Specialization Module

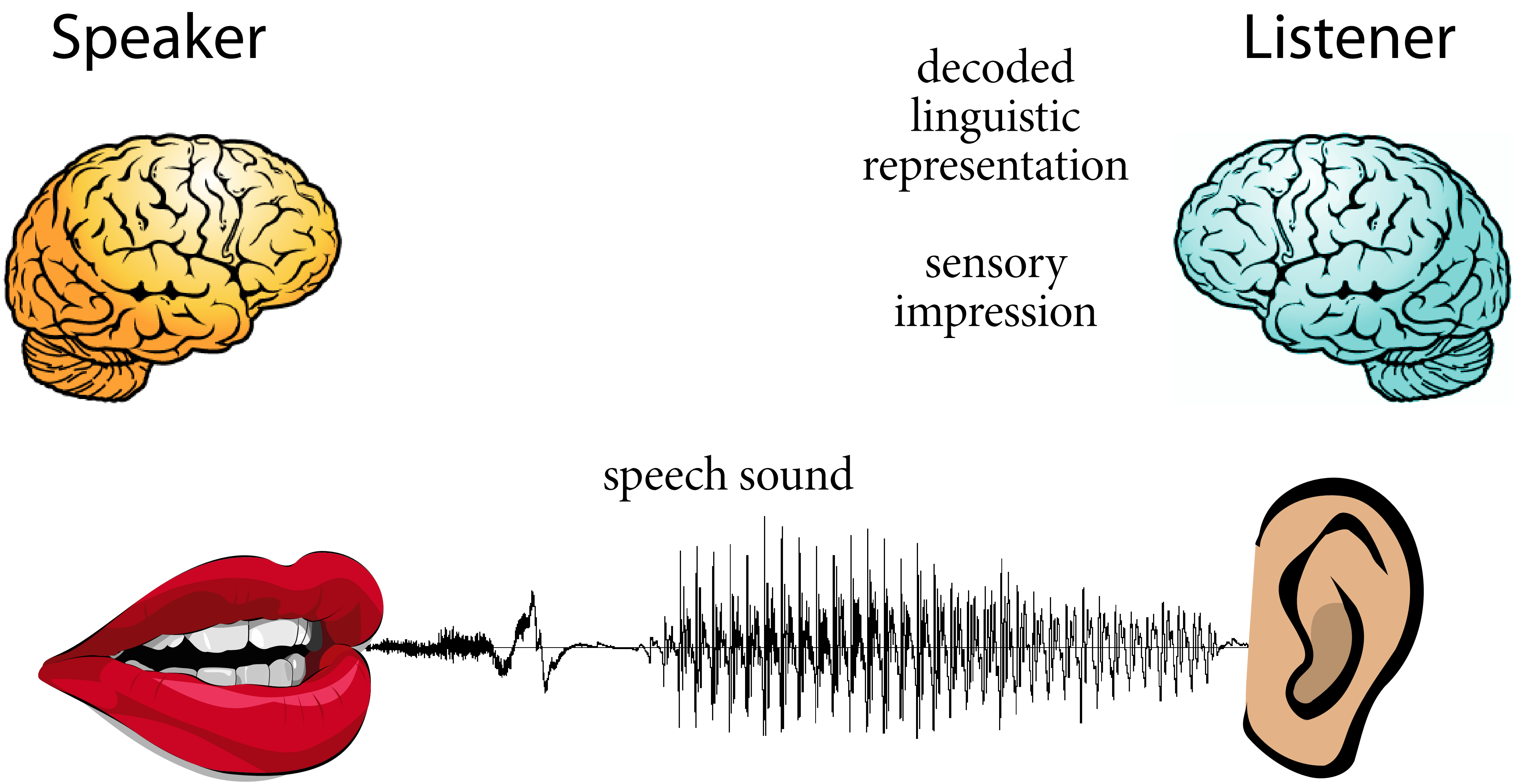
Speech Technology

Timo Baumann
baumann@informatik.uni-hamburg.de

UNIVERSITÄT HAMBURG, DEPARTMENT OF INFORMATICS
NATURAL LANGUAGE SYSTEMS GROUP

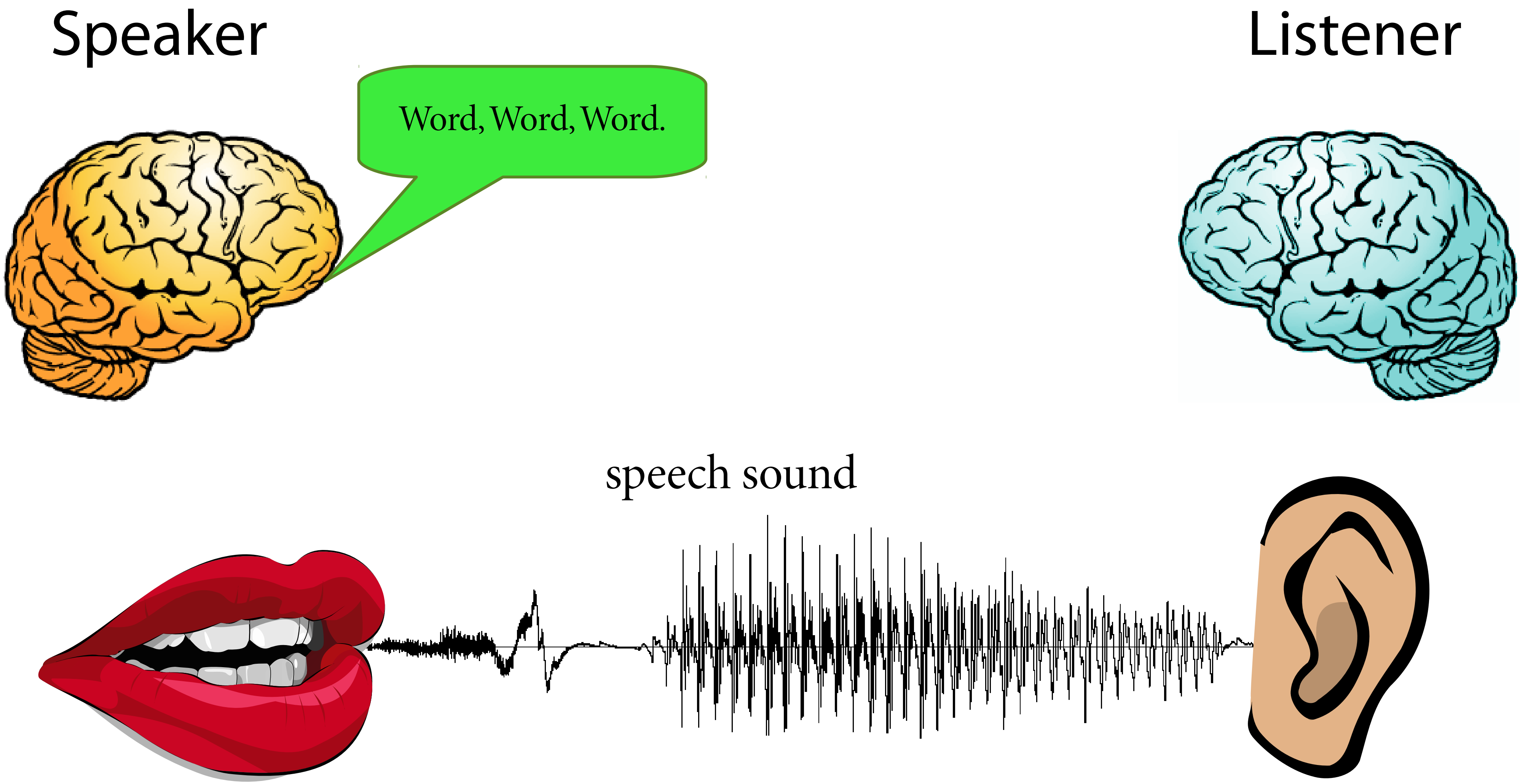
Speech Recognition

The Chain Model of Communication



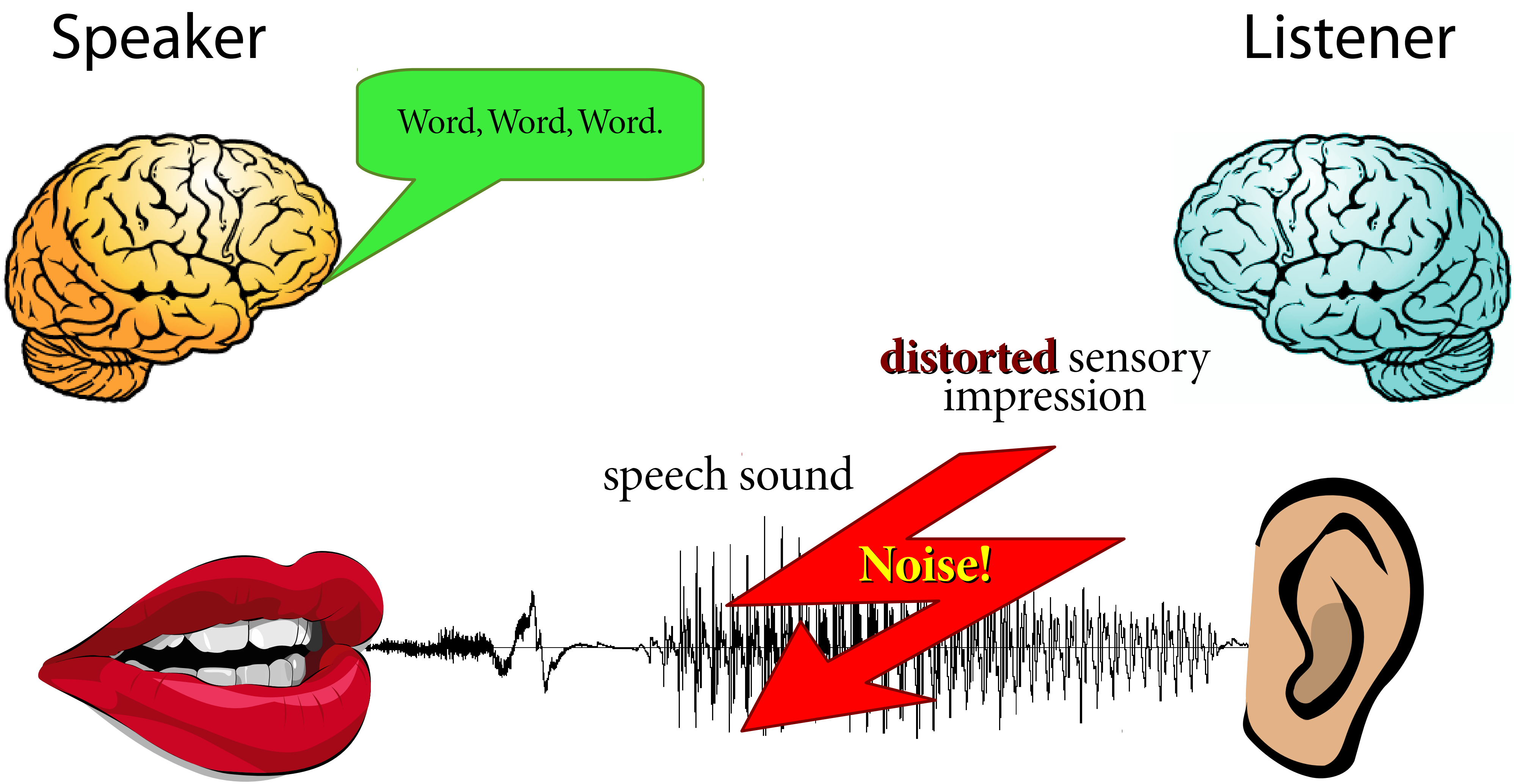
derived from: Pétursson/Neppert: Elementarbuch der Phonetik, 1996.

