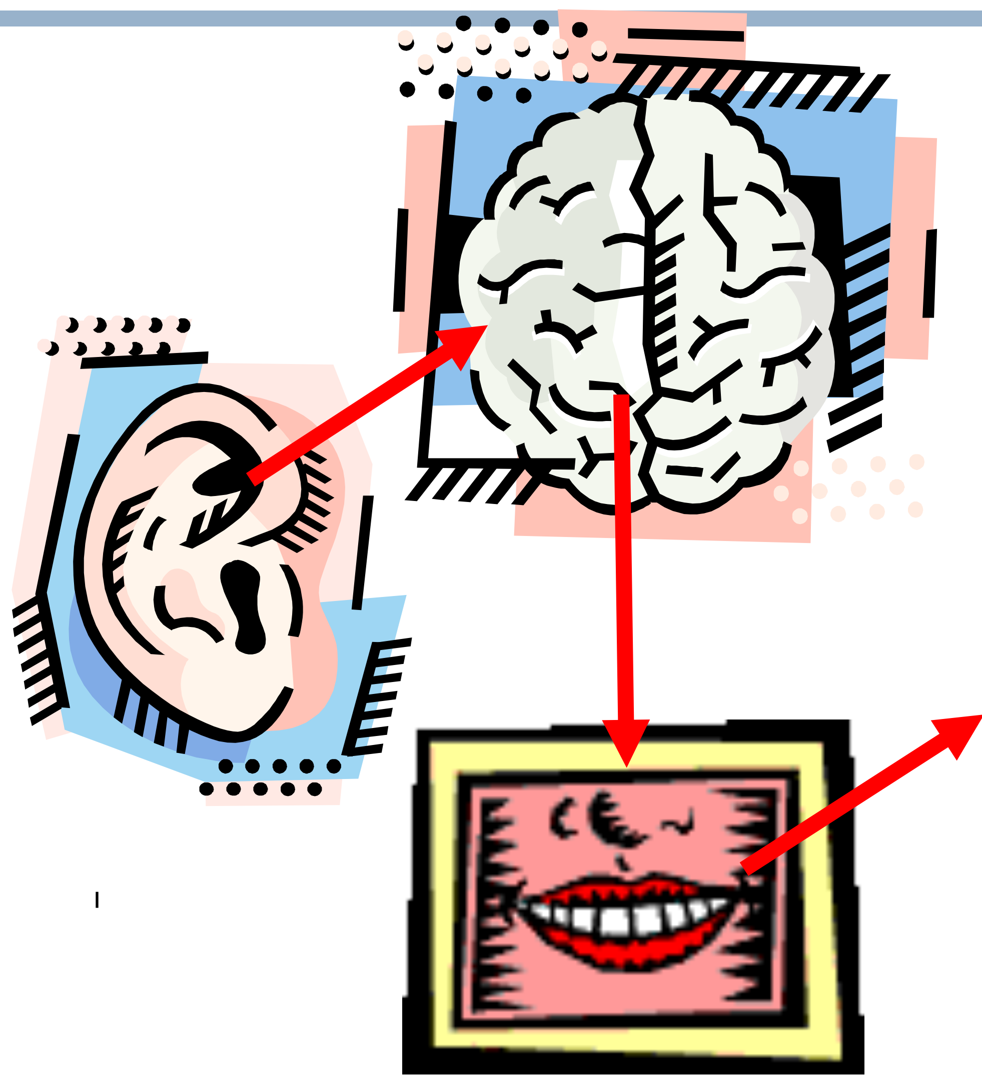
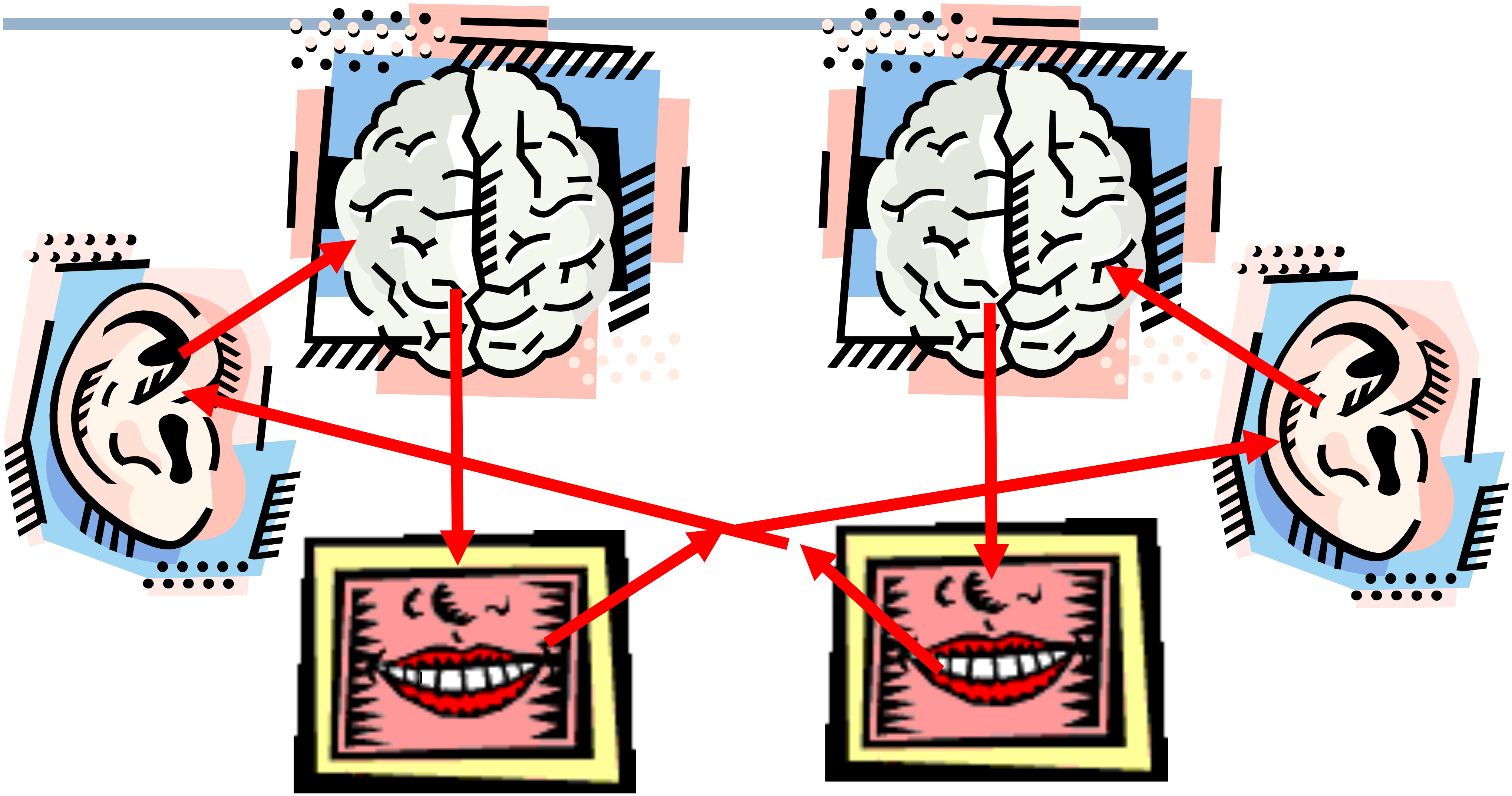
Unit 3: Application Areas

