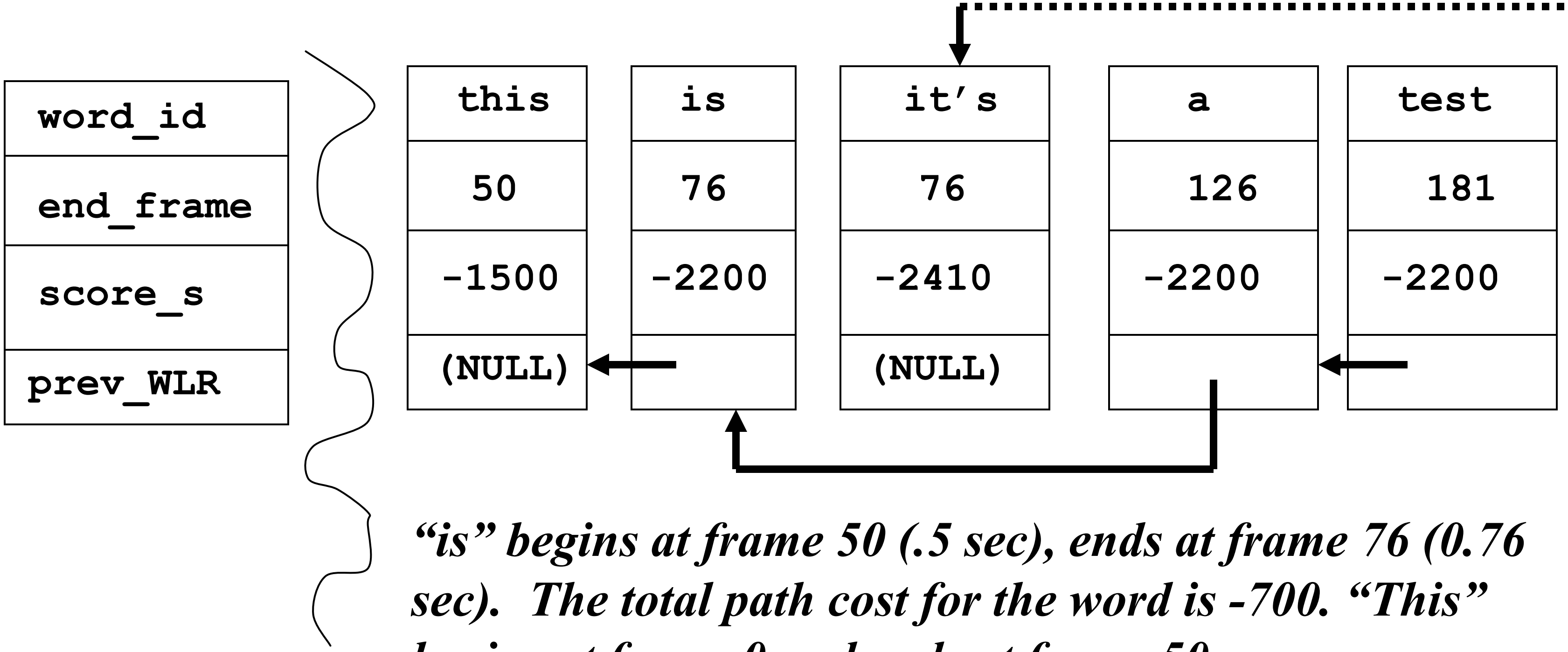
- **<u>Word-Link Record (WLR)</u>: data structure created each time a token exits a word. Contains,**

  - ❑ Word Identifier (e.g., “hello”)
  - ❑ Word End Frame (e.g., “time=t”)
  - ❑ Viterbi Path Score at time t.
  - ❑ Pointer to previous WLR