Noisy-Channel Model



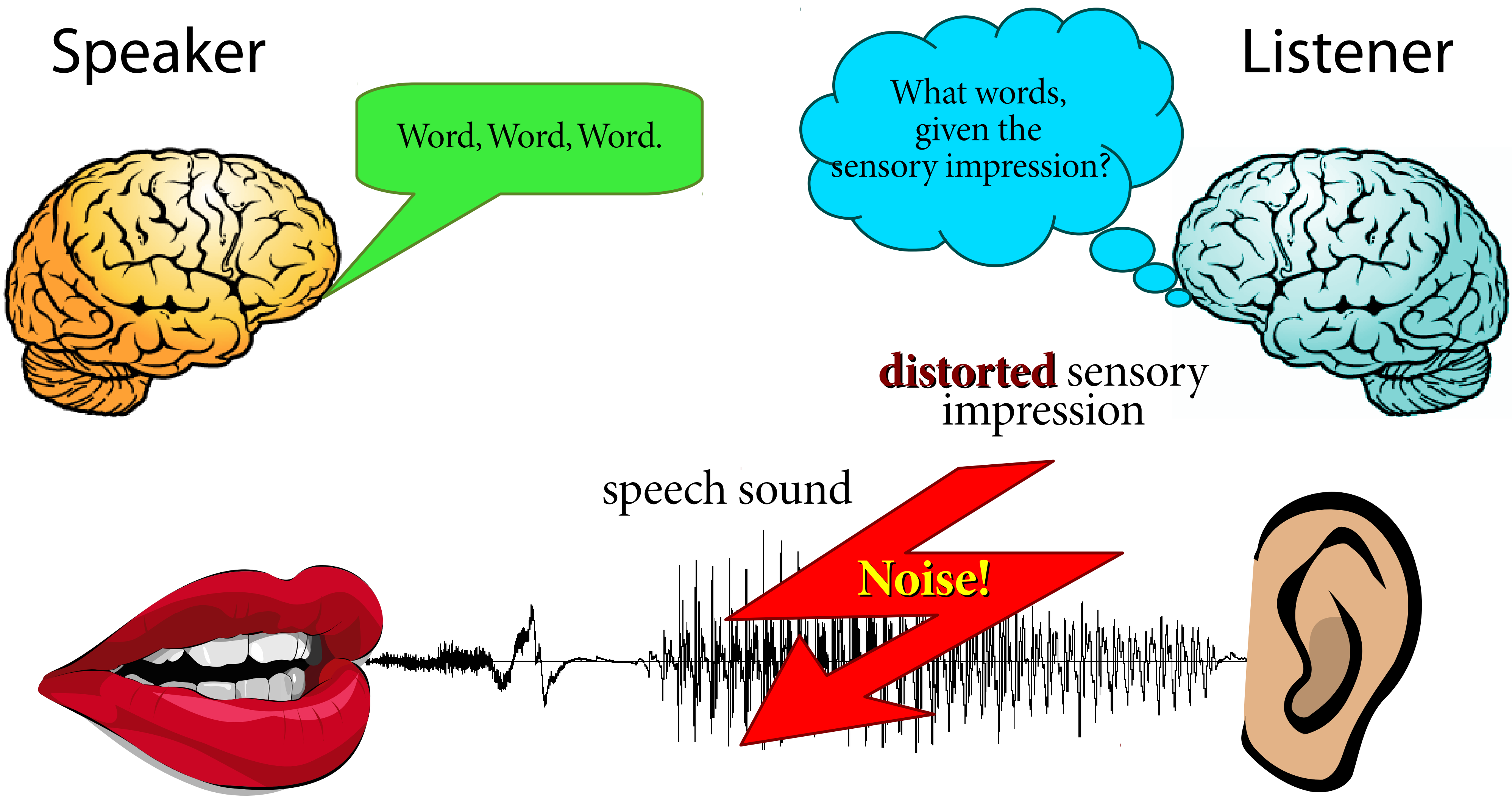
C. Shannon, W. Weaver: The mathematical theory of communication, 1949.

Noisy-Channel Model



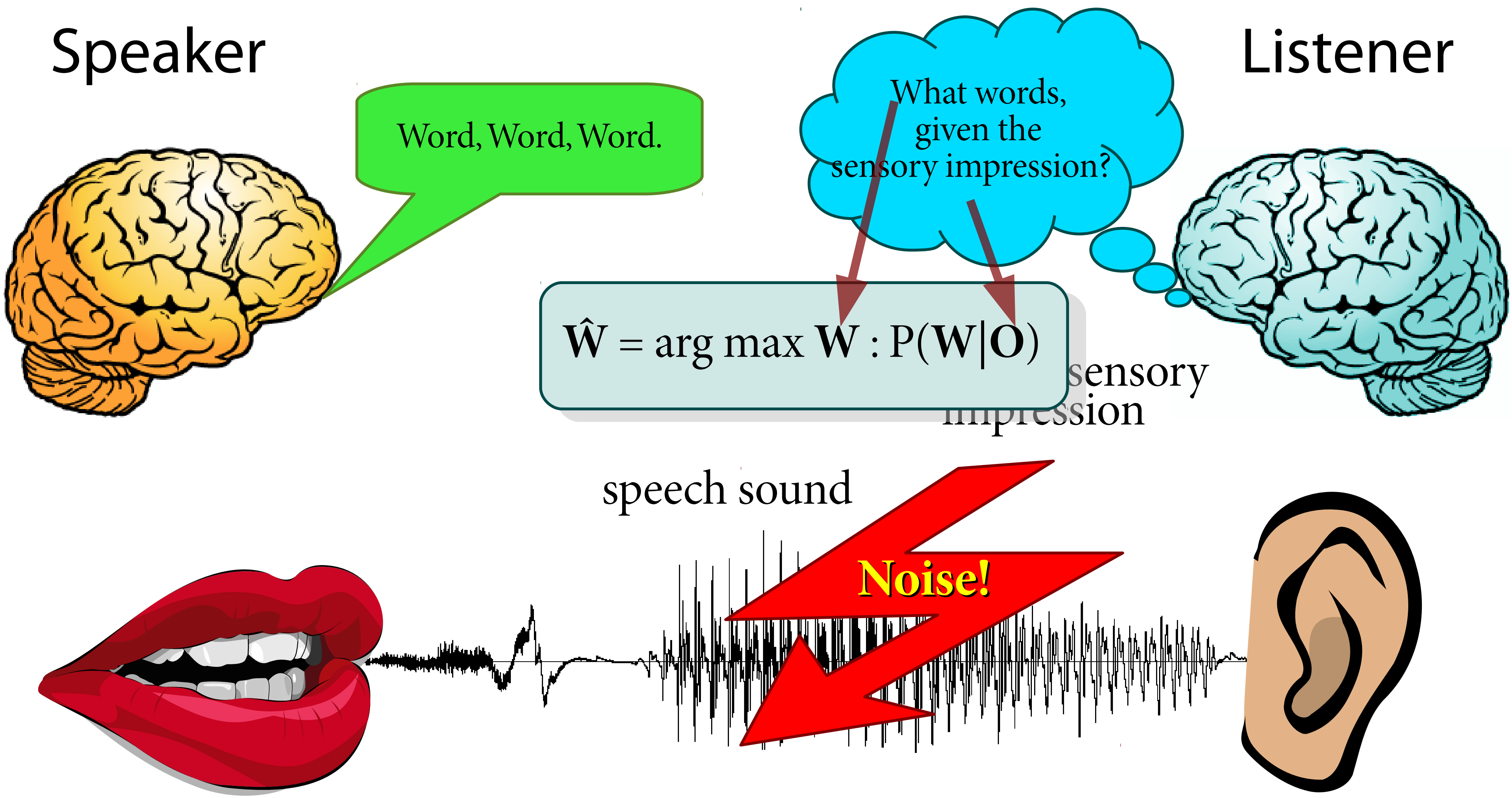
C. Shannon, W. Weaver: The mathematical theory of communication, 1949.

Noisy-Channel Model



C. Shannon, W. Weaver: The mathematical theory of communication, 1949.

Noisy-Channel Model



C. Shannon, W. Weaver: The mathematical theory of communication, 1949.

The Speech Recognition Task

- Given a language $\mathcal{L}$
- and a sensory impression (observation) $\mathbf{O}$
  - sequence of (MFCC) parameters over sliding windows
- we search $\hat{\mathbf{W}}$ in $\mathcal{L}$ such that
  - $\hat{\mathbf{W}} = \arg\max \mathbf{W} : P(\mathbf{W}|\mathbf{O})$
    the *most likely* word sequence given the observation
  - maximum-likelihood principle

- how to determine $P(W|O)$?
- how to organize the search?

Bayes' Rule

Given conditional probabilities A and B:

- $P(A|B) = \dfrac{P(B|A) \times P(A)}{P(B)}$

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

- our formula uses arg max $\rightarrow$ the denominator P(B) does not matter, we can ignore it:
- $P(A|B) \sim P(B|A) \times P(A)$

The Speech Recognition Task (II)

- $\hat{W} = \arg\max W : P(W|O)$

- applying Bayes' rule: $P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$
  - $\hat{W} = \arg\max W : \mathbf{P(O|W)} \times \mathbf{P(W)}$
  - $P(O|W)$: **acoustic model**
    - observation likelihood given a word sequence
    - *What do words sound like?*
  - $P(W)$: **language model**
    - a priori probability for word sequences
    - *What word sequences are likely?*

Words or Phonemes?

