Return to Main

Objectives

Review:
Motivation

Decoder Basics:
Combining Scores
Continuous Speech

N-grams:
Unigram
Bigram
Trigram

Time Synchronous:
Level Building
Viterbi Beam

On-Line Resources:
Tutorial
Hierarchical
AJR: Search
SR: Search

LECTURE 35: TIME SYNCHRONOUS SEARCH

- Objectives:
    - No endpointing!
    - N-gram-specific search
    - Time synchronous search
    - Time synchronous Viterbi beam search

This lecture follows the course textbook closely:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

Another good source for some of this information is:

F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, Boston, Massachusetts, USA, ISBN: 0-262-10066-5, 1998.

SPEECH RECOGNITION REQUIRES GOOD PATTERN RECOGNITION AND SEARCH

- Continuous speech recognition is both a pattern recognition and search problem. Why?
- The decoding process of a speech recognizer finds the most probable sequence of words given the acoustic and language models. Recall our basic equation for speech recognition:

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

  Search is the process of finding the most probable word sequence:

$$\hat{W} = \underset{W}{\operatorname{argmax}} \left[\frac{P(W)P(A|W)}{P(A)} \right]$$

$$= \underset{W}{\operatorname{argmax}} [P(W)P(A|W)]$$

- The complexity of the search algorithm depends heavily on the nature of the search space, which in turn, depends heavily on the language model constraints (e.g., networks vs. N-grams).
- Speech recognition typically uses a hierarchical Viterbi beam search for decoding/recognition, and A^* stack decoding for N-best and word graph generation.

COMBINING SCORES IN THE LOG PROBABILITY SPACE

- Recall our basic equation defining the search problem:

$$\hat{\boldsymbol{W}} = \underset{\boldsymbol{W}}{\operatorname{argmax}} [P(\boldsymbol{W})P(\boldsymbol{A}|\boldsymbol{W})]$$

- It is convenient to process probabilities in the log domain:

$$C(\boldsymbol{W}|\boldsymbol{A}) \approx -\log[P(\boldsymbol{W})P(\boldsymbol{A}|\boldsymbol{W})]$$
$$= -(\log[P(\boldsymbol{W})] + \log[P(\boldsymbol{A}|\boldsymbol{W})])$$
$$\hat{\boldsymbol{W}} = \underset{\boldsymbol{W}}{\operatorname{argmin}} [C(\boldsymbol{W}|\boldsymbol{A})]$$

Why?

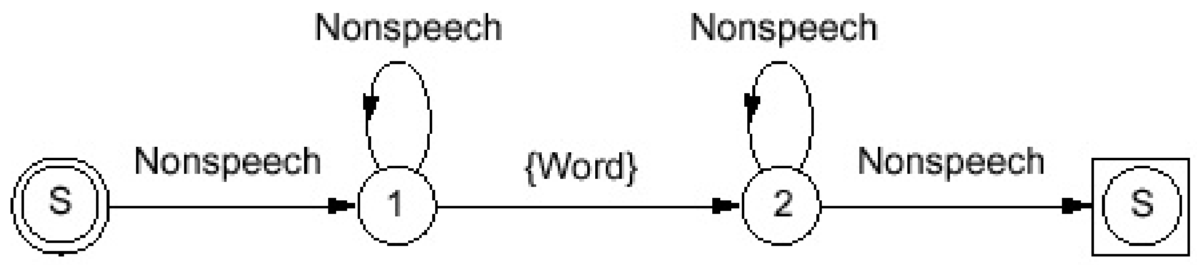
- It is also convenient to combine the language model and acoustic scores using a weighting factor:

$$P(\boldsymbol{W}) \approx P(\boldsymbol{W})^{LW} \mathrm{IP}^{N(\boldsymbol{W})}$$
$$\log(P(\boldsymbol{W})) = LW\log(P(\boldsymbol{W})) + N(\boldsymbol{W})\log(\mathrm{IP})$$

where LW is a language model weight,
$N(\boldsymbol{W})$ is the number of words in the hypothesis,
and IP is a word insertion penalty ($\mathrm{IP} \in [0, 1]$).

ISOLATED WORD RECOGNITION USING NETWORK DECODING

Isolated Word Recognition:

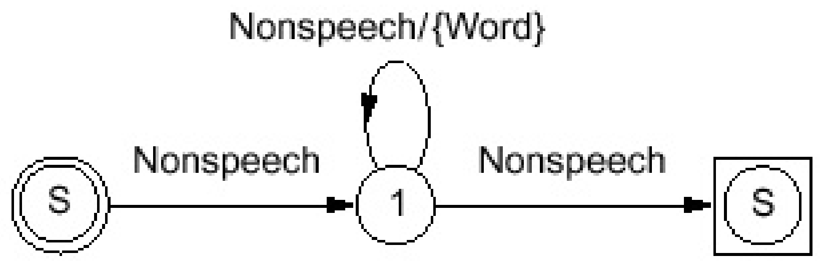


Nonspeech: typically an acoustic model of one frame in duration that models the background noise.