`word_id`
`end_frame`
`path_score_s`
`previous_WLR`

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

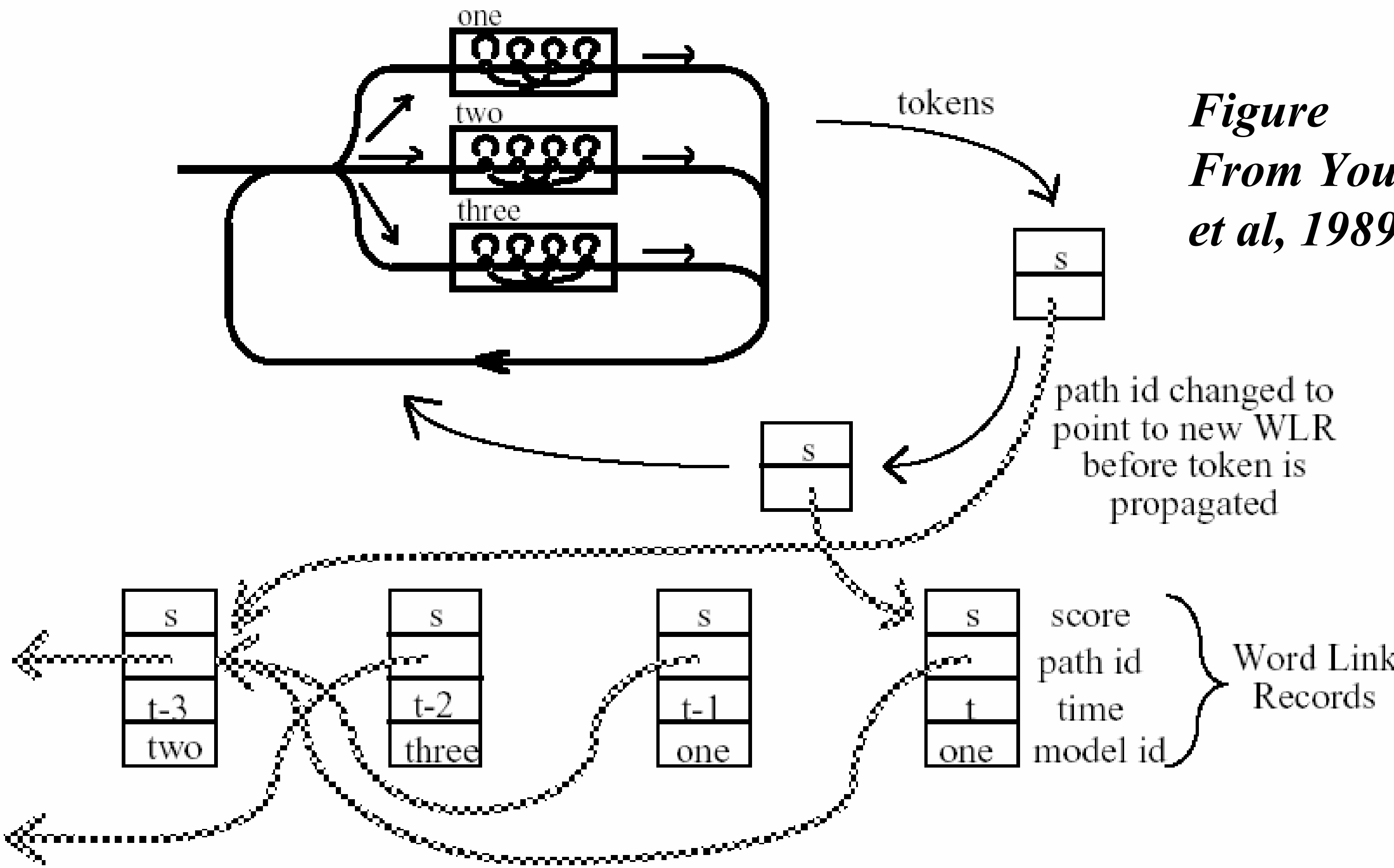
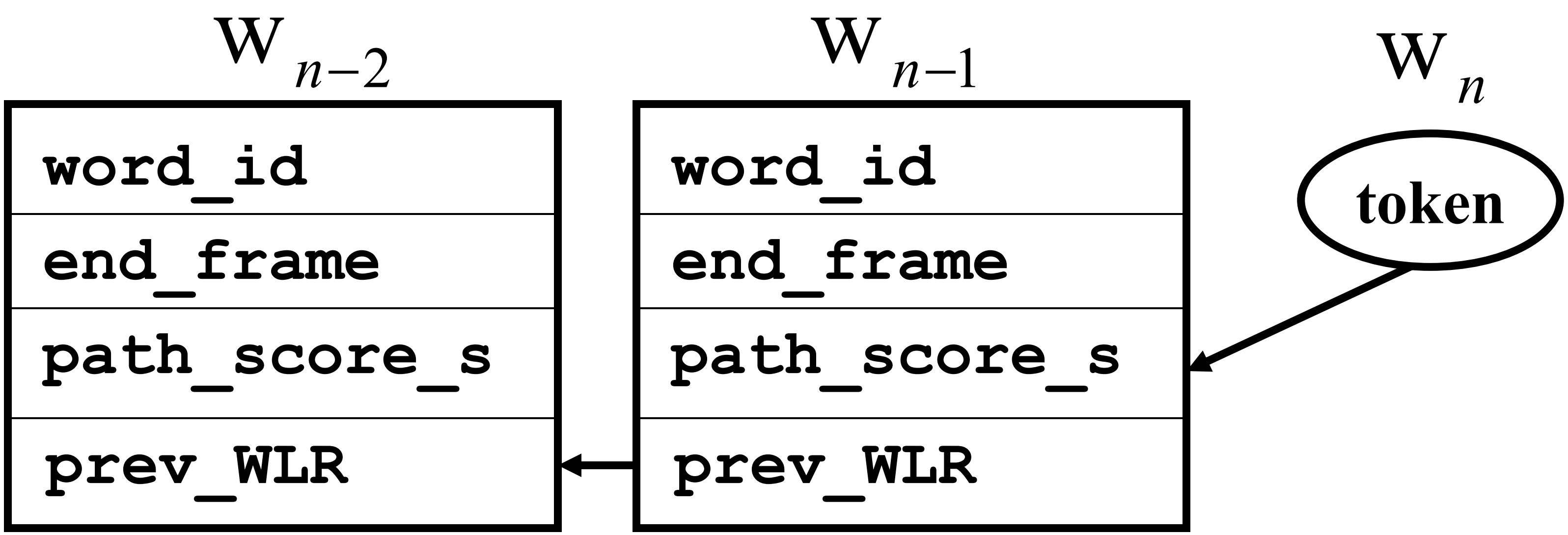


Figure From Young et al, 1989.

