ISCAS 2012 Tutorial 09: Advances in Speech Coding, Recognition and Applications

Tokunbo Ogunfunmi, (SCU)
M.J. (Sim) Narasimha, (Stanford, SCU)
Roberto Togneri, (UWA)

Tutorial Presentation
Date: Sunday May 20, 2012
Outline

• Introduction (Authors, Subject, etc.)
• Part 1 : Principles of Waveform Speech Coding
  – Nyquist, Waveform Coding, PCM, ADPCM, etc.
• Part 2 : Parametric Speech Coding, Standards & Applications
  – LPC, CELP, ITU G.729A, iLBC, DSP Impl., etc.
• Part 3 : Speech Recognition & Applications
  – Feature Extraction, Modeling, HMMs, etc.
• Summary, Conclusions
Part 3: Speech Recognition and Applications

Dr Roberto Togneri
EEC Engineering
The University of Western Australia
Contents

• Feature Extraction

• Acoustic Modelling

• Language Modelling and Scoring

HTK: [http://htk.eng.cam.ac.uk/](http://htk.eng.cam.ac.uk/)
**Feature Extraction**

**Sampling and Windowing**

- **Sampled Speech (*.wav files → an utterance)**
  - **Pre-emphasis**: High-Pass filter to compensate for human speech production attenuation
  - **Framing**: 25ms frames overlapped by 10ms, each frame will be transformed to a feature vector, each utterance will consist of a sequence of feature vectors
  - **Windowing**: Tapering of segment edges to remove spectral artefacts arising from Fourier Transform (FT) operation.
Feature Extraction

Filterbank Analysis (Mel Spacing)

- FT operation yields N complex co-efficients per frame (N=256 or 512, very large!)
- N complex co-efficients $\rightarrow$ N magnitude values uniformly spaced from 0 to $Fs/2$ Hz ($Fs =$ sampling frequency)
- N magnitude values transformed to K melspec values ($K=26$) via mel spacing

$$f_{MEL} = 2595 \log_{10}(1 + \frac{f_{LIN}}{700})$$
Feature Extraction
Cepstral Features

- K melspec ($S_k$) $\rightarrow$ log ($S_k$) \textit{fbank}

- Transform from K log-spectral fbank values to L cepstral domain (\textit{Mel-Frequency Cepstral Co-efficients} or \textit{MFCC}) via Discrete Cosine Transform (DCT) operation (L=12)

$$c_n = \sum_{k=1}^{K} \log(S_k) \cos \left[ \frac{n(k - 0.5)\pi}{K} \right], \quad n = 1,2, \ldots, L$$

- Usually also include $c_0$ (hence 13 \textit{MFCC_0}), $c_0$ represents the average log-power of the frame, $c_0 = \sum_{k=1}^{K} \log(S_k)$
Feature Extraction

Delta and Acceleration Features

39-dim feature vector

- The 13 static MFCC_0 ($c_t$) vector at frame $t$, have 13 delta

\[
\mathbf{d}_t = \frac{\sum_{p=1}^{P} p(c_{t+p} - c_{t-p})}{2 \sum_{p=1}^{P} p^2} \quad (P=2)
\]

- and 13 acceleration co-efficients appended:

\[
\mathbf{a}_t = \frac{\sum_{p=1}^{P} p(d_{t+p} - d_{t-p})}{2 \sum_{p=1}^{P} p^2} \quad (P=2)
\]

- to yield 39-dimensional feature vectors (per frame), **MFCC_0_D_A**

- **Why?** A simple, but effective approach to capture/model the temporal dynamic information of the features
Feature Extraction

Why these features?

