

# Automatic Speech Recognition

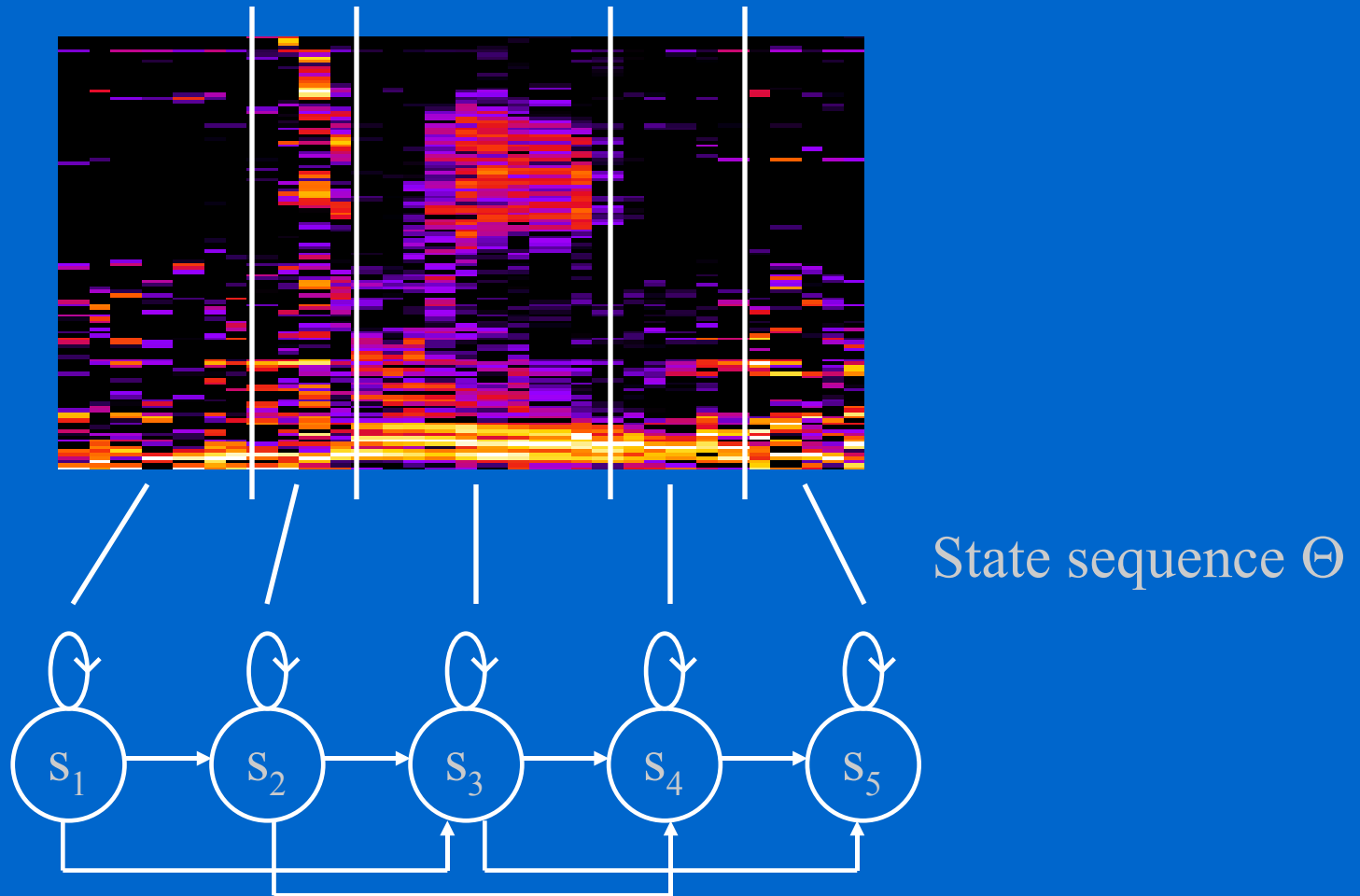
Dr.-Ing. Bernd Plannerer

[plannerer@ieee.org](mailto:plannerer@ieee.org)