3c. Automatic speech recognition

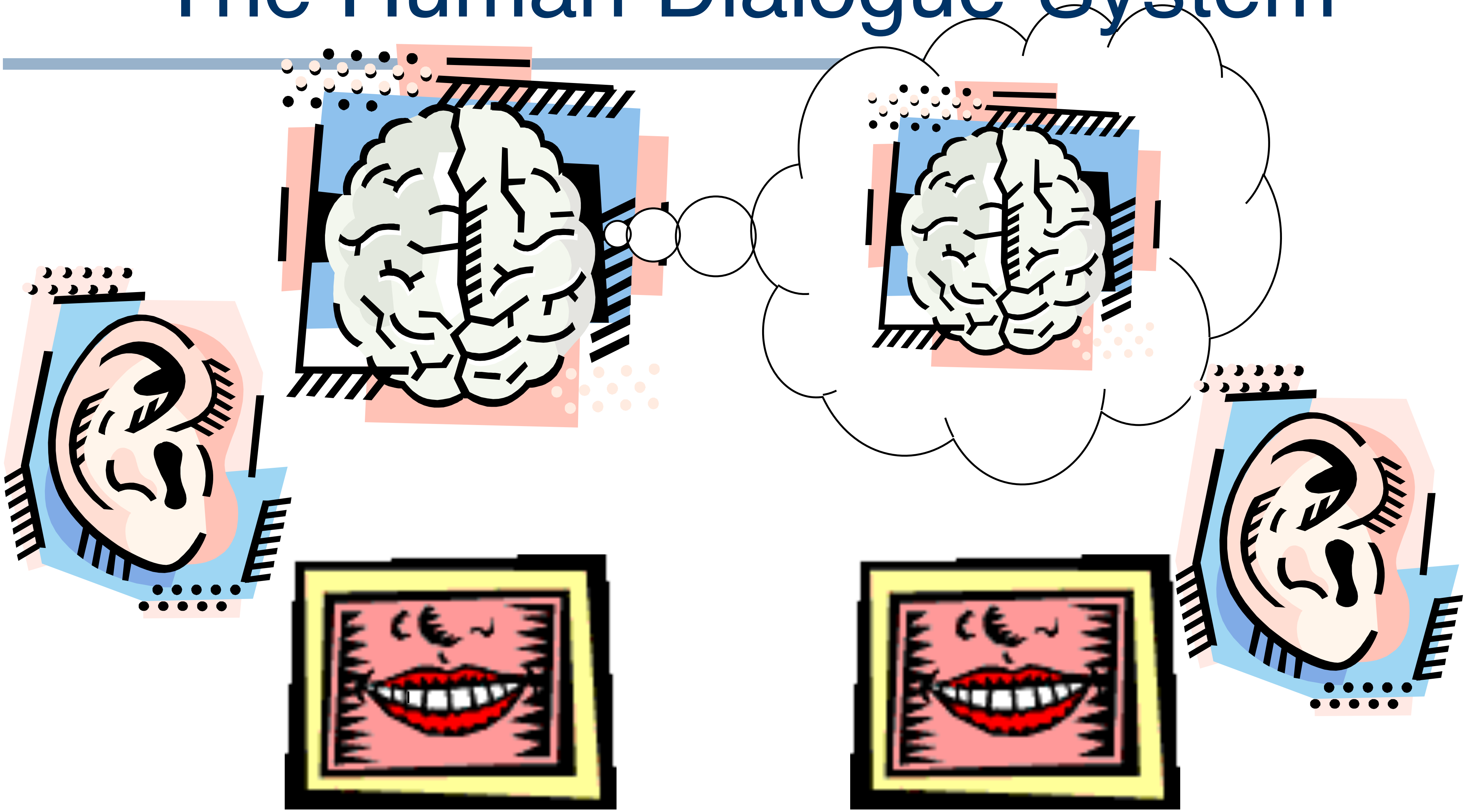
The Human Dialogue System



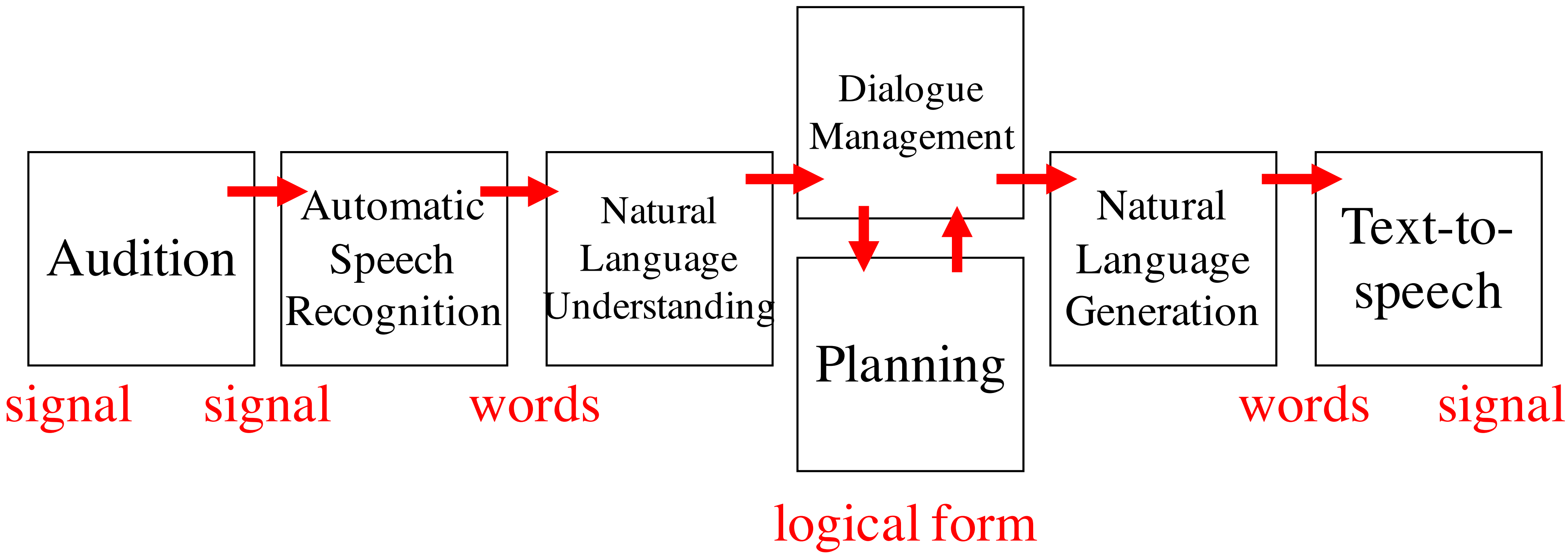
The Human Dialogue System



The Human Dialogue System

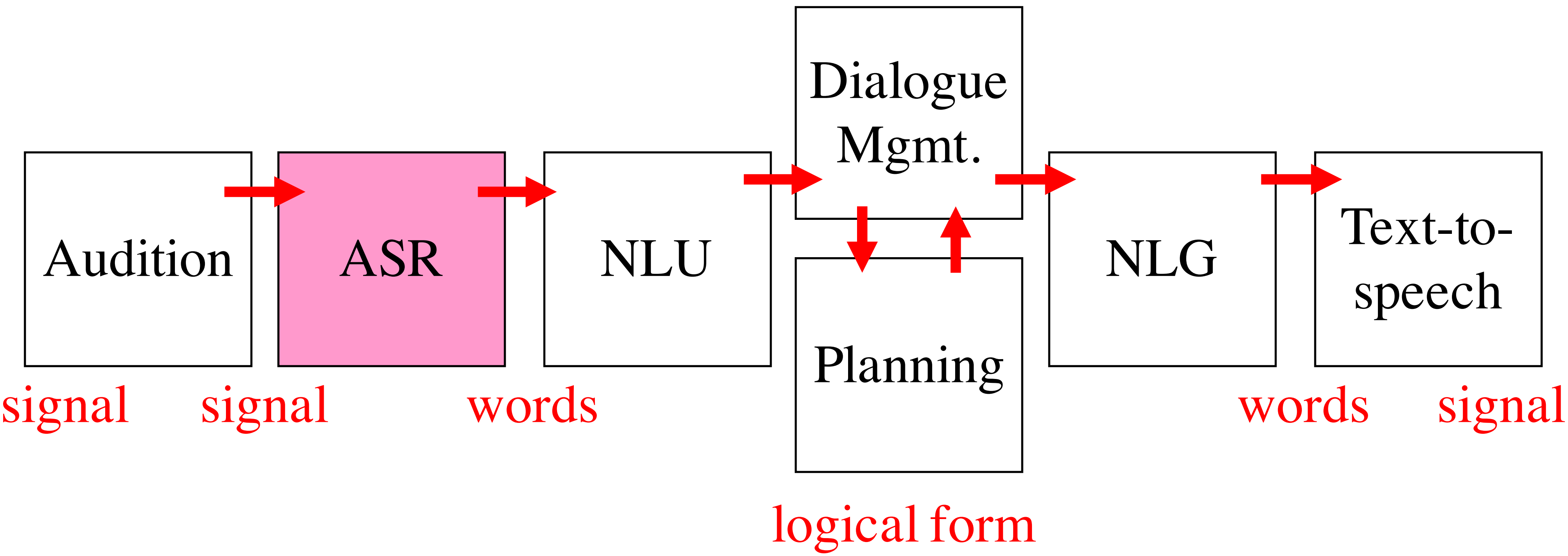