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



Incorporating N-gram Language Models During Token Passing Search

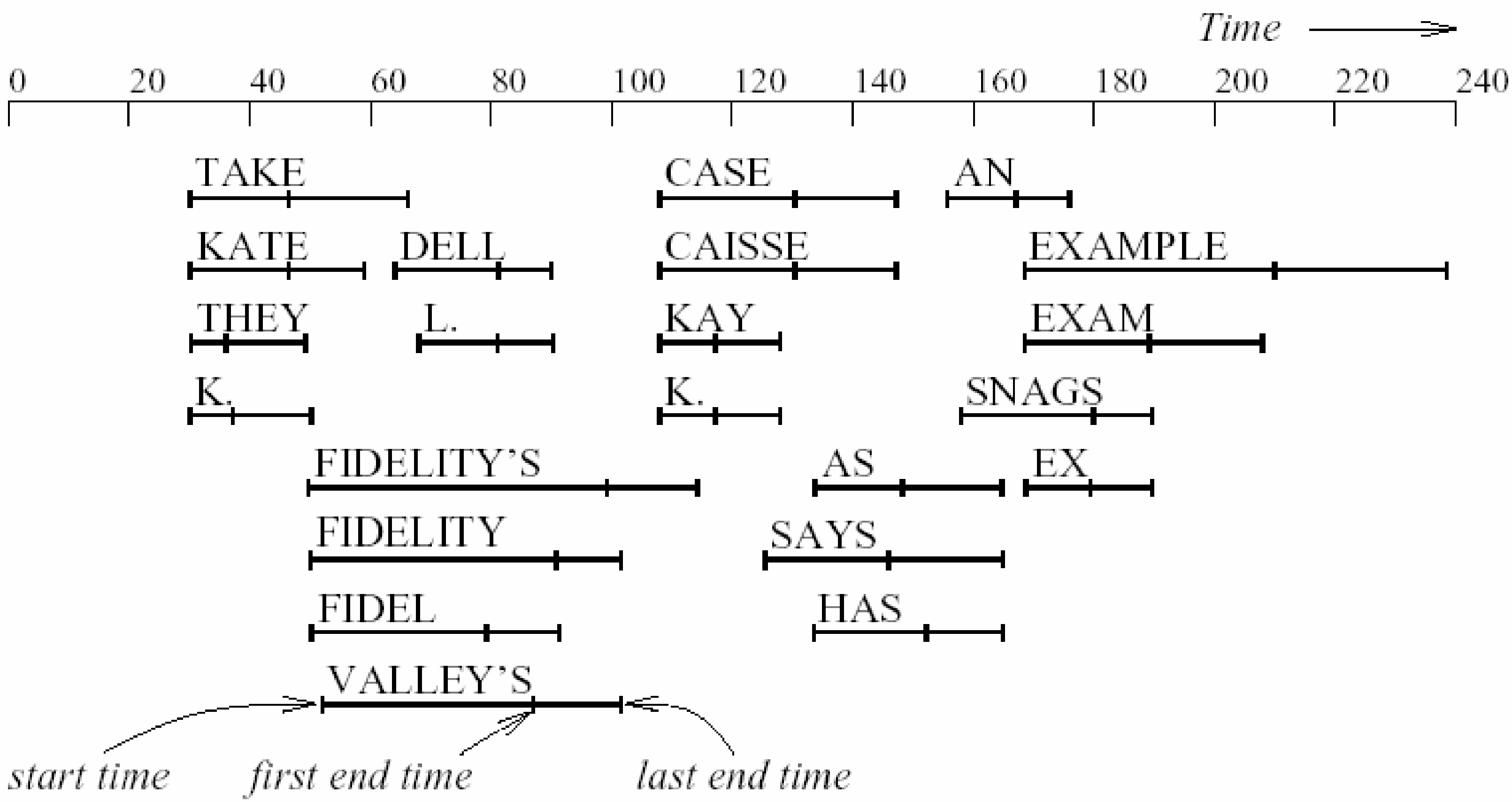
- When a token exits a word and is about to propagate into a new word, we can augment the token's path cost with the LM score.
- Upon exit, each token contains pointer to a word link record. Can obtain previous word(s) from WLR
- Therefore, update the path with,

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

Word-Link Records & Lattices

- Word Link Records encode the possible word sequences seen during search
- Words can overlap in time
- Words can have different path scores
- Can generate a “lattice” of word confusions from WLR’s.

Lattice Representation



"take fidelity's case as an example"

Recovering the "Best" Word String

- **Scan through Word-Link Records created at final time "T" and find WLR corresponding to word with best path score (s).**
- **Follow link from current WRL to previous WRL. Extract word identity.   Repeat until current WRL does not point to any previous WRL (null).**
- **Reverse decoded word sequence**
  - Word begin/end times determined from WRL sequence
  - Word score determined by taking between path scores

Token Passing Search Issues

- **How to correctly apply language model which may depend on *multiple* previous words?**
- **How to prune away tokens which represent unpromising paths?**
- **How can we implement cross-word acoustic models into the token passing search?**

Language Modeling & Token Passing

- Tokens entering a particular state are merged by keeping the token with maximum partial path score s (Viterbi path assumption)
- When N-gram language models are used, must consider merging tokens if they have the <u>same</u> word histories
- <u>Trigram LM</u>: given a token in a state of word n, pick max over all competing tokens which share same 2 previous words

Implications

- Tokens represent partial paths which have unique word histories.
- Tokens must be propagated and merged carefully
- Each HMM state may have multiple tokens assigned to it at any given time.
- Each assigned token should represent a unique word-history

(Practically Speaking)

