Text-to-speech synthesis

- What is a TTS system?
  - A system which takes as input a sequence of words and converts them to speech

- Applications
  - Services for the hearing impaired
  - Reading email aloud
  - Dialogue Systems

- Commercial systems
  - Festival
  - Bell Labs TTS
  - Google speech synthesis API
Text-to-speech synthesis

- High-level TTS architecture

**NATURAL LANGUAGE PROCESSING**
- Linguistic Formalism
- Inference Engines
- Logical Inferences

**DIGITAL SIGNAL PROCESSING**
- Mathematical Models
- Algorithms
- Computations

Text → Phonemes → Prosody → Speech
From words to linguistic specification

- From multi-level / tiered linguistic information to a linear string of context-dependent units

phoneme: ax
left context: sil dh
right context: k ae
position in phrase: initial
syllable stress: unstressed
Speech synthesis as a regression problem

Regression

phrase initial

pitch accent

phrase final

sil dh ax k ae t s ae t sil

"the cat sat"

DET NN VB

((the cat) sat)

sil^dh-ax+k=ae, "phrase initial", "unstressed syllable", ...
Speech synthesis methods

- Expert-based (rule-based) approach
- Corpus-based approach
  - Diphone concatenation
    - A concatenative speech synthesis method in which just one example of each diphone is stored in the database
  - Unit Selection
    - A concatenative speech synthesis method in which multiple examples of each diphone are stored in the database
- Statistical parametric synthesis
Speech synthesis methods

1970s: Formant synthesis
Model based (parametric)

1980s-90
Diphone synthesis

1990s-2000s
Unit selection

2000s
HMM based synthesis

Improved quality

Unit-based (concatenative)

Improved control
Diphone synthesis

- Most important for natural sounding speech is to get the transitions right (allophonic variation, coarticulation effects)
- These are found at the boundary between phoneme segments
- Diphones are the second half of one phone plus the first half of the following phone
- There is one join per phone
- Concatenation points (joins) are in the mid-phone position
- If a language has P phones, then number of diphones is \( \sim P^2 \) (some combinations impossible) – eg 800 for Spanish, 1200 for French, 2500 for German)
Diphone synthesis

➢ If a language has P phones, then number of diphones is \( \sim P^2 \) (some combinations impossible) – eg 800 for Spanish, 1200 for French, 2500 for German

➢ Most systems use diphones because they are
  ➢ Manageable in number
  ➢ Can be automatically extracted from recordings of human speech
  ➢ Capture most inter-allophonic variants

➢ But they do not capture all coarticulatory effects, so some systems include triphones, as well as fixed phrases and other larger units (= USS)
Concatenative synthesis

**Input data** → **Unit segmentation** → **Unit database**

Speech recordings

**Text input** → **Unit selection** → **Concatenation + smoothing** → **Synthesised speech**

... + [signal] + ...
Concatenative synthesis

Whole speech unit database

Target cost is measured by a heuristic distance between contexts

Selected speech units

Target cost

Concatenation cost
Concatenative synthesis

Unit Selection: Viterbi Search

Text
Phones

Symbolic Features

Acoustic Tree
Leave IDs

Acoustic Candidates

Selected Segments
Concatenation vs generation from a model

- **Concatenation** builds up the utterance from units of recorded speech.

- **Generation** uses a sequence of models to generate speech.
Statistical parametric speech synthesis

- From speech coding to speech synthesis

Speech input

- extract spectrum, F0, aperiodic energy
- transmit, compress, modify, ...
- reconstruct

Speech output
Statistical parametric speech synthesis

From speech coding to speech synthesis

Speech input
- extract spectrum, F0, aperiodic energy
- learn model
- reconstruction

Text input
- generate from model
- reconstruct

Stored model

Speech output
Modelling a coded representation of speech

Waveform is not suitable for direct modelling, so use another representation.