{Word}: any word from the set of possible words that can be spoken

- The key point here is that, with such a system, the recognizer finds the optimal start/stop times of the utterance with respect to the acoustic model inventory (a hypothesis-directed search)

Simple Continuous Speech Recognition ("No Grammar"):



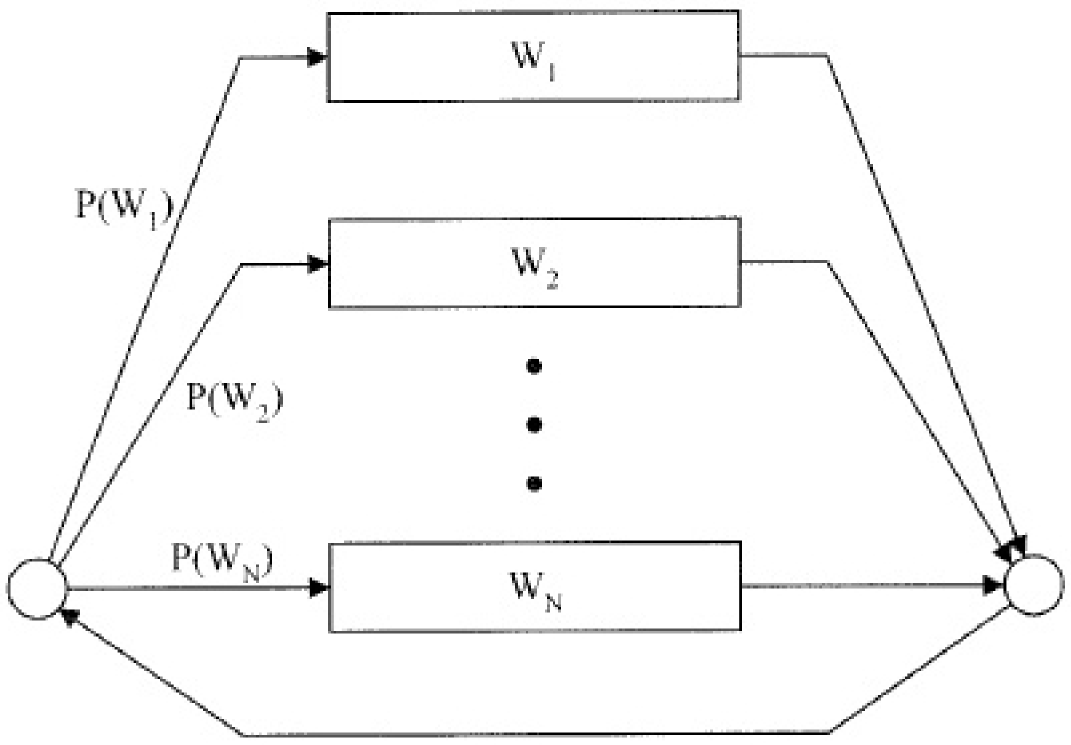
- system recognizes arbitrarily long sequences of words or nonspeech events

UNIGRAM SEARCH: SIMPLE BECAUSE IT IS MEMORYLESS

- The simplest N-gram search is the unigram search, since it is memoryless. The language model probability depends only on the current word:

$$P(W) = \prod_{i=1}^{N} P(w_i)$$

- The grammar network can be viewed as follows:



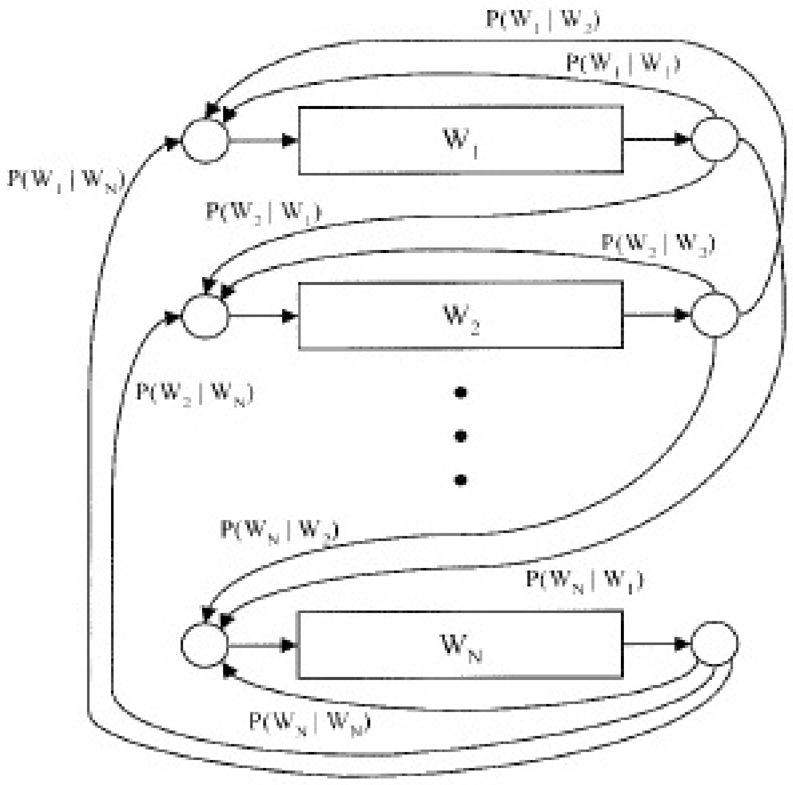
The final state of each word is connect to the collector state by a null transition. The collector state is connected to the start state with another null transition. Word expansion is trivial.