# Training the HMMs

- Optimization criterium
  - Maximum Likelihood Training:  
Find model parameters which maximize  $p(\mathbf{X} | \Lambda_m)$ .
- Training algorithms for ML Training
  - Baum - Welch Training
  - Segmental K - Means Algorithm

# Training the HMMs



# Training the HMMs

- Segmental - K - means Algorithm
- Iteratively maximize  $p(\mathbf{X}, \Theta_{\text{opt}} | \lambda_m)$  by maximization and segmentation:
  - segmentation: Find  $\Theta_{\text{opt}}$  via Viterbi-Algorithm using the given set of parameters  $\lambda_m$
  - maximization: Compute an optimized set of parameters  $\lambda_{m\_opt}$

# Training the HMMs

- Maximize  $p(\mathbf{X}, \Theta_{\text{opt}} | \lambda_m)$  :
  - Segmentation: Compute  $\Theta_{\text{opt}}$ . Therefore,  
 $p(\mathbf{X}, \Theta_{\text{opt}} | \lambda_m) \geq p(\mathbf{X}, \Theta | \lambda_m)$
  - Optimization: Compute  $\lambda_{m\_opt}$ . Therefore,  
 $p(\mathbf{X}, \Theta_{\text{opt}} | \lambda_{m\_opt}) \geq p(\mathbf{X}, \Theta_{\text{opt}} | \lambda_m)$
  - Therefore, for each Iteration,  
 $p(\mathbf{X}, \Theta_{\text{opt}} | \lambda_{m\_opt}) \geq p(\mathbf{X}, \Theta_{\text{opt}} | \lambda_m) \geq p(\mathbf{X}, \Theta | \lambda_m)$

# Training the HMMs

- Segmentation using the Viterbi-Algorithm
- Optimization: Compute  $\lambda_{m\_opt}$
- Optimization can be performed separately for each state

A-posteriori probability,  
of vector  $\vec{x}$  belonging to class  $k$

$$\hat{m}_k = \frac{\sum_{i=1}^N p(k|\vec{x}_i) \cdot \vec{x}_i}{\sum_{i=1}^N p(k|\vec{x}_i)}$$

Normalization

$$\hat{C}_k = \frac{\sum_{i=1}^N p(k|\vec{x}_i) \cdot (\vec{x}_i - \hat{m}_k) \cdot (\vec{x}_i - \hat{m}_k)'}{\sum_{i=1}^N p(k|\vec{x}_i)}$$

# Segmental K-Means Algorithm

- Training algorithm (simplified)
  - Start with single mixture component for each model state
  - Maximize PDF for given number of mixtures
    - Iteration: Segmentation and Optimization
  - Splitting of mean vectors der (as with K-Means Algorithm)
  - Iteratively increase number of Mixtures, until no more mixtures can be trained from the given training data

- 
- 
- 

## Word models and subword units

- Word models: Ideal modelling of within-word coarticulation
- Problem:
  - each word model requires huge amount of training data
  - changing the vocabulary requires recording of new speech samples

## Word models and subword units

- Solution:
  - compose word models from subword units
  - a limited number of subword units can be trained using a limited amount of speech data.
  - compromise: Choose subword units as big as possible given the amount of training data at hand