- acoustics primarily depend on phonemes, not on words
- words have an internal structure (cmp. last week)
  - this was disregarded in early approaches e.g. for single-word recognition. Hence it's almost always ignored in descriptions.
- thus we should rather estimate $P(O|Ph)$, instead of $P(O|W)$

- we need an additional conversion step that relates words to phoneme sequences $P(Ph|W)$

The Lexicon – linking acoustic and language models

- thus, we get:

$$\hat{W} = \arg\max W : \mathbf{P(O|Ph)} \times \mathbf{P(Ph|W)} \times \mathbf{P(W)}$$

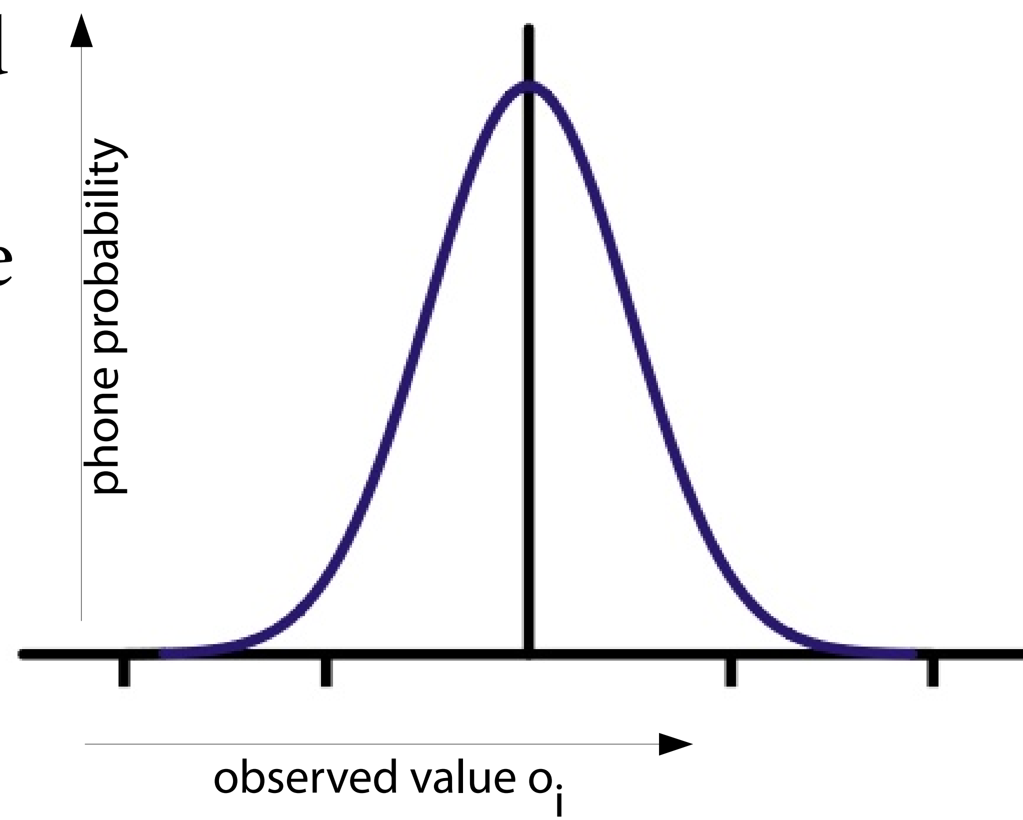
- simple lexicons map each word to a phone sequence
- extensions:
  - pronunciation variants for words
  - adapt lexicon at runtime to speaker's pronunciation (tempo, context, dialect, …)
  - rule-based grapheme-to-phoneme conversion (model phonological rules; may include weighted variants)

The Speech Recognition Task (III)

- $\hat{W} = \arg\max W : \mathbf{P(O|Ph)} \times \mathbf{P(Ph|W)} \times \mathbf{P(W)}$
  - we'll discuss **P(W)** next week. The simplest form could be a list of possible sentences or a simple context-free grammar
  - we skip **P(Ph|W)** (will be dealt with in one of the labs)
- the **acoustic model P(O|Ph)**
  - assesses the observed speech signal wrt. a phoneme hypothesis
  - describes the signal by sequence of acoustic features
- $\mathbf{O} = (o_1, o_2, o_3, o_4, \ldots o_{tmax})$,
  with o_i being the feature vectors (e.g. MFCCs) based on short stretches of audio (previous lecture)

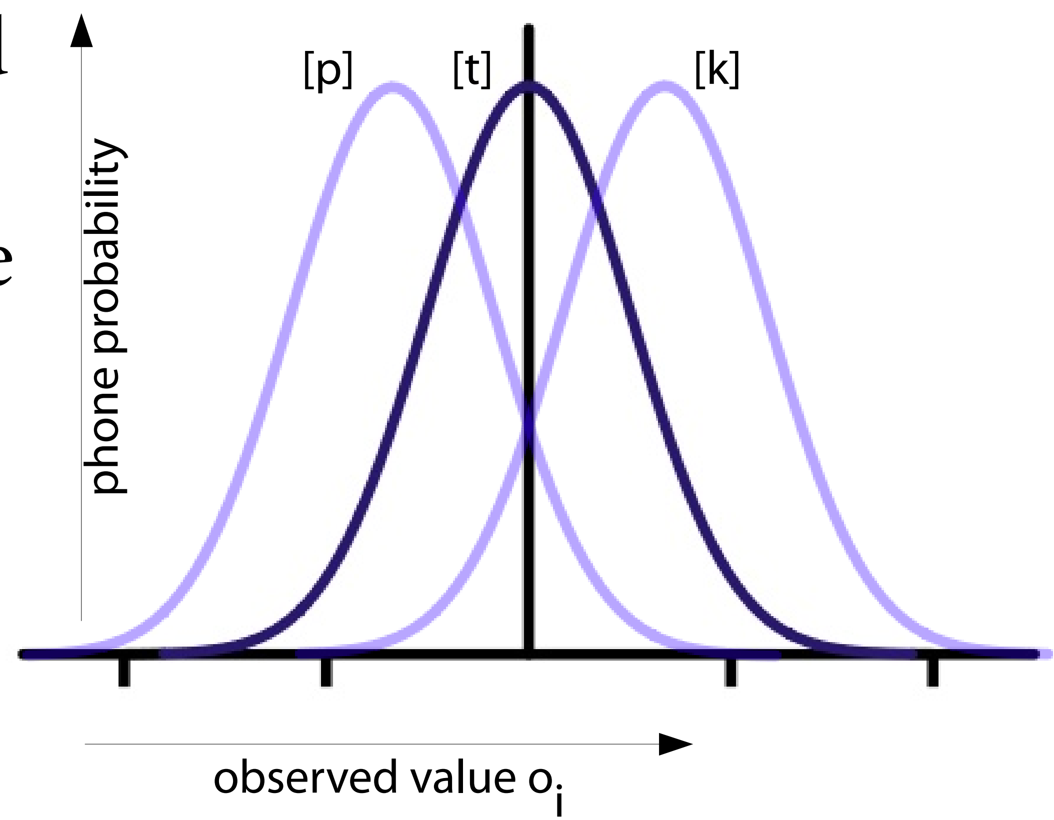
From Observations to Probabilities

- each phone model is associated with an acceptance function to map an observation o_i to a probability
- often based on Gaussian distributions:
  - just two parameters: μ and σ
- probability can be computed based on observed value
- o_i could belong to any phone
  $\rightarrow$ compute distribution
  for all phones



From Observations to Probabilities

- each phone model is associated with an acceptance function to map an observation o_i to a probability
- often based on Gaussian distributions:
  - just two parameters: μ and σ
- probability can be computed based on observed value
- o_i could belong to any phone
  → compute distribution
  for all phones



Phone Models

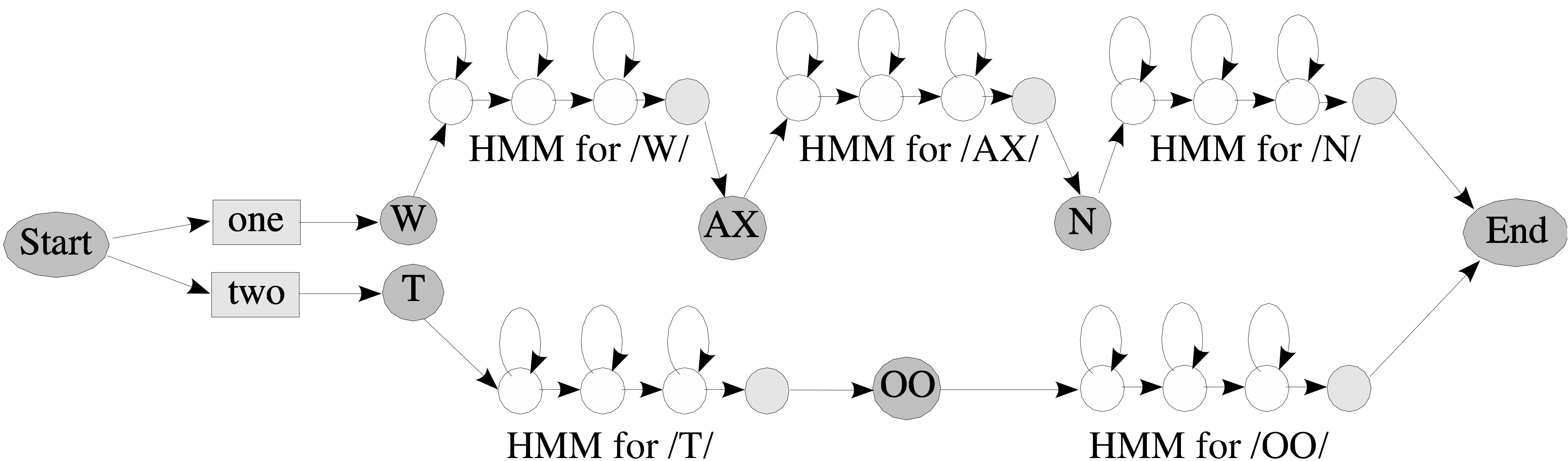
- usually, a speech sound will last longer than one observation
  - but how long exactly?
- we model this using transition probabilities
  - phone(states) differ in *likely* duration
- transition probabilities + observation probabilities
  - … plus Lexicon plus Language Model …

    → Hidden Markov Models to the rescue!