BIGRAM SEARCH: GOOD COMPROMISE BETWEEN PERFORMANCE AND COMPLEXITY

- A bigram search is still relatively simple:

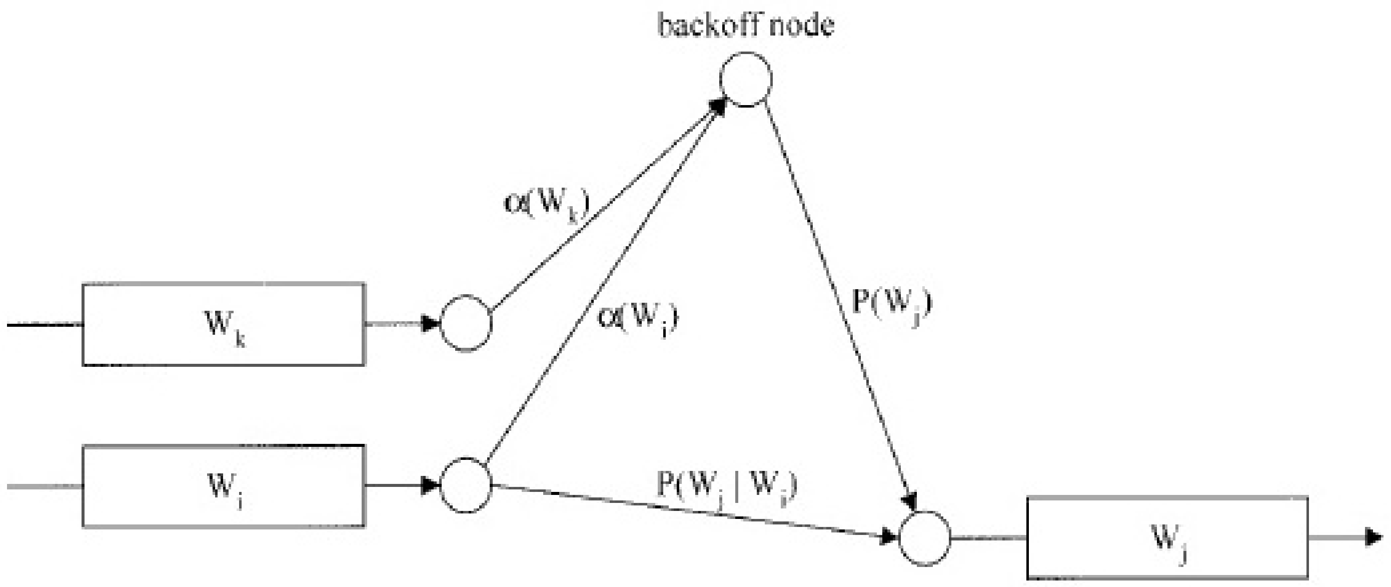
$$P(W) = P(w_1 | \langle s \rangle) \prod_{i=2}^{N} P(w_i | w_{i-1})$$

- A bigram search requires expand and merge:



The search complexity can be N^2 with a backoff model (if any word can follow an other word).

- We can reduce the complexity of a bigram search with backoffs by using a dynamic expansion:

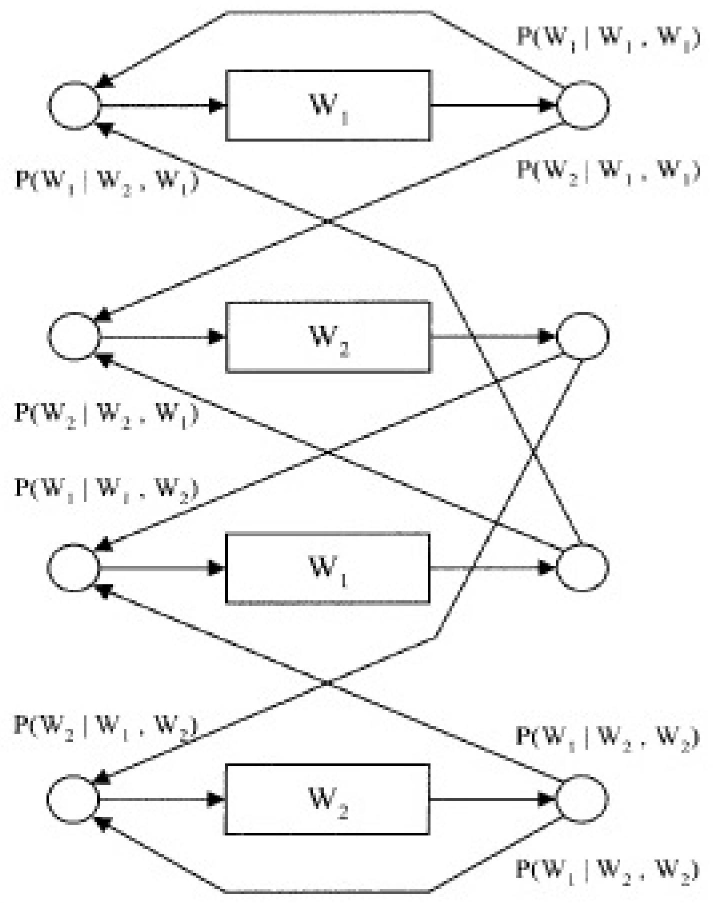


TRIGRAM SEARCH: OFTEN TOO COMPLEX FOR A FORWARD SEARCH

- A trigram search is fairly computationally intensive:

$$P(W) = (P(w_1|\langle s\rangle)P(w_2|\langle s\rangle, w_1)) \prod_{i=3}^{N} P(w_i|w_{i-2}, w_{i-1})$$

