Speech and Language Processing

Chapter 9 of SLP

Automatic Speech Recognition

Outline for ASR

- ASR Architecture
  - The Noisy Channel Model
- Five "easy" pieces of an ASR system
  1) Feature Extraction
  2) Acoustic Model
  3) Language Model
  4) Lexicon/Pronunciation Model (Introduction to HMMs again)
  5) Decoder
- Evaluation

Speech Recognition

- Applications of Speech Recognition (ASR)
  - Dictation
  - Telephone-based Information (directions, air travel, banking, etc)
  - Hands-free (in car)
  - Speaker Identification
  - Language Identification
  - Second language ('L2') (accent reduction)
  - Audio archive searching

LVCSR

- Large Vocabulary Continuous Speech Recognition
- ~20,000-64,000 words
- Speaker independent (vs. speaker-dependent)
- Continuous speech (vs isolated-word)

Current error rates

Ballpark numbers; exact numbers depend very much on the specific corpus

Task	Vocabulary	Error Rate%
Digits	11	0.5
WSJ read speech	5K	3
WSJ read speech	20K	3
Broadcast news	64,000+	10
Conversational Telephone	64,000+	20

HSR versus ASR

Task	Vocab	ASR	Hum SR
Continuous digits	11	.5	.009
WSJ 1995 clean	5K	3	0.9
WSJ 1995 w/noise	5K	9	1.1
SWBD 2004	65K	20	4

- Conclusions:
  - Machines about 5 times worse than humans
  - Gap increases with noisy speech
  - These numbers are rough, take with grain of salt

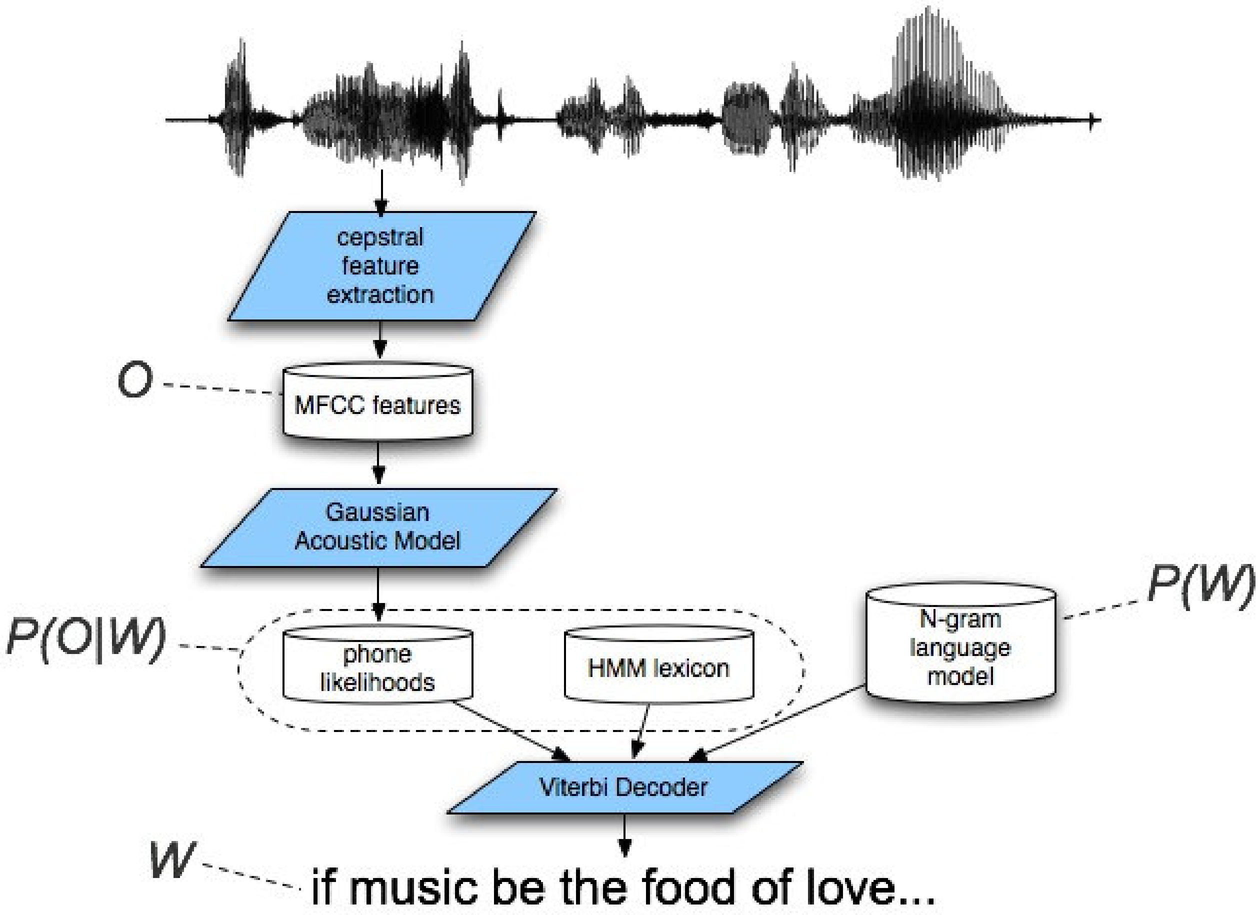
Why is conversational speech harder?

- A piece of an utterance without context
- The same utterance with more context

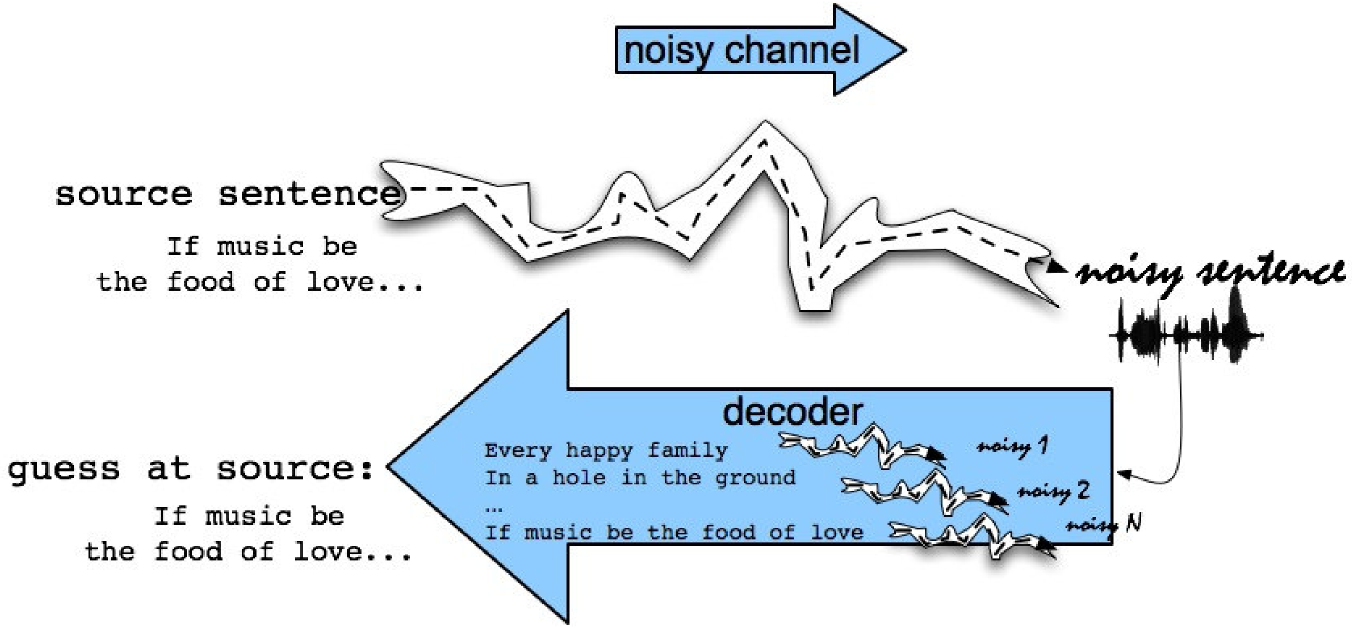
LVCSR Design Intuition

- Build a statistical model of the speech-to-words process
- Collect lots and lots of speech, and transcribe all the words.
- Train the model on the labeled speech
- Paradigm: Supervised Machine Learning + Search

Speech Recognition Architecture



The Noisy Channel Model



- Search through space of all possible sentences.
- Pick the one that is most probable given the waveform.

The Noisy Channel Model (II)

- What is the most likely sentence out of all sentences in the language L given some acoustic input O?
- Treat acoustic input O as sequence of individual observations
  - $O = o_1, o_2, o_3, \ldots, o_t$
- Define a sentence as a sequence of words:
  - $W = w_1, w_2, w_3, \ldots, w_n$

Noisy Channel Model (III)

- Probabilistic implication: Pick the highest prob S = W:

$$\hat{W} = \underset{W \in L}{\operatorname{argmax}} P(W \mid O)$$

- We can use Bayes rule to rewrite this:

$$\hat{W} = \underset{W \in L}{\operatorname{argmax}} \frac{P(O \mid W)P(W)}{P(O)}$$