- **For a trigram language model,**
  - Unpruned tokens with unique 2-word history are merged
  - Results in many tokens assigned to each network state
  - Makes propagation of tokens very costly (slow decoding)
- **Bigram Approximation**
  - merge tokens with unique 1-word previous history
  - Negligible loss in accuracy for English
- **Implemented in CSLR SONIC, CMU Sphinx-II, other recognizers as well.**

Pruning & Efficiency

- The number of tokens will increase in the network as frame count (t) increases.
- Maintaining tokens with unique word histories makes problem worse
- Beam pruning is a useful mechanism for controlling the number of tokens (partial paths) being explored at any given time

Beam Pruning for Token Passing

- Find token with maximum partial path log-score, "s" at time "t".
- Prune away tokens that have score less than a threshold, e.g.,

$$\text{prune} \quad \text{if} \quad s < (s_{\max} - BW)$$

- BW is preset "beam width"
- BW > 0

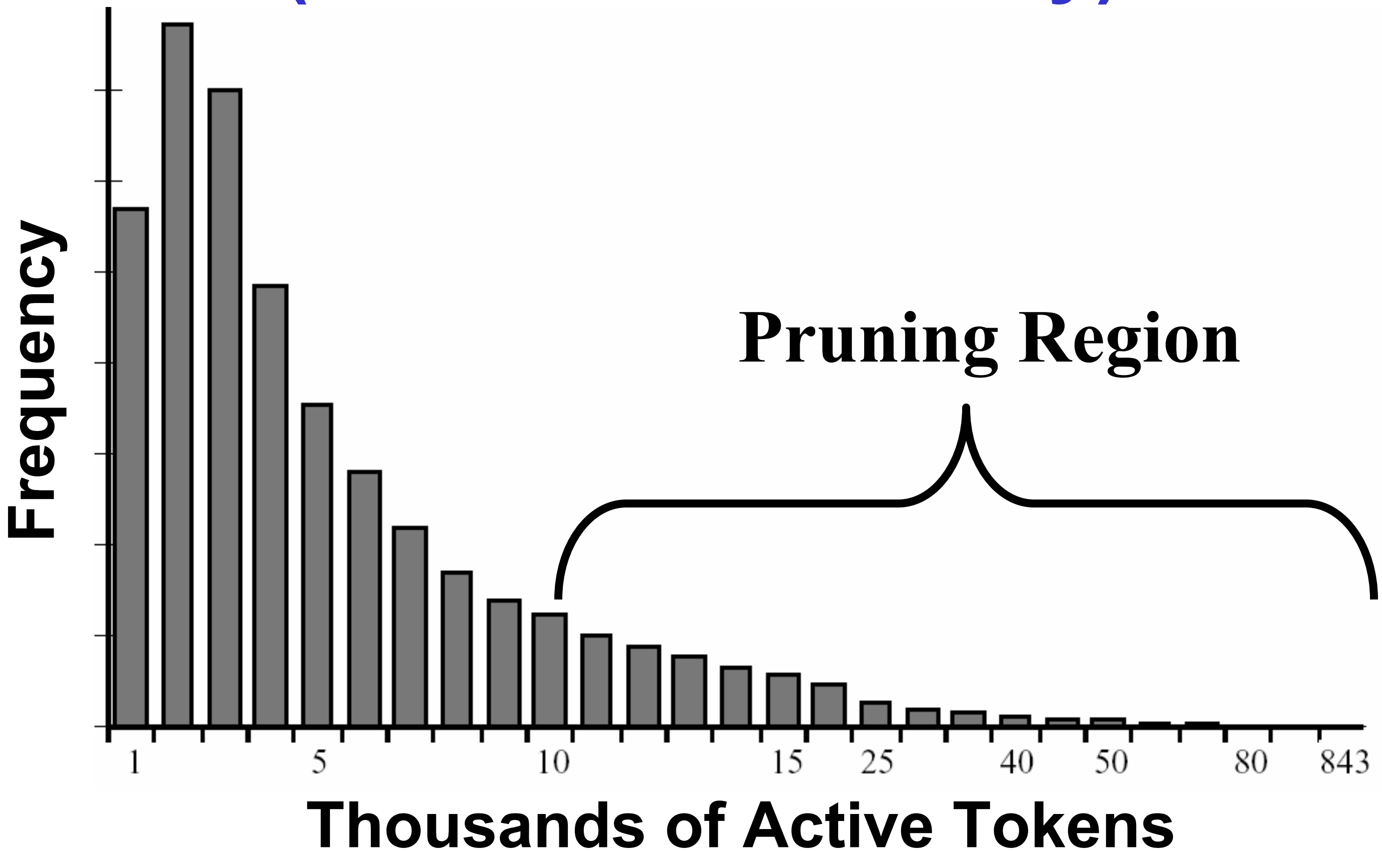
Example Types of Beams

- **Global-Beam:** **overall best token - BW_g**
- **Word-beam:** **best token in word – BW_w**
- **Phone-Beam:** **best token in phone – BW_p**
- **State-beam:** **best token within state - BW_s**