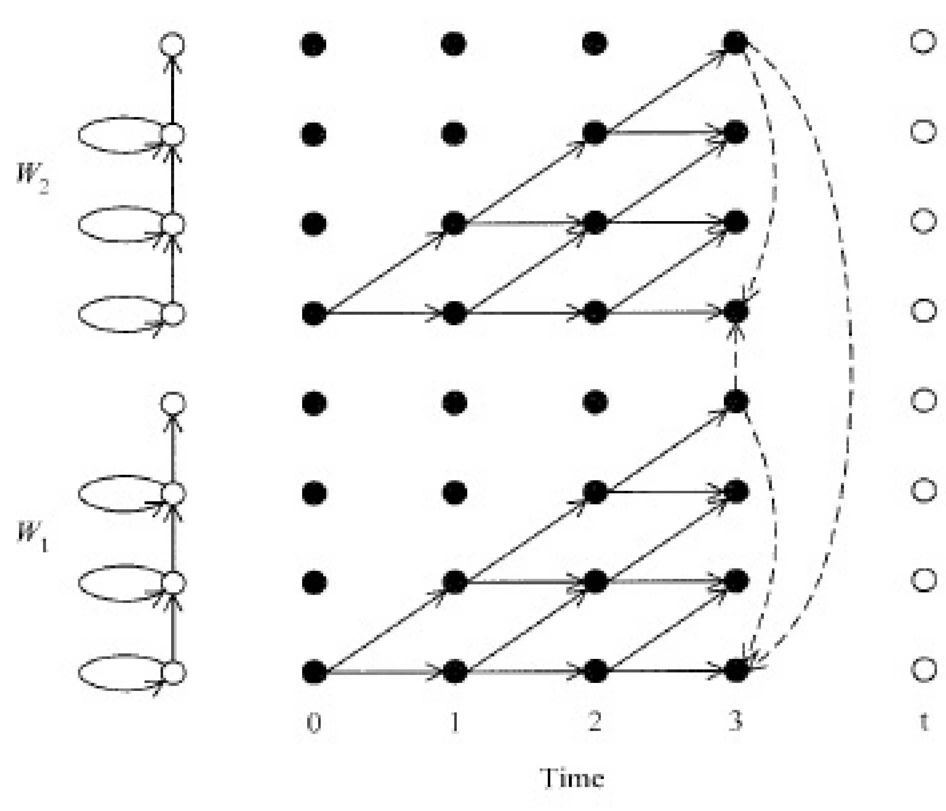
- A trigram search is shown below:



- A trigram search is often too complex for a single-pass forward search, and is instead typically implemented as a postprocessing step (rescoring) after a word graph has been generated.

TIME SYNCHRONOUS DECODING: TRELLIS EXPANSION

- Time synchronous decoding of a network can be viewed as a trellis expansion operation:

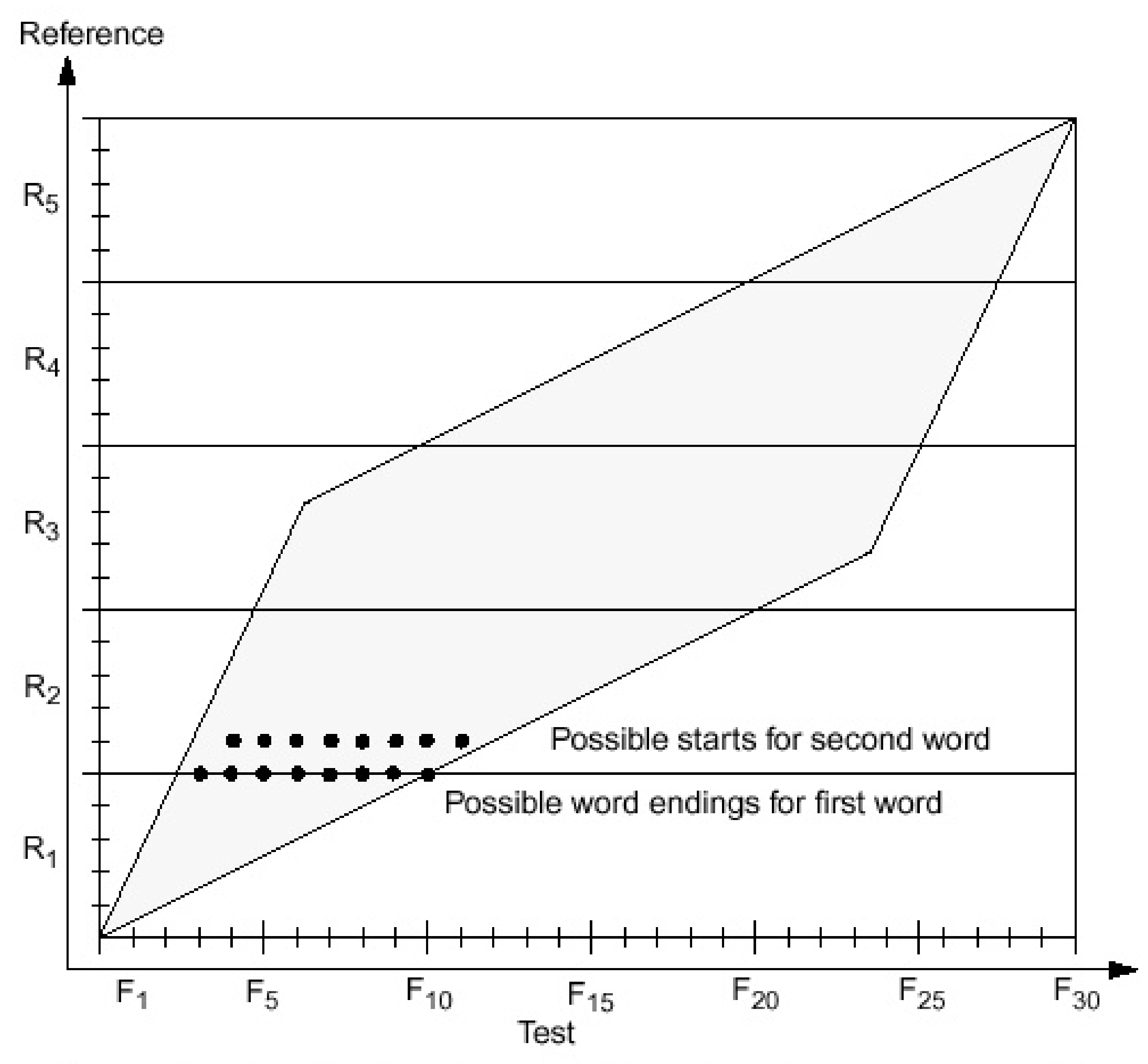


- This algorithm is based on the simple principle of dynamic programming (Sakoe and Chiba):

DTW, Syntactic Constraints, and Beam Search

Consider the problem of connected digit recognition: "325 1739". In the simplest case, any digit can follow any other digit, but we might know the exact number of digits spoken.

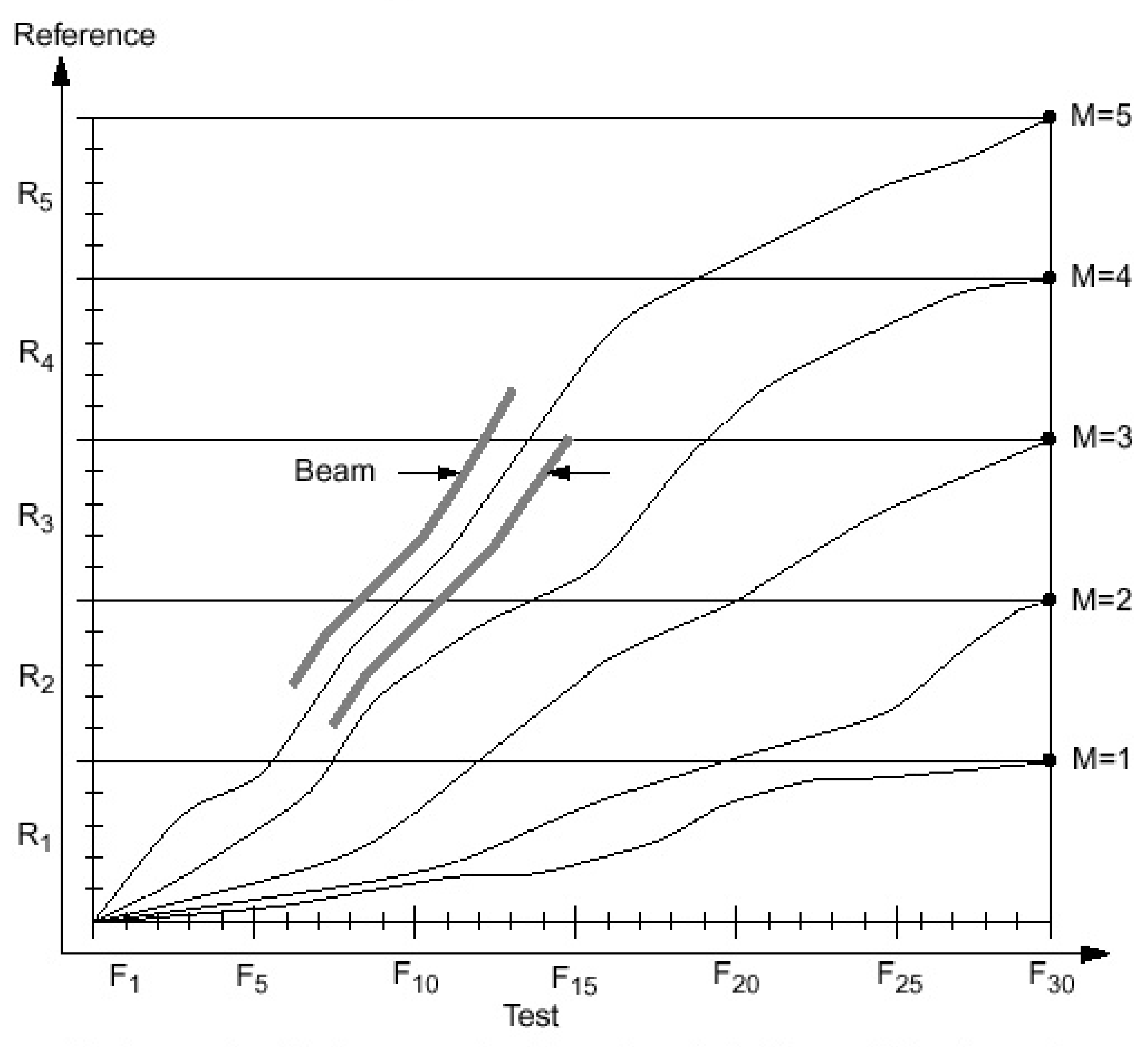
An elegant solution to the problem of finding the best overall sentence hypothesis is known as level building (typically assumes models are same length.