- Since denominator is the same for each candidate sentence W, we can ignore it for the argmax:

$$\hat{W} = \underset{W \in L}{\operatorname{argmax}} P(O \mid W)P(W)$$

Noisy channel model

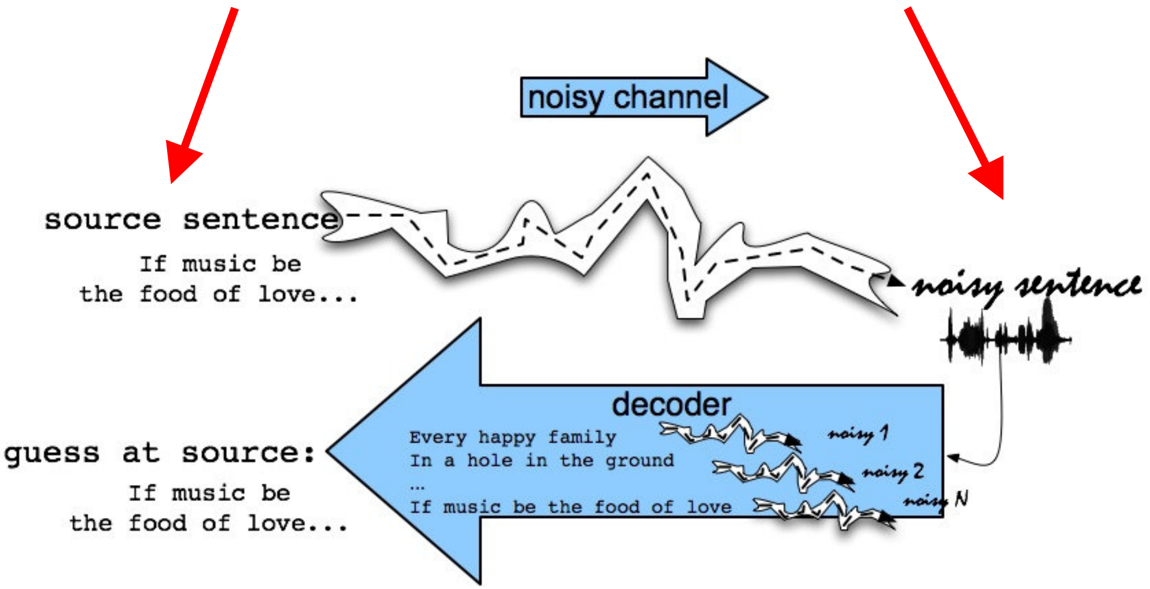
likelihood prior

$$\hat{W} = \underset{W \in L}{\operatorname{argmax}} \underbrace{P(O \mid W)}_{\text{likelihood}} \underbrace{P(W)}_{\text{prior}}$$

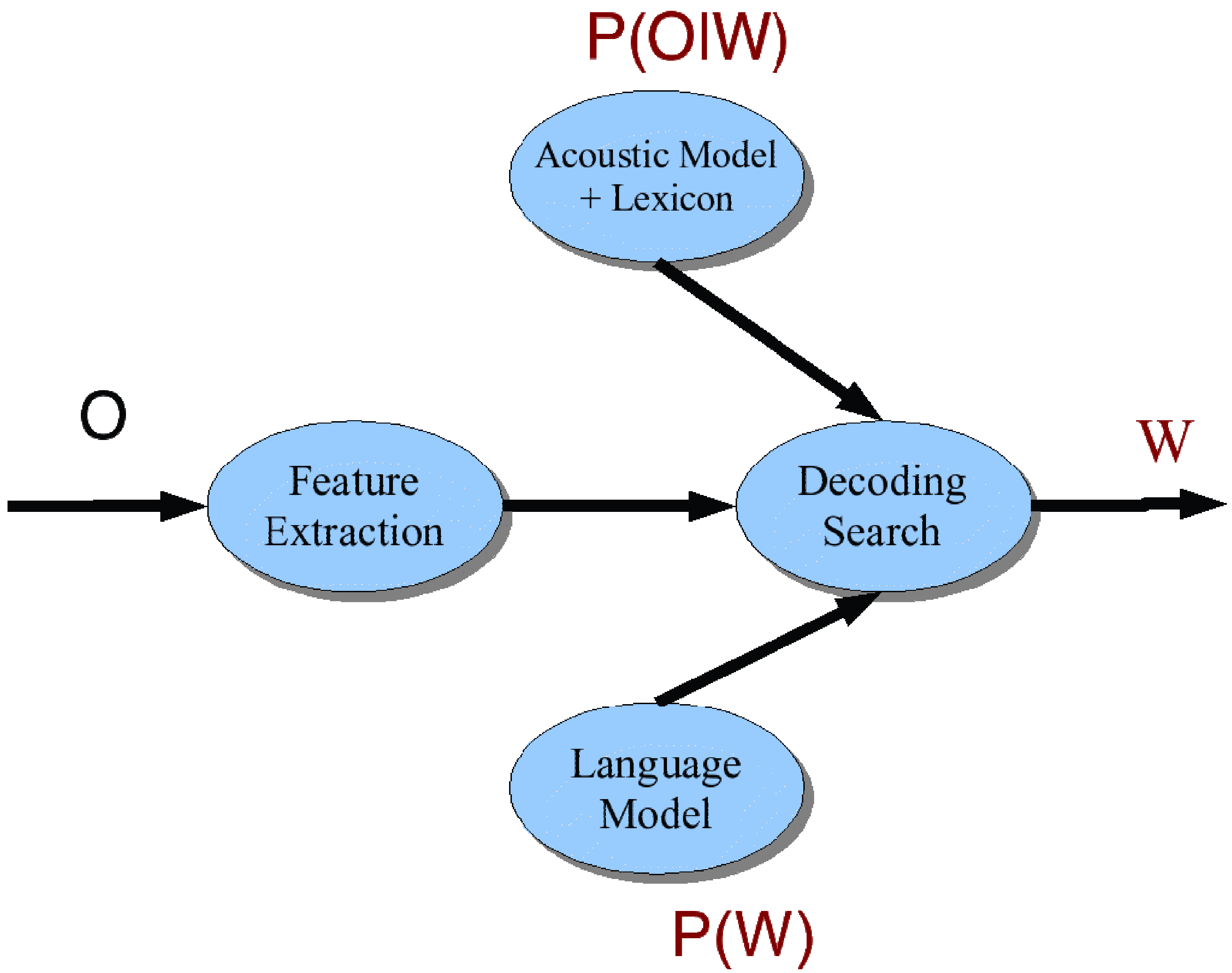
The noisy channel model

- Ignoring the denominator leaves us with two factors:

  P(Source) and P(Signal|Source)



Speech Architecture meets Noisy Channel



Architecture

- HMMs, Lexicons, and Pronunciation
- Feature extraction
- Acoustic Modeling
- Decoding
- Language Modeling (seen this already)

Lexicon

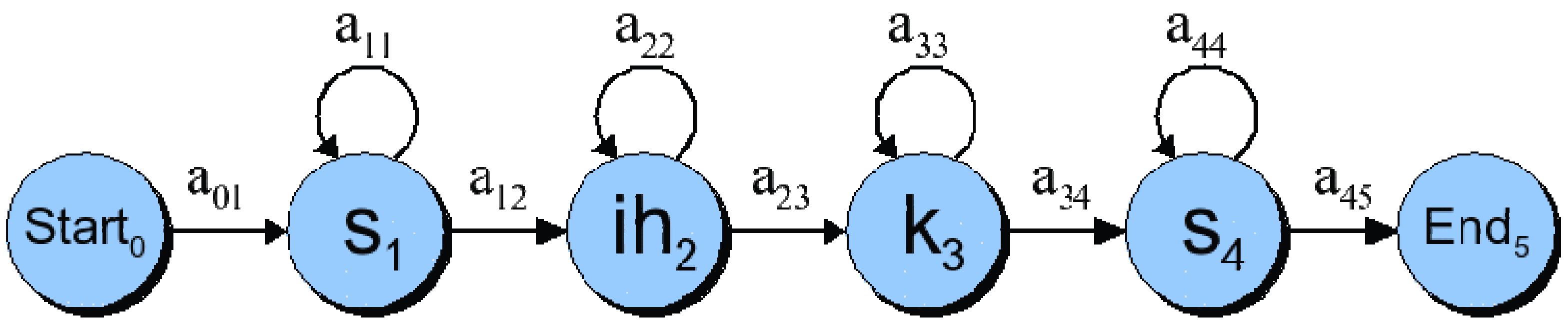
- A list of words
- Each one with a pronunciation in terms of phones
- We get these from on-line pronunciation dictionary, such as CMU dictionary: 127K words
- We'll represent the lexicon as an HMM

HMM Model

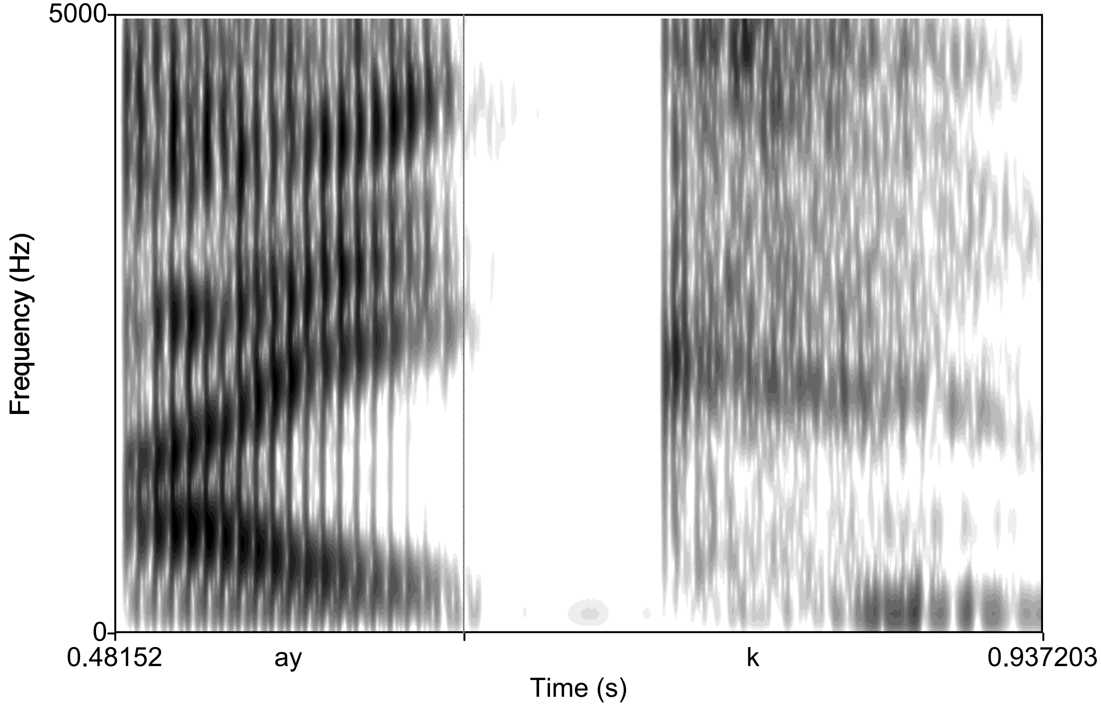
- Per woord
  (aantal afhankelijk van het lexicon)
  - kan alleen voor zeer kleine lexica (cijfers bv)