Computer Dialogue Systems

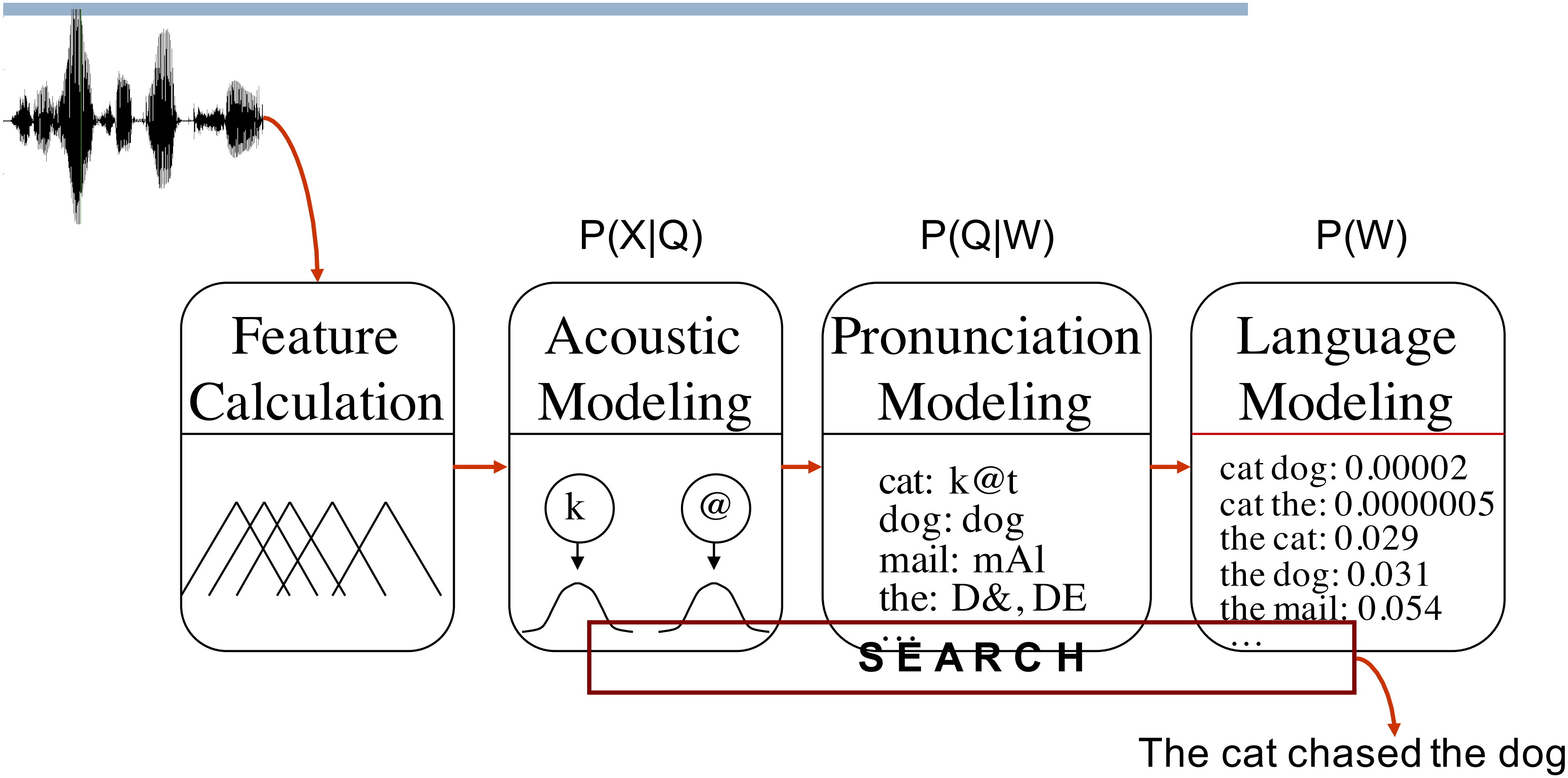


Computer Dialogue Systems

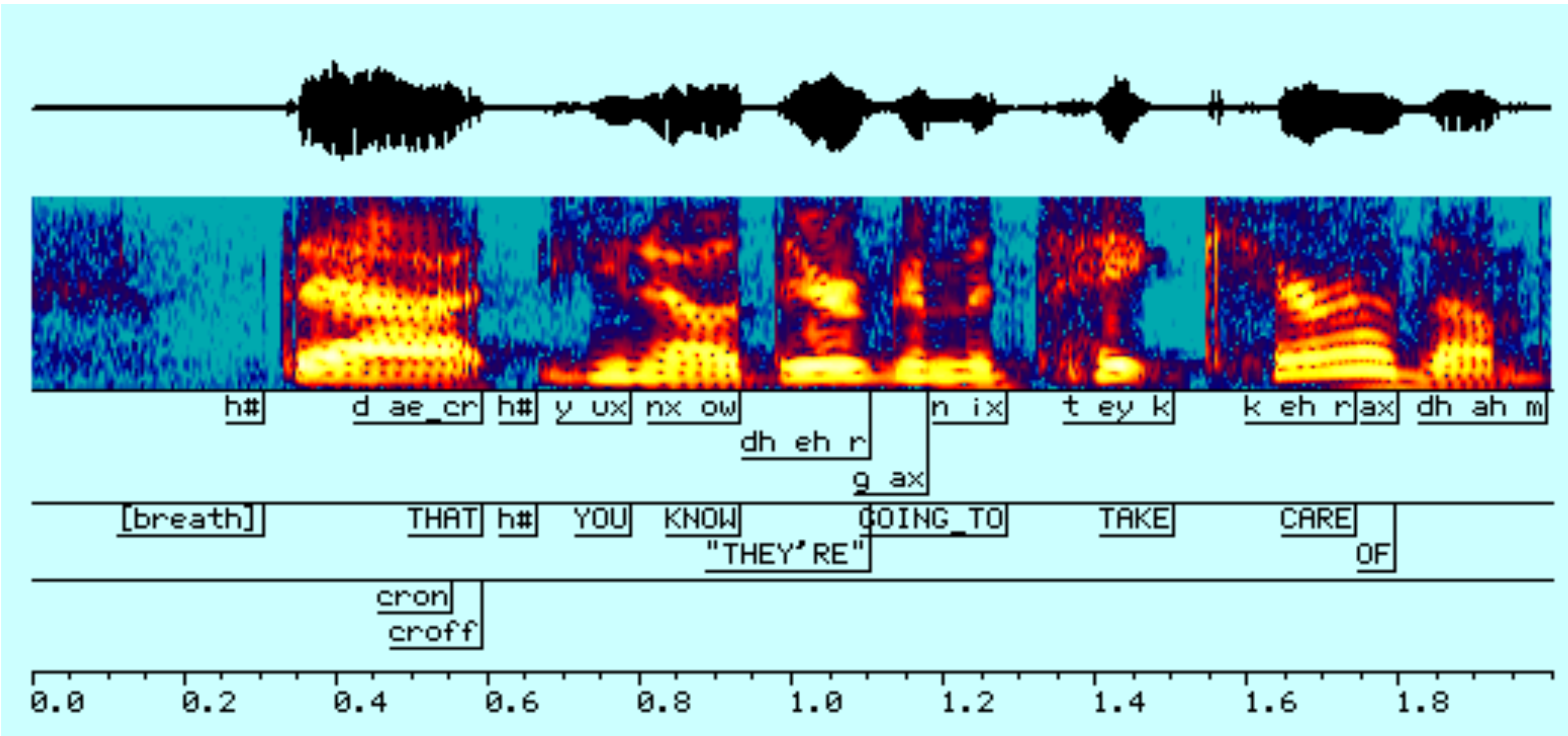




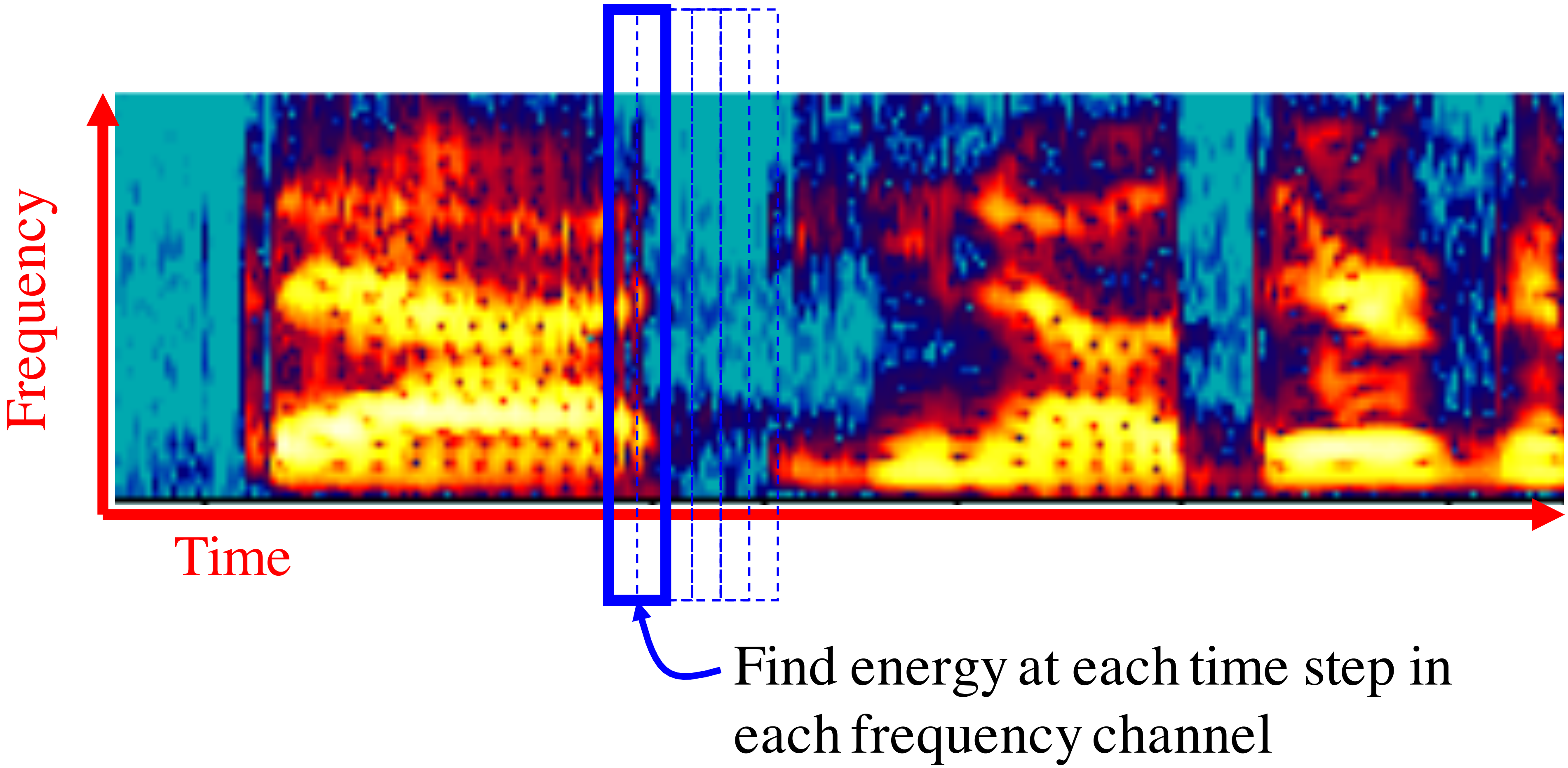
Parts of an ASR System



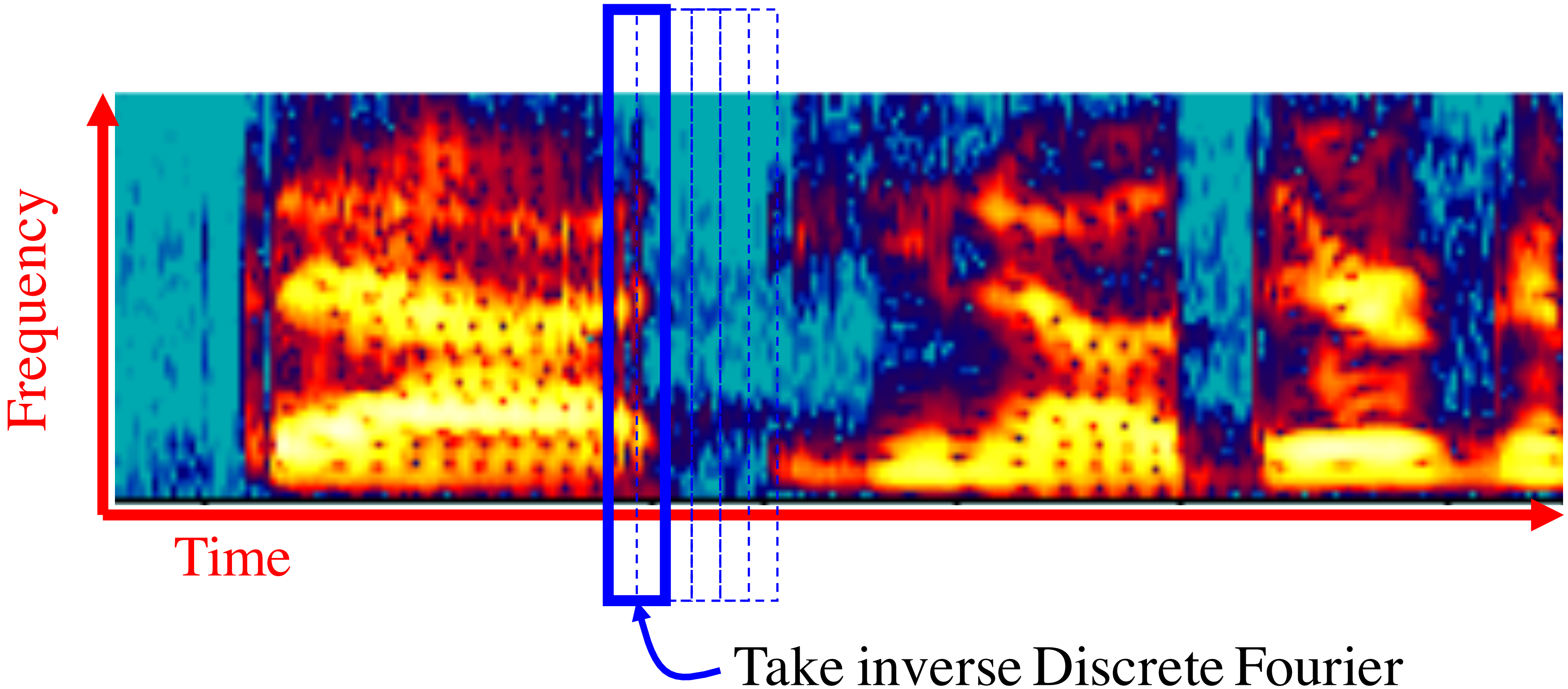
Feature calculation



Feature calculation



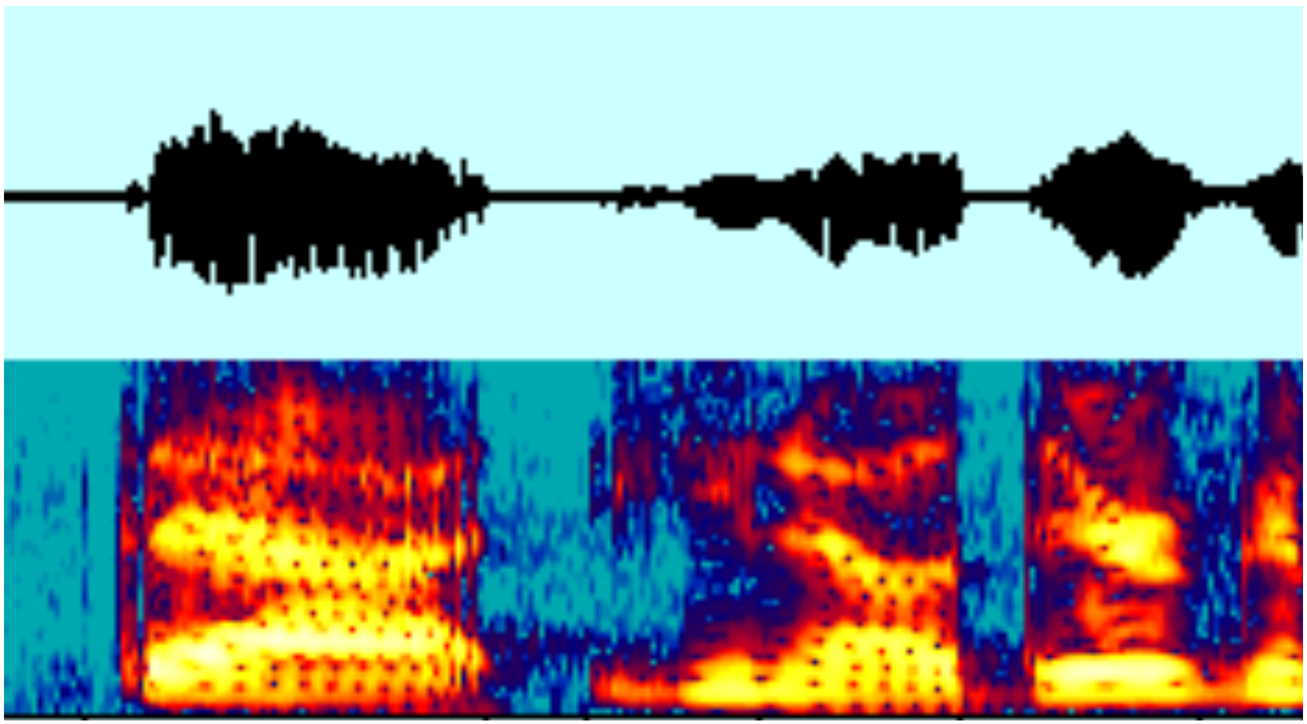
Feature calculation



Take inverse Discrete Fourier Transform to decorrelate frequencies

Feature calculation

Input:



Output:

-0.1	0.2	0.2	-6.1
0.3	0.1	0.0	-2.1
1.4	1.2	1.2	3.1
-1.2	-1.2	-1.2	2.4
2.3	4.4	4.4	1.0
2.6	2.2	2.2	2.2
…	…	…	…

…

Now what?

-0.1	0.2	0.2	-6.1
0.3	0.1	0.0	-2.1
1.4	1.2	1.2	3.1
-1.2	-1.2	-1.2	2.4
2.3	4.4	4.4	1.0
2.6	2.2	2.2	2.2
…	…	…	…

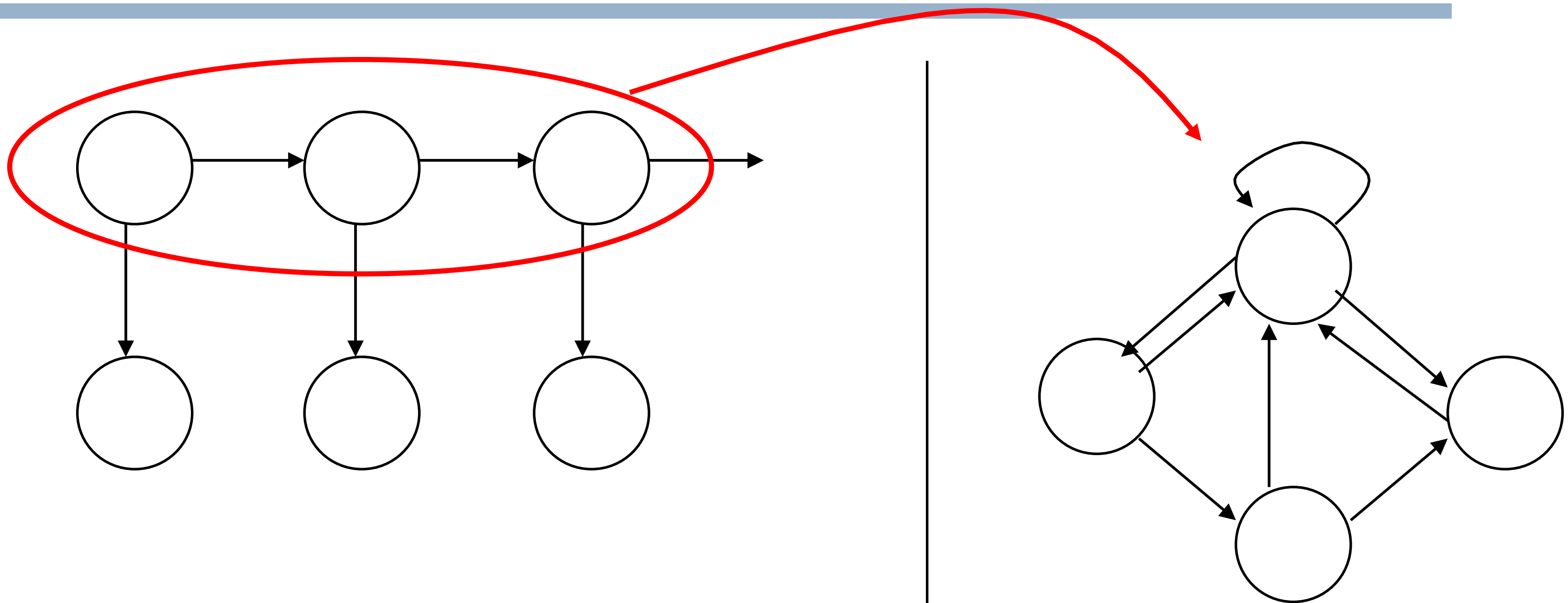
$\xrightarrow{???}$ That you …

Machine Learning!

-0.1	0.2	0.2	-6.1
0.3	0.1	0.0	-2.1
1.4	1.2	1.2	3.1
-1.2	-1.2	-1.2	2.4
2.3	4.4	4.4	1.0
2.6	2.2	2.2	2.2
…	…	…	…

$\xrightarrow[\text{with HMMs}]{\text{Pattern recognition}}$ That you …

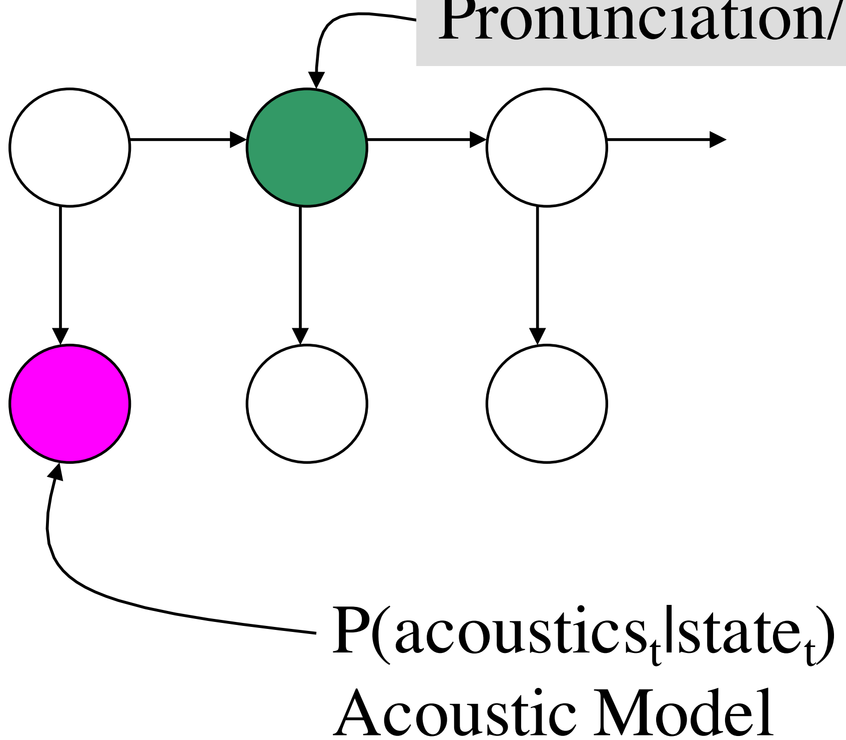
Hidden Markov Models (again!)

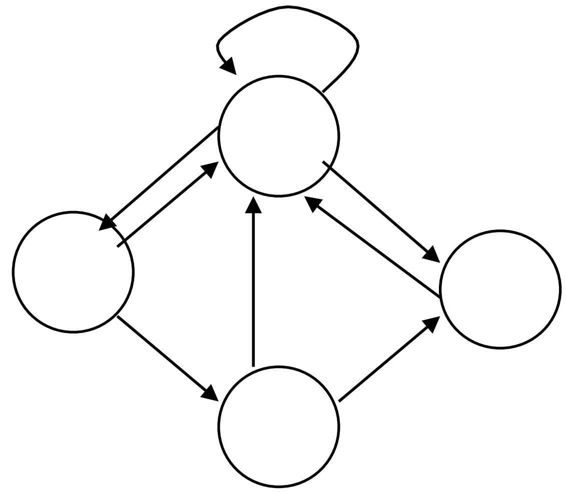


DBN representation

Markov Model representation

Hidden Markov Models (again!)





Acoustic Model

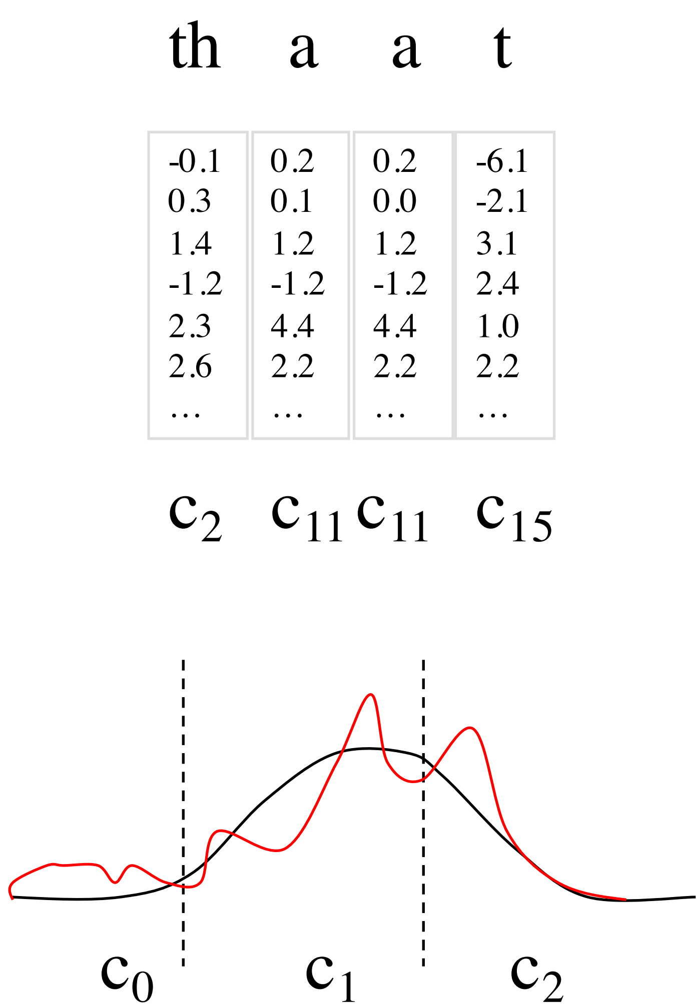
th	a	a	t
-0.1	0.2	0.2	-6.1
0.3	0.1	0.0	-2.1
1.4	1.2	1.2	3.1
-1.2	-1.2	-1.2	2.4
2.3	4.4	4.4	1.0
2.6	2.2	2.2	2.2
…	…	…	…

$N_a(\mu,\Sigma)$

$P(X|state=a)$

- Assume that you can label each vector with a phonetic label
- Collect all of the examples of a phone together and build a Gaussian model

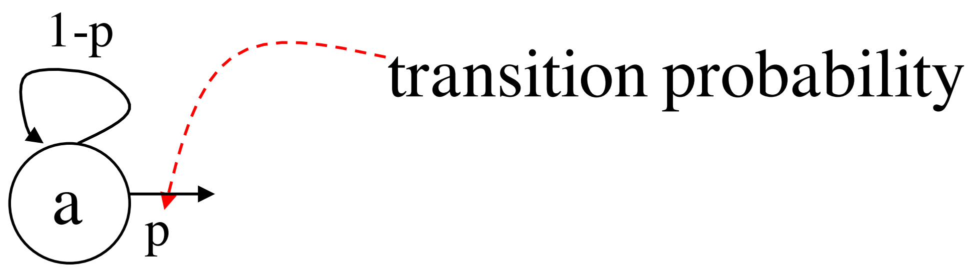
Vector Quantization



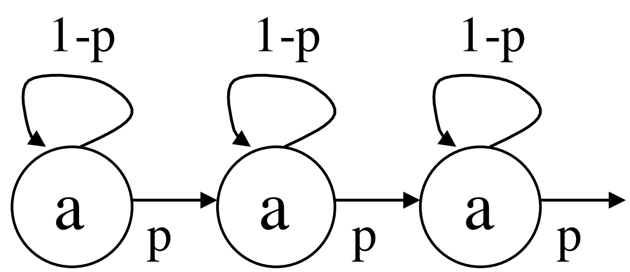
- Discretize the n-dimensional space
- Labels are arbitrary
- Compute P(label|phone) for acoustic model
- Not used much anymore in real systems

Building up the Markov Model

- Start with a model for each phone

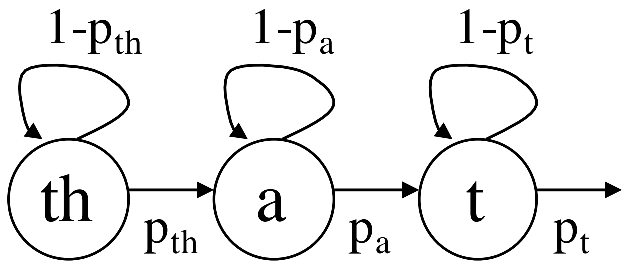


- Typically, we use 3 states per phone to give a minimum duration constraint, but ignore that here…

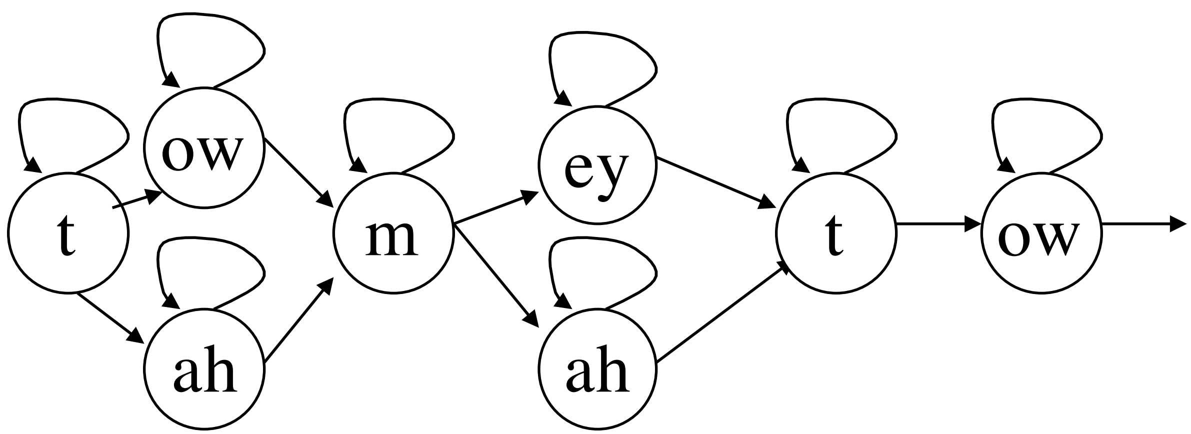


Building up the Markov Model

- Pronunciation model gives connections between phones and words

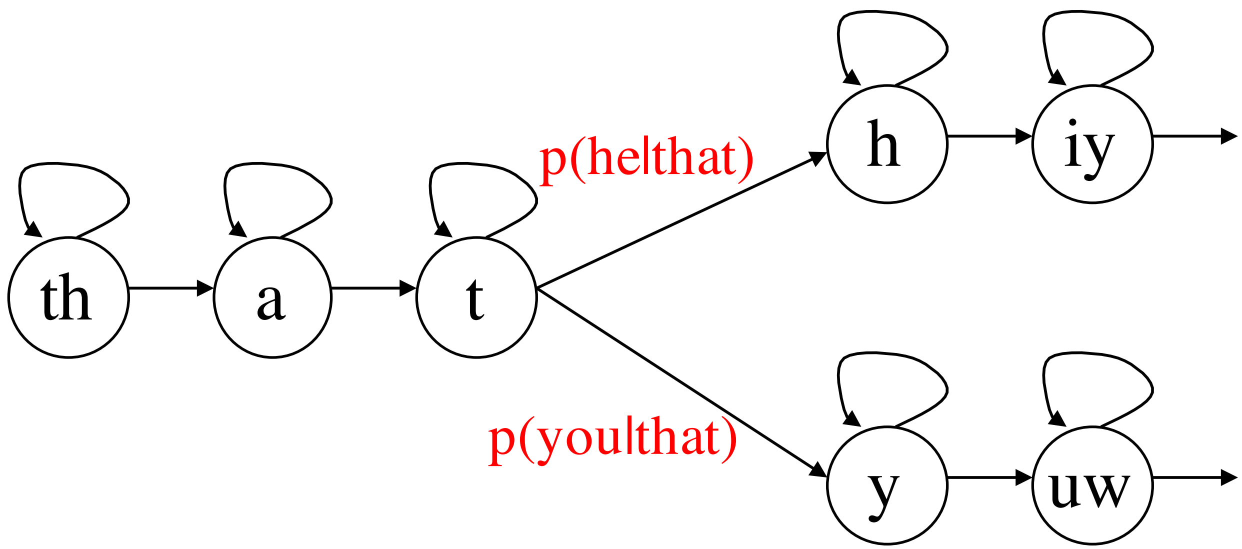


- Multiple pronunciations:



Building up the Markov Model

- Language model gives connections between words (e.g., bigram grammar)



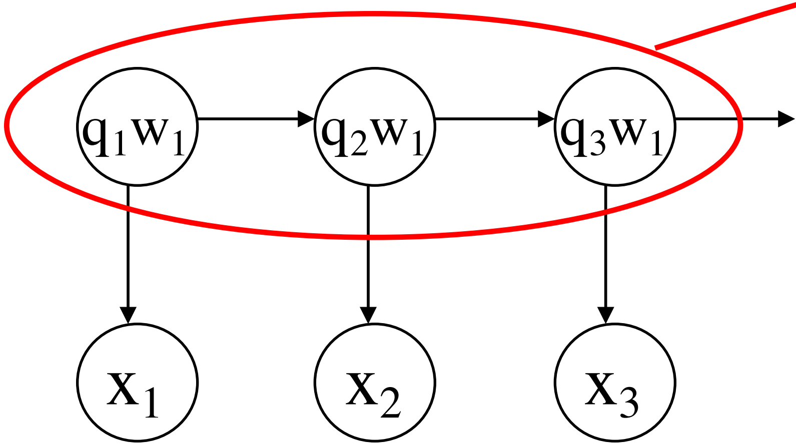
Two questions

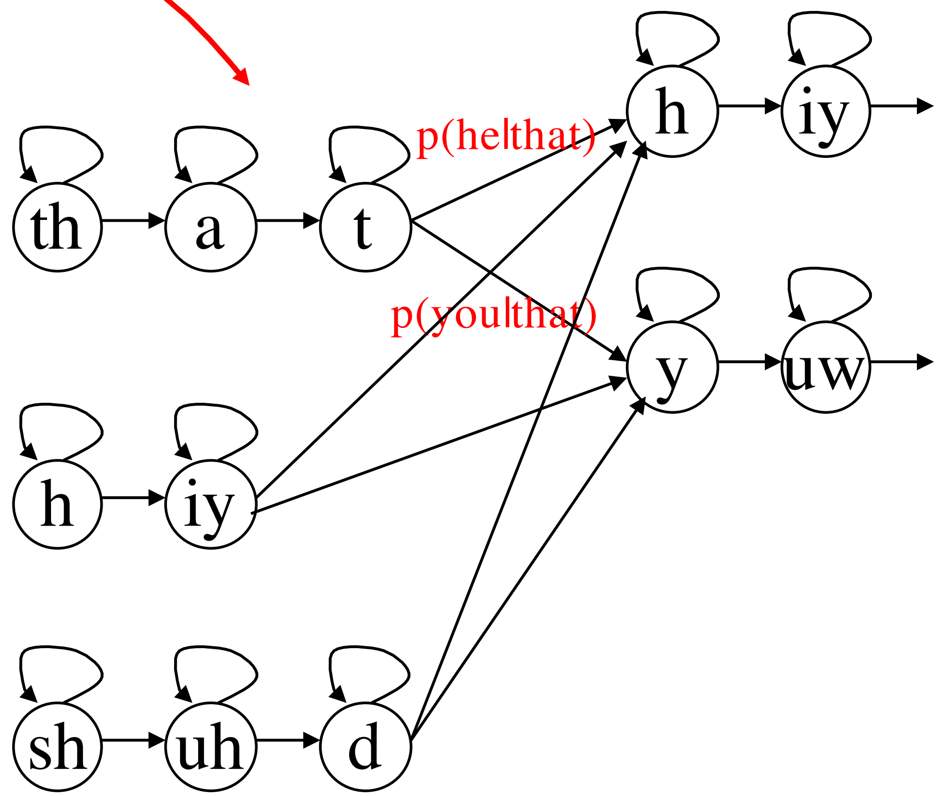
- There are two main questions to be answered:
  - 1: How do we use the ASR system?
  - 2: How do we train the ASR system?

ASR: Decoding

- Three types of sequence information
  - $X=x_1x_2x_3\ldots$ acoustics, $Q=q_1q_2q_3\ldots$ HMM states, $W=w_1w_2w_3\ldots$ words
- Have three probability models
  - $P(X|Q)$: acoustic model
  - $P(Q|W)$: duration/transition/pronunciation model
  - $P(W)$: language model
- Want W that maximizes $P(W|X)$
  - Find the best W through search: “decoding”

ASR as Bayesian Inference





$\text{argmax}_W\ P(W|X)$

$= \text{argmax}_W\ P(X|W)P(W)/P(X)$

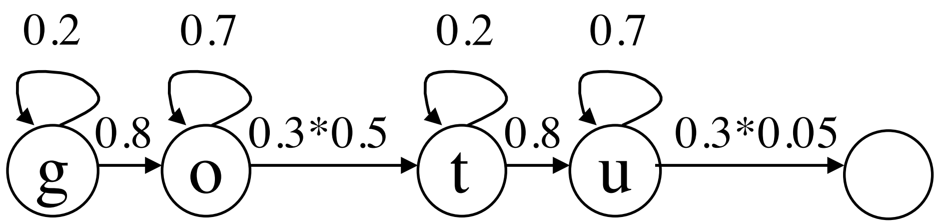
$= \text{argmax}_W\ P(X|W)P(W)$

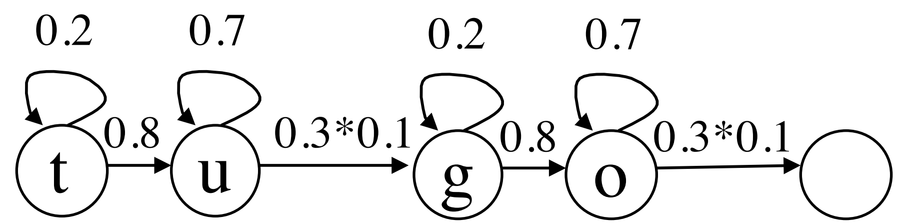
$= \text{argmax}_W\ \Sigma_Q\ P(X,Q|W)P(W)$

$\approx \text{argmax}_W\ \max_Q\ P(X,Q|W)P(W)$

$\approx \text{argmax}_W\ \max_Q\ P(X|Q)\ P(Q|W)\ P(W)$

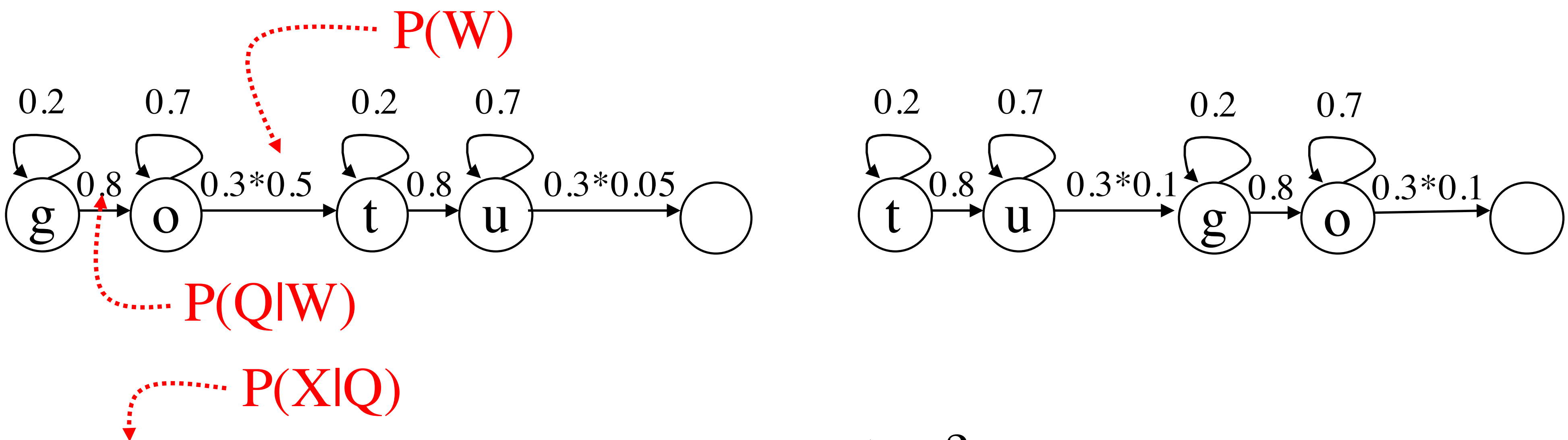
“go to” versus “to go”