Hidden-Markov Models

- unifying model for the speech recognition process
- Markov assumption: we can model the future without looking too far into the past
  - no need for full history to differentiate next observation, the present state is sufficient
- we can construct a state-graph where each state contains the full (relevant) history for determining the next state in the graph

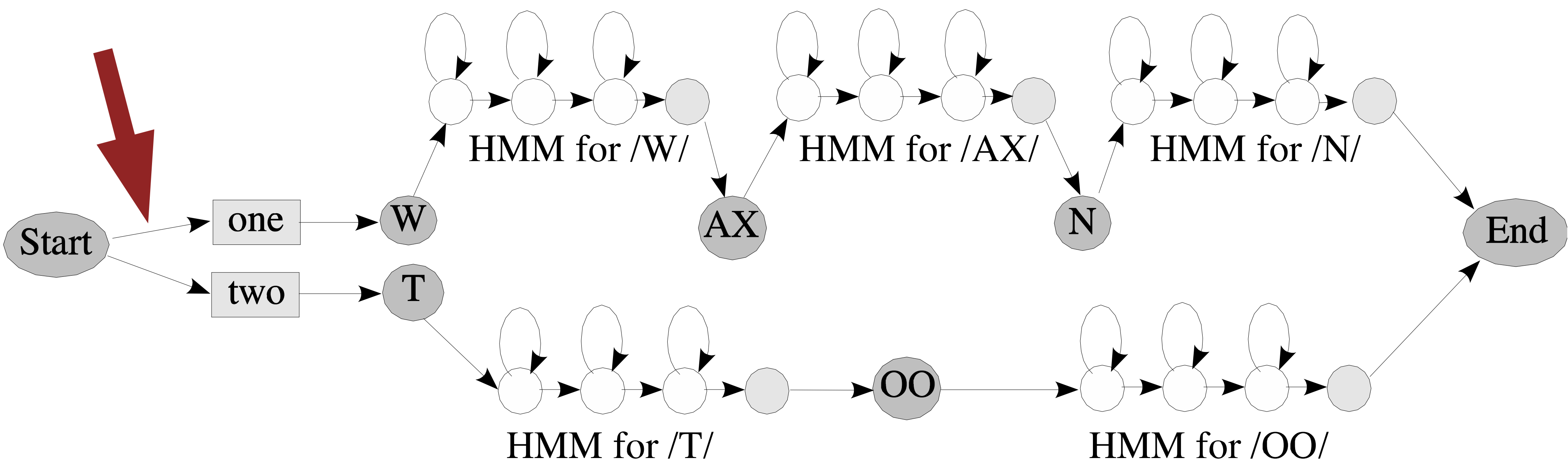
The Search Graph



built from language model (here: S→“one”|“two”), lexicon (one→/W AX N/, two→/T OO/), and phone models

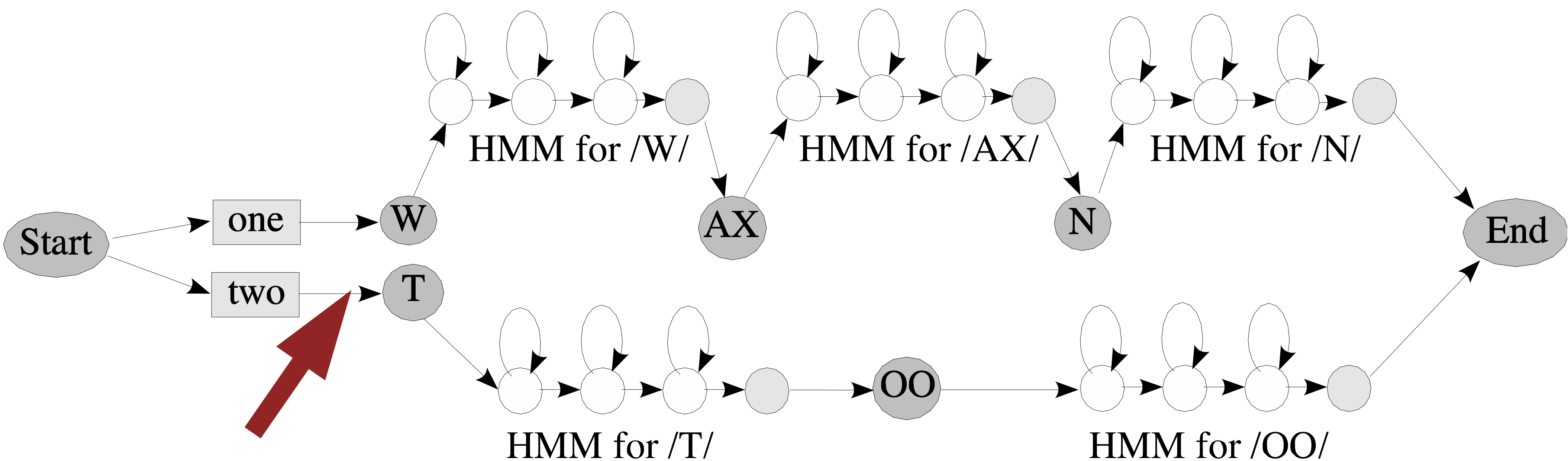
aus: Walker et al., Sphinx-4: A Flexible Open Source Framework for SR, 2004.

The Search Graph



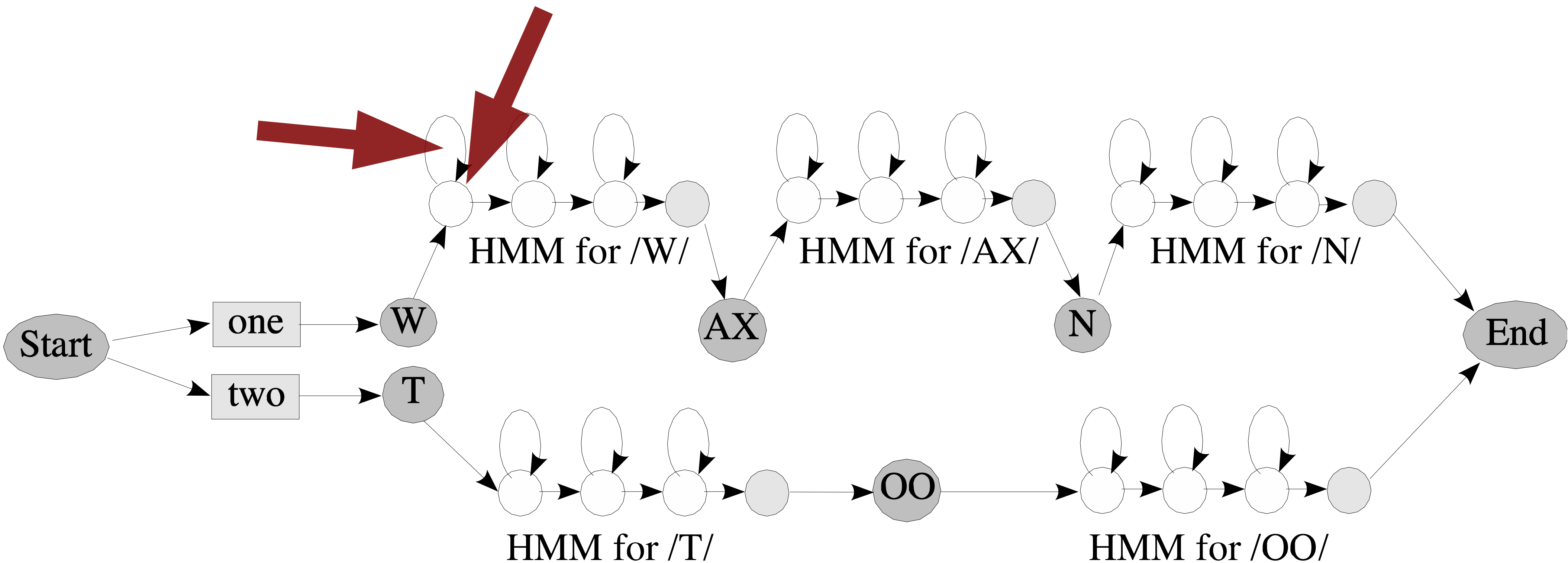
- transition probabilities from language model

The Search Graph



- expansion to sounds from the lexicon

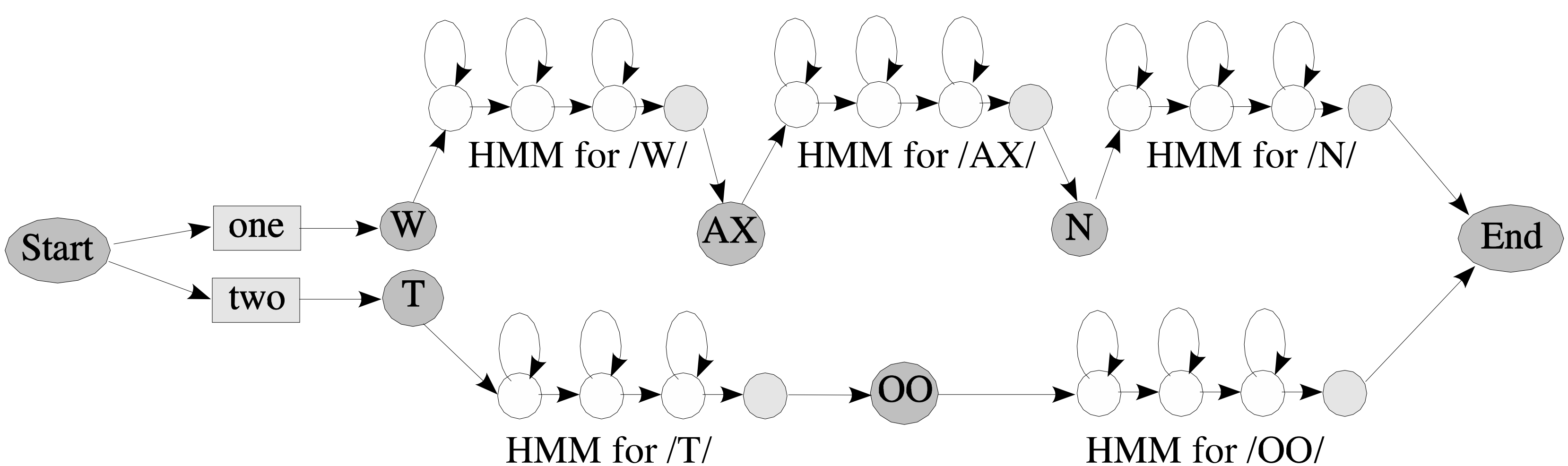
The Search Graph



- acoustic model: transition probabilities (A) and emission/observation probabilities (B) all we need to do is find the most likely path through the graph