• Why mel spacing (melspec)?
  – Mel spacing emulates human perception of frequency (pitch) where there is more accurate/finer perception of lower frequencies relative to higher frequencies

• Why log values (fbank)?
  – Log amplitudes values emulate human logarithmic compression of sound pressure amplitudes

• Why cepstral features (MFCC_0)
  – Although human perception is frequency selective (spectral), the cepstral features have useful modelling characteristics:
    • Uncorrelated making them a more compact representation ➔ can use diagonal covariance matrices
    • Allows separation of short-time (small n) vocal tract speech-specific features from long-time (large n) pitch excitation speaker-specific features ➔ liftering to capture vocal-tract speech-specific features only (i.e. smooth spectral representation)
Feature Extraction
Coping with channel distortion: Cepstral Mean Subtraction

- **Channel Distortion** ($H_C$)
  \[ S_k = H_C X(f) \]
  \[ \log(S_k) = \log H_C + \log X(f) \]
  \[ c_t = c^H + c_t^X \]

- Need to remove unknown channel distortion (compensate for different channels) and estimate the true value of $c_t^X$

- **Cepstral Mean Subtraction (CMS)**
  \[
  \bar{\mu} = \frac{1}{T} \sum_{t=1}^{T} c_t = \frac{1}{T} \sum_{t=1}^{T} c^H + \frac{1}{T} \sum_{t=1}^{T} c_t^X = c^H + \frac{1}{T} \sum_{t=1}^{T} c_t^X
  \]
  \[
  \bar{c}_t = c_t - \bar{\mu} = c_t^X - \frac{1}{T} \sum_{t=1}^{T} c_t^X
  \]
Feature Extraction

Coping with additive noise: Spectral Subtraction

- Additive background noise which is quasi-stationary (e.g. fan noise) can be removed by **Spectral Subtraction (SS):**

\[
|\hat{X}(\omega)|^2 = |Y(\omega)|^2 - |\hat{N}(\omega)|^2
\]

\[
\varphi_{\hat{X}}(\omega) = \varphi_Y(\omega)
\]

where \( Y(\omega) \) is the noisy speech input, \( \hat{X}(\omega) \) is the estimate of the clean speech on output, and \( \hat{N}(\omega) \) is the estimate of the noise by from the non-speech (silence) periods, as provided by a VAD.
Feature Extraction

Why are CMS and SS necessary?

• Mismatch problem
  – Speech Recognition is basically a pattern recognition/matching operation
  – Any mismatch of the statistical/pattern properties of the features used during the model training and system evaluation will severely degrade performance
  – Training is usually conducted with ideal, clean speech
  – Different background noises and channels will create a mismatch

• CMS alleviates channel mismatch by producing features which are independent of the channel

• SS alleviates additive noise mismatch by producing features which are independent of the noise

• Training of models on speech data with CMN/SS applied will be matched with speech data of unknown/differing background noise / channel distortion with CMN/SS applied (used in testing/evaluation)
Feature Extraction

HTK example: HCopy

- **HCopy -C config_data -S codetr.scp**

  **codetr.scp**: lists each utterance speech data filename (input) and utterance feature filename (output) in a text file (one utterance per line):
  
  `/usr/local/data/spka/word1.wav  /var/htk/traindata/spka/word1.mfc`
  `/usr/local/data/spka/word2.wav  /var/htk/traindata/spka/word2.mfc`
  ...

  **config_data**: feature extraction parameters file

  - **SOURCEKIND = WAVEFORM** (sampled data file)
  - **SOURCEFORMAT = NIST** (NIST SPHERE format sampled data file)
  - **TARGETKIND = MFCC_0_Z** (MFCC + C0 with CMS applied (_Z))
  - **TARGETRATE = 100000.0** (feature vector every 10ms)
  - **SAVECOMPRESSED = T**
  - **SAVEWITHCRC = T**
  - **WINDOWSIZE = 250000.0** (windows size of 25ms)
  - **ZMEANSOURCE = T**
  - **USEHAMMING = T**
  - **PREEMCOEF = 0.97**
  - **NUMCHANS = 26** (K=26 fbank features)
  - **CEPLIFTER = 22**
  - **NUMCEPS = 12** (L=12 cepstral features + C0 = 13)
  - **ENORMALISE = F**