P(x|g)	0.2	0	0	0.4	0
P(x|o)	0	0.1	0.1	0	0.2
P(x|t)	0.4	0	0	0.2	0
P(x|u)	0	0.2	0.1	0	0.1
	x_1	x_2	x_3	x_4	x_5

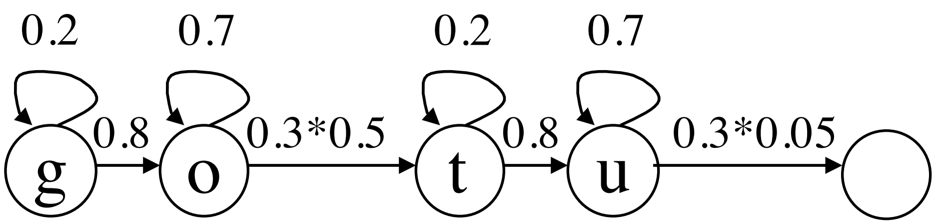
“go to” versus “to go”

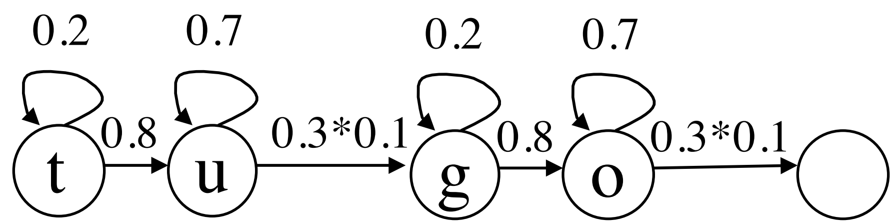


	x_1	x_2	x_3	x_4	x_5
P(x|g)	0.2	0	0	0.4	0
P(x|o)	0	0.1	0.1	0	0.2
P(x|t)	0.4	0	0	0.2	0
P(x|u)	0	0.2	0.1	0	0.1

$g_1o_2o_3t_4u_5$?

“go to” versus “to go”





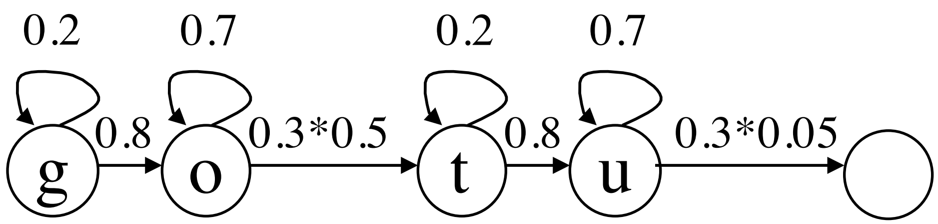
P(x|g)	0.2	0	0	0.4	0
P(x|o)	0	0.1	0.1	0	0.2
P(x|t)	0.4	0	0	0.2	0
P(x|u)	0	0.2	0.1	0	0.1
	x_1	x_2	x_3	x_4	x_5

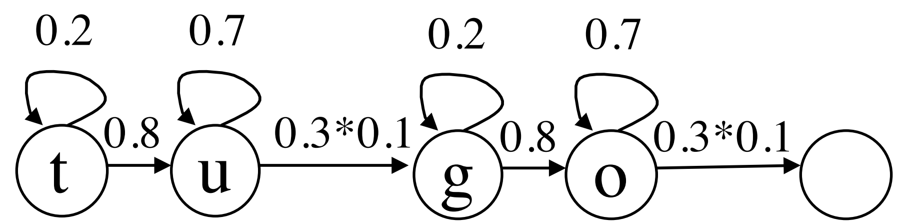
$g_1 o_2 o_3 t_4 u_5$:

(0.2*0.8) * (0.1*0.7) * (0.1*0.3*0.5) *
(0.2*0.8) * (0.1*0.3*0.05) = $4.032*10^{-8}$

$$P(X|W)P(W) = \prod_i P(x_i|q_i)P(q_{i+1}|q_i)$$

“go to” versus “to go”





P(x|g)	0.2	0	0	0.4	0
P(x|o)	0	0.1	0.1	0	0.2
P(x|t)	0.4	0	0	0.2	0
P(x|u)	0	0.2	0.1	0	0.1
	x_1	x_2	x_3	x_4	x_5

$g_1o_2o_3t_4u_5$: $4.032*10^{-8}$

$t_1u_2u_3g_4o_5$: $2.580*10^{-7}$

$(0.4*0.8)*(0.2*0.7)*(0.1*0.3*0.1)*$
$(0.4*0.8)*(0.2*0.3*0.1) = 2.580*10^{-7}$

$$P(X|W)P(W) = \prod_i P(x_i|q_i)P(q_{i+1}|q_i)$$

Training ASR Probability Models

- Three probability models
  - P(X|Q): acoustic model
  - P(Q|W): duration/transition/pronunciation model
  - P(W): language model
- language/pronunciation models inferred from prior knowledge
- Other models learned from data (how?)

Training Language Models

- Get a whole bunch of text data (corpus)

  …go to the market…

  …want to go to…

  …give the ball to him…

  …have to leave now…

  …supposed to go on Monday…

- Simplest model: Learn $P(w_{t+1}|w_t)$ (bigram)

  $P(<\text{the,go,him,leave}>|\text{to}) = <0.2,0.4,0.2,0.2>$

Training Language Models

- Problem: what's the probability of P(have|to)?

  P(<the,go,him,leave>|to) = <0.2,0.4,0.2,0.2>

  With this model, P(have|to)=0!
- Smoothing techniques needed to estimate unseen word pairs
  - Add small counts for all words (bad idea in practice)
  - Backoff smoothing: use P(have) to help estimate P(have|to)

    P(have|to) = P(have) BackoffWeight(to)
  - Interpolation: P(have|to)= α P(have|to) + (1-α) P(have)

EM for ASR: Training models

- Determine “state occupancy” probabilities
  - I.e. assign each data vector to a state
  - Remember: states are hidden variables
- Calculate new transition probabilities, new means & standard deviations using assignments
- What does this remind you of?

HMM Training Algorithm

- E-step: Compute $P(q|x_t)$ for all t
  - Translation: what is the probability of each phone generating the acoustic vector
  - Viterbi training: only consider the most likely phone
    - $P(q^*|x)=1$, $P(q|x)=0$ for other q
- M-step: Calculate $P(q_{t+1}|q_t)$, $P(x_t|q_t)$ to maximize $P(X|Q)$ (and subsequently $P(X|W)$)
  - $P(q_{t+1}|q_t)$: find which states occurred after other states
  - $P(x_t|q_t)$: compute new Gaussians for each state

How to train an ASR system

- Have a speech corpus at hand
  - Should have word (and preferrably phone) transcriptions
  - Divide into training, development, and test sets
- Develop models of prior knowledge
  - Pronunciation dictionary
  - Grammar
- Train acoustic models
  - Possibly realigning corpus phonetically

How to train an ASR system

- Test on your development data (baseline)
- **Think real hard
- Figure out some neat new modification
- Retrain system component
- Test on your development data
- Lather, rinse, repeat **
- Then, at the end of the project, test on the test data.

Judging the quality of a system

- Usually, ASR performance is judged by the word error rate

  ErrorRate = 100*(Subs + Ins + Dels) / Nwords

```
REF: I WANT TO   GO HOME ***
REC: * WANT TWO GO HOME NOW
SC:  D  C     S   C    C    I
100*(1S+1I+1D)/5 = 60%
```

Judging the quality of a system

- Usually, ASR performance is judged by the word error rate
- This assumes that all errors are equal
  - Also, a bit of a mismatch between optimization criterion and error measurement
- Other (task specific) measures sometimes used
  - Task completion
  - Concept error rate

Hybrid DNN-HMM ASR

- Variant: instead of modeling $P(x_t|q_t)$ with a Gaussian, use a deep neural network instead
- Problem: DNN computes $P(q_t|x_t)$
  - Idea: replace $P(x_t|q_t)$ with *scaled likelihood*

$$P(x_t|q_t) \propto \frac{P(q_t|x_t)}{P(q_t)}$$

Divide posterior by prior of phone (or HMM state)