Histogram Pruning

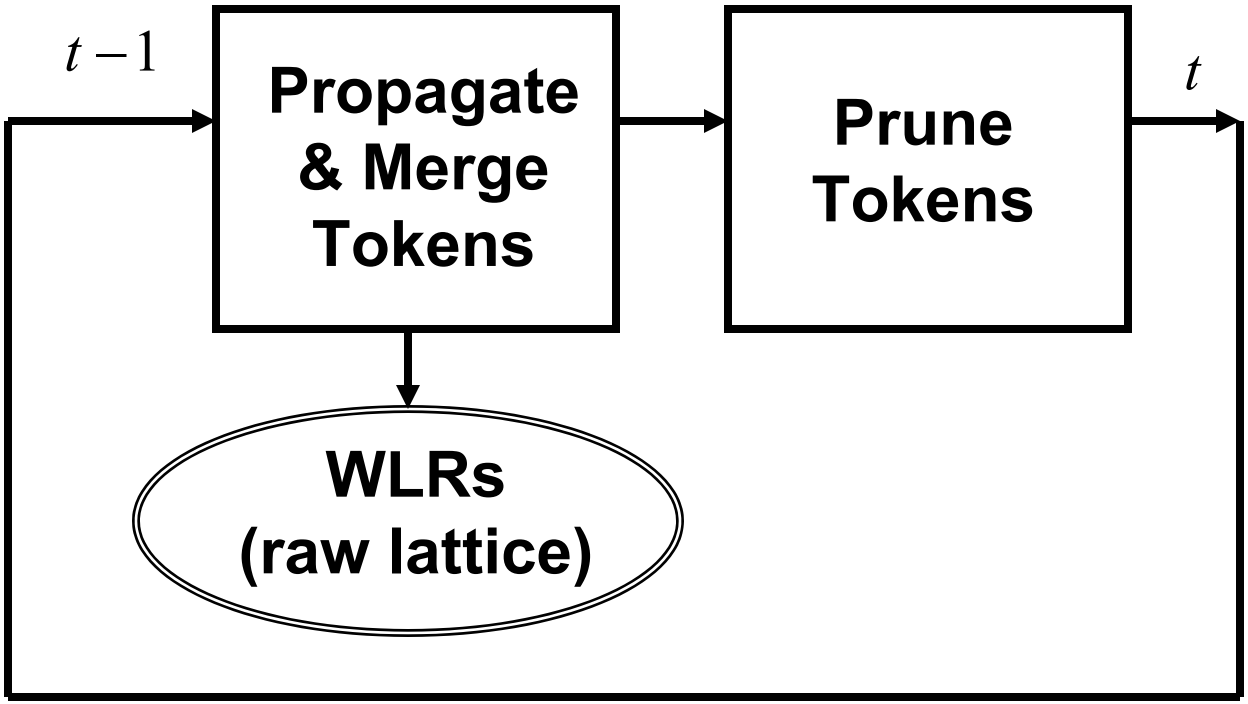
- For each frame, keep top N tokens (based on path score) propagated throughout search network.
- N = 10k → 40k tokens (depends on vocabulary size)
- Smaller N means fewer tokens, faster search speed, possibly more word errors due to accidental pruning of correct path.
- Reduces peak-memory required by decoder to store tokens

Active Tokens Per Frame (WSJ 5k Vocabulary)