- Per foneem
  (zonder contekst: ca 40 modellen)
  - contekst heeft wel veel invloed op uitspraak

HMMs for speech: HMM for the word "six"

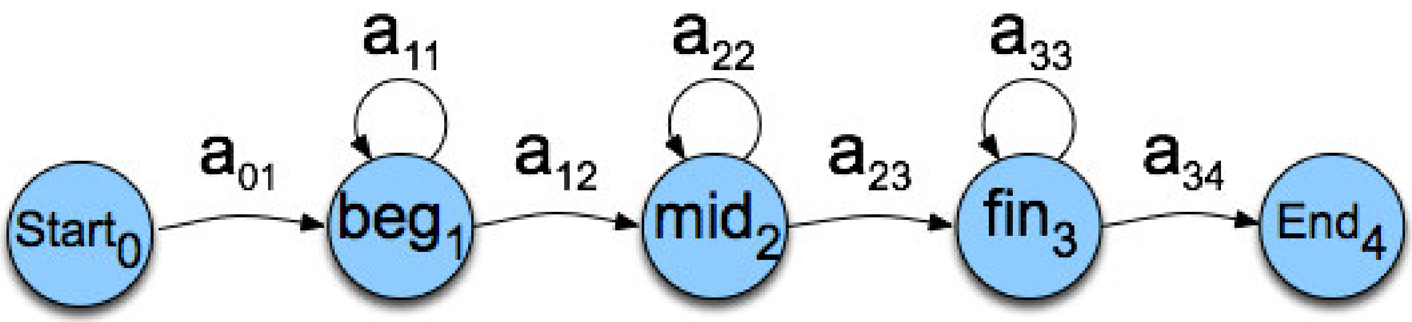


Spectra in phones are not homogeneous!