  **NOTE**: HTK does not provide SS (Spectral Subtraction) processing
# Acoustic Modelling

## Acoustic-Phonetics of the English Language

### Table of English phonemes (from [1])

<table>
<thead>
<tr>
<th>Phonemes</th>
<th>Word Examples</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iy</td>
<td>feel, eve, me</td>
<td>front close unrounded</td>
</tr>
<tr>
<td>ih</td>
<td>fill, hit, lid</td>
<td>front close unrounded (lax)</td>
</tr>
<tr>
<td>ae</td>
<td>at, carry, gas</td>
<td>front open unrounded (tense)</td>
</tr>
<tr>
<td>ah</td>
<td>father, ah, car</td>
<td>back open unrounded</td>
</tr>
<tr>
<td>ao</td>
<td>dog, lawn, caught</td>
<td>open-mid back unrounded</td>
</tr>
<tr>
<td>ay</td>
<td>tie, ice, bite</td>
<td>diphthong with quality: aa + ih</td>
</tr>
<tr>
<td>ax</td>
<td>ago, comply</td>
<td>central close mid (schwa)</td>
</tr>
<tr>
<td>ey</td>
<td>ate, day, tape</td>
<td>front close-mid unrounded (tense)</td>
</tr>
<tr>
<td>eh</td>
<td>pet, berry, ten</td>
<td>front open-mid unrounded</td>
</tr>
<tr>
<td>er</td>
<td>turn, fur, meter</td>
<td>central open-mid unrounded rhoti-</td>
</tr>
<tr>
<td>ow</td>
<td>go, own, tone</td>
<td>back close-mid rounded</td>
</tr>
<tr>
<td>aw</td>
<td>foul, how, a wr</td>
<td>diphthong with quality: ao + uh</td>
</tr>
<tr>
<td>oy</td>
<td>toy, coin, ott</td>
<td>back close-mid unrounded (lax)</td>
</tr>
<tr>
<td>uh</td>
<td>book, pull, good</td>
<td>back close-mid unrounded</td>
</tr>
<tr>
<td>uw</td>
<td>tool, crew, moo</td>
<td>back close round</td>
</tr>
<tr>
<td>b</td>
<td>big, able, tab</td>
<td>voiced bilabial plosive</td>
</tr>
<tr>
<td>p</td>
<td>put, open, tap</td>
<td>voiceless bilabial plosive</td>
</tr>
<tr>
<td>t</td>
<td>talk, sat</td>
<td>voiceless alveolar plosive</td>
</tr>
<tr>
<td>t</td>
<td>meter</td>
<td>alveolar flap</td>
</tr>
<tr>
<td>g</td>
<td>gut, angle, tag</td>
<td>voiceless velar plosive</td>
</tr>
<tr>
<td>k</td>
<td>cut, ken, take</td>
<td>voiceless velar plosive</td>
</tr>
<tr>
<td>f</td>
<td>fork, after, if</td>
<td>voiceless labiodental fricative</td>
</tr>
<tr>
<td>v</td>
<td>vat, over, have</td>
<td>voiceless labiodental fricative</td>
</tr>
<tr>
<td>s</td>
<td>sit, cast, toss</td>
<td>voiceless alveolar fricative</td>
</tr>
<tr>
<td>z</td>
<td>zap, lazy, haze</td>
<td>voiceless alveolar fricative</td>
</tr>
<tr>
<td>th</td>
<td>thin, nothing, truth</td>
<td>voiceless dental fricative</td>
</tr>
<tr>
<td>dh</td>
<td>then, father, scythe</td>
<td>voiced dental fricative</td>
</tr>
<tr>
<td>sh</td>
<td>she, cushion, wash</td>
<td>voiced postalveolar fricative</td>
</tr>
<tr>
<td>zh</td>
<td>genre, azure</td>
<td>voiced postalveolar fricative</td>
</tr>
<tr>
<td>l</td>
<td>lid</td>
<td>alveolar lateral approximant</td>
</tr>
<tr>
<td>r</td>
<td>red, part, far</td>
<td>velar lateral approximant</td>
</tr>
<tr>
<td>y</td>
<td>yacht, yard</td>
<td>retroflex approximant</td>
</tr>
<tr>
<td>w</td>
<td>with, away</td>
<td>palatal sonorant glide</td>
</tr>
<tr>
<td>h</td>
<td>help, ahead, hotel</td>
<td>labiovelar sonorant glide</td>
</tr>
<tr>
<td>m</td>
<td>mat, amid, amm</td>
<td>voiceless glottal fricative</td>
</tr>
<tr>
<td>n</td>
<td>no, end, pan</td>
<td>bilabial nasal</td>
</tr>
<tr>
<td>ng</td>
<td>sing, anger</td>
<td>alveolar nasal</td>
</tr>
<tr>
<td>ch</td>
<td>chin, archer, march</td>
<td>velar nasal</td>
</tr>
<tr>
<td>jh</td>
<td>joy, agile, edge</td>
<td>voiceless alveolar affricate: t + zh</td>
</tr>
</tbody>
</table>