Histogram from Julian Odell's PhD thesis, Cambridge University

Typical Token Passing Search Loop



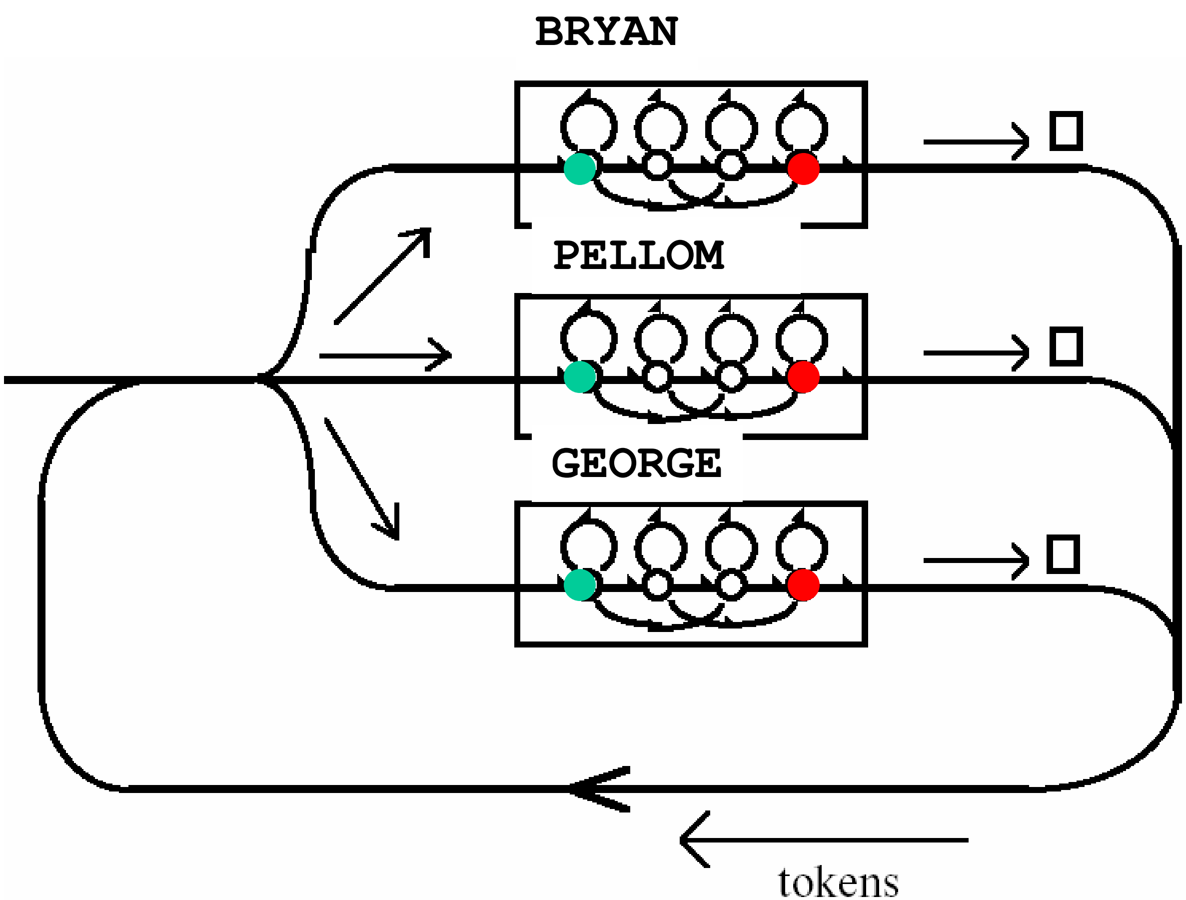
Cross-Word Modeling

- **How to incorporate between-word context dependency within search?**

```
BRYAN PELLOM → ?-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+?

BRYAN GEORGE → ?-B+R B-R+AY R-AY+AX
      AY-AX+N AX-N+JH N-JH+AO JH-AO+R
      AO-R+JH R-JH+?
```

Linear (Flat) Lexicon Search

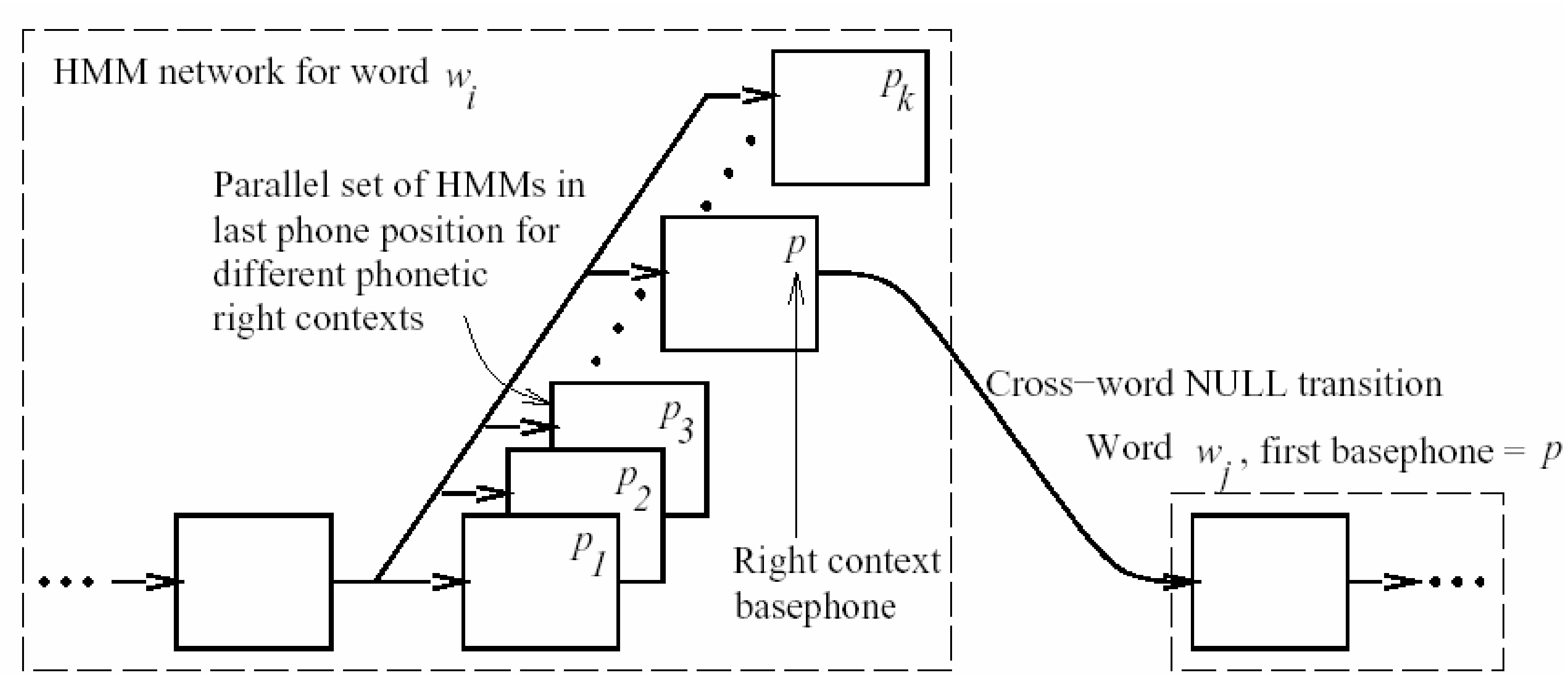


- **Green** = variable left-context (word-entry)
- **Red** = variable right-context (word-exit)

Right-Context Fan-out

- **The right-context of the last base phone of each word is the first base phone of the next word.**
- **Impossible to know the next word in advance of the search; can be several possible next words**
- **Solution: model the last phone of each word using a parallel set of triphone models; one for each possible phonetic right-context**