# Word models and subword units

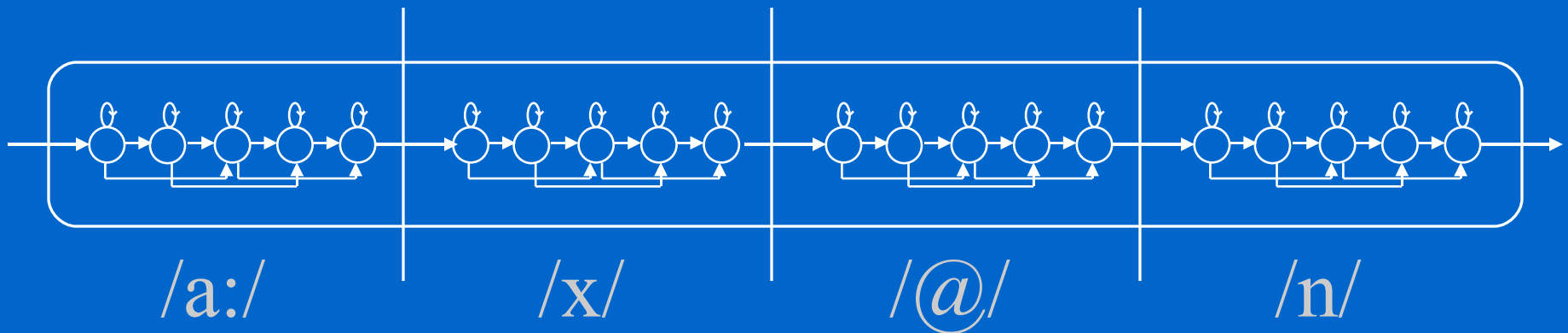
- Choosing the subword units
  - Selection criteria
    - trainability
    - context dependency
  - subword unit examples
    - phonemes (context independent)
    - triphones (left- and right context)
    - demisyllables
    - automatically selected units of any size

# Word models and subword units

- Number of subword units
  - phonemes 47
  - triphones  $47^3 = 103\ 823$
  - demisyllables 750 ID + 880 FD
- Selection based on statistics
  - selection from triphones by trainability
  - automatic selection of subword units (from phonemes up to word models)

# Word models and subword units

- Structure of word models
  - identical phonemes have identical parameters in all words

