EDAN20
Language Technology
http://cs.lth.se/edan20/
Chapter 19: Speech Recognition

Pierre Nugues

Lund University
Pierre.Nugues@cs.lth.se
http://cs.lth.se/pierre_nugues/

October 10, 2016

Speech Recognition

Conditions to take into account:

- Number of speakers
- Fluency of speech.
- Size of vocabulary
- Syntax
- Environment

Structure of Speech Recognition

Words:

$$W = w_1, w_2, \ldots, w_n.$$

Acoustic symbols:

$$A = a_1, a_2, \ldots, a_m,$$

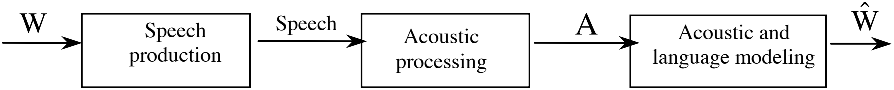
$$\hat{W} = \arg\max_{W} P(W|A).$$

Using Bayes' formula,

$$P(W|A) = \frac{P(A|W)P(W)}{P(A)}.$$

Two-Step Recognition

$$\hat{W} = \arg\max_{W} P(A|W)P(W).$$



Speech Parameters

Recognition devices derive a set of acoustic parameters from speech frames.
Parameters should be related to “natural” features of speech: voiced or unvoiced segments.
A simple parameter giving a rough estimate of it: the energy: the darker the frame, the higher the energy.

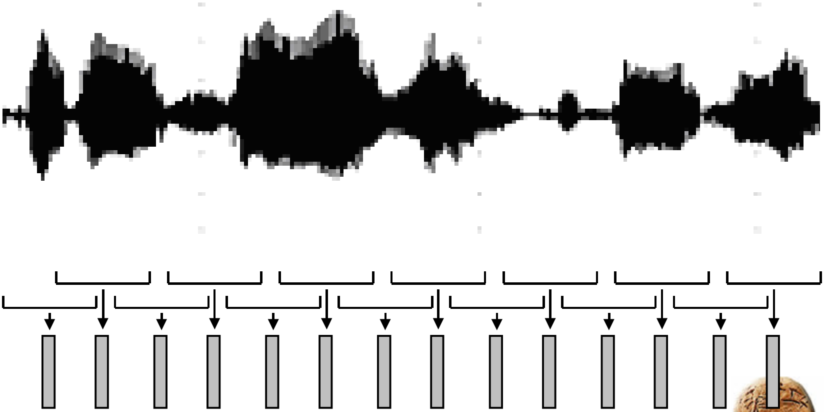
$$E(F_k) = \sum_{n=m}^{m+N-1} s^2(n).$$

Linear prediction coefficients:

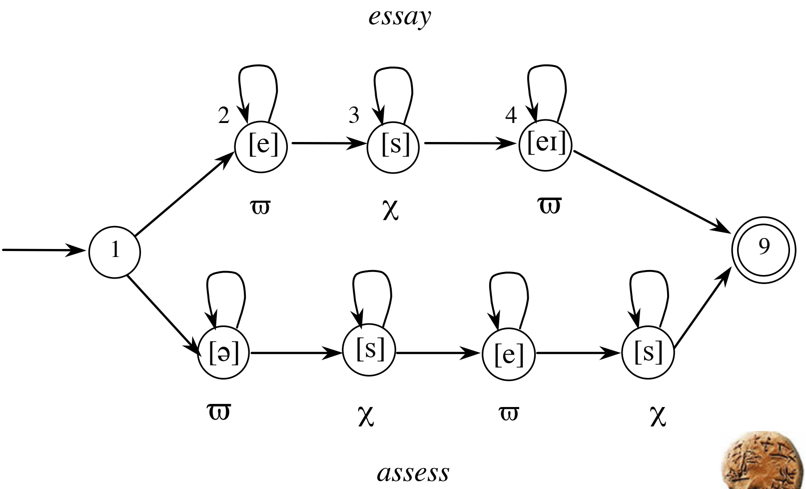
$$\hat{s}(n) = a(1)s(n-1) + a(2)s(n-2) + a(3)s(n-3) + ... + a(m)s(n-m),$$