### Vowels + Diphthongs

Open oral cavity; Sustained sound for vowels; Two-tier sound for diphthongs

### Consonants

Plosives + Nasals + Fricatives + Glides + Liquids + Affricates

Constriction of cavity; Burst of sound for plosives/affricates; Sustained for fricatives/nasals
Acoustic Modelling

Acoustic-Phonetics of the English Language

Major places of consonant constriction (articulation) in human mouth (from [1])

Labiodental: $v, f$

Dental: $th, dh$

Alveolar: $t, d, n, s, z, r, l$

Palatal: $sh, zh, y$

Labial: $m, p, b, w$

Velar: $k, g, ng$
Acoustic Modelling

Acoustic-Phonetics of the English Language

Spectrogram: yeller (left) and pin (right) (from [1])
Acoustic Modelling

Hidden Markov Models: What are they?

Hidden States \((s_t = 0, 1, ..., N)\) at time \(t\), with a Gaussian Mixture Model (GMM) of \(M\) mixtures per state, with \(N\) states per model. At each time \(t\), there is an observation, \(o_t\), which is the \(n\)-length feature vector for frame \(t\). Thus each Gaussian requires an \(n \times n\) covariance matrix, \(\Sigma\), with \(n\)-length mean, \(\mu\).

Transition Probability:

\[ a_{ij} = P(s_t = j | s_{t-1} = i) \]

Observation Probability:

\[ b_j(o_t) = \sum_{k=1}^{M} c_{jk} N(o; \mu_{jk}, \Sigma_{jk}) \]

where:

\[ N(o; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(o - \mu)^T \Sigma^{-1} (o - \mu)} \]
Acoustic Modelling

Hidden Markov Models: How to Use?

Given an utterance, \( \mathbf{O} = \{o_1, o_2, ..., o_T\} \), consisting of \( T \) observations, and a Hidden Markov Model (HMM), \( \lambda = \{A, B, \pi\} \),

\[
A = \{a_{ij}\}_{1 \leq i, j \leq N} \quad B = \{c_{jk}, \mathcal{N}(\mathbf{o}; \mu_{jk}, \Sigma_{jk})\}_{1 \leq j \leq N, 1 \leq k \leq M} \quad \pi = \{\pi_i = P(s_1 = i)\}_{1 \leq i \leq N}
\]

we can derive the forward probability which is expected to be maximum only for the utterance generated from the corresponding model: \( P(\mathbf{O}|\lambda) \).

Thus we can recognise an unknown utterance by associating the HMM which has the maximum probability where the HMM is either a specific phoneme or word:

\[
\lambda_0 = \arg\max_\lambda P(\mathbf{O}|\lambda)
\]

If an utterance consists of a string of phonemes (for a word) or a string of words (for a sentence) then we need to segment into a sequence of HMM’s yielding the maximum overall probability score.