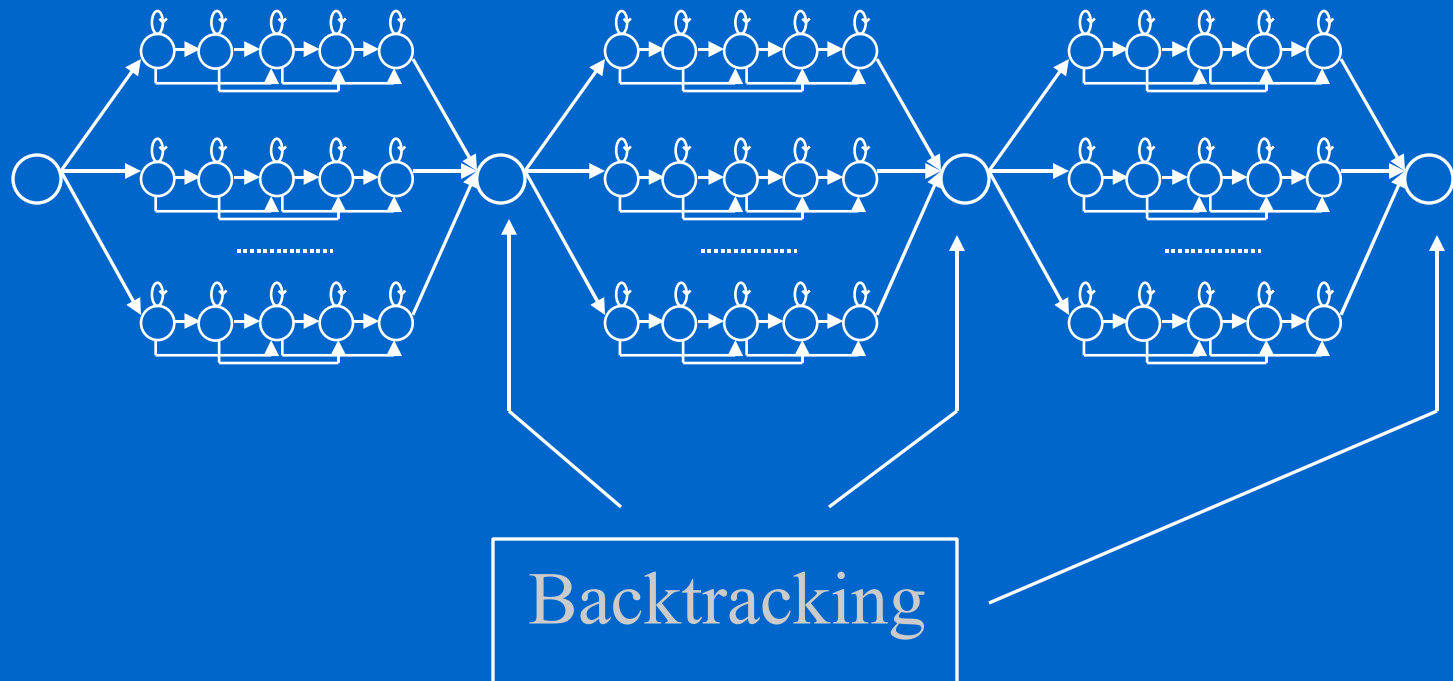


## Recognition of continuous speech

- So far: Recognition of connected words by searching the optimal state sequence
- Computation of  $p(\mathbf{X}|w_1, w_2, \dots, w_N)$
- All word transitions were assumed to be equally likely
- Acoustic modelling is the only „knowledge source“ during recognition



# Example: PIN recognition



## Example: PIN recognition

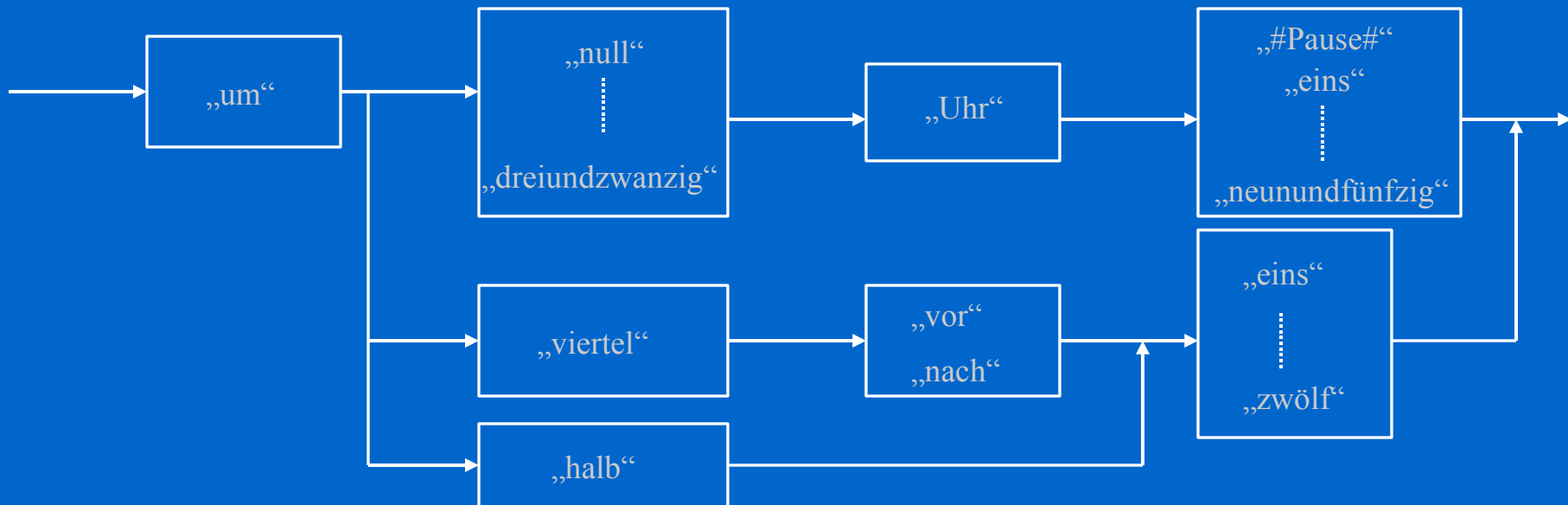
- Recombination of paths takes only place for paths containing an equal number of digits
- In the terminal node, only paths containing exactly 3 digits are recombined
- Generally, a „memory of length  $N$ “ can be modelled by splitting the generation model into  $N$  partitions.

# Continuous Speech Recognition

- Exploiting additional knowledge sources
  - Finite State Grammar (FSG)
    - phrases can be modelled using FSG
    - many „ready to use“ grammars are available
    - semantic analysis is possible
  - Language Model
    - statistic approach
    - automatic training procedure

# Continuous Speech Recognition

- Finite State Grammar
  - example: time of day



# Continuous Speech Recognition

- Language Model
  - represents the transition probabilities between words
  - formulation of recognition task:
    - given a vector sequence  $\mathbf{X}$ , find the word sequence  $W_{\text{opt}}$  with length  $L$  according to the equation:
    - $W_{\text{opt}} = \text{argmax}_W \{P(W|\mathbf{X})\}$
    - search for the word sequence with the highest *a posteriori probability*  $P(W|\mathbf{X})$

# Continuous Speech Recognition

- Sentence of Bayes

- $W_{\text{opt}} = \operatorname{argmax}_W \{P(W|\mathbf{X})\}$   
 $= \operatorname{argmax}_W \{P(W)p(\mathbf{X}|W)\}$

- $P(W)$  : (a priori) probability of word sequence  $W$   
 $= \{w_1, w_2, \dots, w_L\}$

- $p(\mathbf{X}|W)$  : emission probability of  $\mathbf{X}$  given the word sequence  $W$

- Approximation using the dominant state sequence:  
 $W_{\text{opt}} = \operatorname{argmax}_W \{P(W)\max_{\Theta} p(\mathbf{X}, \Theta|W)\}$

# Continuous Speech Recognition

- Language Model

- $P(W) = P(w_1)P(w_2|w_1)P(w_3|w_2, w_1) \dots$   
 $\dots P(w_L|w_{L-1}, w_{L-2}, \dots, w_1)$

- n-Gramm-Model

- $P(w_\tau | w_{\tau-1}, \dots, w_1) = P(w_\tau | w_{\tau-1}, \dots, w_{\tau-n+1})$

- typical models

- bigram (one predecessor word)

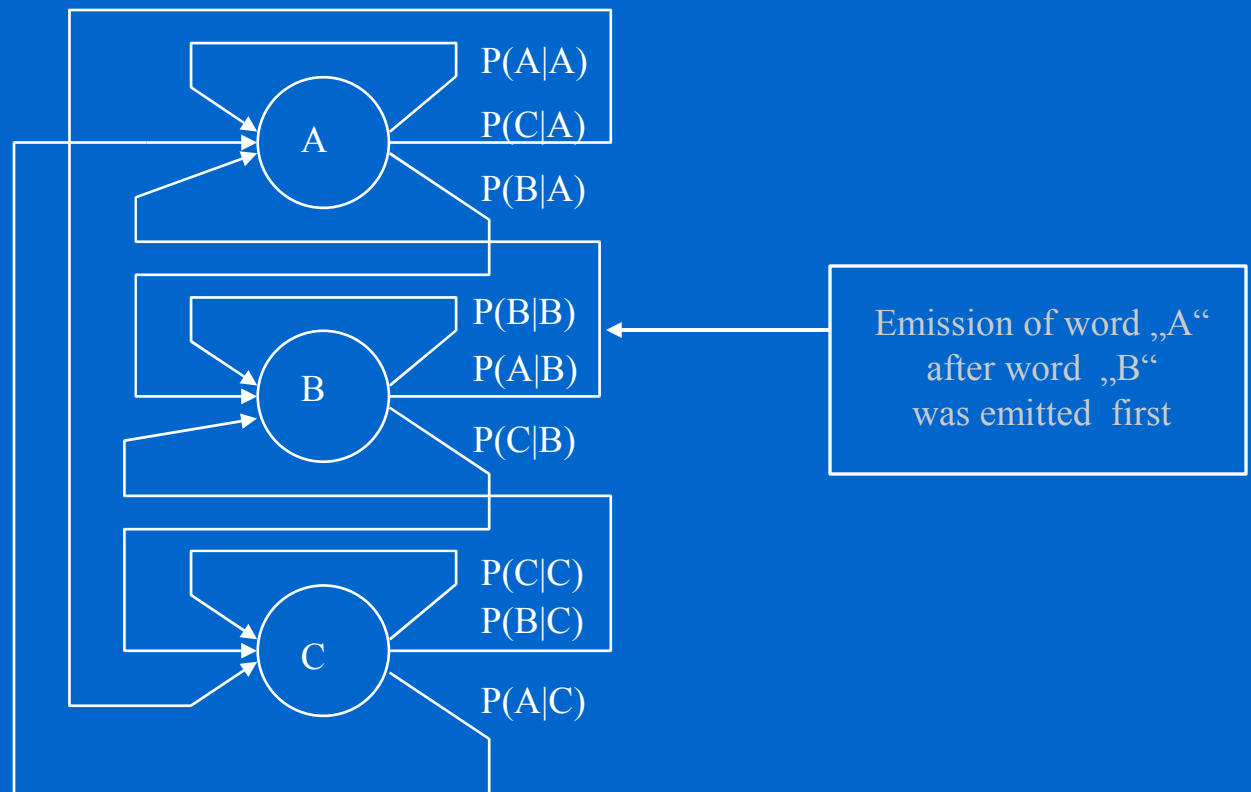
- trigram (two predecessor words)

# Continuous Speech Recognition

- Language Model
  - requires huge text corpora for training (millions of words)
  - only the orthographic representation is required
  - interpolation techniques for „smoothing“
  - word categories can be defined with a-priori probabilities within a category (unigram) and with n-gram probabilities for category transitions

# Continuous Speech Recognition

- Structure of a recognizer using a bigram-LM



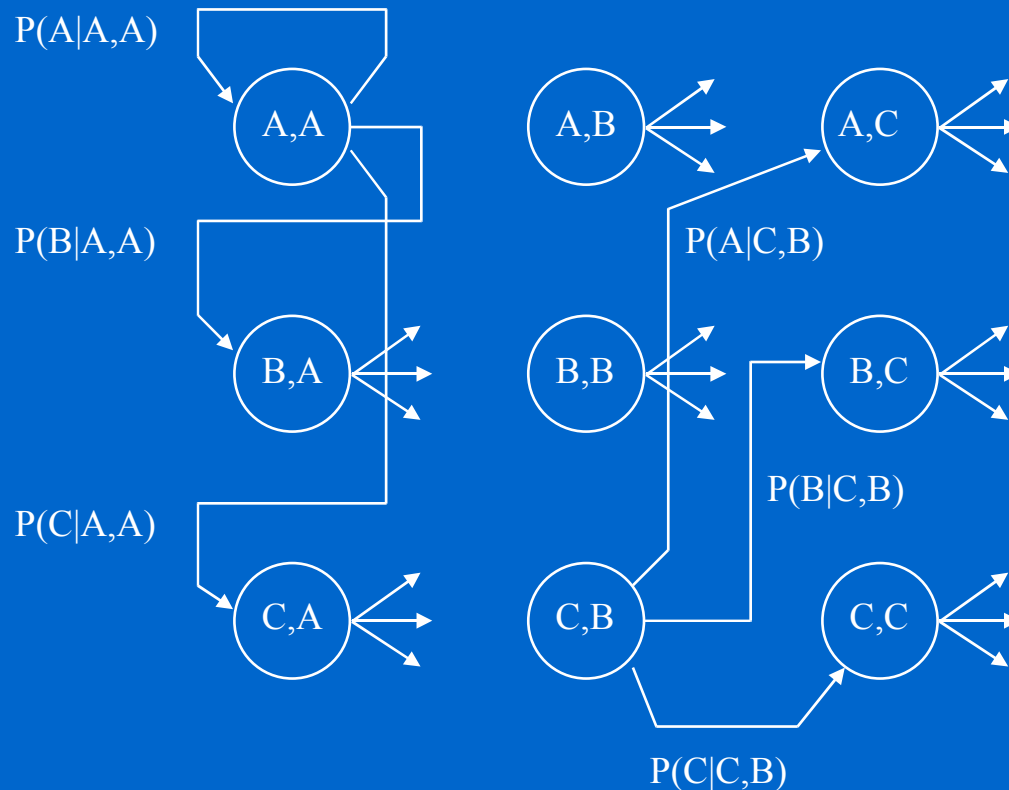
•  
•  
•

# Continuous Speech Recognition

- Exercise
- What would the structure of a recognizer using a trigram language model look like ?