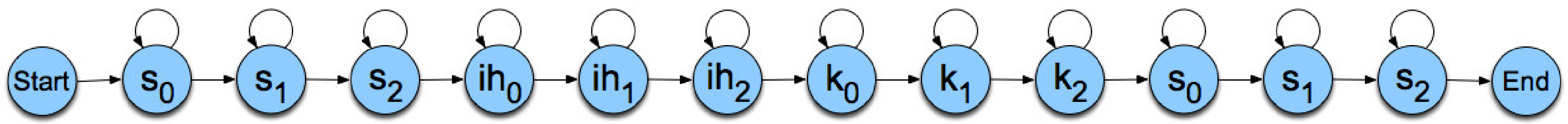
ay (Ike) k

Each phone has 3 subphones

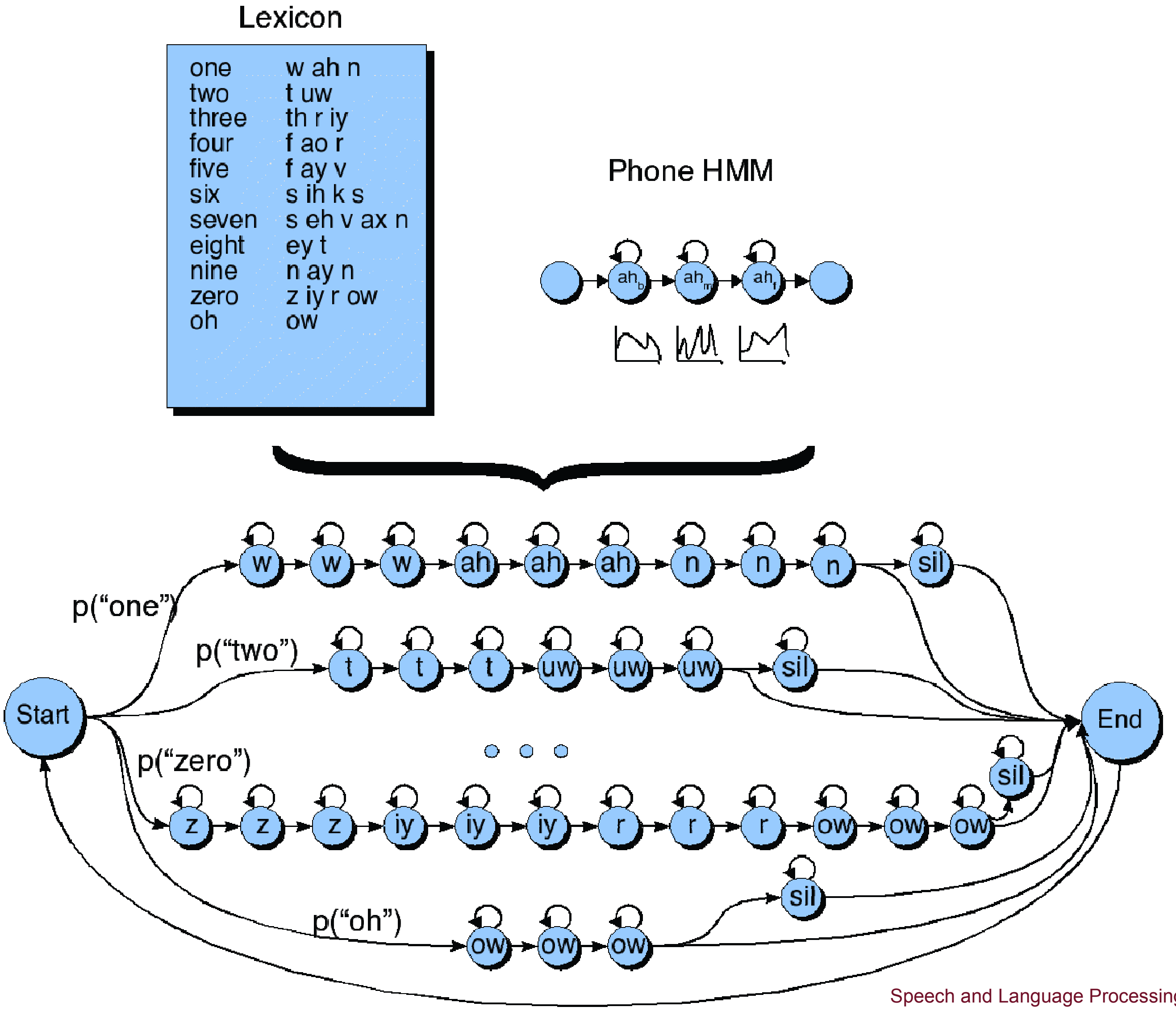


HMM van een foneem, met drie states:
begin + midden + eind

Resulting HMM word model for "six" with their subphones



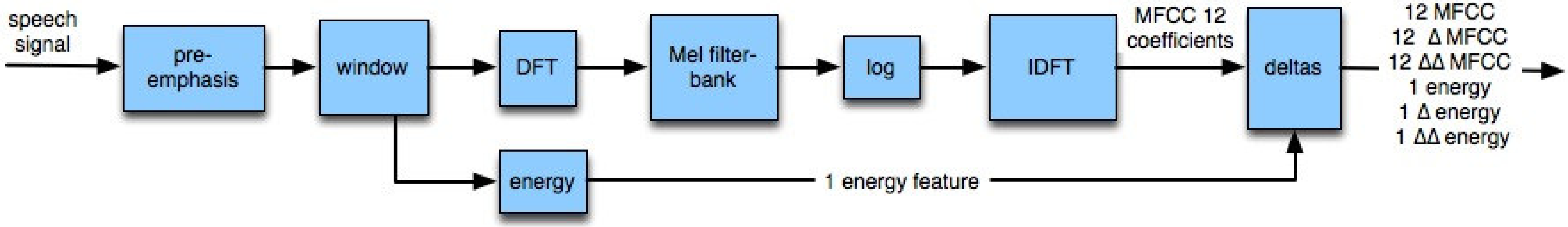
HMM for the digit recognition task



Detecting Phones

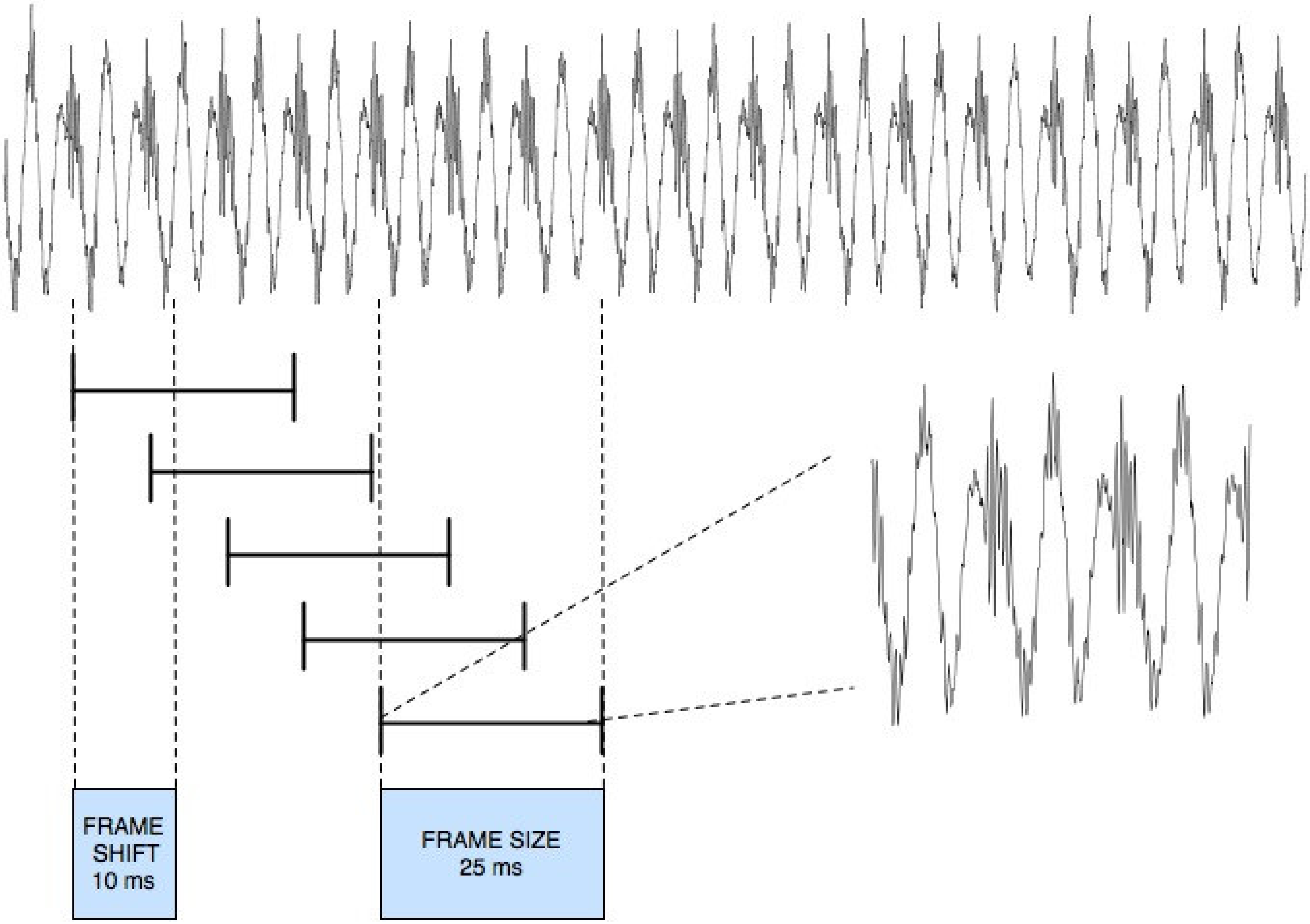
- Two stages
  - Feature extraction
    - Relevante eigenschappen uit het akoestisch spraaksignaal extraheren
    - Basically a slice of a spectrogram
  - Building a phone classifier

Feature extraction MFCC: Mel-Frequency Cepstral Coefficients

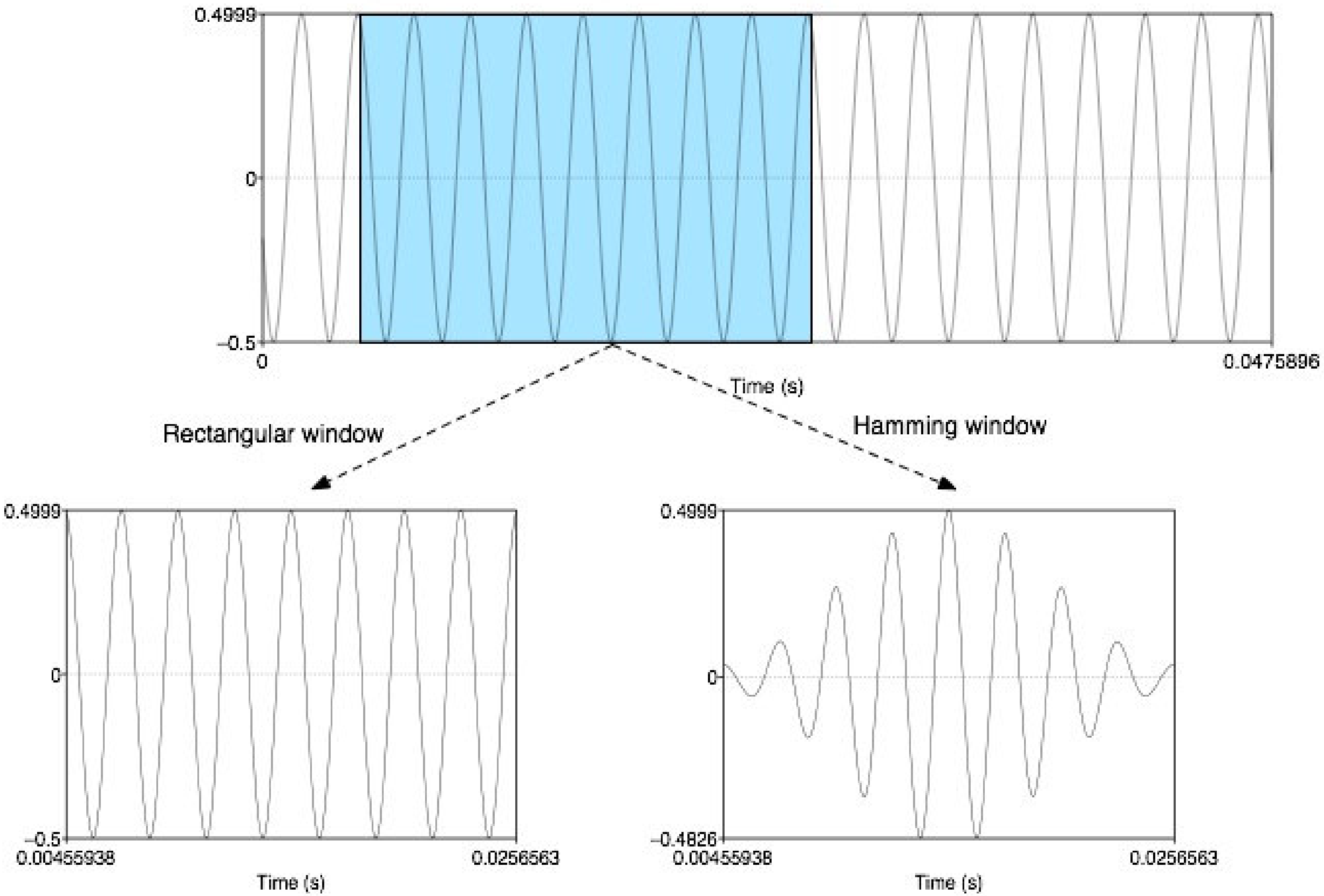


Geïnspireerd door menselijke geluidverwerking:
- ~logaritmische frequentie as: mel-schaal
- Spectrale omhullende met formanten: berekenen als cepstrumcoëfficienten (laten we verder buiten beschouwing)

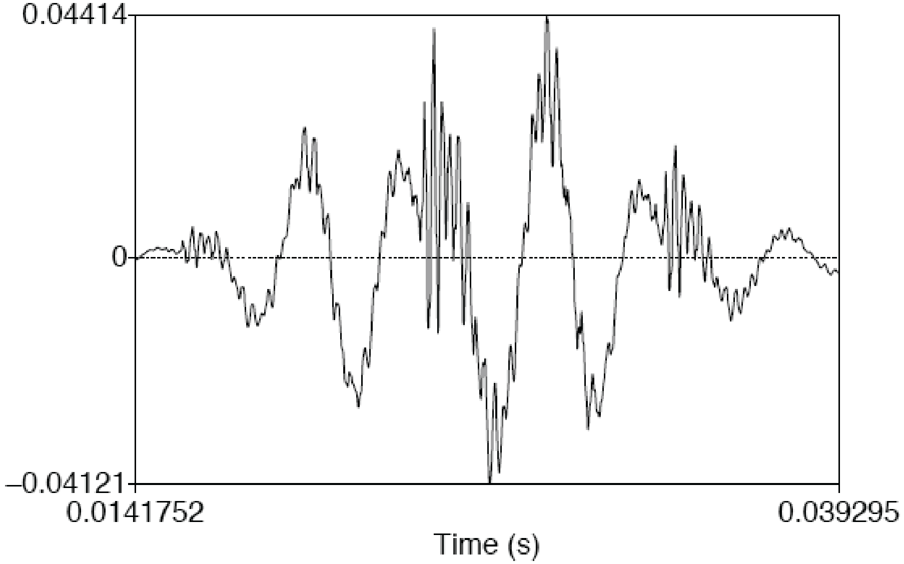
MFCC process: windowing

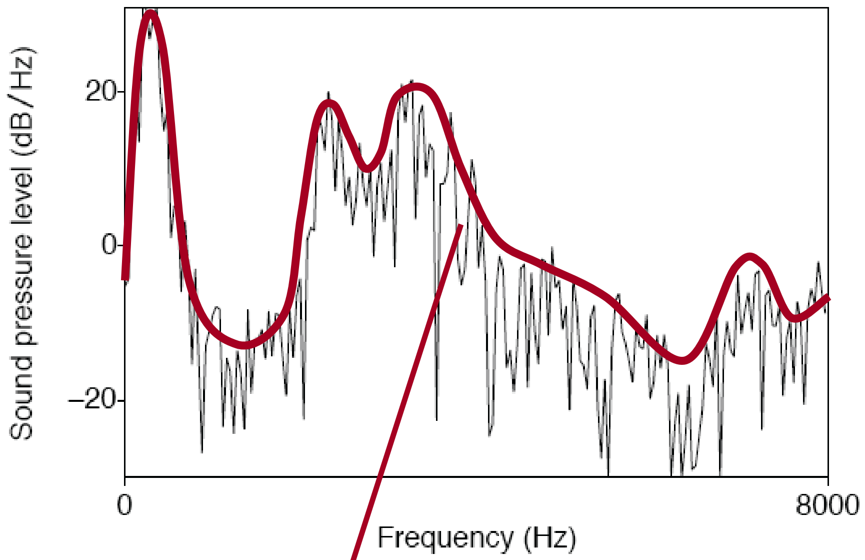


MFCC process: windowing



Hamming window on the signal, and then computing the spectrum





Het cepstrum beschrijft het omhullende spectrum

Eigenschappen per frame

- 12 MFCC coefficienten (spectrum)
- 1 energie niveau

- Om variatie in tijd te "vangen" ook het verschil met het vorige frame opnemen: Delta-MFCC, Delta-energie

- En om variatie in de variatie mee te nemen, dat nogmaals doen: Delta-Delta-MFCC, Delta-Delta-energie

Final Feature Vector

- 39 Features per 10 ms frame:
  - 12 MFCC features
  - 12 Delta MFCC features
  - 12 Delta-Delta MFCC features
  - 1 (log) frame energy
  - 1 Delta (log) frame energy
  - 1 Delta-Delta (log frame energy)
- So each frame represented by a 39D vector

Acoustic Modeling (= Phone detection)

- Given a 39-dimensional vector corresponding to the observation of one frame o_i
- And given a phone q we want to detect
- Compute $p(o_i|q)$
- Most popular method:
  - GMM (Gaussian mixture models)
- Other methods
  - Neural nets, CRFs, SVM, etc

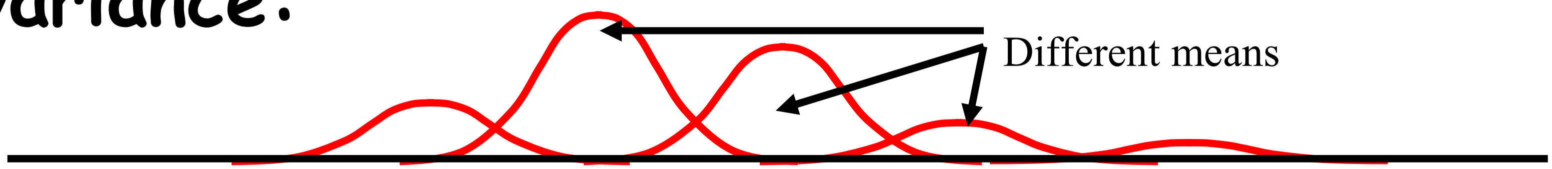
Gaussische verdeling

- Ook wel Normale verdeling genoemd
- Kenmerken:
  - Gemiddelde
  - Spreiding

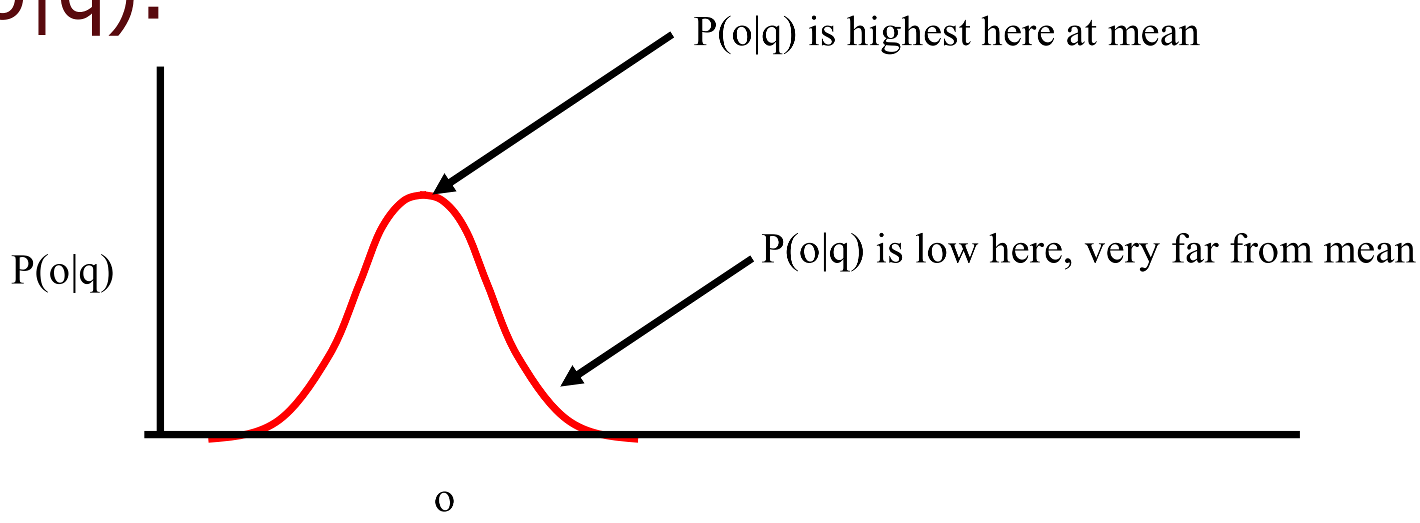
- Een variabele in de spraakvector heeft geen vaste waarde, maar spreidt rond een gemiddelde (in een HMM state)

Gaussians for Acoustic Modeling

A Gaussian is parameterized by a mean and a variance:



- P(o|q):



Complex!

- Niet 1 maar 39 variabelen (en verdelingen)
- Per variabele is 1 normale verdeling vaak niet genoeg, maar moeten er meer zijn (mixture)

Where we are

- Given: A wave file
- Goal: output a string of words
- What we know: the **acoustic model**
  - How to turn the wavefile into a sequence of acoustic feature vectors, one every 10 ms
  - If we had a complete phonetic labeling of the training set, we know how to train a gaussian "phone detector" for each phone.
  - We also know how to represent each word as a sequence of phones
- What we knew from Chapter 4: **the language model**
- To do:
  - Seeing all this back in the context of HMMs
  - Search: how to combine the language model and the acoustic model to produce a sequence of words

Decoding

- In principle:

$$\hat{W} = \underset{W \in \mathscr{L}}{\operatorname{argmax}} \; \overbrace{P(O|W)}^{\text{likelihood}} \; \overbrace{P(W)}^{\text{prior}}$$

- In practice:

$$\hat{W} = \underset{W \in \mathscr{L}}{\operatorname{argmax}} \, P(O|W) P(W)^{LMSF}$$

scale factor

(spraak frames zijn niet onafhankelijk)

$$\hat{W} = \underset{W \in \mathscr{L}}{\operatorname{argmax}} \, P(O|W) P(W)^{LMSF} \, WIP^{N}$$

word insertion penalty

(lange zinnen worden anders minder waarschijnlijk)

$$\hat{W} = \underset{W \in \mathscr{L}}{\operatorname{argmax}} \log P(O|W) + LMSF \times \log P(W) + N \times \log WIP$$

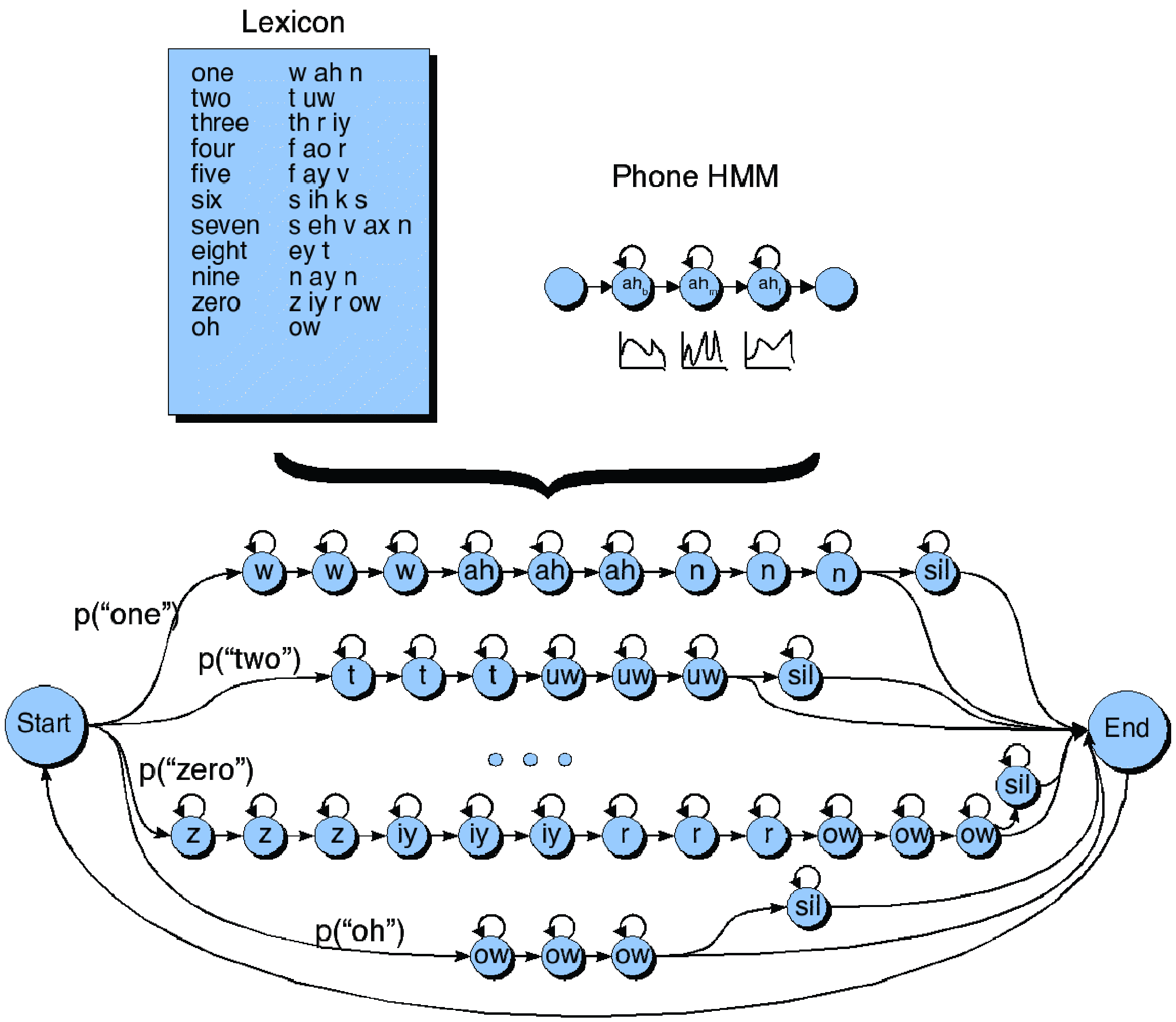
Why is ASR decoding hard?