• Though this algorithm is no longer widely used, it gives us a glimpse into the complexity of the syntactic pattern recognition problem.

- This algorithm goes by many names including level building:

Level Building For An Unknown Number Of Words



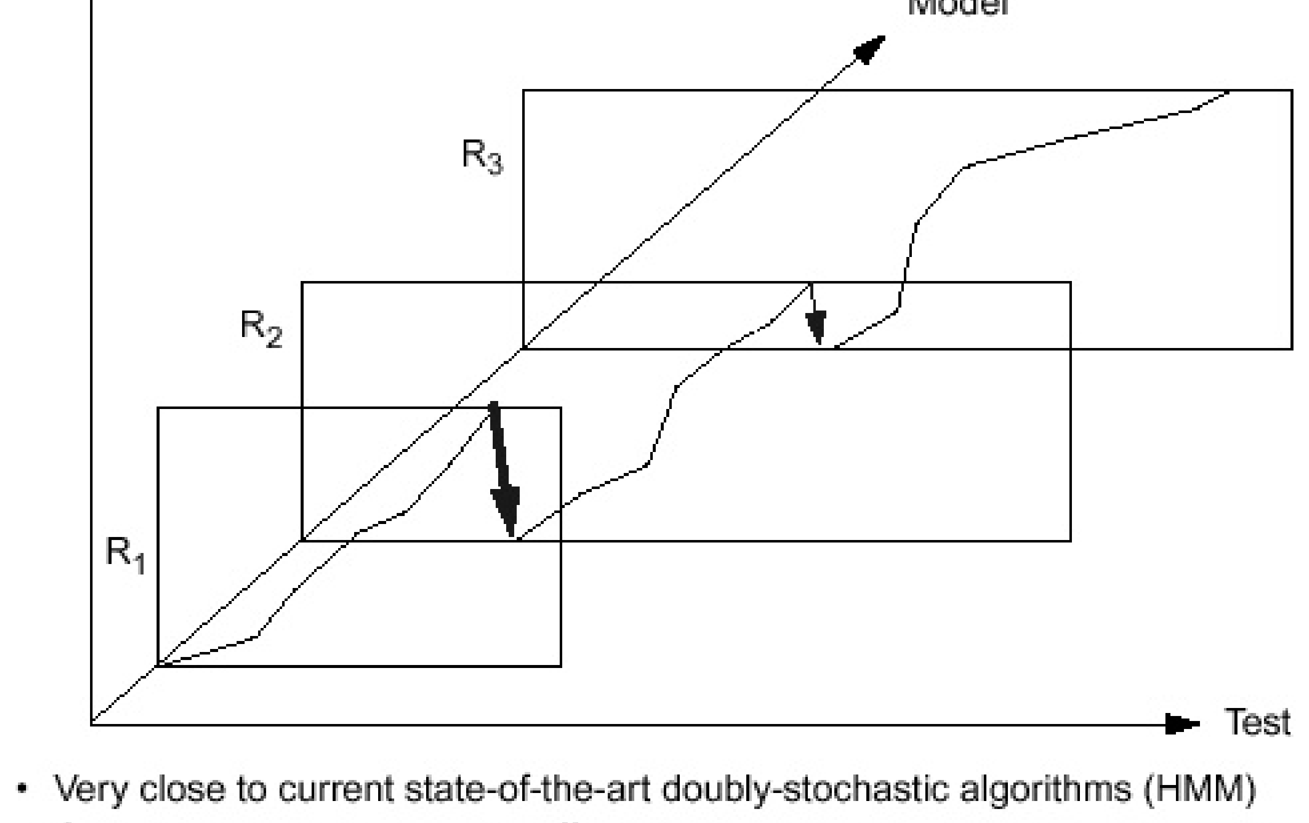
- Paths can terminate on any level boundary indicating a different number of words was recognized (note the significant increase in complexity)
- A search band around the optimal path can be maintained to reduce the search space
- Next-best hypothesis can be generated (N-best)
- Heuristics can be applied to deal with free endpoints, insertion of silence between words, etc.
- Major weakness is the assumption that all models are the same length!