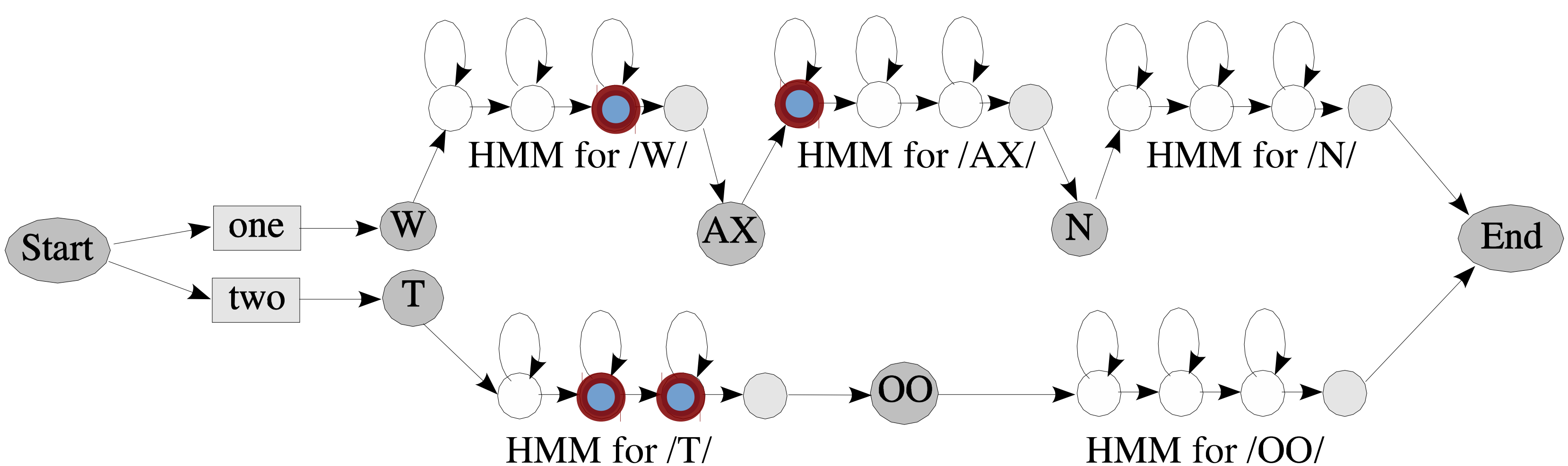
Decoding: Searching the Graph

- we're looking for the path in the graph that
  - distributes the observations to (emitting) phone states
  - while keeping costs at a minimum
    (identical to the highest probability)



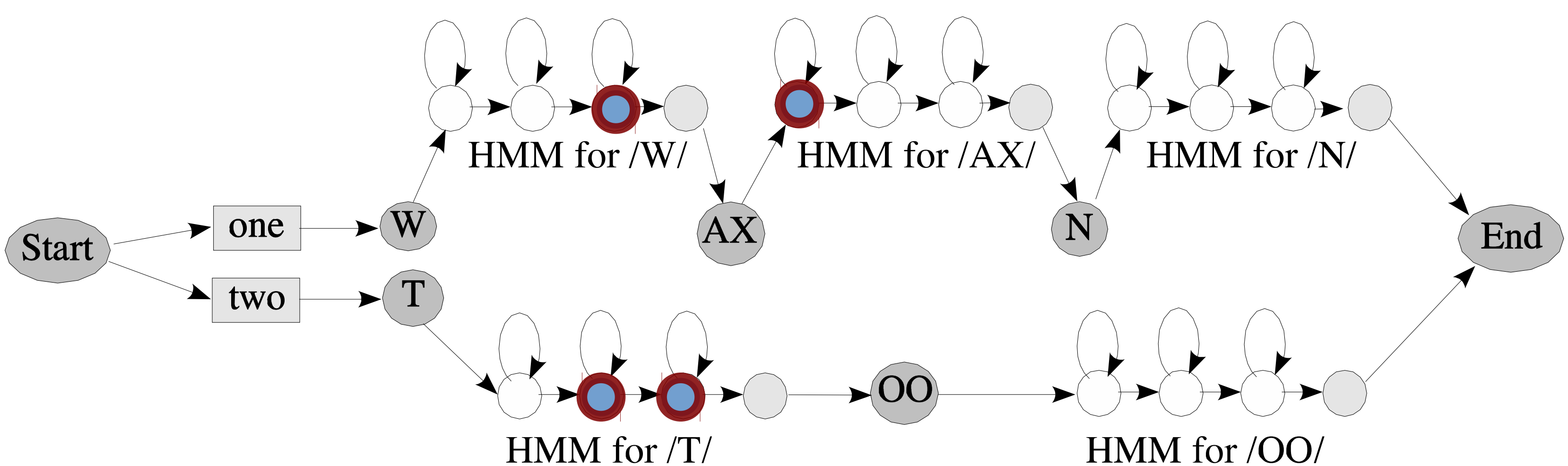
Token-Pass Algorithm: Basic Idea

- time-synchronous search of the observations
  - at every point in time, keep a number of hypotheses, that are represented each by a token
  - generate new tokens from old tokens in every step
  - the winner: best token that reaches the final state in the end



Token-Pass Algorithm: Basic Idea

- every *token*
  - stores the current state in the graph
  - the sum of costs incurred so far
    - possibly differentiated for LM and AM costs
  - details to preceding token (necessary to recover path)

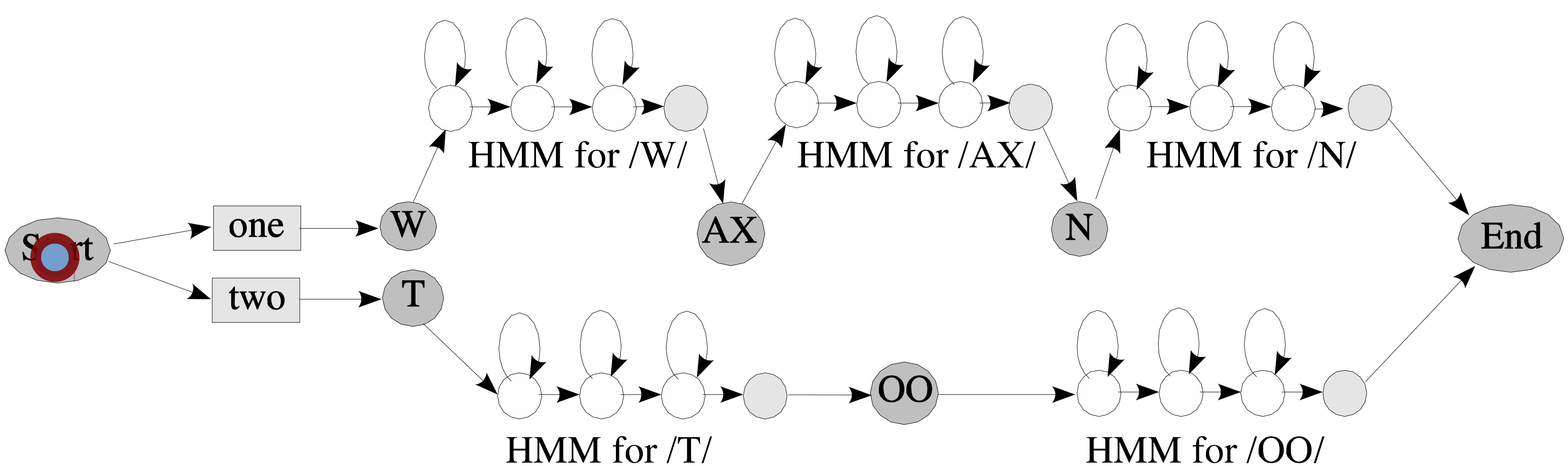


Token-Pass Algorithm en détail

- start with an empty token in the initial state
- for all tokens
  - take the next observation
  - generate all successor tokens from the current state
  - add costs (transition, observation)
  - of all token that are in one state keep only the best token
    - principle of *dynamic programming*: the best path leading here is the only relevant path in the globally best path

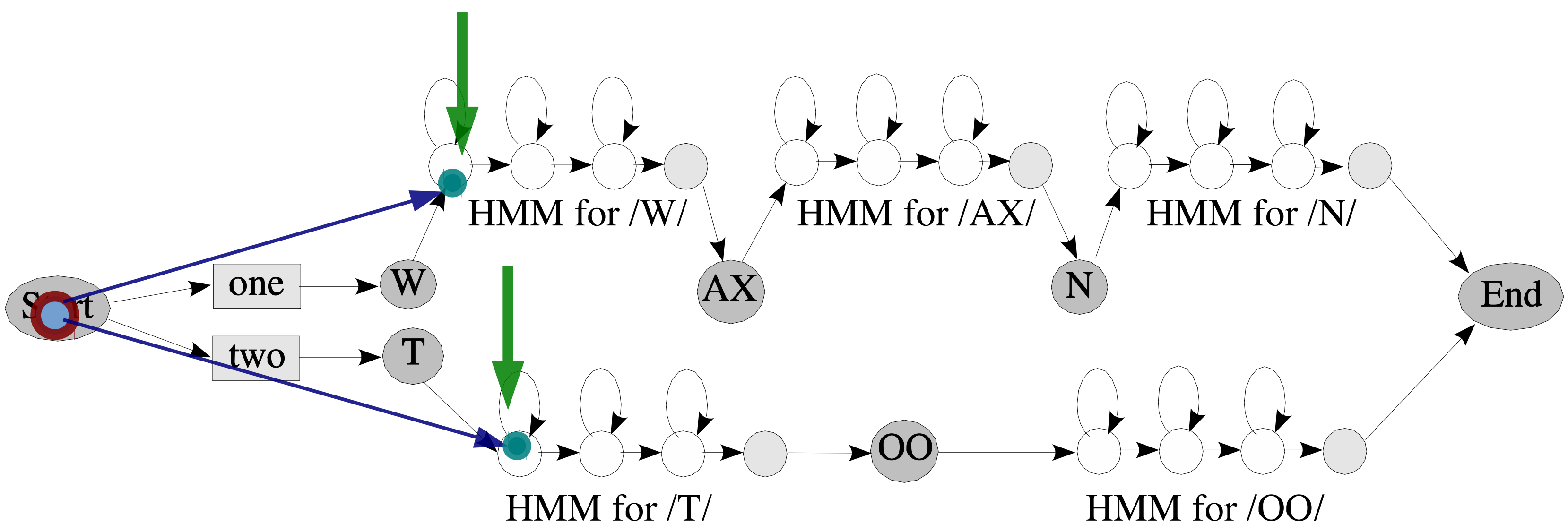
Token-Pass Algorithm

- Initialization: put a token into initial state
- find next tokens (forward to next emitting state)
  - add transition costs for edges
  - add emission/acceptance cost of observation