Speech waveform

Speech parameters

Speech models

model 1  model 2  model 3  model 4
Speech production mechanism

Modulation of carrier wave by speech information

Frequency transfer characteristics
Magnitude start--end
Fundamental frequency

Speech
Sound source
Voiced: pulse
Unvoiced: noise

air flow
Source-filter model

Source excitation part

- Pulse train
- White noise

Vocal tract resonance part

- Excitation $e(n)$
- Linear time-invariant system $h(n)$

Speech

$x(n) = h(n) * e(n)$

Fourier transform

$X(e^{j\omega}) = H(e^{j\omega})E(e^{j\omega})$

$H(e^{j\omega})$ should be defined by the state-output vector of HMMs or LDMs (e.g., mel-cepstral coefficients, LSP coefficients)
Overview of speech vocoding

Original speech

- F0
- Unvoiced / voiced
- Mel-cepstrum

Synthesis filter $h(n)$

Excitation $e(n)$

Pulse train

White noise

Vocoded speech $x(n)$

These speech parameters modeled by HMMs and LDMs
Speech parameters

F0

![Graph showing voiced and unvoiced sections](image-url)
Speech parameters

- The F0 of an utterance has two components:
  - The global pitch contour shape
  -Localised pitch accents

- Most utterances show an overall downward trend in F0 called declination. We run out of breath, so air flow and pressure decrease and the vocal folds vibrate.
Speech parameters

- The F0 of an utterance has two components:
  - The global pitch contour shape
  - Localised pitch accents

![Graph showing the global pitch contour](image)

a: Rise-Fall  b: Y/N-Question  c: Continuation

Three accent types
Speech parameters

- mcep (40 coefficients)
HMM and LDM synthesis

➢ Every phoneme+context is represented by a model
➢ Acoustic features extracted: F0, spectrum, duration
➢ Train a model with these examples.
Sequential processes

Consider a system which can occupy one of \( N \) discrete states or categories:

\[ x_t \in \{1, 2, \ldots, N\} \rightarrow \text{state at time } t \]

We are interested in stochastic systems, in which state evolution is random.

Any joint distribution can be factored into a series of conditional distributions:

\[
p(x_1, x_2, \ldots, x_T) = p(x_1) \prod_{t=2}^{T} p(x_t | x_1, \ldots, x_{t-1})
\]
Markov processes

- For a Markov process, the next state depends only on the current state:

\[ p(x_t | x_1, \ldots, x_{t-1}) = p(x_t | x_{t-1}) \]

- This property in turn implies that

\[ p(x_1, \ldots, x_{t-1}, x_{t+1}, \ldots, x_T | x_t) = p(x_1, \ldots, x_{t-1} | x_t) p(x_{t+1}, \ldots, x_T | x_t) \]

"Conditioned on the present, the past & future are independent"
Markov processes

- Stationarity Assumption
  - Probabilities independent of t when process is “stationary”

\[
p(x_t = j | x_{t-1} = i) = a_{ij}
\]

- This means that if system is in state i, the probability that the system will next move to state j is \(a_{ij}\), no matter what the value of t is
Markov processes

- State transition matrices
  - A stationary Markov chain with N states is described by an N x N transition matrix:

\[
A = \begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{bmatrix}
\]

- Constraints on valid transition matrices:
  \[
a_{ij} \geq 0 \quad \sum_{j=1}^{N} a_{ij} = 1 \quad \text{for all } i
\]
Speech synthesis methods

- State transition diagrams
- Think of a particle randomly following an arrow at each discrete time step
- Most useful when N small, and A sparse

\[ a_{ij} = p(x_t = j|x_{t-1} = i) \]

\[ A = \begin{bmatrix} 0.5 & 0.3 & 0.2 \\ 0.1 & 0.0 & 0.9 \\ 0.0 & 0.4 & 0.6 \end{bmatrix} \]
Markov chains

\[ p(x_1, x_2, \ldots, x_T) = p(x_1) \prod_{t=2}^{T} p(x_t | x_{t-1}) \]

- Graph interpretation differs from state transition diagrams:
  - **Nodes**: state values at particular times
  - **Edges**: Markov properties

\[
A = \begin{bmatrix}
0.5 & 0.3 & 0.2 \\
0.1 & 0.0 & 0.9 \\
0.0 & 0.4 & 0.6
\end{bmatrix}
\]
In many applications, it is more natural to define states in some continuous, Euclidean space:

\[ x_t \in \mathbb{R}^n \]
HMMs

- Few realistic time series directly satisfy the assumptions of Markov processes:
  