Illustration of Right-Context Fan-out



- **Illustration from CMU Sphinx-II Recognizer**

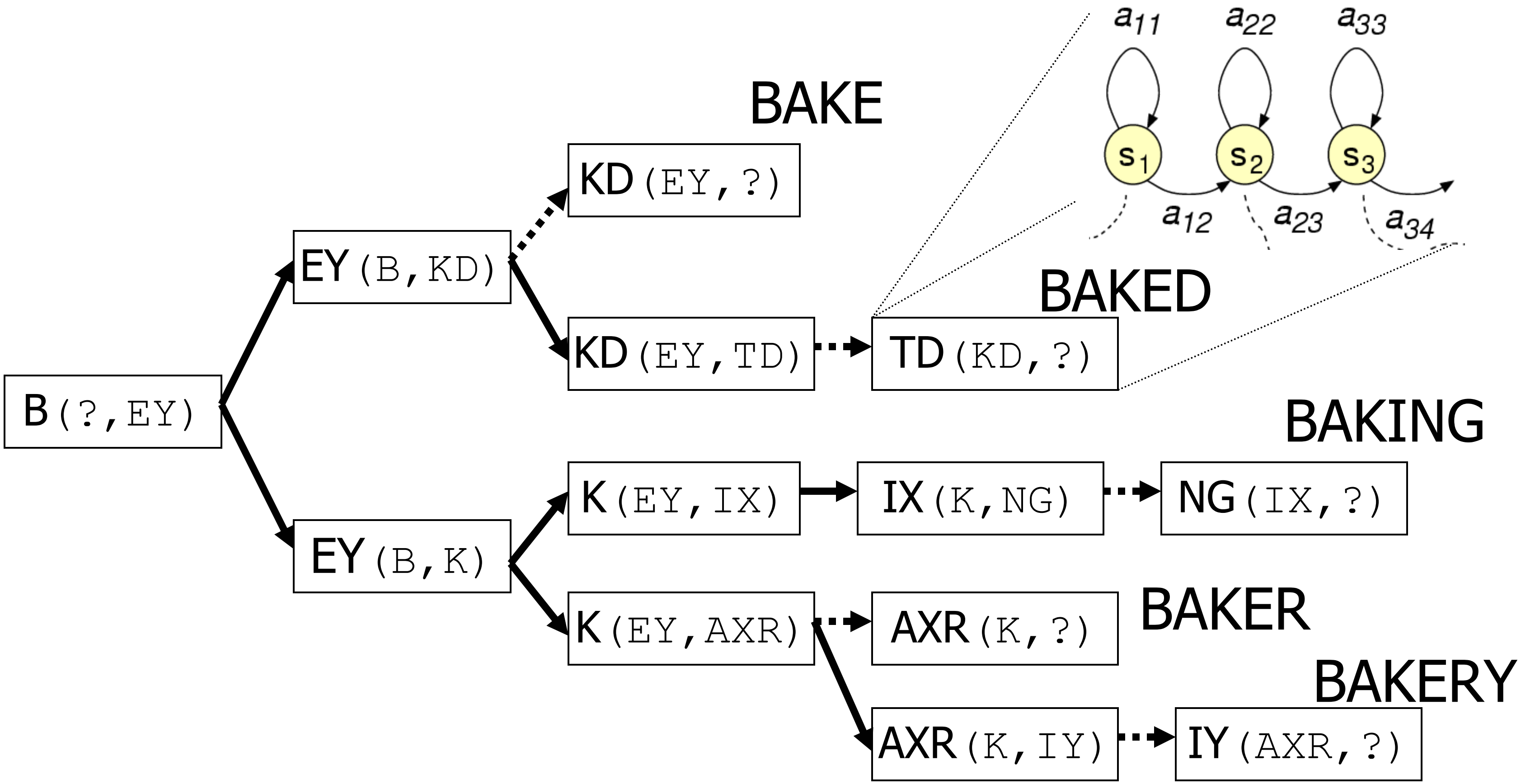
Left-Context Fan-Out

- **The phonetic left-context for the first phone position in a word is the last base phone from the previous word**
- **During search, no unique predecessor word**
- **Can fan-out at initial phone just as in the case of the right-context fan out;**
  - However, *word initial states are evaluated quite often*.
  - Some recognizers do suboptimal things. CMU Sphinx-II performs "Left-Context Inheritance"
  - Dynamically Inherit the left-context from the competing word with the highest partial path score.

Lexical Prefix Tree Search

- **As vocabulary size increases:**
  - Number of states needed to represent the flat search network increases linearly
  - Number of cross-word transitions increases rapidly
  - Number of language model calculations (required at word boundaries) increases rapidly
- **Solution: Convert Linear Search Network into a Prefix Tree.**