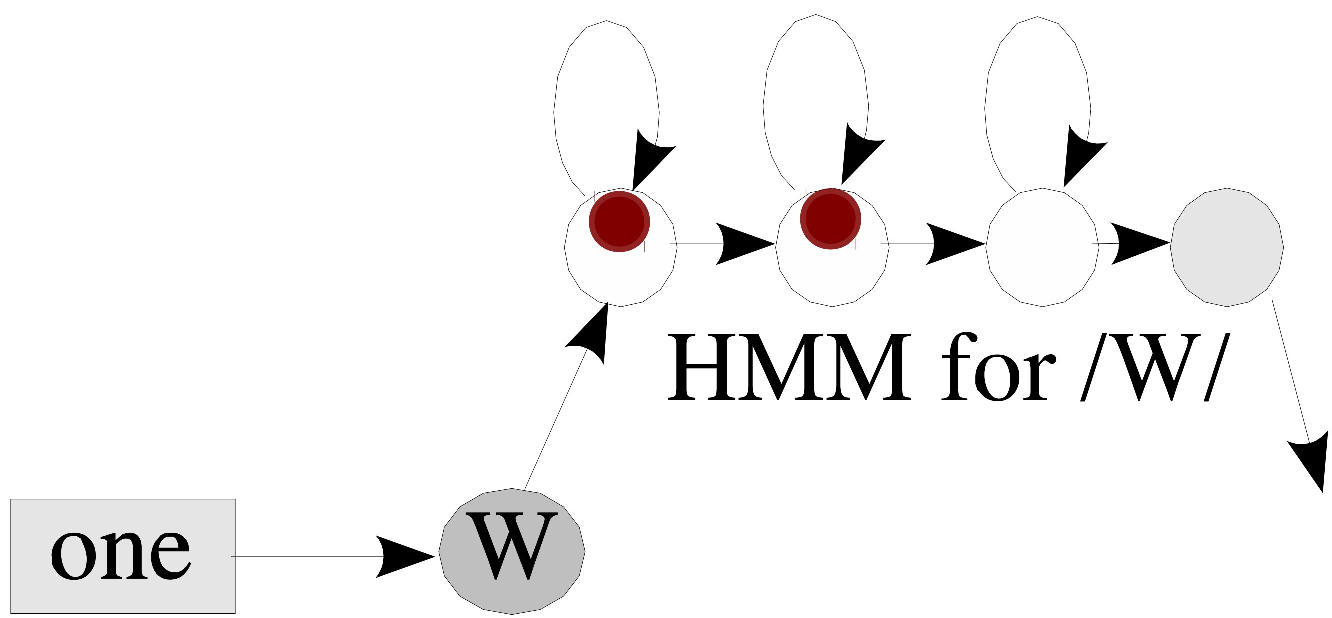
Token-Pass Algorithm

- Initialization: put a token into initial state
- find next tokens (forward to next emitting state)
  - add transition costs for edges
  - add emission/acceptance cost of observation



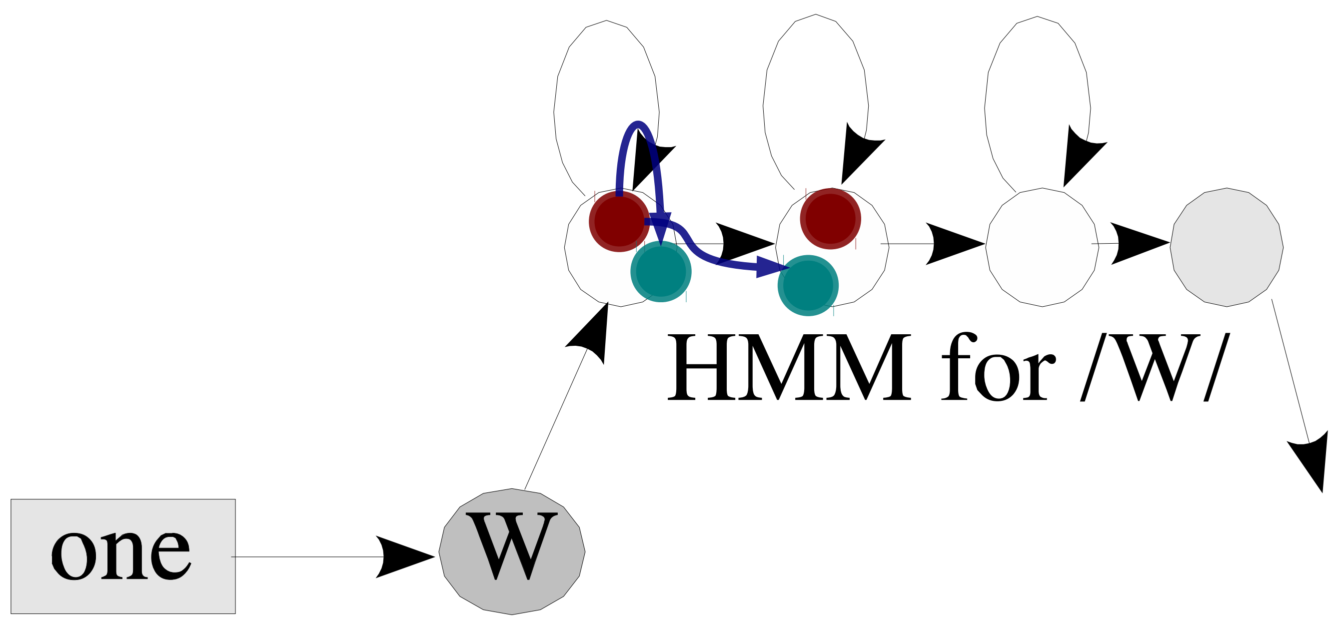
Token-Pass Algorithm: Multiple Tokens in the Same State

- different alignments of observations to one state path
- only the best path needs to be kept
  - all others can't be on the best final path



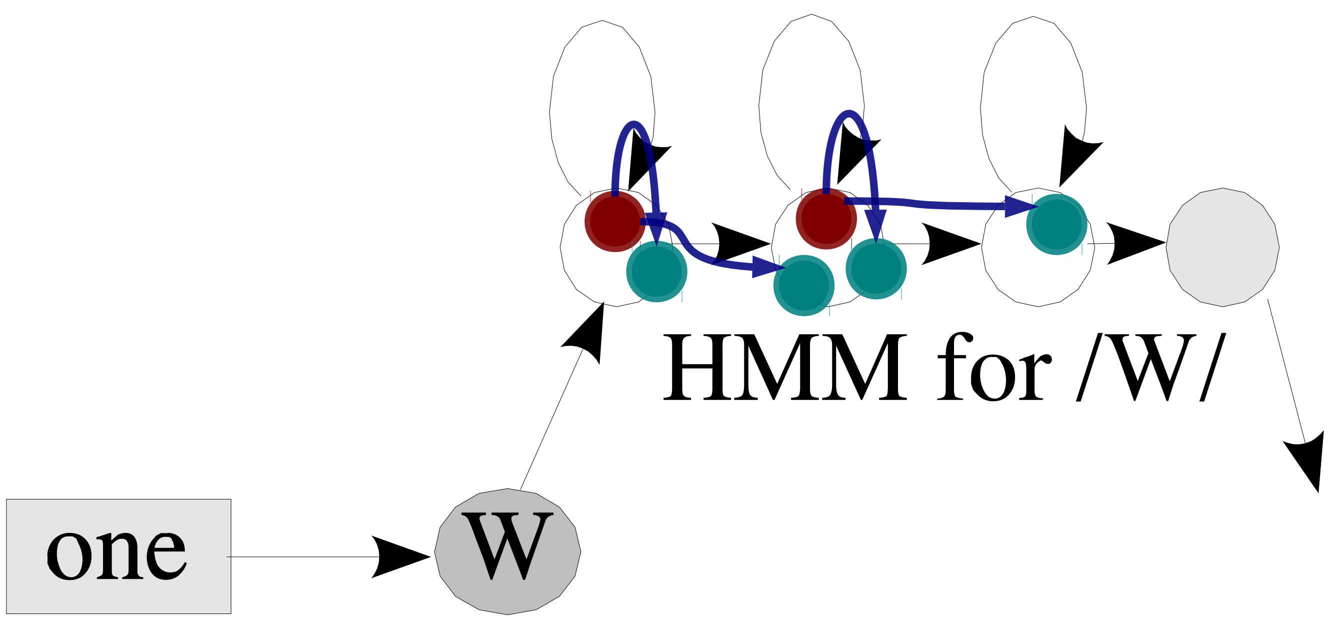
Token-Pass Algorithm: Multiple Tokens in the Same State

- different alignments of observations to one state path
- only the best path needs to be kept
  - all others can't be on the best final path



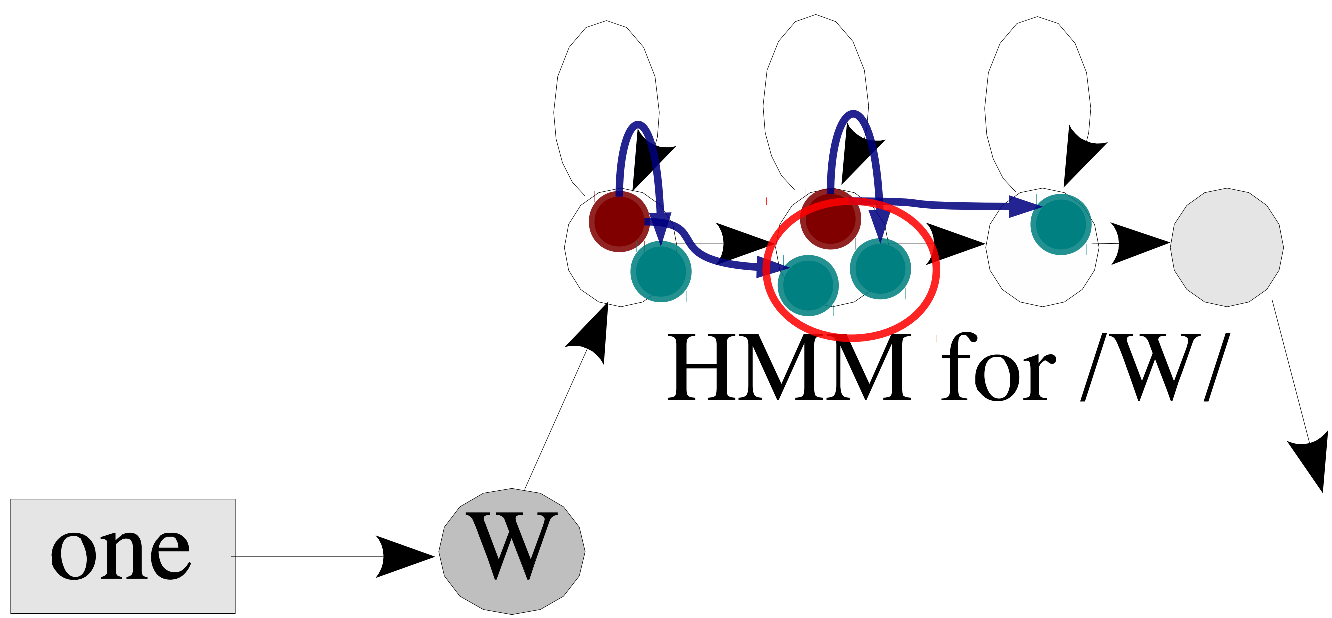
Token-Pass Algorithm: Multiple Tokens in the Same State