  "Conditioned on the present, the past & future are independent"

- Motivates hidden Markov models (HMM):

$$p(x_1, \ldots, x_T, y_1, \ldots, y_T) = p(x_1)p(y_1|x_1) \prod_{t=2}^{T} p(x_t|x_{t-1})p(y_t|x_t)$$
HMMs

Given $x_t$, earlier observations provide no additional information about the future:

$$p(y_t, y_{t+1}, \ldots | x_t, y_{t-1}, y_{t-2}, \ldots) = p(y_t, y_{t+1}, \ldots | x_t)$$
HMMs

» Where do states come from?
» Analysis of a physical phenomenon:
  » Dynamical models of an aircraft
» Discovered from training data:
  » Recorded examples of spoken English
HMMs

\[ x_t \in \{1, 2, \ldots, N\} \]

- Associate each of the N hidden states with a different observation distribution:

- Observation densities are typically chosen to encode domain knowledge

- Observation densities are typically chosen to encode domain knowledge
HMMs: Observations

Discrete Observations

\[ p(y_t \mid x_t = 1) = \begin{bmatrix} 0.3 \\ 0.1 \\ 0.5 \\ 0.1 \end{bmatrix} \]

\[ p(y_t \mid x_t = 2) = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.1 \\ 0.5 \end{bmatrix} \]

Continuous Observations

\[ y_t \in \{1, 2, \ldots, M\} \]

\[ y_t \in \mathbb{R}^k \]
Specifying an HMM

- Observation model: $p(y_t|x_t)$
- Transition model: $p(x_t|x_{t-1})$
- Initial state distribution: $p(x_1)$
HMMs

- $a_{ij}$ : State transition probability
- $b_q(y_t)$ : Output probability

The Markov chain whose state sequence is unknown
⇒ Estimating state sequence by the observation
Main issues using an HMM

- **Evaluation problem.** Given the HMM $\theta=(A, B, \pi)$ and the observation sequence $Y=y_1 y_2 ... y_T$, calculate the probability that model $M$ has generated sequence $Y$.

- **Decoding problem.** Given the HMM $\theta=(A, B, \pi)$ and the observation sequence $Y=y_1 y_2 ... y_T$, calculate the most likely sequence of hidden states $x_t$ that produced this observation sequence $Y$.

- **Learning problem.** Given some training observation sequences $Y=y_1 y_2 ... y_T$ and general structure of HMM (numbers of hidden and visible states), determine HMM parameters $\theta=(A, B, \pi)$ that best fit training data.
Filtering & smoothing

- For online data analysis, we seek filtered state estimates given earlier observations:
  \[ p(x_t \mid y_1, y_2, \ldots, y_t) \quad t = 1, 2, \ldots \]

- In other cases, find smoothed estimates given earlier and later of observations:
  \[ p(x_t \mid y_1, y_2, \ldots, y_T) \quad t = 1, 2, \ldots, T \]
HMMs: Filtering

\[ \alpha_t(x_t) \triangleq p(x_t \mid y_1, \ldots, y_t) \]

\[ = \frac{1}{Z_t} p(y_t \mid x_t) \sum_{x_{t-1}} p(x_t \mid x_{t-1}) \alpha_{t-1}(x_{t-1}) \]

**Normalization constant**

**Prediction:** \( p(x_t \mid y_1, \ldots, y_{t-1}) \)

**Update:** \( p(x_t \mid y_1, \ldots, y_t) \)

**Incorporates** \( T \) **observations in** \( \mathcal{O}(TN^2) \) **operations**
HMMs: Smoothing

\[ p(x_t \mid y) \propto p(x_t \mid y_1, \ldots, y_t) p(y_{t+1}, \ldots, y_T \mid x_t) \]

- The **forward-backward** algorithm updates filtering via a **reverse-time** recursion:

\[ \beta_t(x_t) = \frac{1}{Z_t} \sum_{x_{t+1}} p(x_{t+1} \mid x_t) p(y_{t+1} \mid x_{t+1}) \beta_{t+1}(x_{t+1}) \]
Optimal state estimation