[ay d ih s hh er d s ah m th ih ng ax b aw m uh v ih ng r ih s en l ih]

HMMs for speech

$Q = q_1 q_2 \ldots q_N$	a set of **states** corresponding to **subphones**
$A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability for each subphone of taking a **self-loop** or going to the next subphone. Together, Q and A implement a **pronunciation lexicon**, an HMM state graph structure for each word that the system is capable of recognizing.
$B = b_i(o_t)$	A set of **observation likelihoods:**, also called **emission probabilities**, each expressing the probability of a cepstral feature vector (observation o_t) being generated from subphone state i.

HMM for digit recognition task



The Evaluation (forward) problem for speech

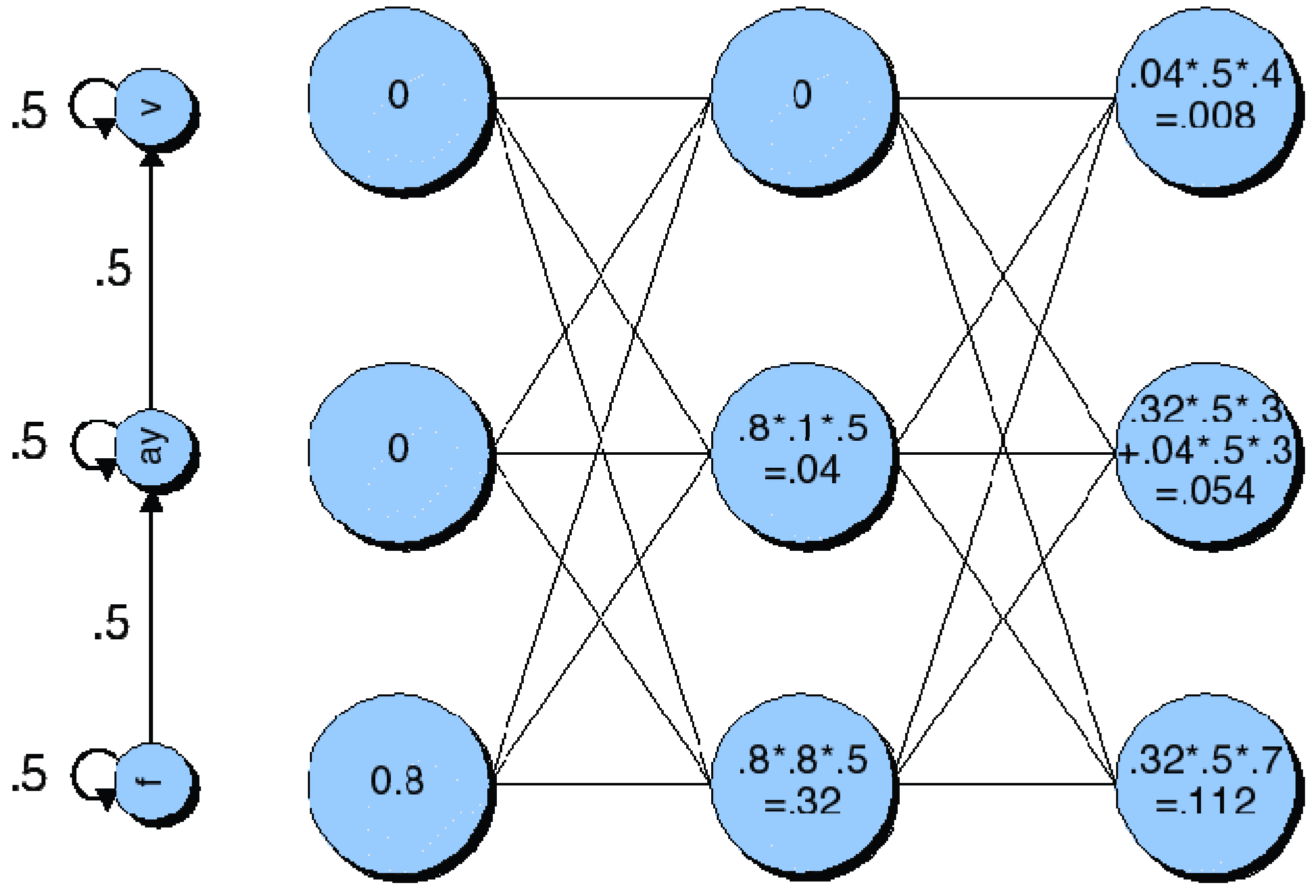
- The observation sequence O is a series of MFCC vectors
- The hidden states W are the phones and words
- For a given phone/word string W, our job is to evaluate P(O|W)
- Intuition: how likely is the input to have been generated by just that word string W

  [dit is een pad maximalisatie probleem]

Evaluation for speech: Summing over all different paths!

- f ay ay ay ay v v v v
- f f ay ay ay ay v v v
- f f f f ay ay ay ay v
- f f ay ay ay ay ay ay v
- f f ay ay ay ay ay ay ay ay v
- f f ay v v v v v v v

The forward lattice for “five”



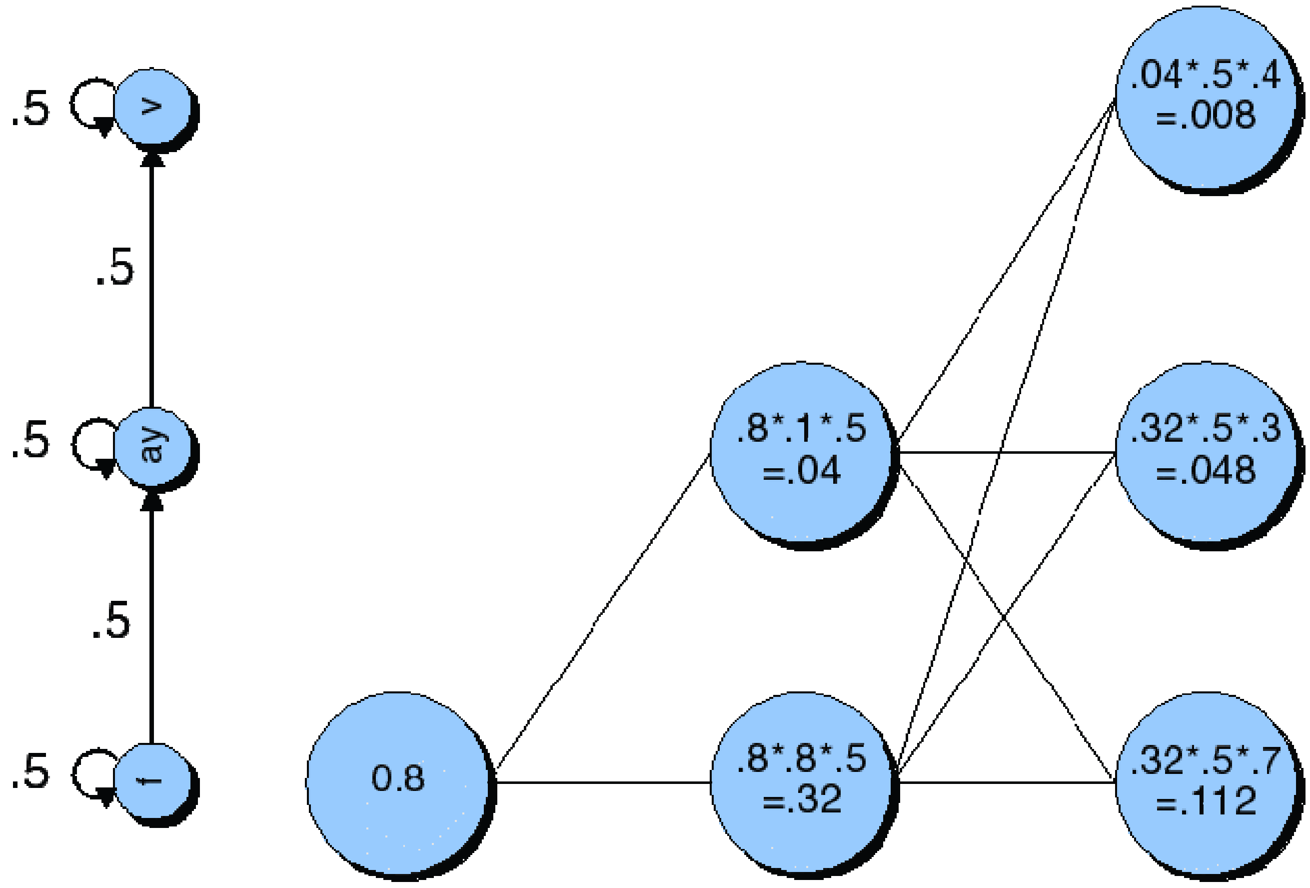
The forward trellis for “five”

Waarden voorwaardse variabele

V	0	0	0.008	0.0093	0.0114	0.00703	0.00345	0.00306	0.00206	0.00117
AY	0	0.04	0.054	0.0664	0.0355	0.016	0.00676	0.00208	0.000532	0.000109
F	0.8	0.32	0.112	0.0224	0.00448	0.000896	0.000179	4.48e-05	1.12e-05	2.8e-06
Time	1	2	3	4	5	6	7	8	9	10
B	*f* 0.8 *ay* 0.1 *v* 0.6 *p* 0.4 *iy* 0.1	*f* 0.8 *ay* 0.1 *v* 0.6 *p* 0.4 *iy* 0.1	*f* 0.7 *ay* 0.3 *v* 0.4 *p* 0.2 *iy* 0.3	*f* 0.4 *ay* 0.8 *v* 0.3 *p* 0.1 *iy* 0.6	*f* 0.4 *ay* 0.8 *v* 0.3 *p* 0.1 *iy* 0.6	*f* 0.4 *ay* 0.8 *v* 0.3 *p* 0.1 *iy* 0.6	*f* 0.4 *ay* 0.8 *v* 0.3 *p* 0.1 *iy* 0.6	*f* 0.5 *ay* 0.6 *v* 0.6 *p* 0.1 *iy* 0.5	*f* 0.5 *ay* 0.5 *v* 0.8 *p* 0.3 *iy* 0.5	*f* 0.5 *ay* 0.4 *v* 0.9 *p* 0.3 *iy* 0.4

Output waarschijnlijkheden

Viterbi trellis for “five”



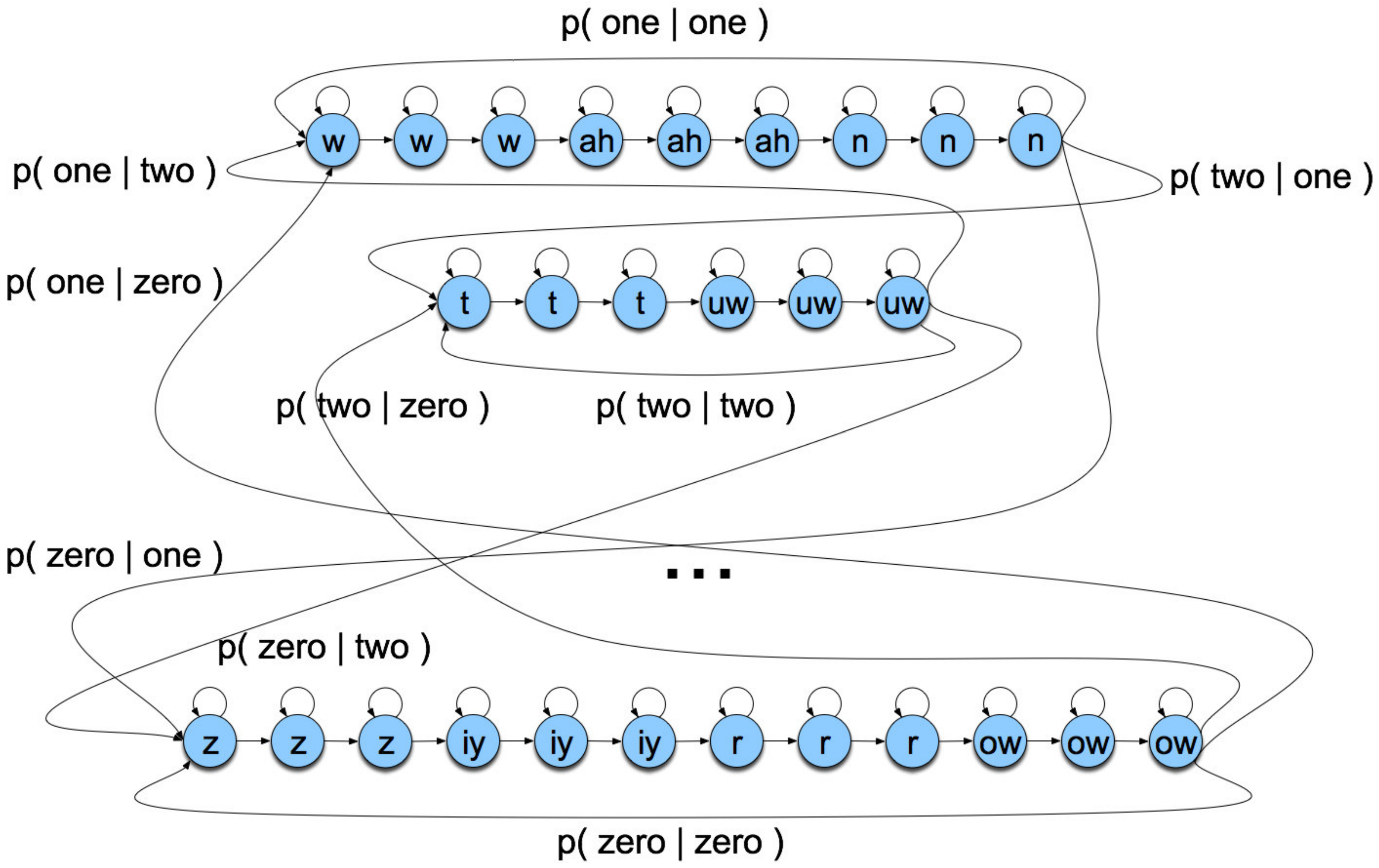
Viterbi trellis for "five"

Waarden viterbi variabele

V	0	0	0.008	0.0072	0.00672	0.00403	0.00188	0.00161	0.000667	0.000493
AY	0	0.04	0.048	0.0448	0.0269	0.0125	0.00538	0.00167	0.000428	8.78e-05
F	0.8	0.32	0.112	0.0224	0.00448	0.000896	0.000179	4.48e-05	1.12e-05	2.8e-06
Time	1	2	3	4	5	6	7	8	9	10
B	*f* 0.8 *ay* 0.1 *v* 0.6 *p* 0.4 *iy* 0.1	*f* 0.8 *ay* 0.1 *v* 0.6 *p* 0.4 *iy* 0.1	*f* 0.7 *ay* 0.3 *v* 0.4 *p* 0.2 *iy* 0.3	*f* 0.4 *ay* 0.8 *v* 0.3 *p* 0.1 *iy* 0.6	*f* 0.4 *ay* 0.8 *v* 0.3 *p* 0.1 *iy* 0.6	*f* 0.4 *ay* 0.8 *v* 0.3 *p* 0.1 *iy* 0.6	*f* 0.4 *ay* 0.8 *v* 0.3 *p* 0.1 *iy* 0.6	*f* 0.5 *ay* 0.6 *v* 0.6 *p* 0.1 *iy* 0.5	*f* 0.5 *ay* 0.5 *v* 0.8 *p* 0.3 *iy* 0.5	*f* 0.5 *ay* 0.4 *v* 0.9 *p* 0.3 *iy* 0.4

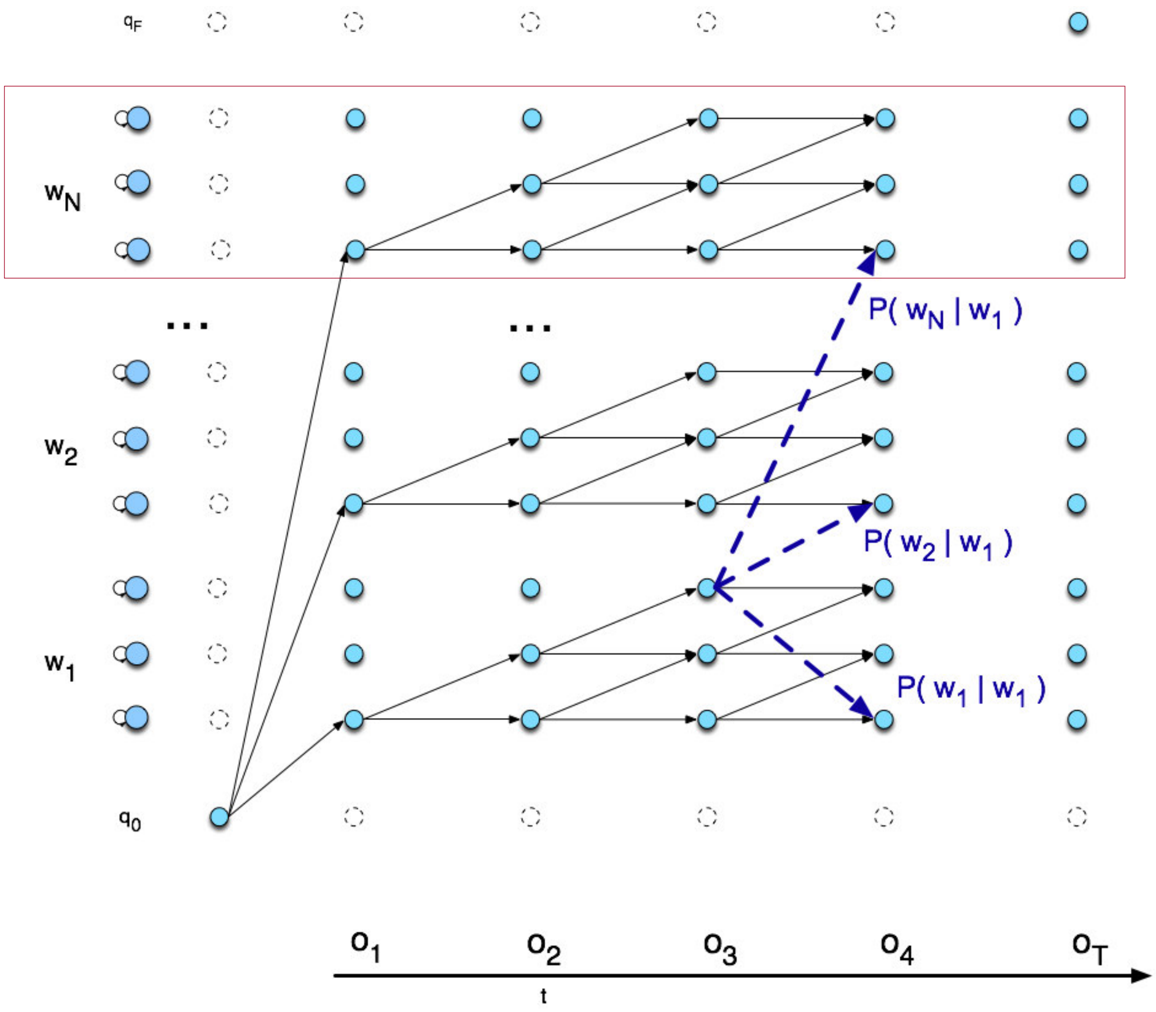
Search space with bigrams

[verbindingen tussen woorden]