# Continuous Speech Recognition

- Recognizer with trigram-LM

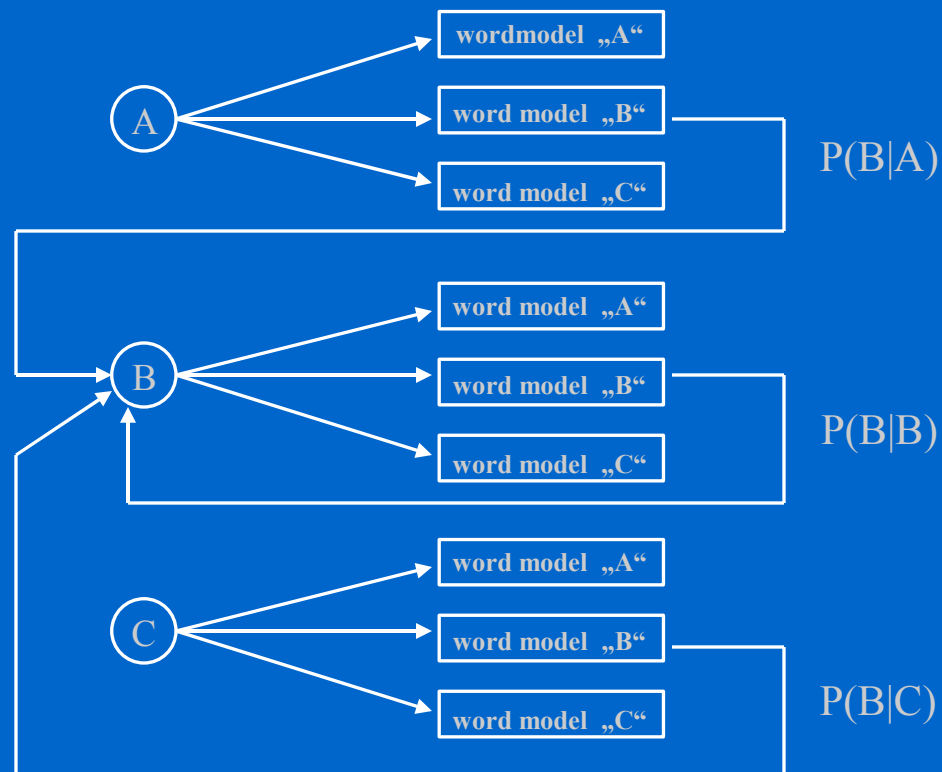


# Continuous Speech Recognition

- Structure of recognizers with language-model
  - words are emitted during the transitions between the nodes
  - LM nodes represent the „history“ (predecessor words)
  - number of states: number of predecessor combinations:  $(N-1)$ -Tupel from vocabulary  $V$ :  $V^{(N-1)}$
  - all edges leading to a given node have emitted the last word  $(w_{\tau-1})$  of this node's history

# Continuous Speech Recognition

- „Naive“ integration of the acoustic models

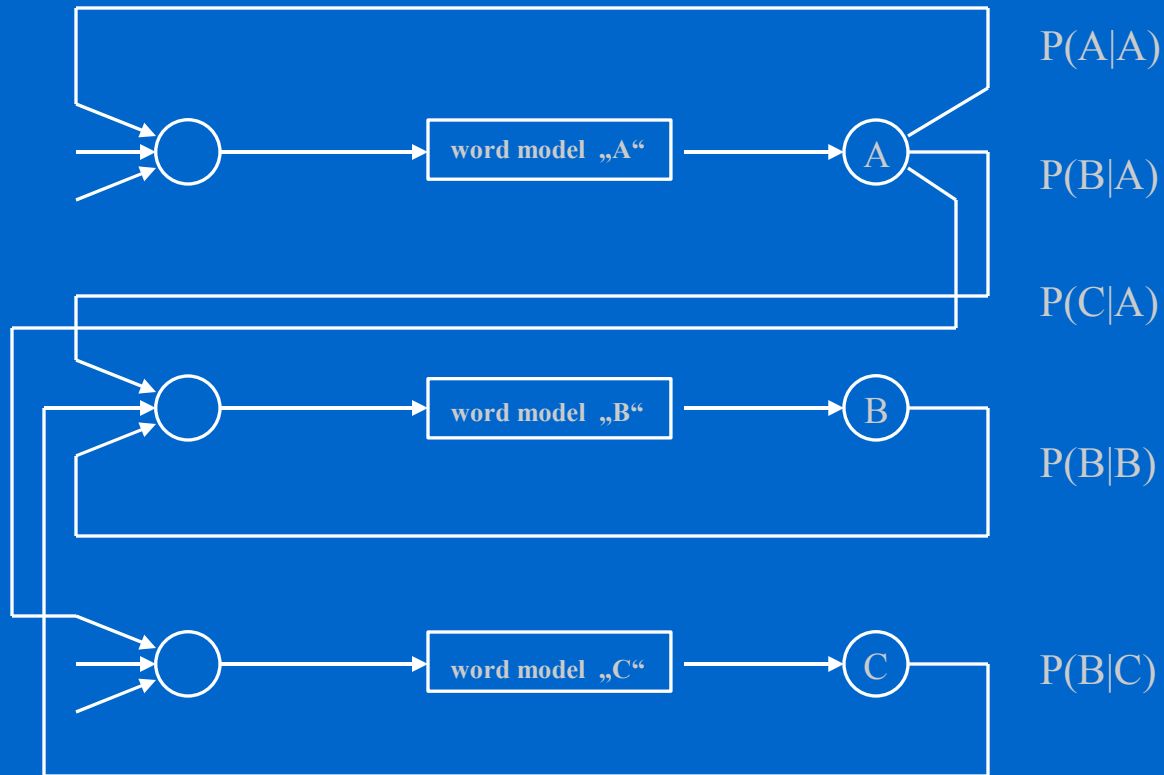


# Continuous Speech Recognition

- DP Algorithm: Only comparable paths are recombined.
- A new word  $w_\tau$  starts from several word ends  $w_\tau$
- Word transition score is added *before* the path recombination at the nodes of the LM. Then the new words are started („early recombination“).

# Continuous Speech Recognition

- „early recombination“



# Continuous Speech Recognition

- Structure using a bigram model
  - in analogy with connected word recognition
  - for the word transition  $v \rightarrow w$ , the word transition score is added, then the best predecessor is selected
  - search for th best word sequence is performed by searching the optimum state sequence (DP-algorithm)
- Structure using a trigram model
  - requires  $V$  independent copies of the word models

# Continuous Speech Recognition

- Estimation of computational effort (bigram)
  - vocabulary  $V$ : 20K
  - subword units per word approx.: 7
  - model states per subword unit: 6
  - number of instances per word model : 1
  - search space :  $20K \times 7 \times 6 \times 1 = 840000$  states
  - time frame: 10 ms
  - number of state hypotheses per second: 84 000 000  
 $= 8.4 \times 10^7$

# Continuous Speech Recognition

- Estimation of computational effort (trigram)
  - vocabulary  $V$ : 20K
  - subword units per word approx.: 7
  - states per subword unit: 6
  - number of instances per word model 20K
  - search space:  $20K \times 7 \times 6 \times 20K = 16,8 \times 10^9$  states
  - time frame: 10 ms
  - number of state hypotheses per second:  
 **$16,8 \times 10^{11}$**

# Continuous Speech Recognition

- Estimation of computational effort
  - Problems
    - CPU time
    - memory
  - Solution
    - disregard non-promising paths („pruning“)
    - only the „surviving“ paths will be computed and stored
    - data driven construction of search space: The whole theoretic search space will be never generated in memory

# END OF CHAPTER 4