- Probabilities measure the posterior confidence in the true hidden states.
- For the learning problem, the following marginal probabilities are inferred from the observations:

\[ p(x_t | y) = \frac{1}{Z_t} \hat{\alpha}_t(x_t) \hat{\beta}_t(x_t) \]

\[ p(x_{t-1}, x_t | y) = \frac{1}{Z_t} \hat{\alpha}_{t-1}(x_{t-1}) p(x_t | x_{t-1}) p(y_t | x_t) \hat{\beta}_t(x_t) \]

- What about the state sequence with the highest joint probability?
Algorithm 3: Forward Recursions in HMM

Data: Observations conditional probabilities, $b_t$, $t \in \{1, \ldots, T\}$, and model parameters: $A$, $\pi$
Result: $\log L = \log(p(y_{1:T}))$, $c_t$ and $\hat{\alpha}_t$, $t \in \{1, \ldots, T\}$

/* Initialization */
/* Computation of $\alpha_1(i)$ */
for $i = 1:K$ do
  $\alpha_1(i) = \pi(i)b_1(i)$

[$\hat{\alpha}_1, c_1] = \text{normalize}(\alpha_1)$
$\log L = \log(c_1)$

/* Induction */
for $t = 2:T$ do
  /* Computation of $\hat{\alpha}_t'(i) = c_t \cdot \hat{\alpha}_t(i)$ */
  for $i = 1:K$ do
    $\hat{\alpha}_t'(i) = A(:, i)^\top \cdot \hat{\alpha}_{t-1}b_t(i)$
  [$\hat{\alpha}_t, c_t] = \text{normalize}(\hat{\alpha}_t')$
  $\log L = \log L + \log(c_t)$
Algorithm 4: Backward Recursions in HMM

Data: Observations conditional probabilities, $b_t, t \in \{1, \ldots, T\}$, scaling factors $c_t$ and model parameters: $A$

Result: $\hat{\beta}_t, t \in \{1, \ldots, T\}$

/* Initialization */
for $i = 1:K$ do
$\beta_T(i) = 1$ /* which is the probability of a non-event, $p(\emptyset|x_T = i)$ */
/* Induction */
for $t = T-1:2$ do
$\hat{\beta}_{t-1} = A \cdot (\frac{1}{c_t} \hat{\beta}_t \odot b_t)$ /* where $\odot$ is the Hadamard product */
Viterbi algorithm

• Decoding problem. Given the HMM $M = (Q, B, \pi)$ and the observation sequence $Y = y_1 y_2 \ldots y_T$, calculate the most likely sequence of hidden states $x_t$ that produced this observation sequence.

• We want to find the state sequence $X = x_1 \ldots x_T$ which maximizes $P(X \mid y_1 y_2 \ldots y_T)$, or equivalently $P(X, y_1 y_2 \ldots y_K)$.

• Brute force consideration of all paths takes exponential time. Use efficient **Viterbi algorithm** instead.

• Define variable $\delta_t(i)$ as the maximum probability of producing observation sequence $y_1 y_2 \ldots y_t$ when moving along any hidden state sequence $x_1 \ldots x_{t-1}$ and getting into $x_t = i$.

\[
\delta_k(i) = \max P(x_1 \ldots x_{t-1}, x_t = i, y_1 y_2 \ldots y_t)
\]

where max is taken over all possible paths $x_1 \ldots x_{t-1}$.
Viterbi algorithm

Algorithm 5: Viterbi Algorithm

Data: Observations conditional probabilities, \( b_t, t \in \{1, \ldots, T\} \), and model parameters: \( A, \pi \)
Result: The most probable state sequence \( x_{1:T}^* \) and \( P^* = P(x_{1:T}^* | y_{1:T}) \)

/* Initialization */
for \( i = 1:K \) do
  \[ \delta_1(i) = \pi(i)b_1(i) \]
  \[ \psi_1(i) = 0 \]