Extraction of Speech Parameters

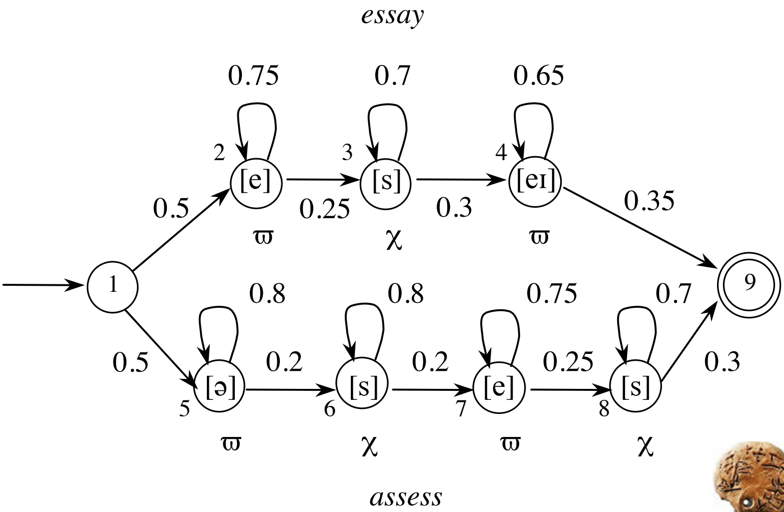
Features are extracted every 10 ms over a 20 s frame



Automata



Markov Chains



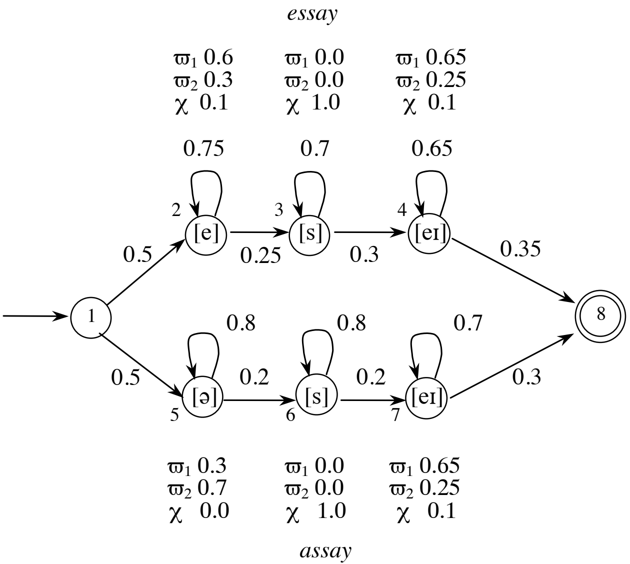
A Markov Chain in Prolog

```
start(q1). final(q9).
transition(q1, o, q2, 0.5).
transition(q2, o, q2, 0.75).
transition(q2, k, q3, 0.25).
transition(q3, k, q3, 0.7).
transition(q3, o, q4, 0.3).
transition(q4, o, q4, 0.65).
transition(q1, o, q5, 0.5).
transition(q5, o, q5, 0.8).
transition(q5, k, q6, 0.2).
transition(q6, k, q6, 0.8).
transition(q6, o, q7, 0.2).
transition(q7, o, q7, 0.75).
transition(q7, k, q8, 0.25).
transition(q8, k, q8, 0.7).

silent(q4, q9, 0.35). silent(q8, q9, 0.3).

accept(Symbols, Probability) :-
  start(StartState),
  accept(Symbols, StartState, 1.0, Probability).
```

A Markov Chain in Prolog

```
accept([], State, Probability, Probability) :-
  final(State).
accept([Symbol | Symbols], State, ProbIn, ProbOut) :-
  transition(State, Symbol, NextState, ProbTrans),
  NextProb is ProbIn * ProbTrans,
  write(NextProb), nl,
  accept(Symbols, NextState, NextProb, ProbOut).
accept(Symbols, State, ProbIn, ProbOut) :-
  silent(State, NextState, ProbTrans),
  NextProb is ProbIn * ProbTrans,
  accept(Symbols, NextState, NextProb, ProbOut).
```

Hidden-Markov Models



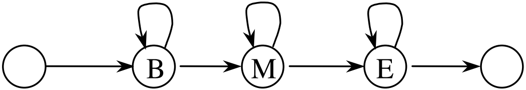
Solving Problems with Hidden-Markov Models

Given a hidden-Markov model, the main problems to solve are to:

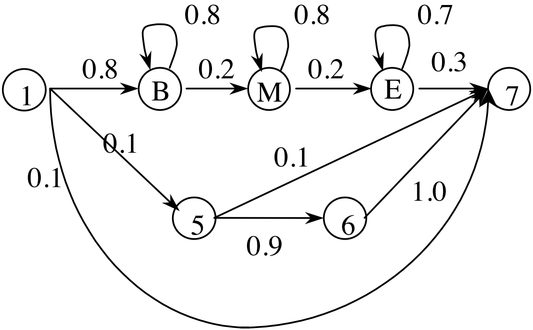
- Estimate the probability of an observed sequence. It corresponds to the sum of all the paths producing the observation. It is solved using the forward algorithm.
- Determine the most likely path of an observed sequence. It is a decoding problem. It is solved using the Viterbi algorithm.
- Determine (learn) the parameters given a set of observations. It is used to build models to recognize speech. It is solved using the forward-backward algorithm.