Forward Algorithm to calculate \( P(\mathbf{O}|\lambda) \)

Step 1: \( \alpha_1(i) = \pi_i b_i(o_1) \quad 1 \leq i \leq N \)

Step 2: \( \alpha_t(j) = \left[ \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \right] b_j(o_t) \quad 2 \leq t \leq T; \ 1 \leq j \leq N \)

Step 3: \( P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \)

The forward algorithm derives \( P(\mathbf{O}|\lambda) \) over all possible state paths. In practice for recognition/decoding it is just as effective and more computationally efficient (using the Viterbi algorithm) to derive over the best state path only.
Acoustic Modelling

Hidden Markov Models: Baum-Welch algorithm

- **Given**: A collection of $L$ utterances for each HMM, $\{O_l\}_{1 \leq l \leq L}$ (e.g. $L=20$ utterances of the same spoken digit “one”, $\{O_l^{\text{one}}\}_{1 \leq l \leq L}$, for HMM, $\lambda_{\text{one}}$)

- **Parameter Estimation (aka Training)**: Need to estimate, $\lambda_{\text{one}}$, to maximise: $\prod_{l=1}^{L} P(O_l^{\text{one}} | \lambda_{\text{one}}) 
\rightarrow$ an EM algorithm $\rightarrow$ Baum-Welch algorithm

- **Baum-Welch Algorithm**: Very involved! Refer to pages 389-393 and 398-405 of [1]
Acoustic Modelling

Acoustic Models: whole words, mono-phones, tri-phones

- Whole word:
  - Easiest and most reliable, but you need as many HMMs as words in the English language?
  - Limited to small vocabulary systems “command and control” type applications (e.g. digit recognition of PIN, control action words, etc.)

- Mono-phones:
  - Only 50 phonemes in English language, so can cover any size vocabulary system
  - As phonemes sound different depending on context due to co-articulation mono-phone systems yield poor recognition performance
  - Training requires optimal segmentation of utterance into HMM phone boundaries
  - Example: butter → /b/ /ah/ /t/ /er/

- Tri-phones:
  - Consists of pre- and post- phone contexts for context dependent phone models
  - Provides good recognition but need to deal with large number of tri-phone models (50x50x50) via model clustering
  - Example: butter → ( )/-b/+(ah) (b)-/ah+/+(t) (ah)-/t+/+(er) (t)-/er/+( )
Acoustic Modelling

Tri-phone models: parameter tying, model clustering

- Two related problems
  - Too many tri-phone models $\Rightarrow$ not enough data to train so many Gaussians!
  - Not all tri-phone models will be present in the training data $\Rightarrow$ unseen and untrained tri-phones in test data

- Two solutions
  - Fair: Tie together the centre mono-phone state so that the same Gaussians are shared/pooled for the same phone (but mixed differently depending on context)
    - Ok but may be too severe and doesn’t deal effectively with unseen tri-phones
    - Example: Pool all /aw/ contexts: (*)-/aw/+(*)
  - Better: As above but cluster the pool depending on the left- and right contexts (decision tree-based state tying)
    - More precise training and can deal effectively with unseen tri-phones by synthesising from cluster representation
Acoustic Modelling

Tri-phone models: parameter tying, model clustering

Example of decision tree clustering for phone /aw/ (from [2])

- Leafs of tree represent the clusters, one HMM state is associated per cluster
- Unseen contexts are synthesised by traversing tree and using the leaf cluster for that context
Acoustic Modelling

HTK Example

• Train mono-phones first (several iterations):
  – HERest -C config -I phones.mlf -S train.scp -H hmm/macros -H hmm/hmmdefs -M hmm1 monophones

  • config: TARGETKIND = MFCC_0_Z_D_A
    – delta + acceleration created on the fly as data is read in
  • phones.mlf: lists the monophone sequence of each utterance
  • train.scp: list of training utterances
  • monophones: list of monophone HMMs to train
  • -H <hmm files>: the current / initial HMM parameters
  • -M <hmm dir>: the next updated HMM parameters
Acoustic Modelling

HTK Example

• Generate the tri-phone labels / transcriptions:
  – HLEd -n triphones -l "*" -i triphones.mlf mktri.led phones.mlf
    • triphones.mlf: the tri-phone sequence of each utterance
    • triphones: list of tri-phone HMMs
    • mktri.led: instructions to generate the labels