- and the Bridle algorithm (one-stage DP):

The One-Stage Algorithm ("Bridle Algorithm")

The level building approach is not conducive to models of different lengths, and does not make it easy to include syntactic constraints (which words can follow previous hypothesized words).

An elegant algorithm to perform this search in one pass is demonstrated below:

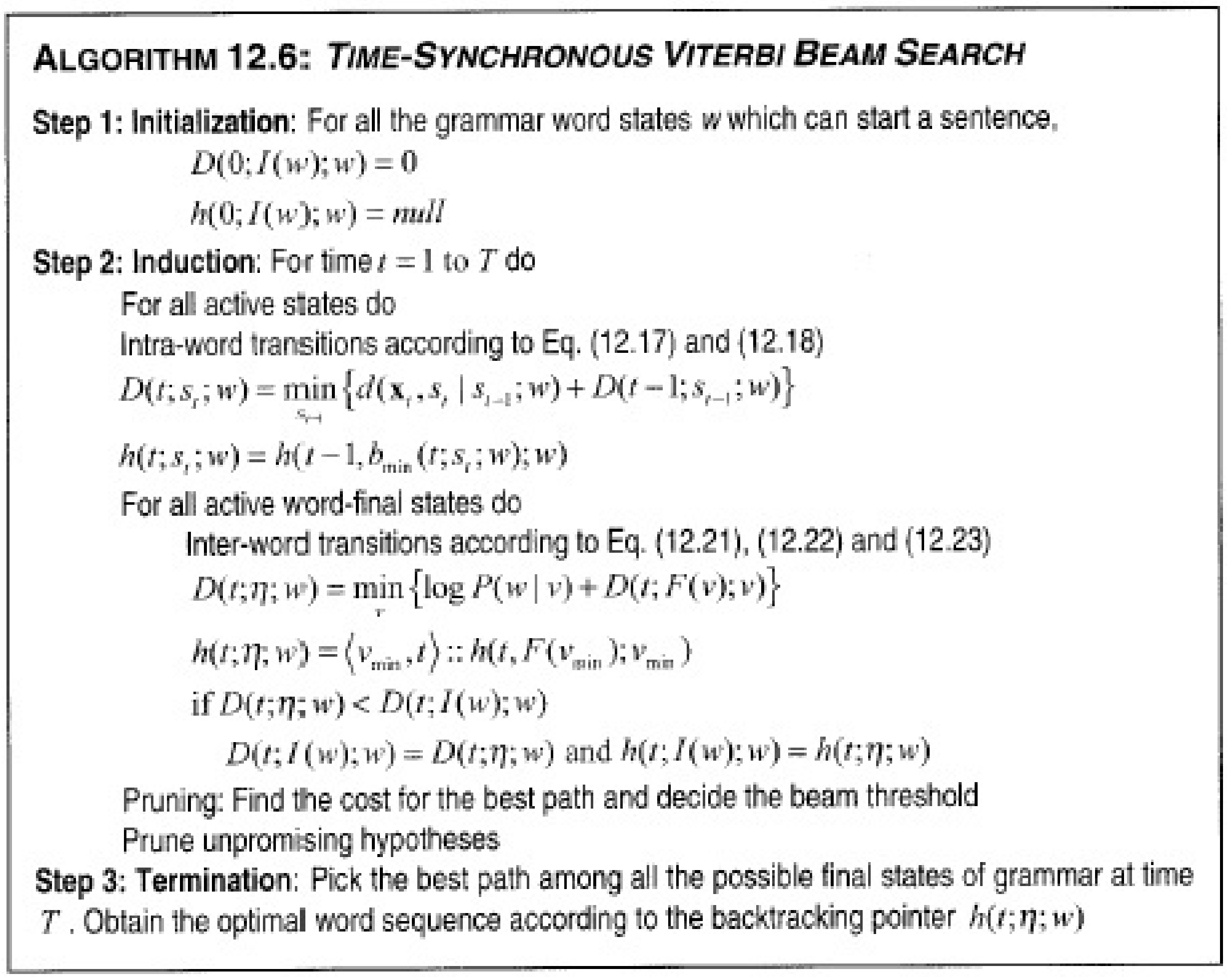


- Very close to current state-of-the-art doubly-stochastic algorithms (HMM)
- Conceptually simple, but difficult to implement because we must remember information about the interconnections of hypotheses
- Amenable to beam-search concepts and fast-match concepts
- Supports syntactic constraints by limited the choices for extending a hypothesis
- Becomes complex when extended to allow arbitrary amounts of silence between words
- How do we train?

It was first introduced for dynamic time-warping (DTW) systems.

TIME SYNCHRONOUS VITERBI BEAM SEARCH

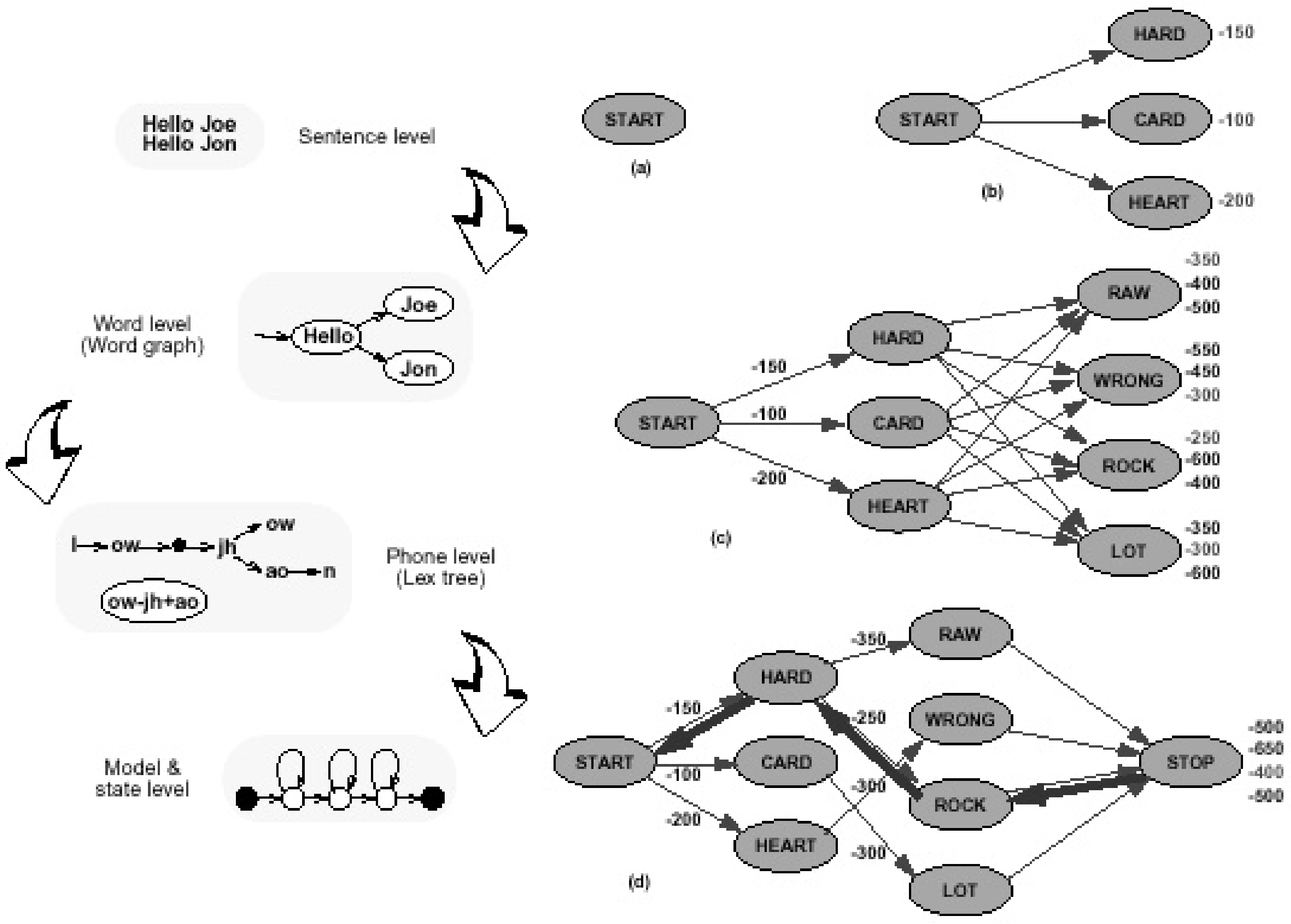
- We can define many of these search concepts into a single algorithm: expansion operation:



ALGORITHM 12.6: *TIME-SYNCHRONOUS VITERBI BEAM SEARCH*

```
Step 1: Initialization: For all the grammar word states w which can start a sentence,
        $D(0; I(w); w) = 0$
        $h(0; I(w); w) = null$
Step 2: Induction: For time $t = 1$ to $T$ do
    For all active states do
    Intra-word transitions according to Eq. (12.17) and (12.18)
    $D(t; s_t; w) = \min_{s_{t-1}} \{ d(\mathbf{x}_t, s_t \mid s_{t-1}; w) + D(t-1; s_{t-1}; w) \}$
    $h(t; s_t; w) = h(t-1, b_{\min}(t; s_t; w); w)$
    For all active word-final states do
        Inter-word transitions according to Eq. (12.21), (12.22) and (12.23)
        $D(t; \eta; w) = \min_{v} \{ \log P(w \mid v) + D(t; F(v); v) \}$
        $h(t; \eta; w) = \langle v_{\min}, t \rangle :: h(t, F(v_{\min}); v_{\min})$
        if $D(t; \eta; w) < D(t; I(w); w)$
            $D(t; I(w); w) = D(t; \eta; w)$ and $h(t; I(w); w) = h(t; \eta; w)$
    Pruning: Find the cost for the best path and decide the beam threshold
    Prune unpromising hypotheses
Step 3: Termination: Pick the best path among all the possible final states of grammar at time
$T$. Obtain the optimal word sequence according to the backtracking pointer $h(t; \eta; w)$
```

- This algorithm uses dynamic expansion of the network to minimize memory requirements:



We will have more to say about this algorithm later.