HMM and Phones

Modeling phones:
Simple model



A more complex model due to Lee



Word Decoding

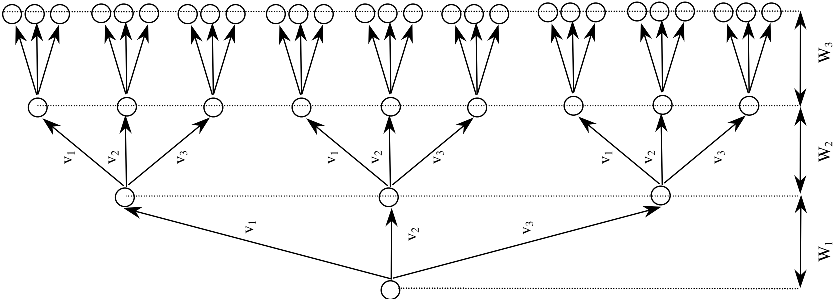
Markov models are a probabilistic mapping of a string of acoustic symbols $a_1, a_2, ..., a_m$ onto a string of phonemes $\varphi_1, \varphi_2, ..., \varphi_m$.
A language model applies a second probability to a word sequence.
The complete speech recognition then consists in decoding word sequences $w_1, w_2, ..., w_n$ from phonemic strings and weighting them using the language model.

$$
\begin{array}{lccccccccc}
\textit{words} & \multicolumn{3}{c}{\overbrace{\hspace{8em}}^{w_1}} & \multicolumn{3}{c}{\overbrace{\hspace{8em}}^{w_2}} & & \multicolumn{2}{c}{\overbrace{\hspace{6em}}^{w_j}} \\
\textit{phonemes} & \varphi_1^1 & \varphi_1^2 & \varphi_1^{m1} & \varphi_2^1 & \varphi_2^2 & \varphi_2^{m2} & & \varphi_j^1 & \varphi_j^{mj} \\
\textit{ac.symbols} & a_1^1, & a_1^2, \ldots & a_1^{m1}, & a_2^1, & a_2^2, \ldots & a_2^{m2}, & \ldots, & a_j^1, \ldots, & a_j^{mj}, \ldots,
\end{array}
$$

Searching Words

A hypothesis search.

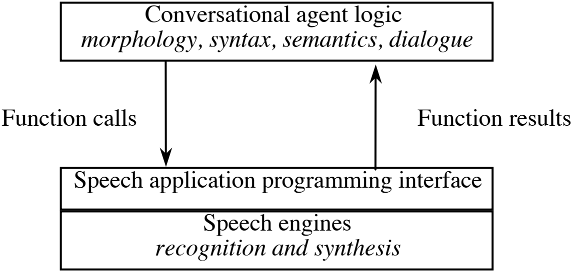
If the vocabulary contains k words $v_1, v_2, ..., v_k$, w_1 is to be selected amongst k possibilities, w_2 amongst k possible choices again and so on.



Decoding uses the A* algorithm

Commercial Systems

Speech recognition systems are accessible using an API



In addition to a language model, speech engines often give the possibility to use a phrase–structure grammar

A Phrase–Structure Grammar for the IBM ViaVoice

```
<kiosk> =  <greeting1>? <greeting2>? <sentence1>
  | <greeting1>? <sentence2> .
<greeting1> =  hello | excuse me | excuse me but .
<greeting2> =  can you tell me | I need to know
  | please tell me .
<sentence1> =  where <destination1> is located
  | where is <destination1>
  | where am I
  | when will <transportation> <destination2>? arrive
  | when <transportation> <destination2>? will arrive
  | what time it is
  | the local time
  | the phone number of <destination1>
  | the cost of <transportation> <destination2>? .
```

A Phrase–Structure Grammar for the IBM ViaVoice

```
<sentence2> =  I am lost
  | I need help
  | please help me
  | help
  | help me
  | help me please .
<destination1> = a restaurant
  | the <RestaurantType> restaurant
  | <BusinessType>? <BusinessName> .
<RestaurantType> = best | nearest | cheapest | fastest .
<BusinessType> = a | the nearest .
<BusinessName> = filling station
  | public rest room
  | police station .
```

A Phrase–Structure Grammar for the IBM ViaVoice

```
<transportation> = the <TransportType>? <TransportName> .
<TransportType> = next | first |last .
<TransportName> = bus | train .
<destination2> = to metro central
  | to union station
  | to downtown
  | to national airport .
```