• Generate the tri-phone models:
  – HHEd -H hmm/macros -H hmm/hmmdefs -M hmm mktri.hed monophones
    • -H <hmm files>: original monophone HMMs
    • -M <hmm dir>: tied mixture triphone HMMs
    • mktri.hed: instructions to generate triphones
Acoustic Modelling

HTK Example

• Perform Decision Tree Clustering:
  – HHEd -H hmm/macros -H hmm/hmmdefs -M hmm tree.hed triphones
    • -H <hmm files>: original triphone HMMs
    • -M <hmm dir>: clustered triphone HMMs
    • tree.hed: list of questions to generate the decision tree

• Now train the clustered triphone HMMs:
  – HERest -C config -I triphones.mlf -S train.scp -H hmm/macros -H hmm/hmmdefs -M hmm tiedlist
    • tiedlist: List of tied clustered triphone HMMs to train
Language Modelling and Scoring
Isolated, Connected and Continuous

• **Decoding Problem**: What is the sequence of HMMs which yield the largest probability for the given unknown test utterance?
  – Sequence of HMMs \(\rightarrow\) Sequence of words/phones \(\rightarrow\) Utterance is recognised!

• **Isolated**: Utterance a word, HMMs are word models (trivial word network) \(\rightarrow\) easy!

• **Connected**: Utterance is a sequence of words, HMMs are word models (simple word network) \(\rightarrow\) OK

• **Continuous**: Utterance is sequence of words, HMMs are triphone models (complex word network) \(\rightarrow\) tricky!
Language Modelling and Scoring

Word Networks: Isolated and Connected

- **Isolated (a,b):** only need to search over collection of words for the word which provides the maximum probability
- **Connected (c,d):** need to loop over a sequence of words for the sequence which provides the maximum probability
- **Insertion/Deletion with connected:** Final sequence may have more (insertion) or less (deletion) words than actual; not a problem for well trained word models (coarse granularity of models)

Figure taken from [2]
Language Modelling and Scoring
Dictionaries and Networks: Continuous

Word to Phone Dictionary [2]

- Continuous needs Dictionary: Map words to valid phone sequence; makes sense to only search over sequence of phones which constitute a valid word!
- Triphone word network: Each possible word is expanded to sequence of triphones using dictionary
Language Modelling and Scoring

Language Models: n-gram

- Language Models capture English grammar structure
  - Statistical analysis of word sequences
    - Most likely sequences have high probability
      Words “apple”, “man” can follow word “The”
    - Unlikely sequence have low probability
      Words “sun”, “reflect” rarely follow word “An”

- Simplest are n-gram models, especially \textbf{bigram} $P(w_i | w_{i-1})$ and \textbf{trigram} $P(w_i | w_{i-2}, w_{i-1})$ models
  - Probability of word $w_i$ occurring given the previous word sequence of $w_{i-1}$ or $w_{i-2}, w_{i-1}$
    - $P($man $|$ happy$) \rightarrow$ very high probability
    - $P($sat $|$ is$) \rightarrow$ very low probability
Language Modelling and Scoring

Decoding Continuous: Language Models

- Final probability score:
  HMM Acoustic Model (AM) score from triphone word network \( \times \)
  Language Model (LM) score from n-gram model

- Can weight the two differently:
  More weight to AM: random sequence of words, no grammar structure
  More weight to LM: strict grammar structure, possibly wrong words
Language Modelling and Scoring

Decoding and Searching

• Given an unknown utterance we need to find the sequence of HMM “word” models \( \hat{w} = w_1, w_2, \ldots, w_m \) from all permitted and most likely sequences (as determined by the word network and language models) that maximises:

\[
\hat{w} = \arg\max_w P(w|O) = \arg\max_w P(w) P(O|w)
\]

  – \( P(w) \): Language Model score (with word network)
  – \( P(O|w) \equiv P(O|\lambda) \): Acoustic Model score (with dictionary)