/* Induction */
for \( t = 2:T \) do
  for \( i = 1:K \) do
    \[ \delta_t(i) = \max_{1 \leq j \leq N} (\delta_{t-1}(j)A_{ji})b_t(i) \]
    \[ \psi_t(i) = \arg \max_{1 \leq j \leq N} (\delta_{t-1}(j)A_{ji}) \]
    \[ [\delta_t, c_t] = \text{normalize}(\delta_t) \]

/* Termination */
\[ P^* = \max_{1 \leq i \leq N} (\delta_T(i)) \]
\[ x_T^* = \arg \max_{1 \leq i \leq N} (\delta_T(i)) \]

/* Path (state sequence) backtracking */
for \( t = T-1:-1:1 \) do
  \[ x_t^* = \psi_{t+1}(x_{t+1}^*) \]
Learning problem for HMM

- **Learning problem.** Given some training observation sequences $Y=y_1y_2...y_T$ and general structure of HMM (numbers of hidden and visible states), determine HMM parameters $\theta=(A, B, \pi)$ that best fit training data, that is maximizes $P(Y | \theta)$.

- There is no algorithm producing optimal parameter values.

- Use iterative expectation-maximization algorithm to find local maximum of $P(Y | \theta)$ - Baum-Welch algorithm.
Learning problem for HMM

- Log-likelihood:
  \[ L(\theta) = \log p(Y|\theta) = \log \left( \int_X p(X,Y|\theta)\,dX \right) \]

  \[ L(\theta) \geq \int_X q(X) \log(X,Y|\theta)\,dX - \int_X q(X) \log q(X)dX \]

  \[ F(q, \theta) = \int_X q(X) \log(X,Y|\theta)\,dX - \int_X q(X) \log q(X)dX \]

- EM-algorithm

  E-step: \[ q_{i+1} \leftarrow \arg \max_q F(q, \theta_i) \]

  M-step: \[ \theta_{i+1} \leftarrow \arg \max_\theta F(q_{i+1}, \theta) \]
Learning problem for HMM

Algorithm 8: EM Algorithm for HMM

Data: Data \( D_y \) and parameters \( \pi, A, P_{obs} = \{(\mu_k, \Sigma_k) \mid k = 1, \ldots, K\} \), \( numIter \)

Result: New parameters \( \pi, A, (\mu_k, \Sigma_k), k \in \{1, \ldots, K\} \) and log-likelihood \( \log L \)

for \( i = 1 : numIter \) do
  % E-step
  Clear the sufficient statistics: \( \gamma_1, \gamma_2, \gamma_3, \Gamma_1, \Gamma_2 \)
  \( \log L = 0 \)
  Let \( N \) be the number of trajectories in \( D_y \)
  for \( l = 1:N \) do
    \( Y = D_y[l] \)
    \( b = \text{calculateObservationProb}(Y, P_{obs}) \)
    \([\alpha, c, \log L_l] = \text{forwardRecursions}(A, \pi, b)\)
    \( \log L = \log L + \log L_l \)
    \( \beta = \text{backwardRecursions}(c, A, b) \)
    \([\gamma'_1, \gamma'_2, \gamma'_3, \Gamma'_1, \Gamma'_2] = \text{sufficientStatistics}(Y, \alpha, \beta, c, A, b)\)
    Sum sufficient statistics (Eq. (34)). E.g., \( \gamma_1 = \gamma_1 + \gamma'_1 \)
  % M-step
  \([\pi, A, P_{obs}] = \text{updateParameters}(N, \gamma_1, \gamma_2, \gamma_3, \Gamma_1, \Gamma_2)\)
HMM-based speech synthesis system

**Training part**

1. **Speech signal**
2. **Speech signal**
3. **Labels**
4. **Speech signal**
5. **Training HMMs**
6. **Parameter generation from HMMs**
7. **Synthesis filter**
8. **Synthesized speech**

**Synthesis part**

1. **Text**
2. **Text analysis**
3. **Labels**
4. **Excitation generation**
5. **Excitation parameters**
6. **Excitation parameters**
7. **Parameter generation from HMMs**
8. **Excitation parameters**
9. **Spectral parameters**
10. **Spectral parameters**
11. **Context-dependent HMMs & state duration models**