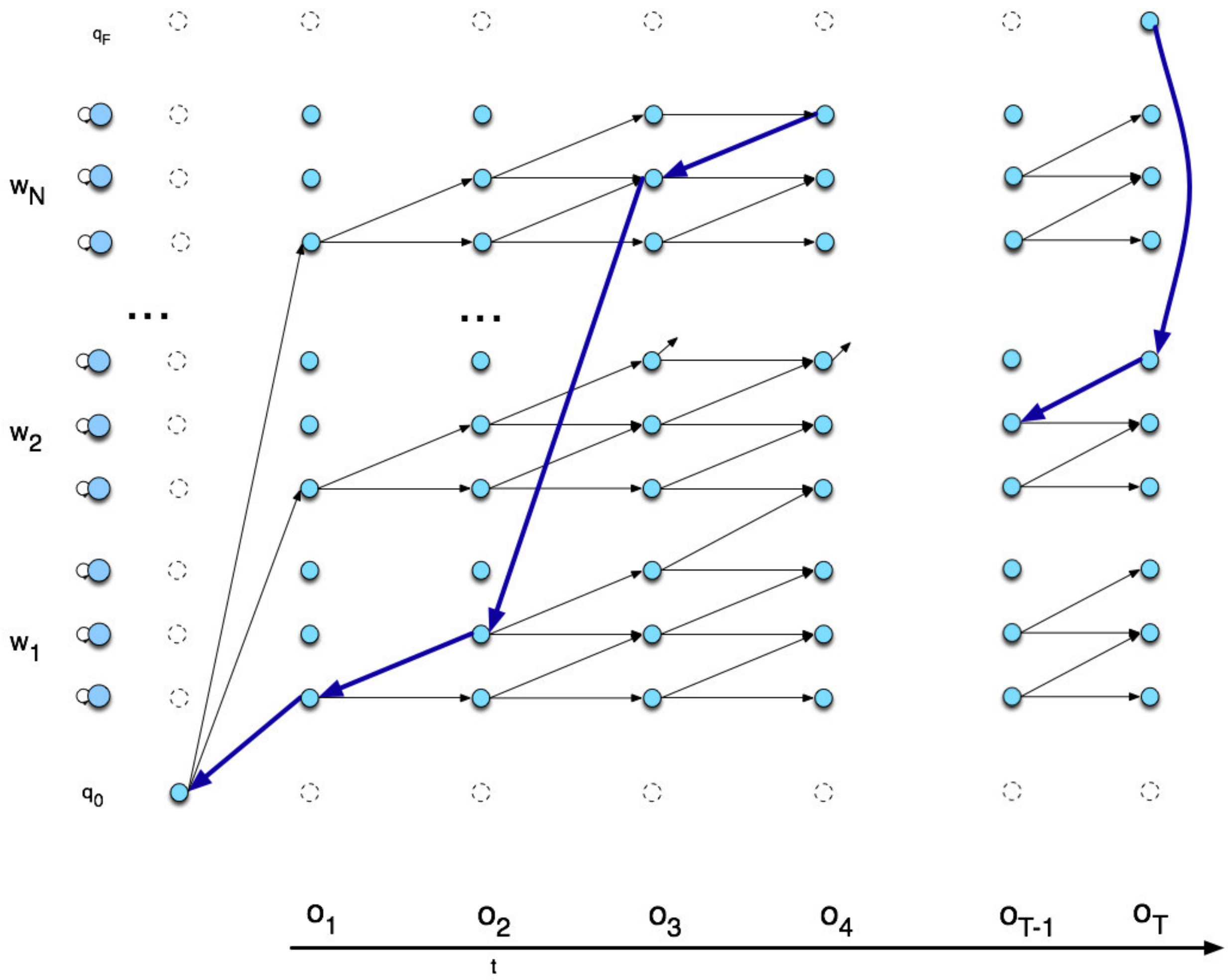
Viterbi trellis

[bij meerdere verbonden woorden]



netwerk per woord

Viterbi backtrace

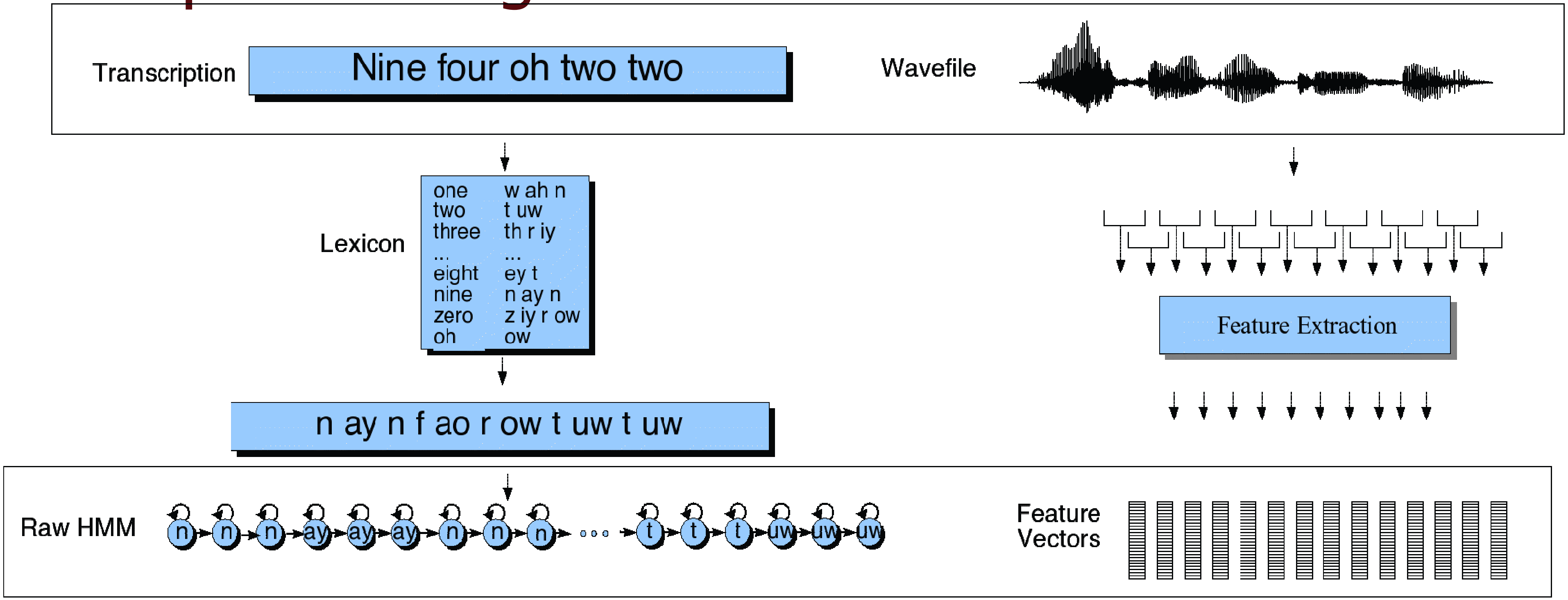


Training

- Gelabelde spraak
  - tijdrovend en duur
  - zeker voor subphones niet mogelijk
  - maar ‘tellen’ zou dan wel voldoende zijn
- Embedded training
  - Op basis van
    - Tekst
    - Spraaksignaal
    - +uitspraaklexicon
    - +model

Embedded training

- Forward-backward algoritme vindt optimale segmentatie zelf



Evaluation

- How to evaluate the word string output by a speech recognizer?

Word Error Rate

- Word Error Rate =

$$\frac{100 * (\text{Insertions}+\text{Substitutions} + \text{Deletions})}{\text{Total Word in Correct Transcript}}$$

Aligment example:

REF: portable **** PHONE UPSTAIRS last night so

HYP: portable FORM OF STORES last night so

Eval I S S

WER = 100 (1+2+0)/6 = 50%

NIST sctk-1.3 scoring softare: Computing WER with sclite

- http://www.nist.gov/speech/tools/
- Sclite aligns a hypothesized text (HYP) (from the recognizer) with a correct or reference text (REF) (human transcribed)

```
id: (2347-b-013)
Scores: (#C #S #D #I) 9 3 1 2
REF:  was an engineer SO I   i was always with **** **** MEN UM   and they
HYP:  was an engineer ** AND i was always with THEM THEY ALL THAT and they
Eval:                 D  S                     I    I    S   S
```

Oplijnen zelf is weer een dynamic programming taak!

Sclite output for error analysis

```
CONFUSION PAIRS                  Total                 (972)
                                 With >=  1 occurances (972)

   1:    6  ->  (%hesitation) ==> on
   2:    6  ->  the ==> that
   3:    5  ->  but ==> that
   4:    4  ->  a ==> the
   5:    4  ->  four ==> for
   6:    4  ->  in ==> and
   7:    4  ->  there ==> that
   8:    3  ->  (%hesitation) ==> and
   9:    3  ->  (%hesitation) ==> the
  10:    3  ->  (a-) ==> i
  11:    3  ->  and ==> i
  12:    3  ->  and ==> in
  13:    3  ->  are ==> there
  14:    3  ->  as ==> is
  15:    3  ->  have ==> that
  16:    3  ->  is ==> this
```

Better metrics than WER?

- WER has been useful
- But should we be more concerned with meaning (“semantic error rate”)?
    - Good idea, but hard to agree on
    - Has been applied in dialogue systems, where desired semantic output is more clear

Summary: ASR Architecture

- Five "easy" pieces: ASR Noisy Channel architecture

1) Feature Extraction:
   39 "MFCC" features
2) Acoustic Model:
   Gaussians for computing p(o|q)
3) Lexicon/Pronunciation Model
   - HMM: what phones can follow each other
4) Language Model
   - N-grams for computing $p(w_i|w_{i-1})$
5) Decoder
   - Viterbi algorithm: dynamic programming for combining all these to get word sequence from speech!