Lexical Prefix Tree

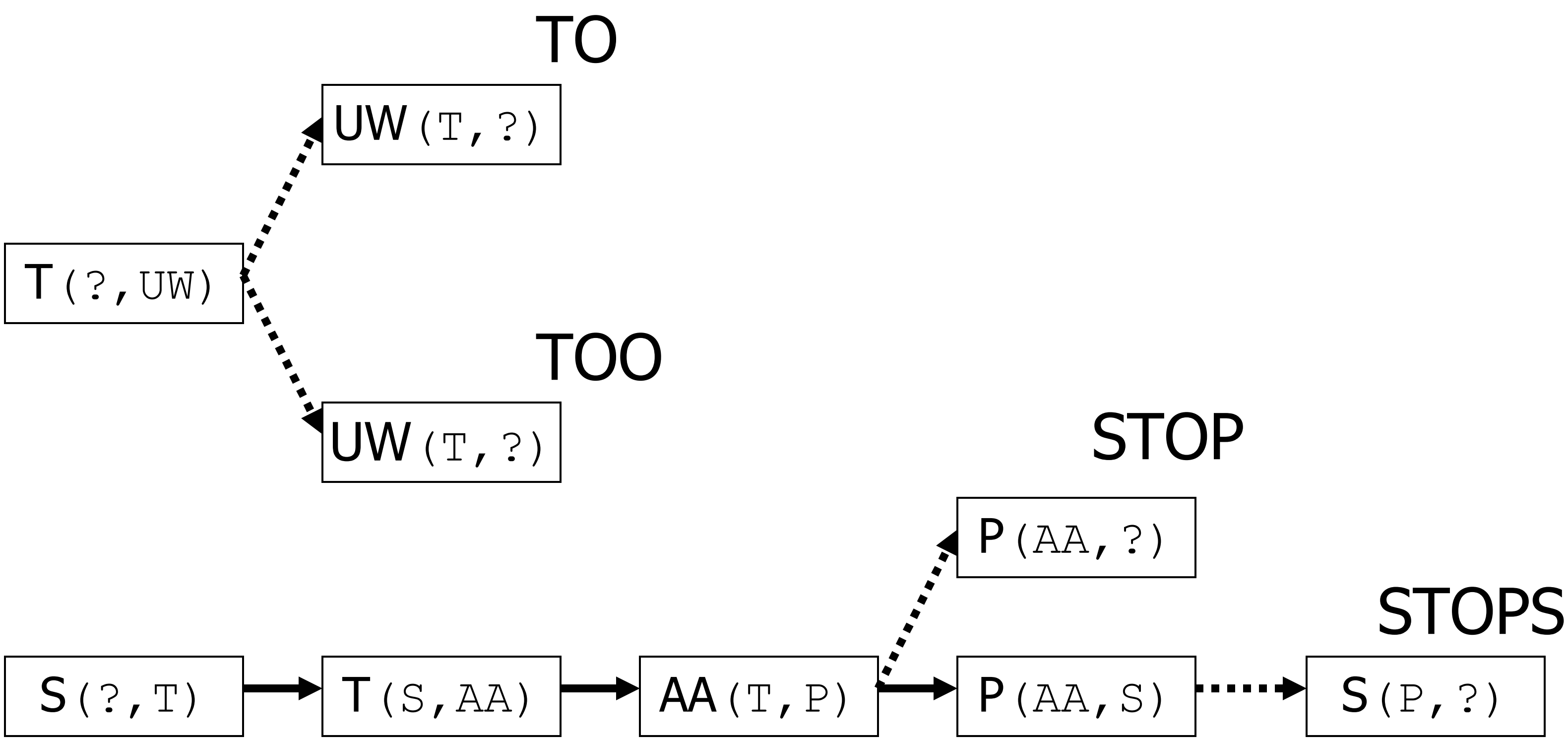


** Figure adapted from Huang et al., Spoken Language Processing, Prentice Hall*

Leaf Node Construction

- **Leaf Nodes ideally should have unique word identity**
- **Allows for efficient application of language model**
- **Handles instances such as,**
  - When word is the prefix of another word [“stop”, “stops”].
  - Homophones like “two” and “to”.

Leaf Node Construction



Advantages of Lexical Tree Search

- High degree of sharing at the root nodes reduces the number of word-initial HMMs needed to be evaluated in each frame
- Reduces the number of cross-word transitions
- Number of active HMM states and cross-word transitions grow more slowly with increasing vocabulary size

Advantages of Lexical Tree Search

- **Savings in the number of nodes in the search space [e.g., 12k vocabulary, 2.5x less nodes].**
- **Memory savings; fewer paths searched**
- **Search effort reduced by a factor of 5-7 over linear lexicon [since most effort is spent searching the first or second phone of each word due to ambiguities at word boundaries].**

Comparing Flat Network and Tree Network in terms of # of HMM states

	58K		
Level	Tree	Flat	Ratio
1	851	61657	1.4%
2	5782	61007	9.5%
3	18670	57219	32.6%
4	26382	49390	53.4%
5	24833	38254	64.9%
6	18918	26642	71.0%
7	13113	17284	75.9%
8	8129	10255	79.3%

Speed Comparison between Flat and Tree Search

Task	*Dev93*	*Dev94*	*Eval94*	*Mean*
20K	4.8	4.7	4.7	4.7
58K	5.2	4.8	4.5	4.9

- **CMU Sphinx-II : Speed Improvements of tree search compared to flat search for 20k and 58k word vocabularies [speed is about 4-5x faster!]**
- **Accuracy is about 20% relative worse for tree search.**

Disadvantages of Lexical Tree

- Root nodes model the beginnings of several words which have similar phonetic sequences
- Identity of word not known at the root of the tree

→ Can not apply language model until tree represents a unique word identity. "Delayed Language Modeling"

→ Delayed Language Modeling implies that pruning early on is based on acoustics-alone. This generally leads to increased pruning errors and loss in accuracy

T-61.184

Next Week

- **More search issues**
  - N-best Lists
  - Lattices / Word-Graphs

- **Pronunciation Lexicon Development & Prediction of Word Pronunciations from orthography. A review of approaches**

- **Practical aspects of training, testing, tuning speech recognition systems**