• **Problem**: Need a search algorithm to efficiently calculate \( P(O|w) \) over a sequence of candidate HMM word models
  – The forward algorithm considers all possible state sequences \( \rightarrow \) no good
  – Consider only best state / optimal path sequence when “searching” \( \rightarrow \) **Viterbi algorithm** \( \rightarrow \) Can use basic search strategies (path pruning, beam search, etc.)
Language Modelling and Scoring

Viterbi Algorithm

Given: Observation sequence of length $T$ and HMM with $N$ states

**Step 1: Initialisation**

$$
\delta_1(i) = \pi_i b_i(o_1) \quad 1 \leq i \leq N
$$

$$
\psi_1(i) = 0
$$

**Step 2: Induction**

$$
\delta_t(j) = \max_{1 \leq i \leq N} \left[ \delta_{t-1}(i) a_{ij} \right] b_j(o_t) \quad 2 \leq t \leq T; \quad 1 \leq j \leq N
$$

$$
\psi_t(j) = \arg\max_{1 \leq i \leq N} \left[ \delta_{t-1}(i) a_{ij} \right] \quad 2 \leq t \leq T; \quad 1 \leq j \leq N
$$

**Step 3: Termination**

$$
P^* = \max_{1 \leq i \leq N} \delta_T(i)
$$

$$
s_T^* = \arg\max_{1 \leq i \leq N} \psi_T(i)
$$

**Step 4: Backtracking**

$$
s_t^* = \psi_{t+1}(s_{t+1}^*) \quad t = T - 1, T - 2, \ldots, 1
$$

$$
s^* = (s_1^*, s_2^*, \ldots, s_T^*) \quad \text{is the best state sequence}
$$
Language Modelling and Scoring

Scoring: Word Error Rate

• Evaluating recognised output sequence against reference/correct sequence
  – Trivial for isolated word (it either matches or it doesn’t!)
  – Connected/Continuous: Need to consider mismatches:
    • Number of (Ins) insertions, (Del) deletions, (Sub) substitutions over (N) total words (in reference)

• Word Error Rate (WER) = \( \frac{\text{Sub} + \text{Del} + \text{Ins}}{N} \times 100\% \)
  – Reference: An apple a day keeps the doctor away or [is] [it] the dentist?
  – Recognised: An awful a day keeps the dock tore away or * * the dentist?
  – WER = (Sub=2 + [Del]=2 + Ins=1) / (N=13) \( \Rightarrow 38.4\% \)
  – Accuracy = 100 – WER = 61.6%
Language Modelling and Scoring

HTK Example

• Bigram Language Model
  – HLStats -o -b bgraml wordlist words-train.mlf
    • wordlist: dictionary list of all words in training data
    • words-train.mlf: word sequence of each training utterance
    • bgraml: output bi-gram language model probabilities based on word sequence statistics

• Build the Word Network (HTK lattice) from the Language Model
  – HBuild -b -n bgraml wordlist wdnet
    • wdnet: output HTK lattice word network
Language Modelling and Scoring

HTK Example

• Do the recognition (using Viterbi algorithm)
  – HVite -C config -H hmm/macros -H hmm/hmmdefs -S
    test.scp -i recog.mlf -w wdnet -t 250.0 dictionary
tiedlist
    • -H <hmm files>: HMM triphone models
    • -t 250: beam search threshold for pruning
    • test.scp: list of test utterances (to be recognised)
    • recog.mlf: list of test utterance recognised transcriptions
    • dictionary: list of word to phone mappings

• Evaluating performance (Accuracy)
  – HResults –I reference.mlf tiedlist recog.mlf
    • reference.mlf: list of test utterance known/reference
      transcriptions
    • Output → List of sentence and word accuracy values
Key References


   http://htk.eng.cam.ac.uk/docs/docs.shtml


4. Collection of scripts for HTK:
   http://www.ee.uwa.edu.au/~roberto/research/archive/HTKscripts/
   and other speech recognition resources: