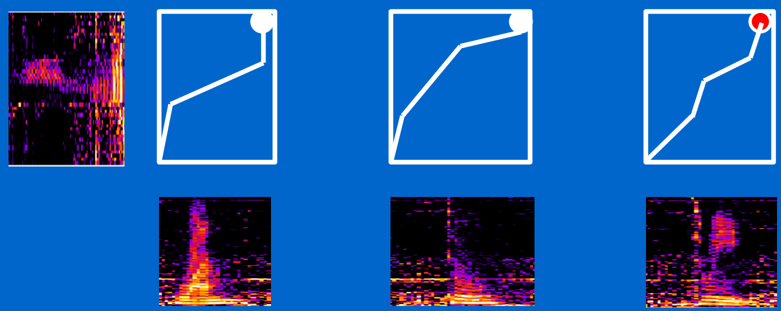
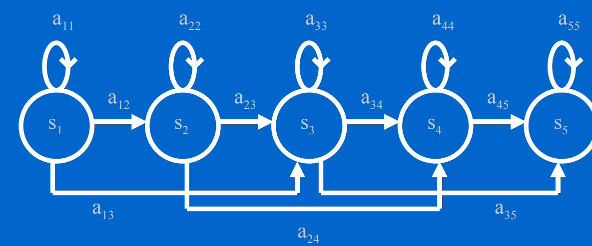
Automatic Speech Recognition

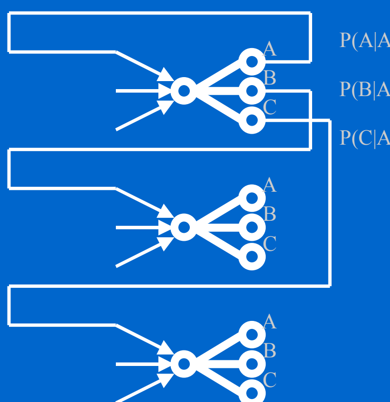
Dr.-Ing. Bernd Plannerer
plannerer@ieee.org

CHAPTER 5

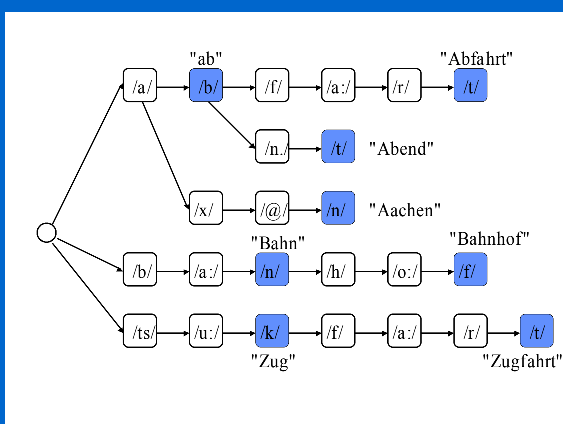




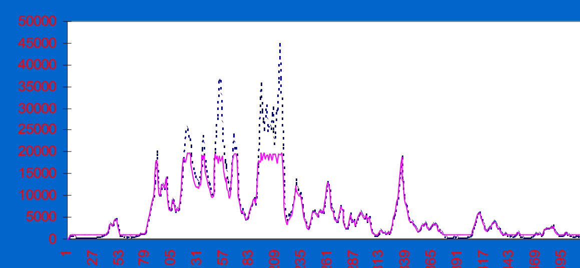
a_{11} a_{22} a_{33} a_{44} a_{55}
s_1 s_2 s_3 s_4 s_5
a_{12} a_{23} a_{34} a_{45}
a_{13} a_{24} a_{35}



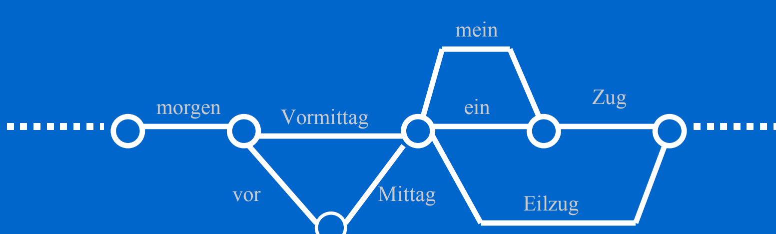
P(A|A)
P(B|A)
P(C|A)
A B C



"ab" "Abfahrt"
/a/ /b/ /f/ /a:/ /r/ /t/
/n./ /t/ "Abend"
/x/ /@/ /n/ "Aachen"
"Bahn" "Bahnhof"
/b/ /a:/ /n/ /h/ /o:/ /f/
/ts/ /u:/ /k/ /f/ /a:/ /r/ /t/
"Zug" "Zugfahrt"



50000 45000 40000 35000 30000 25000 20000 15000 10000 5000 0



mein
morgen
Vormittag
ein
Zug
vor
Mittag
Eilzug

Beam Search Algorithm

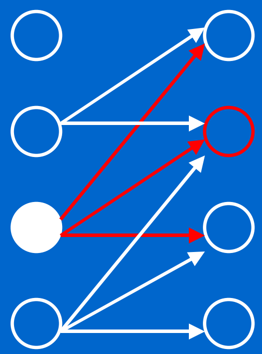
- Pruning
  - for every time frame t, search the globally optimum state hypothesis with score $D_{opt}(t)$
  - remove all state hypotheses with:
    $D(t) > D_{opt}(t) + H(t)$
  - H(t): Pruning-threshold (mostly, a constant value is chosen)

Beam Search Algorithm

- Dynamic construction of the search space
  - dynamic bookkeeping of state hypotheses , e.g., using lists
  - only activated states are held in memory
  - expansion from active states into successor states to achieve dynamic generation of the search space

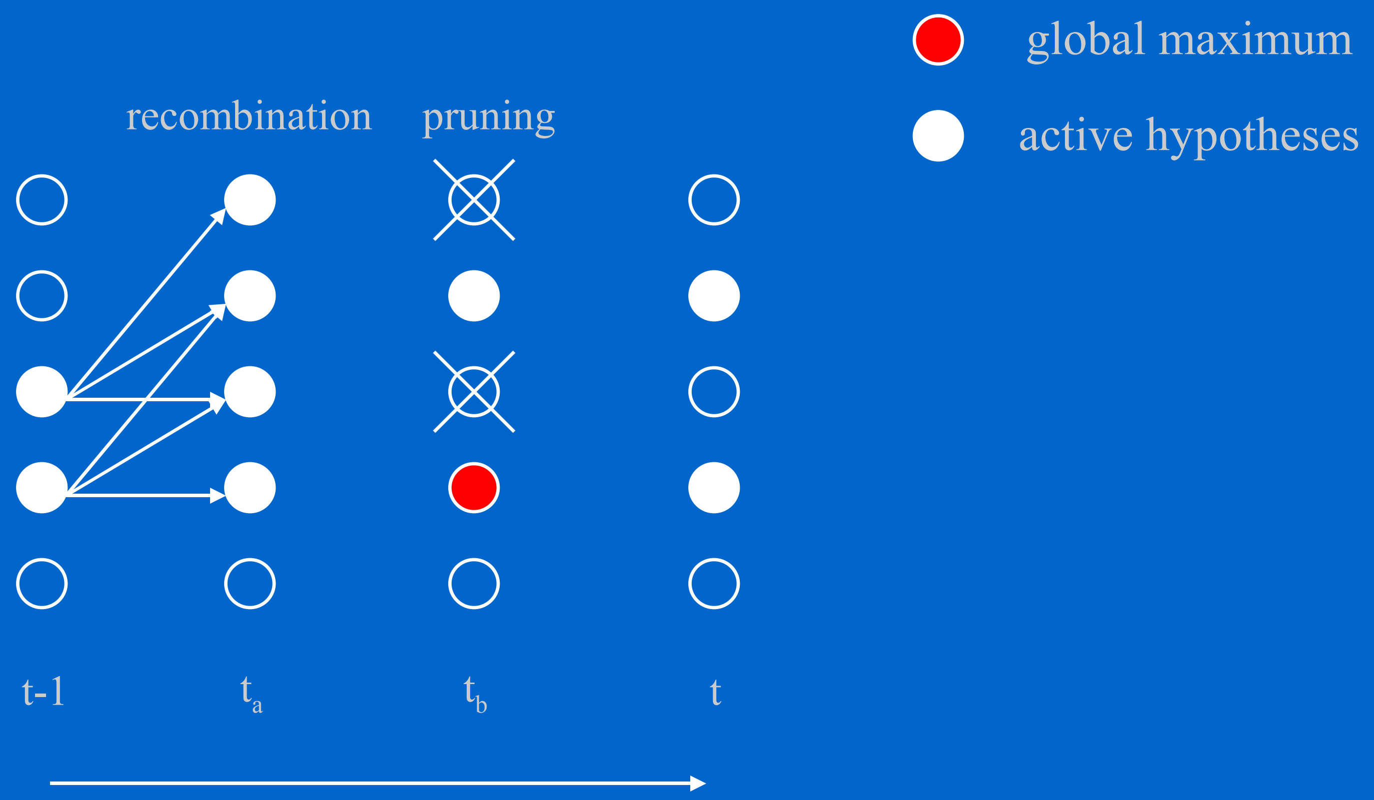
Beam Search Algorithm

- Data driven construction of the search space
  - forward oriented recombination
  - change score of successor, if new value would be better than the value already entered



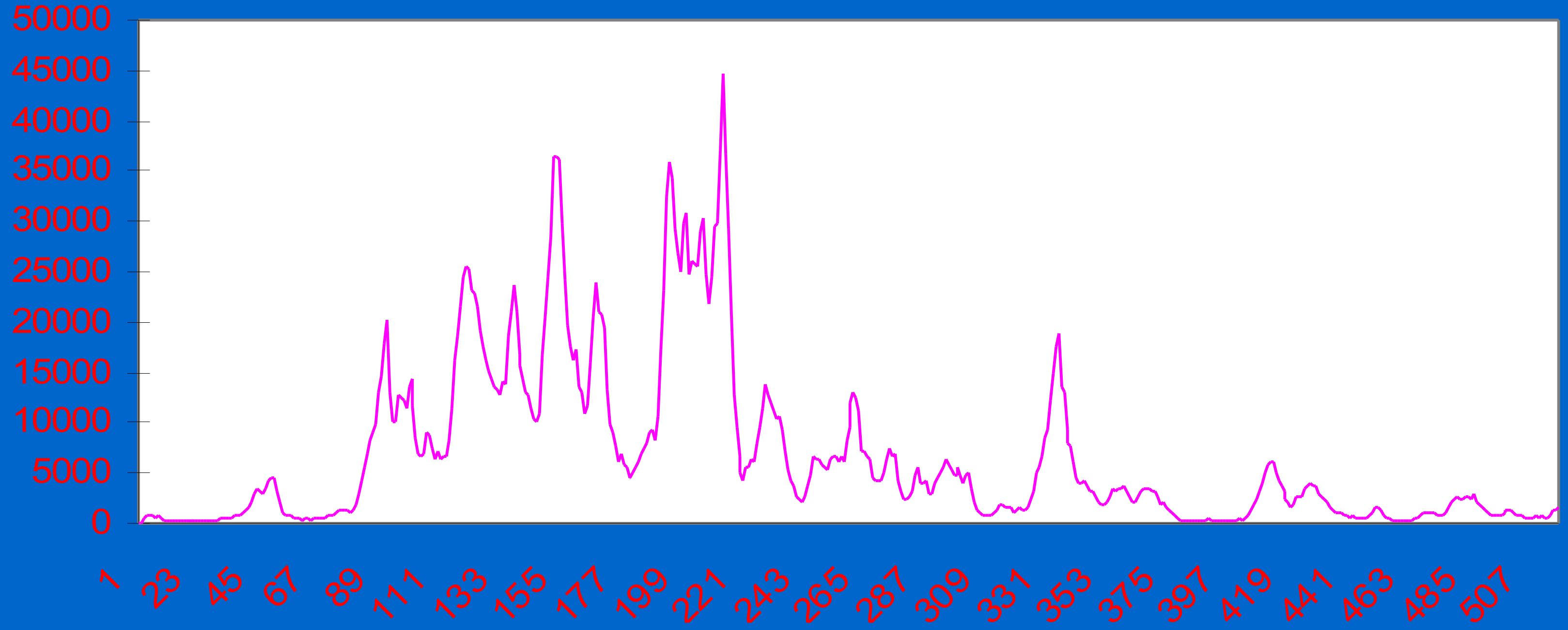
Beam Search Algorithm

- Principle of operation



Beam Search Algorithm

- Number of active state hypotheses during an utterance
  - „Guten Tag, wann geht morgen vormittag ein Zug nach Frankfurt?“

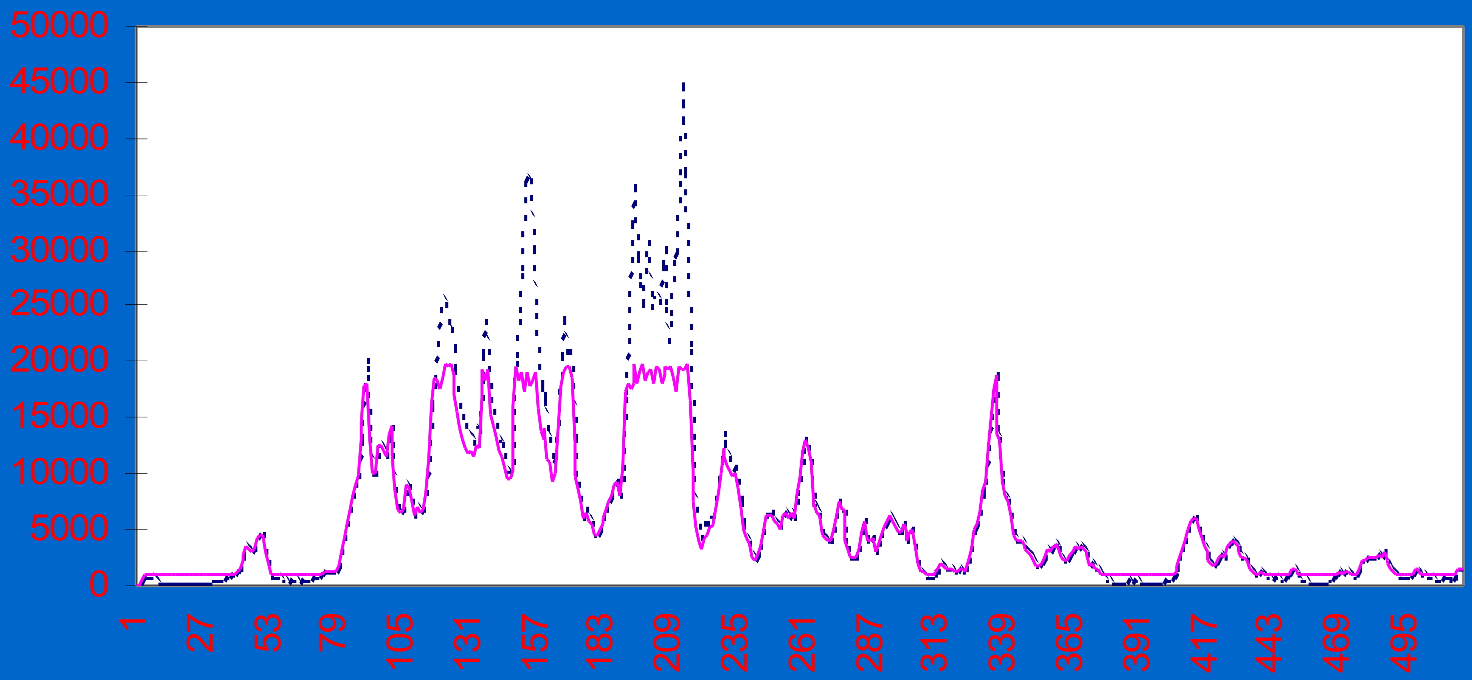


Beam Search Algorithm

- Histogram-Pruning
  - objective: Limit the number of active state hypotheses to a given number
  - define the pruning threshold necessary to keep the number of state hypotheses within th given limit.
  - this cannot be achieved by sorting
  - solution: A histogram of scores can easily be computed
  - use the histogram of scores to find the appropriate pruning threshold

Beam Search Algorithm

- Number of state hypotheses using histogram pruning

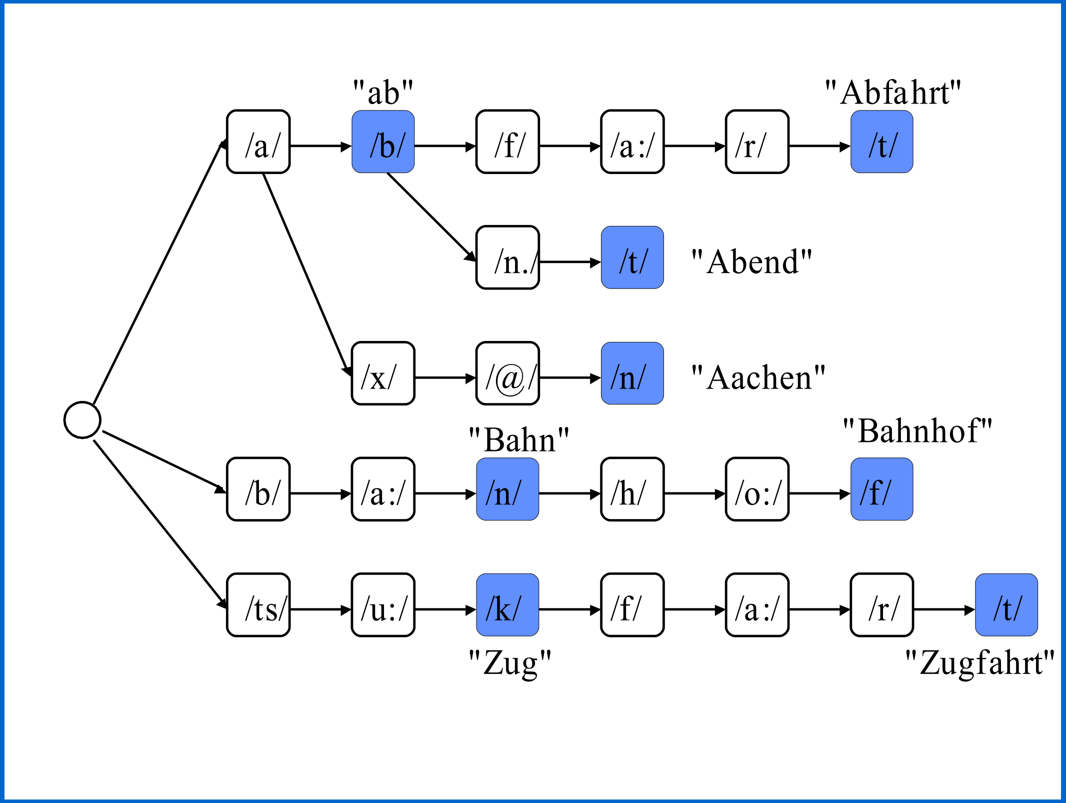


Beam Search Algorithm

- Further ways to limit the search space
  - lexicon tree
    - redundant computation of identical beginnings of words
    - organize vocabulary as a lexicon tree
    - leaves: Word ends
    - there are word ends within the branches (e.g. „ab“ und „Abfahrt“)
    - several leaves may correspond to one word and several words may belong to one leave

Beam Search Algorithm

- Lexicon tree

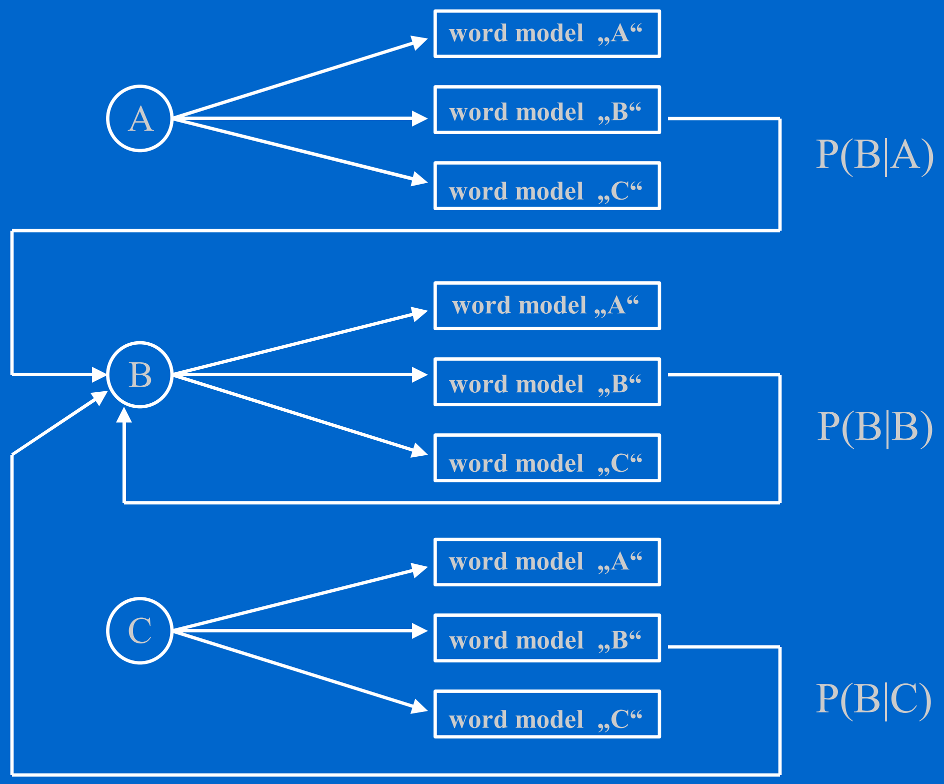


Beam Search Algorithm

- A speech recognizer using a lexicon tree
  - remember the structure with bigram LM: all paths generating the same word were recombined before entering the word model (early recombination)
  - Problem: Now all words start at the root. The word identity is only known at the leaves of the tree
  - Solution: Provide copies for each predecessor word (do not use early recombination)

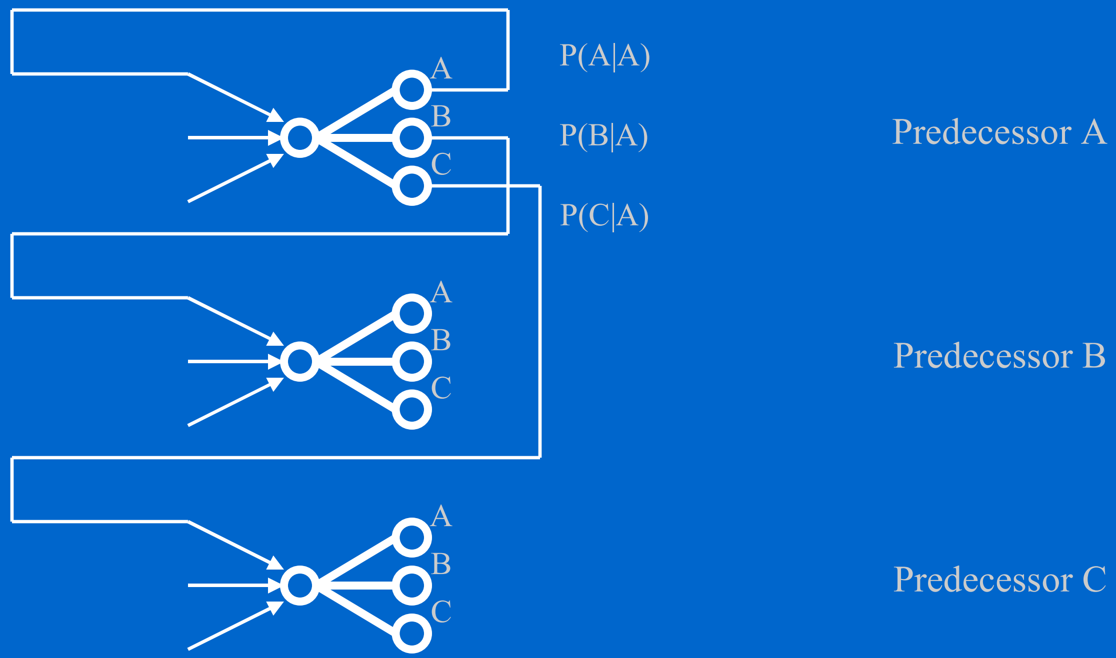
Beam Search Algorithm

- Structure using a bigram LM (principle)



Beam Search Algorithm

- Structure with lexicon tree



Beam Search Algorithm

- Given a vocabulary with V words, V instances of the lexicon tree are necessary for a bigram LM
- theoretically, the resulting search space is much bigger than for a linearly organized lexicon
- BUT: Using beam search, only a small fraction of the tree is activated
- therefore, a significant reduction of the active search space is possible

Beam Search Algorithm

- Reduction of state hypotheses: LM pruning
  - after recombination on the LM nodes,search for the optimum score $D_{LMopt}(t)$ among all tree roots
  - only those trees are (re-) started, which are below the pruning threshold H_{LM} :
    start up a tree, if $D_{LM}(t) < D_{LMopt}(t) + H_{LM}$
  - this reduces the number of active trees and therefore the number of active state hypotheses

N best search

- So far: Find the best word sequence using the Viterbi algorithm
- New: Find several acoustically similar sentence hypotheses for postprocessing with more complex LM´s
- Approaches
  - N best search
  - generation of word hypotheses graphs

N best search

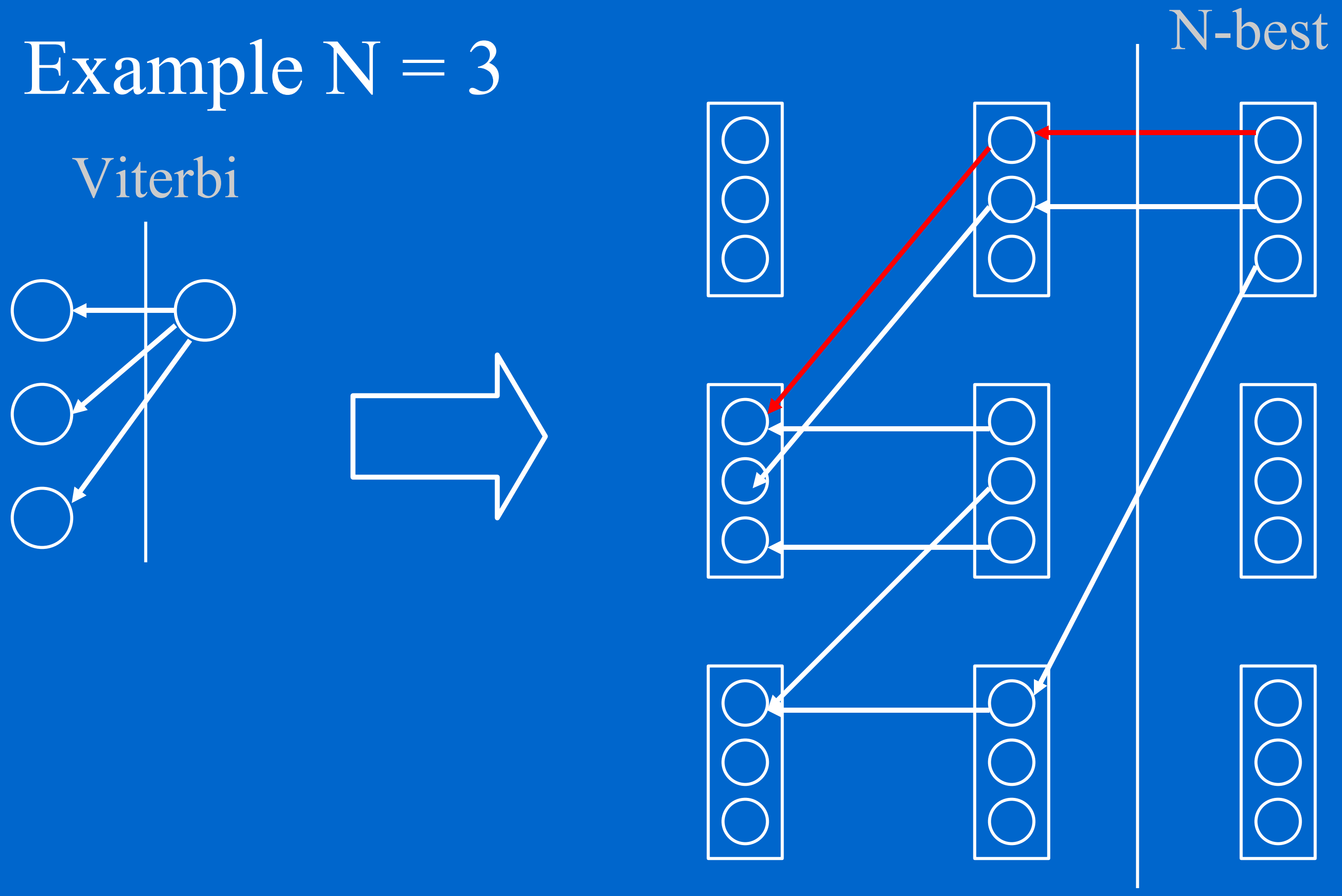
- Extended Viterbi search
  - Objective: Find the N best paths / state hypotheses
  - For each state the N best paths leading to this state have to be considered
  - a path which is not among the local N best path list can never be part of the globally N best paths

N best search

- DP recursion: For all predecessors, consider all locally N best partial paths and among those select the N best paths to the current point
- Boundary condition: Only select paths whose state sequence is not yet contained in the top-N best list (to avoid trivial solutions)

N best search

- Example N = 3



N best search

- N best search finds exactly the N best *state* sequences
- However, the difference among these are often only the time-alignment of the words
- We are interested in the N best *word* sequences !

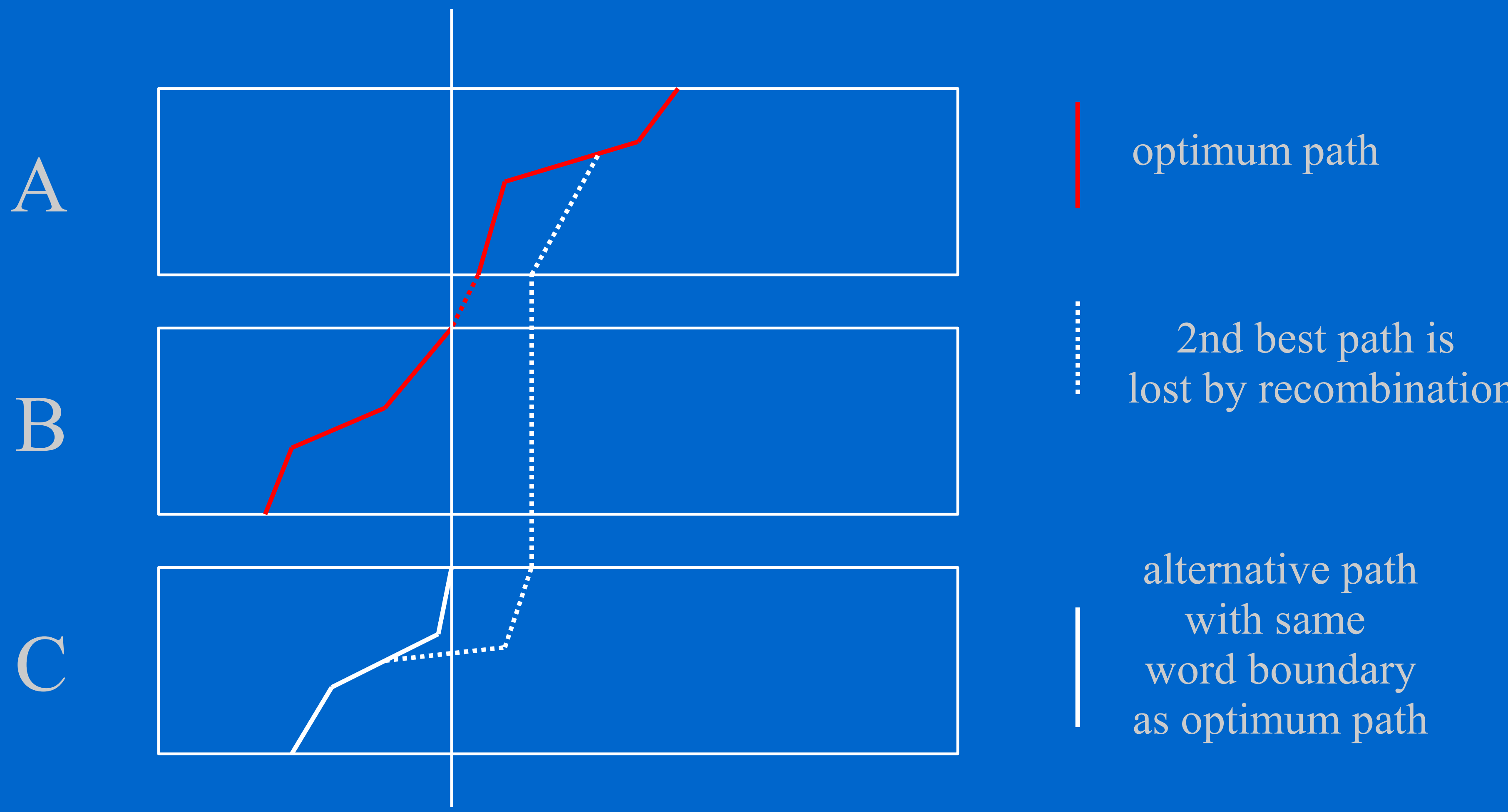
N best search

- Boundary condition during DP-recursion
  - only select paths whose corresponding word sequence is not yet contained in the N best list
  - this requires for each state the direct access to the word sequences defined by the N partial paths leading to the state. (e.g., tree structure for word sequences, states keep pointer to nodes in the tree)
  - High computational efforts !!!

N best search

- Approximation: The N best alternatives are only considered at the word boundaries
- Within the words, use standard DP recursion
- Therefore, only the possible word boundaries defined by the optimum path are considered

N best search (approximation)



N best search

- Summary
  - approximations are acceptable
  - N: Usually around 10
  - complex postprocessing of hypotheses on the word level
    - speech understanding
    - M gram LM
    - etc.

Word Graph

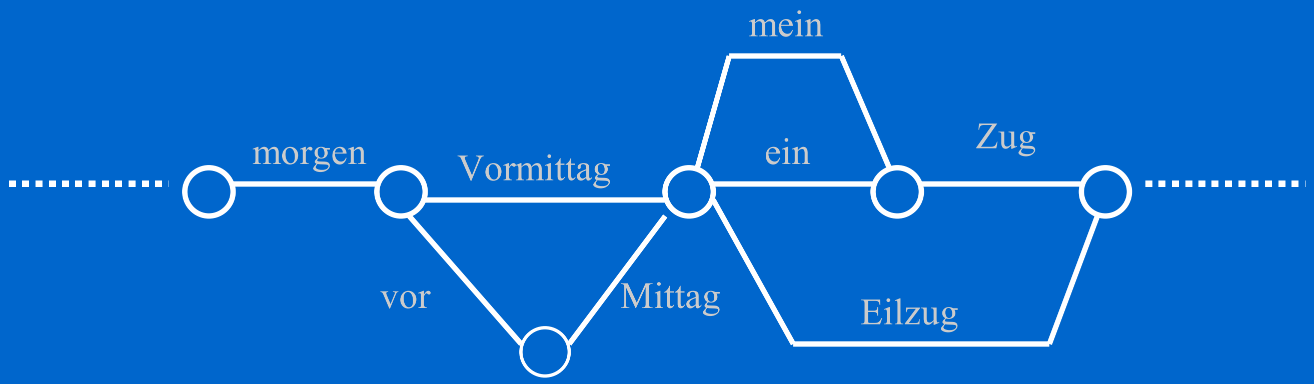
- Problem of N best search: All sequences of words differ often only in the permutations of the most ambiguous words
- Solution: Representation of alternative paths using a graph of words

Word Graph

- A word graph may contain many paths (sentence hypotheses)
- It is a compact representation of the sentence hypotheses
- Can be used as interface between acoustic phonetic and semantic speech processing

Word Graph

- Example



Word Graph

- Typical format
  - node := <A E word score ta te infostring> where:
  - A: node where word begins
  - B: node whrer word ends
  - wort: word hypothesis
  - ta: start frame
  - te: end frame
  - infostring: additional information

Word Graph

- Example
  - (comment ....)
    BEGIN_LATTICE

1	2	#PAUSE#	11.5	1	110
2	3	ich	7.05	111	125
3	4	#NIB#	7.45	126	194
4	5	muss_am	8	195	208
4	6	kam	8.2	195	210

(...)
END_LATTICE

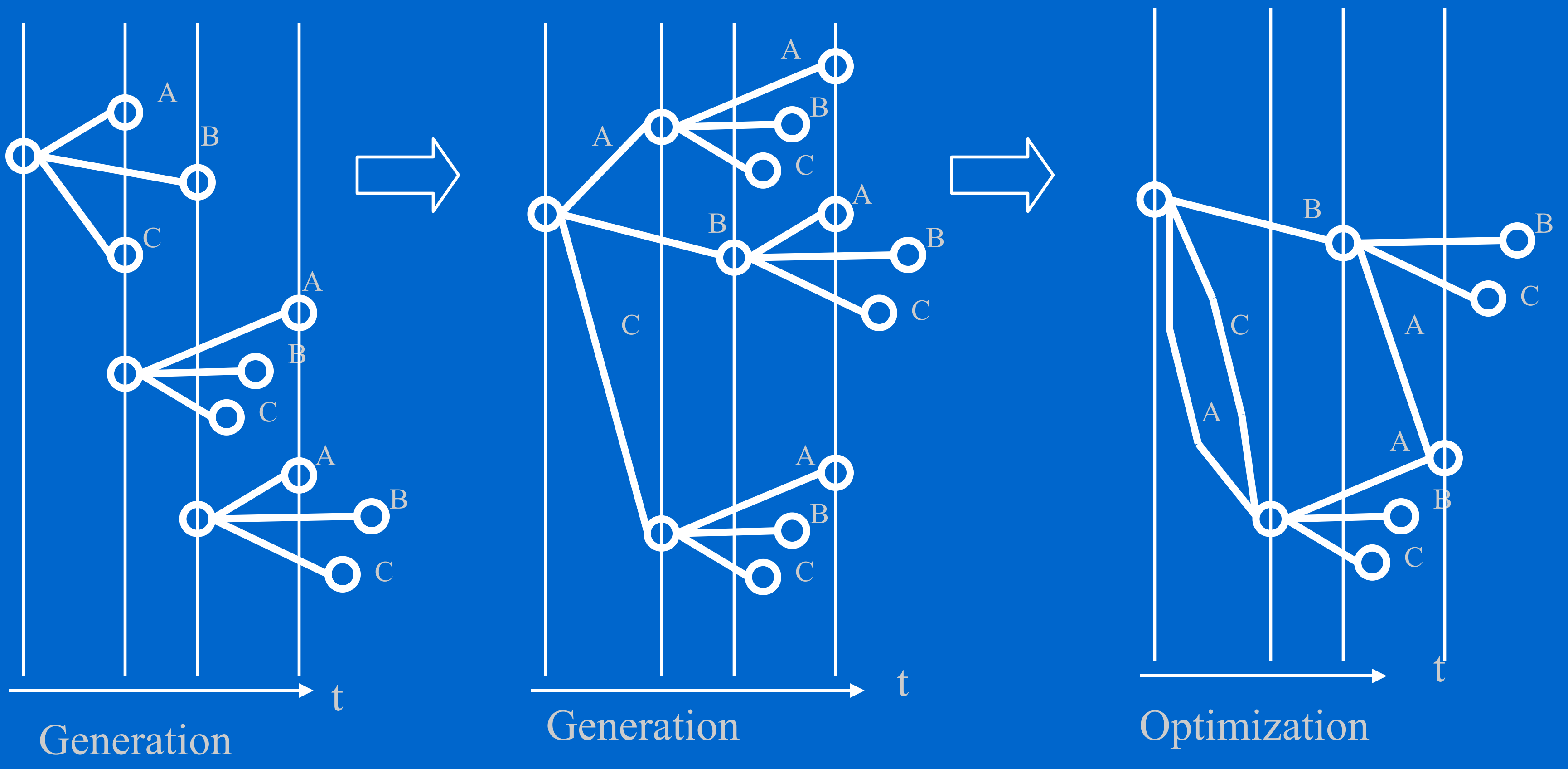
Word Graph

- Generation of word graphs
  - two stage approach
    - generation of word hypotheses
    - optimization of word graph
  - generation of word hypotheses
    - for each time frame t, start a new lexicon tree
    - for all time frames t, for all leaves, compute: word, startframe, endframe (=current time frame), and score

Word Graph

– Reduction of search space
  - pruning of state hypotheses relatively to the global optimum
  - a new tree is started only every 20 or 30 ms
– Word graph optimization
  - merge nodes with identical time index
  - merge subgraphs with identical start and stop time frames and words. All successors of the graph with the lower score will be appended to the graph with the better score.

Word Graph

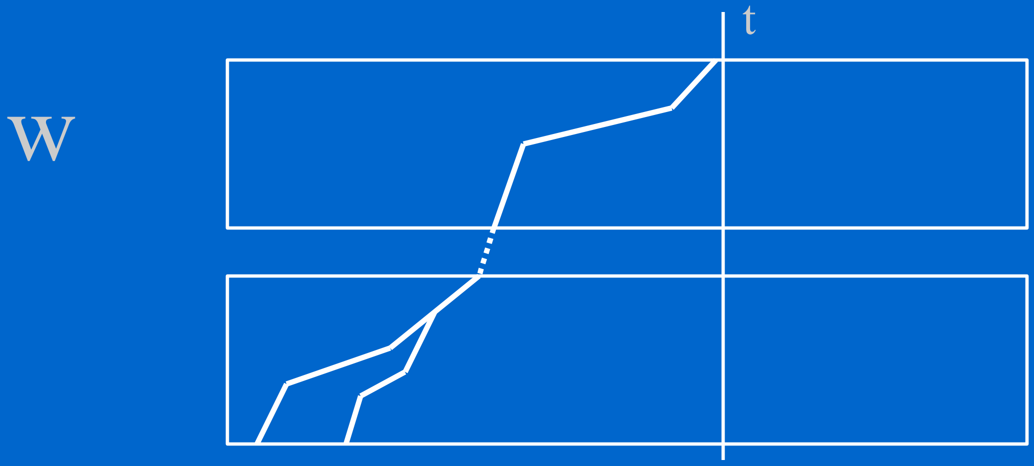


A
B
C
A
B
C
A
B
C
t
Generation
A
A
B
C
B
A
B
C
C
A
B
C
t
Generation
B
B
C
C
A
A
A
B
C
t
Optimization

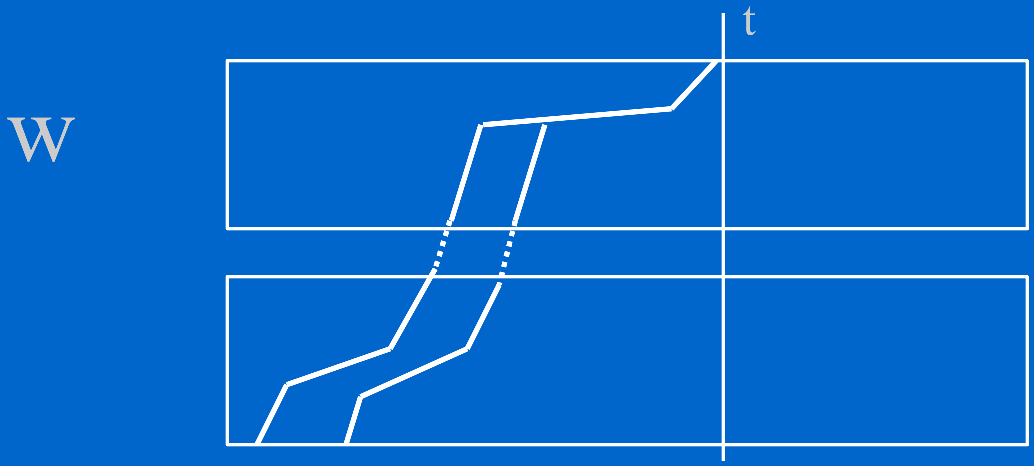
Word Graph

- Alternative approach
  - Task: For a given word*end* w at time t, compute a limited number of possible predecessor words
  - Problem: *Begin* of w depends on the predecessor words (and possibly on their predecessors, too)

Word Graph



Begin of word w depends only on its pedecessor



Begin of word w depends not only on its predecessor

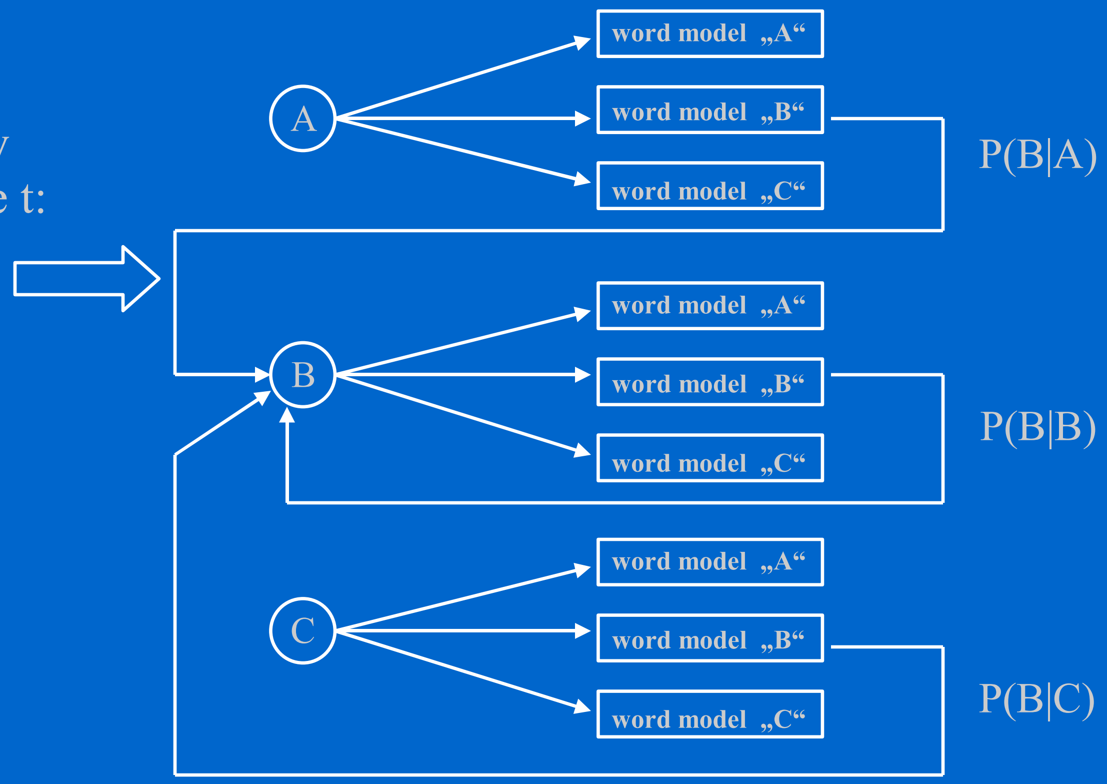
Word Graph

- word pair approximation
  - Assumption: word boundary depends only on predecessor word
  - for every time frame t consider all word pairs (w,v) (beam search)
  - for every time frame t and every word pair (w,v) store the word boundary between w and v and the word score for word w
  - At the end of the utterance, generate the word graph via backtracking

Word Graph

Generation of word graphs using lexicon trees and bigram LM

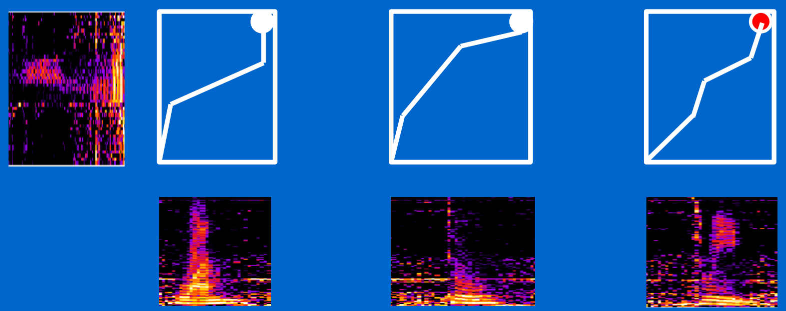
For every
time frame t:
$\tau(B,A,t)$
$s(B,t,\tau)$

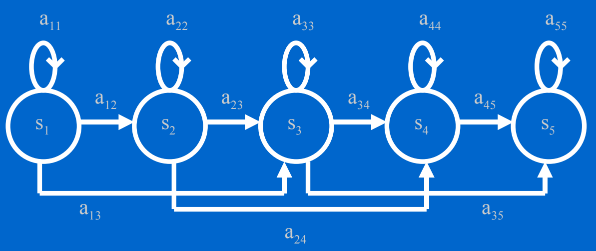


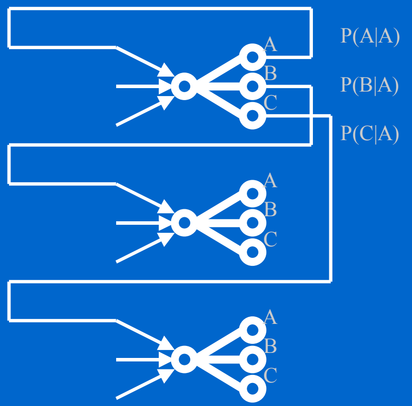
Wort Graph

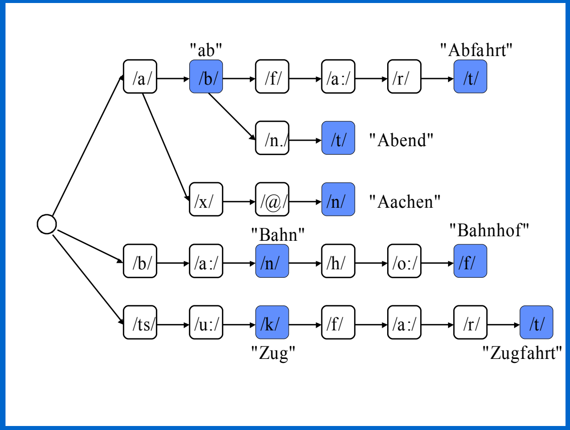
- Word Pair Approximation yields good results
- Easy modification of existing system
- Quality of the recognizer is not the recognition rate, but the recognition rate related to the graph density (number of edges per spoken word)

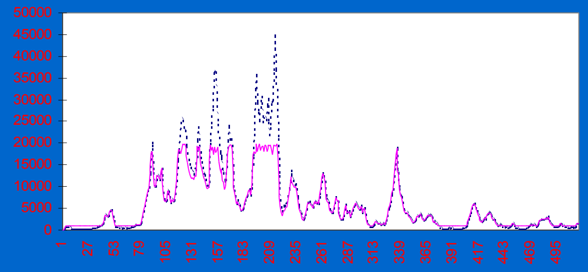
THE END

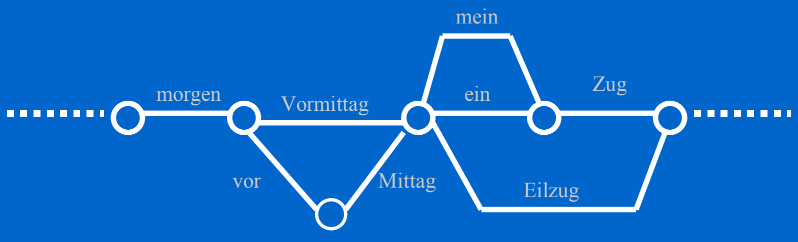



a_{11} a_{22} a_{33} a_{44} a_{55}
s_1 s_2 s_3 s_4 s_5
a_{12} a_{23} a_{34} a_{45}
a_{13} a_{24} a_{35}


A
B
C
P(A|A)
P(B|A)
P(C|A)


"ab"
"Abfahrt"
/a/ /b/ /f/ /a:/ /r/ /t/
/n./ /t/ "Abend"
/x/ /@/ /n/ "Aachen"
"Bahn"
"Bahnhof"
/b/ /a:/ /n/ /h/ /o:/ /f/
/ts/ /u:/ /k/ /f/ /a:/ /r/ /t/
"Zug"
"Zugfahrt"


50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
1 27 53 79 105 131 157 183 209 235 261 287 313 339 365 391 417 443 469 495


mein
morgen
Vormittag
ein
Zug
vor
Mittag
Eilzug