- different alignments of observations to one state path
- only the best path needs to be kept
  - all others can't be on the best final path



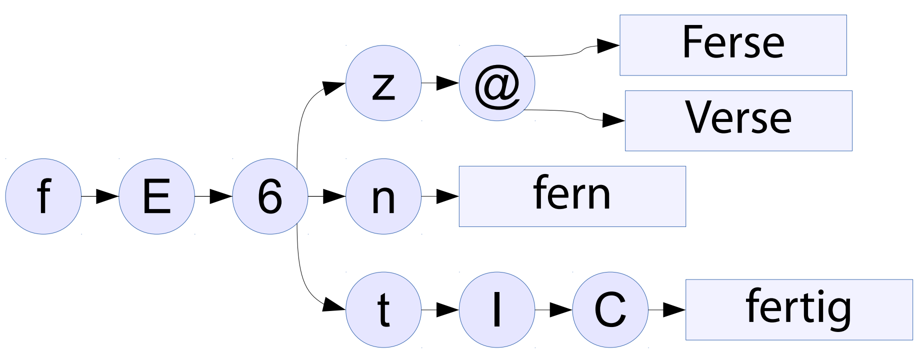
Token-Pass Algorithm: Multiple Tokens in the Same State

- different alignments of observations to one state path
- only the best path needs to be kept
  - all others can't be on the best final path



Limiting the Search

- The search graph may become very large
- remedy:
  - dynamically expand the search graph during recognition
  - only expand where hypotheses are likely
    - purge unlikely hypotheses
  - make the graph more compact by sharing common prefixes



Token-Pass Algorithm: Extensions

- sort tokens by cost in every step and
  - prune list to a maximum of N tokens at every time step
  - keep only tokens that are `good' relative to the best token
  - ➔ reduces search space but may result in non-optimal path
- it's not necessary to operate time-synchronously
  - could e.g. also use A* search
- more administrative complexity when using dynamic search graph, LexTree, Triphones, …

Training the HMM-parameters: Baum-Welch Algorithm

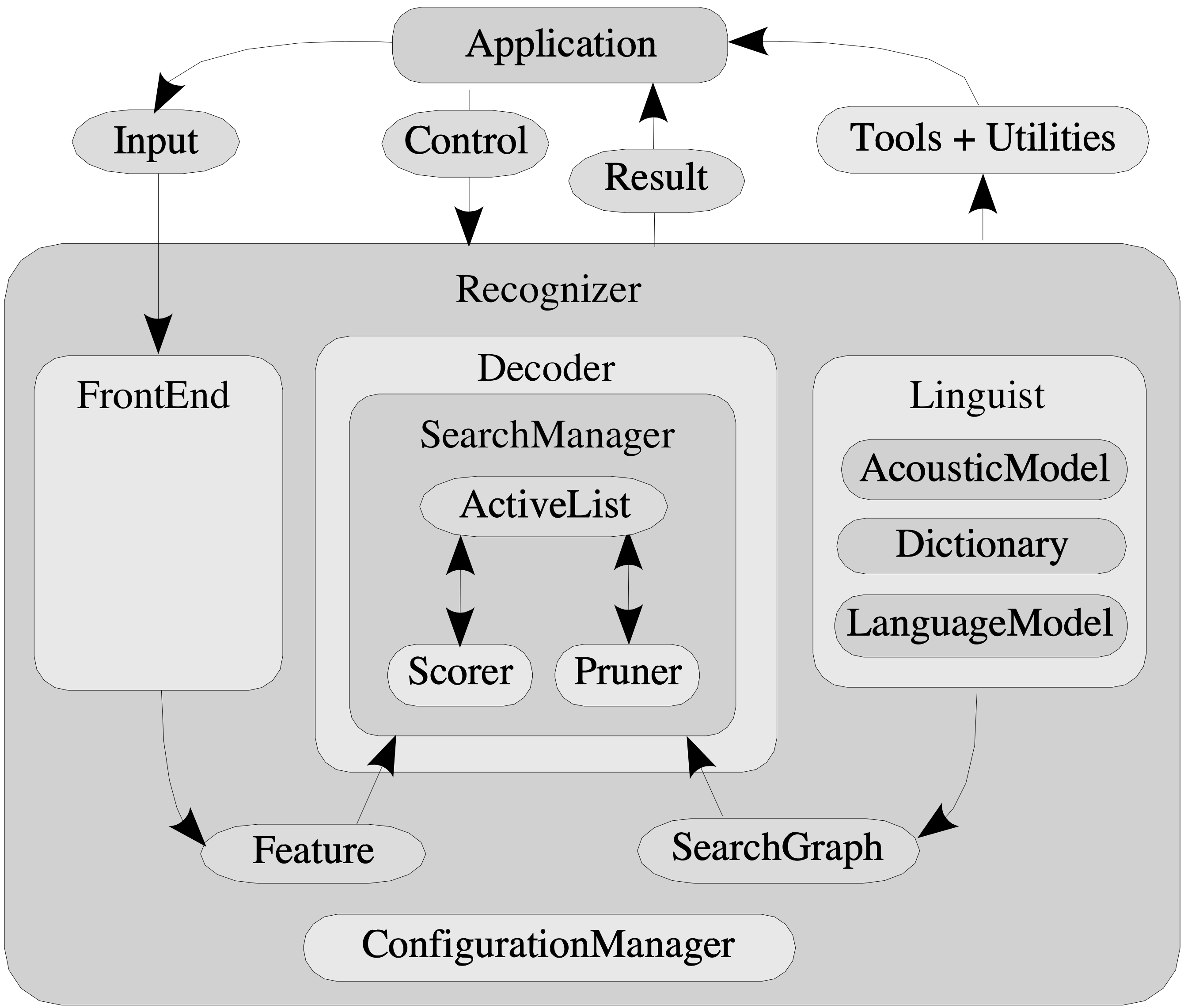
- computing Gaussian μ and σ is straightforward from training data
  - … if we know phoneme/state boundaries beforehand
- in practice we only have texts and corresponding audio
  1) turn text into phoneme/state sequence
  2) split audio into as many parts as there are states in the sequence
  3) estimate parameters based on these state boundaries
  4) use parameters to re-align state boundaries
  5) goto 3) until convergence

Phone Models (II)

reality is slightly more complex:

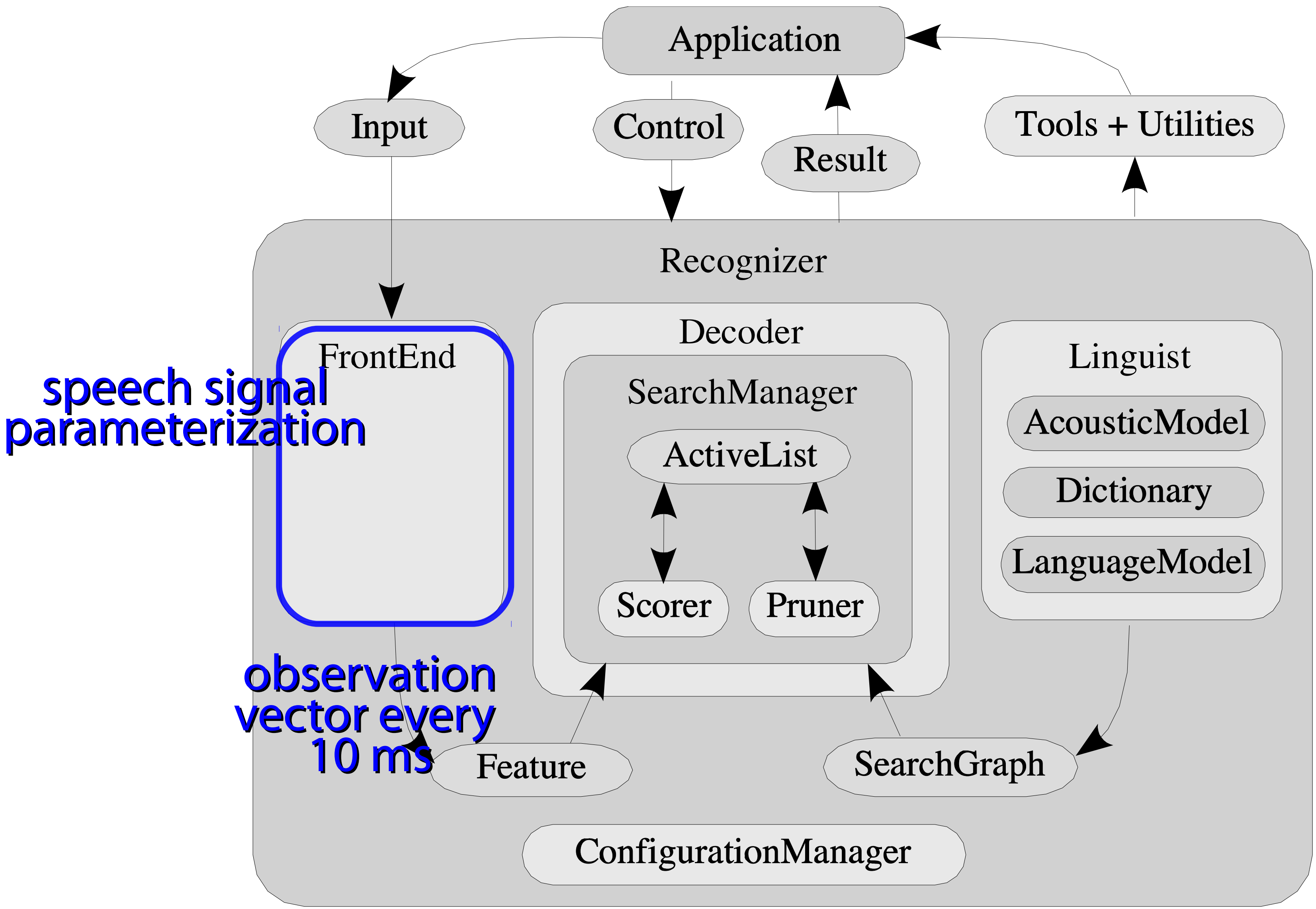
- the observation vector is multi-dimensional
  → multi-dimensional Gaussian
- there are usually three states per phone
  (transition/stable phase/next transition) → more states
- phone context shapes acoustics → use Triphone contexts → more states
- probability distribution is not necessarily Gaussian in practice
  - complex distributions can be modelled by mixing multiple Gaussians
    → more parameters per state
- drawback: need to estimate many parameters during training
  - remedy: share mixtures between some phonemes
    (sharing strategy is determined from training data)

Sphinx-4: A Flexible Open Source Framework for Speech Recognition



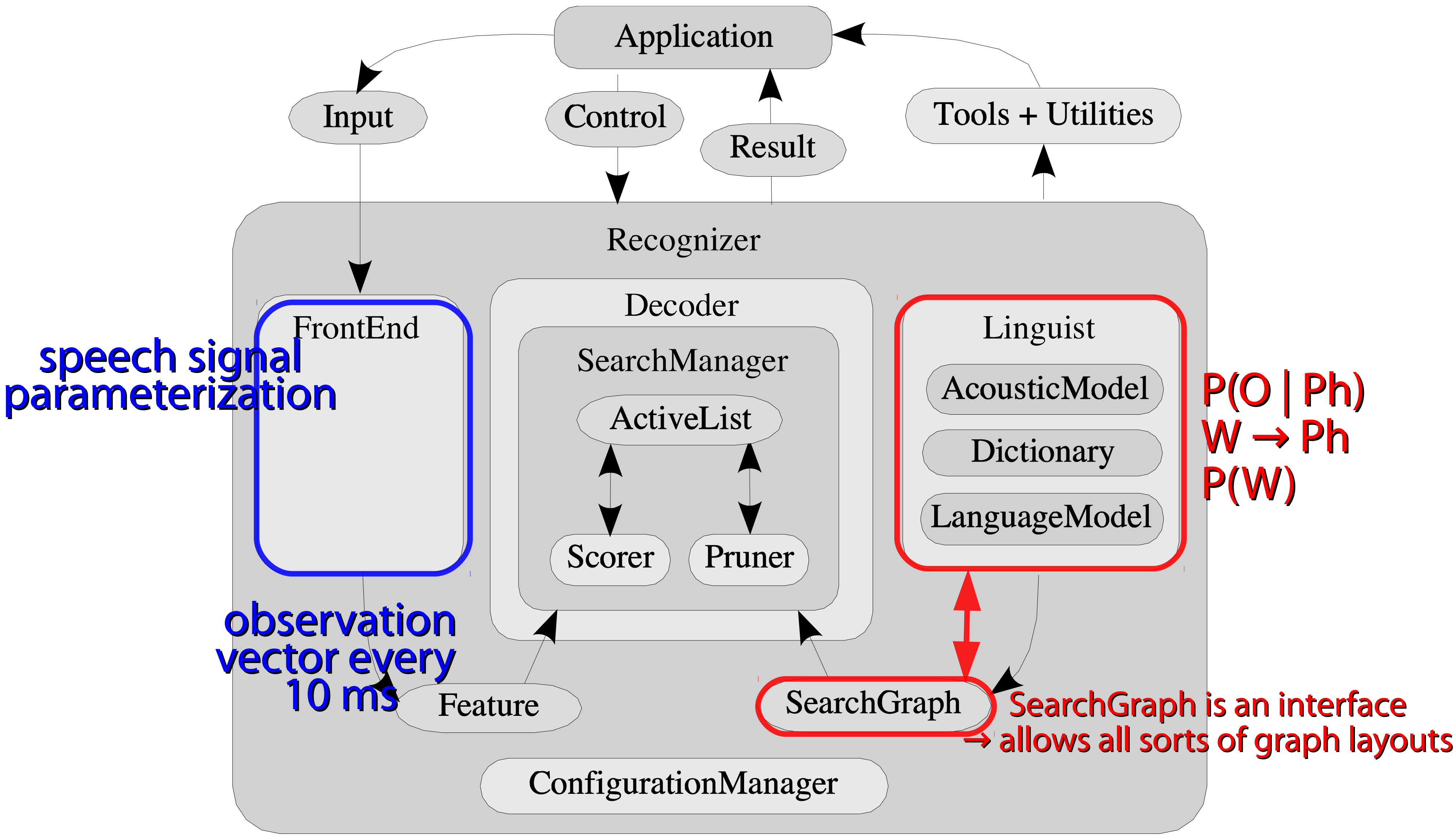
Walker et al., Sphinx-4: A Flexible Open Source Framework for SR, 2004.

Sphinx-4: A Flexible Open Source Framework for Speech Recognition



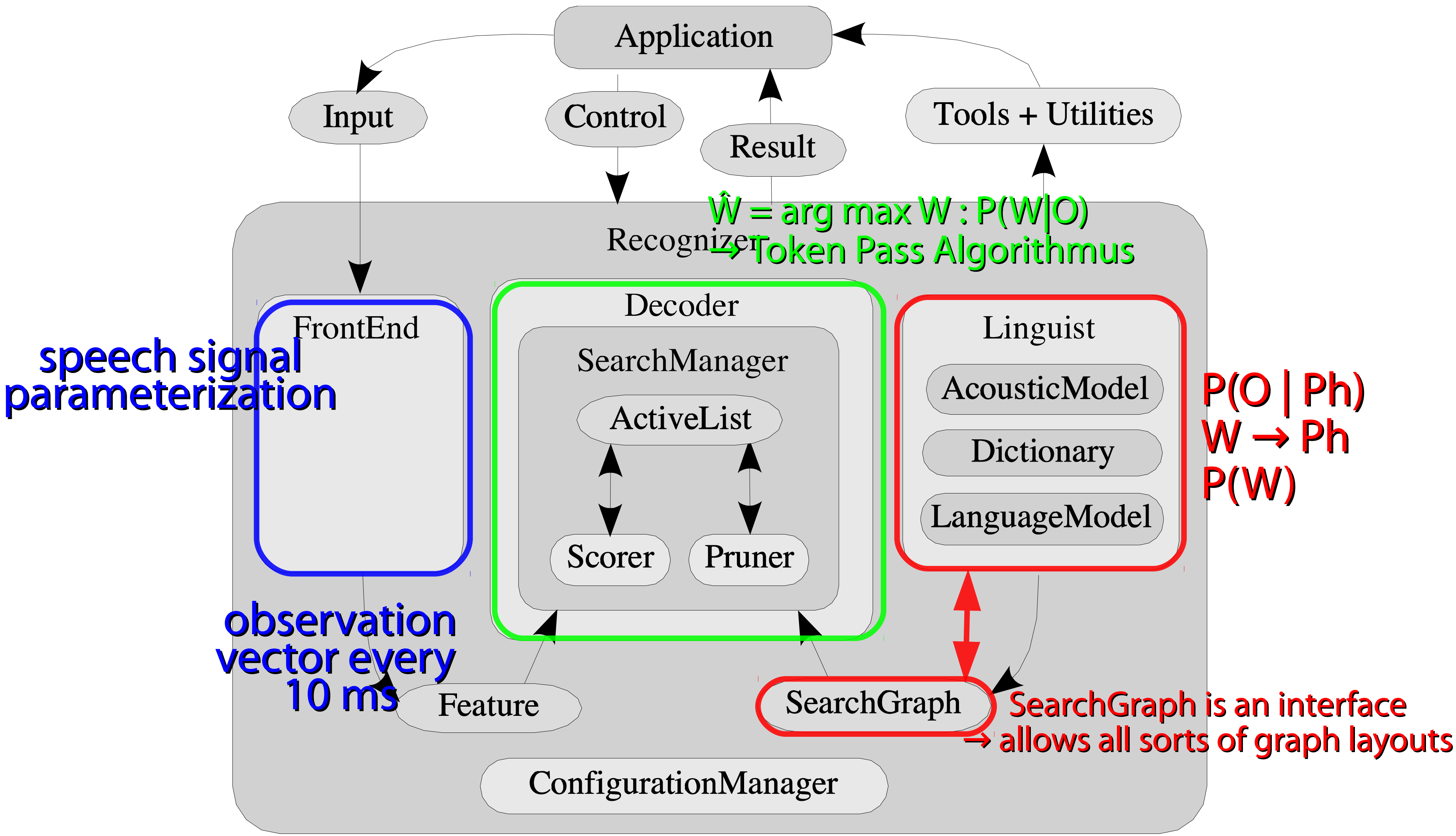
Walker et al., Sphinx-4: A Flexible Open Source Framework for SR, 2004.

Sphinx-4: A Flexible Open Source Framework for Speech Recognition



Walker et al., Sphinx-4: A Flexible Open Source Framework for SR, 2004.

Sphinx-4: A Flexible Open Source Framework for Speech Recognition



Walker et al., Sphinx-4: A Flexible Open Source Framework for SR, 2004.

Summary

- Noisy-channel model
- Problem: $\hat{W} = \arg\max W : P(W|O)$
- Solution: $\hat{W} = \arg\max W : P(O|Ph) \times P(Ph|W) \times P(W)$
  - $P(W)$: Word Sequence Model → N-Gram, (weighted) Grammar
  - $P(Ph|W)$: Pronunciation Model → e.g. table lookup, rules, ...
  - $P(O|Ph)$: Allophone Model → Hidden Markov Models
- Search Problem
  - time-synchronous search, dynamic programming
  - Token Pass Algorithmus
  - idea of Baum-Welch training

Thank you.

baumann@informatik.uni-hamburg.de

https://nats-www.informatik.uni-hamburg.de/SLP16

Further Reading

- Speech Recognition in General:
  - D. Jurafsky & J. Martin (2009): *Speech and Language Processing*. Pearson International. InfBib: A JUR 4204x
- Token-Pass Algorithm:
  - Young, Russel, Thornton (1989): "Token Passing: A Simple Conceptual Model for Connected Speech Recognition Systems", *Tech.Rep. CUED/F-INFENG/TR*, Cambridge University.
- The Sphinx-4 Speech Recognizer:
  - Walker et al. (2004): "Sphinx-4: A Flexible Open Source Framework for Speech Recognition", *Tech.Rep. SMLI TR2004-0811*, Sun Microsystems.

Notizen

Desired Learning Outcomes

- understand the optimization target of speech recognition and see implications on the whole-system perspective
- know and understand the details of the basic speech decoding algorithm based on token-passing, as well as be able to discuss its properties