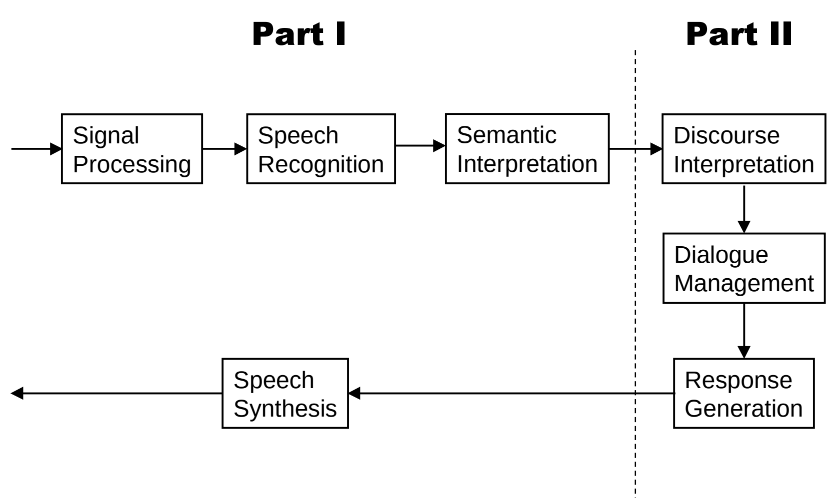
# Spoken Dialogue Systems

**Bob Carpenter** and **Jennifer Chu-Carroll**

June 20, 1999



## Tutorial Overview: Data Flow



## Speech and Audio Processing

- Signal processing:
  - Convert the audio wave into a sequence of feature vectors
- Speech recognition:
  - Decode the sequence of feature vectors into a sequence of words
- Semantic interpretation:
  - Determine the meaning of the recognized words
- Speech synthesis:
  - Generate synthetic speech from a marked-up word string



## Tutorial Overview: Outline

### Part I

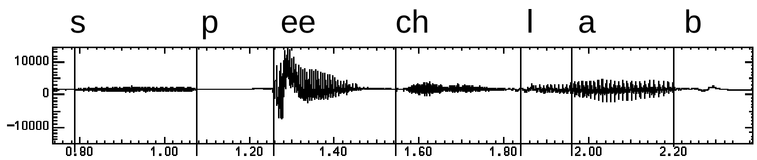
- Signal processing
- Speech recognition
  - acoustic modeling
  - language modeling
  - decoding
- Semantic interpretation
- Speech synthesis

### Part II

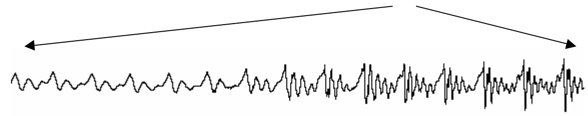
- Discourse and dialogue
  - Discourse interpretation
  - Dialogue management
  - Response generation
- Dialogue evaluation
- Data collection

## Acoustic Waves

- Human speech generates a wave
  - like a loudspeaker moving
- A wave for the words “speech lab” looks like:




“l” to “a” transition:

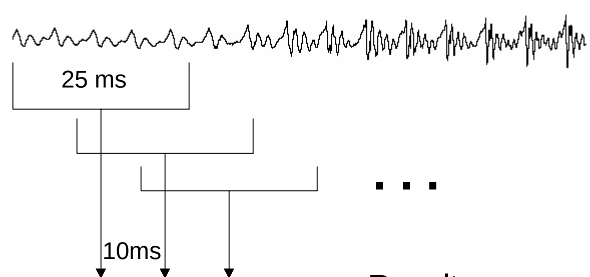


Graphs from Simon Arnfield’s web tutorial on speech, Sheffield:
http://lethe.leeds.ac.uk/research/cogn/speech/tutorial/



## Acoustic Sampling

- 10 ms frame (ms = millisecond = 1/1000 second)
- ~25 ms window around frame to smooth signal processing

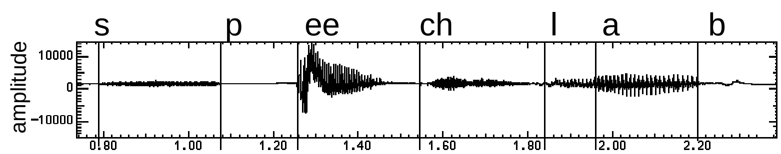



$\mathbf{a}_1$ $\mathbf{a}_2$ $\mathbf{a}_3$

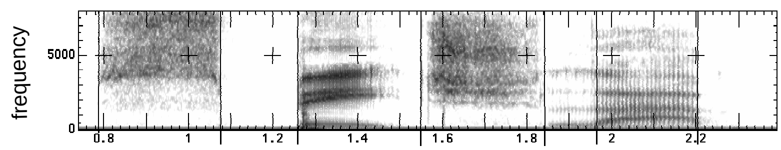
Result:
**Acoustic Feature Vectors**

## Spectral Analysis

- Frequency gives pitch; amplitude gives volume
  - sampling at ~8 kHz phone, ~16 kHz mic (kHz=1000 cycles/sec)



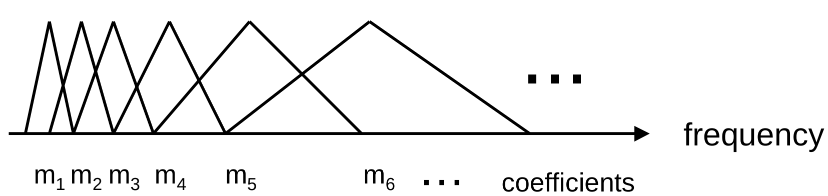

- Fourier transform of wave yields a spectrogram
  - darkness indicates energy at each frequency
  - hundreds to thousands of frequency samples






## Acoustic Features: Mel Scale Filterbank

- Derive Mel Scale Filterbank coefficients
- Mel scale:
  - models non-linearity of human audio perception
  - mel(f) = 2595 $\log_{10}(1 + f / 700)$
  - roughly linear to 1000Hz and then logarithmic
- Filterbank
  - collapses large number of FFT parameters by filtering with ~20 triangular filters spaced on mel scale



## Cepstral Coefficients

- Cepstral Transform is a discrete cosine transform of log filterbank amplitudes:

$$c_i = (2/N)^{1/2} \sum_{j=1}^{N} \log m_j \cos\left(\frac{\pi i}{N}(j-0.5)\right)$$

- Result is ~12 Mel Frequency Cepstral Coefficients (MFCC)
- Almost independent (unlike mel filterbank)
- Use Delta (velocity / first derivative) and Delta$^2$ (acceleration / second derivative) of MFCC (+ ~24 features)



## Additional Signal Processing

- **Pre-emphasis** prior to Fourier transform to boost high level energy
- **Liftering** to re-scale cepstral coefficients
- **Channel Adaptation** to deal with line and microphone characteristics (example: cepstral mean normalization)
- **Echo Cancellation** to remove background noise (including speech generated from the synthesizer)
- Adding a **Total (log) Energy** feature (+/- normalization)
- **End-pointing** to detect signal start and stop

## Tutorial Overview: Outline

### Part I

- Signal processing
- Speech recognition
  - acoustic modeling
  - language modeling
  - decoding
- Semantic interpretation
- Speech synthesis

### Part II

- Discourse and dialogue
  - Discourse interpretation
  - Dialogue management
  - Response generation
- Dialogue evaluation
- Data collection



## Properties of Recognizers

- **Speaker Independent** vs. Speaker Dependent
- **Large Vocabulary** (2K-200K words) vs. Limited Vocabulary (2-200)
- **Continuous** vs. Discrete
- **Speech Recognition** vs. Speech Verification
- **Real Time** vs. multiples of real time
- **Spontaneous Speech** vs. Read Speech
- Noisy Environment vs. Quiet Environment
- High Resolution Microphone vs. Telephone vs. Cellphone
- Adapt to speaker vs. non-adaptive
- Low vs. High Latency
- With online incremental results vs. final results

## The Speech Recognition Problem

- **Bayes' Law**
  - $P(a,b) = P(a|b)\,P(b) = P(b|a)\,P(a)$
  - Joint probability of *a* and *b* = probability of *b* times the probability of *a* given *b*

- The **Recognition Problem**
  - Find most likely sequence **w** of "words" given the sequence of acoustic observation vectors **a**
  - Use Bayes' law to create a **generative model**
  - $\text{ArgMax}_{\mathbf{w}}\ P(\mathbf{w}|\mathbf{a}) = \text{ArgMax}_{\mathbf{w}}\ P(\mathbf{a}|\mathbf{w})\,P(\mathbf{w}) / P(\mathbf{a})$
    $= \text{ArgMax}_{\mathbf{w}}\ P(\mathbf{a}|\mathbf{w})\,P(\mathbf{w})$

- **Acoustic Model:** $P(\mathbf{a}|\mathbf{w})$
- **Language Model:** $P(\mathbf{w})$



## Tutorial Overview: Outline

### Part I

- Signal processing
- Speech recognition
  - acoustic modeling
  - language modeling
  - decoding
- Semantic interpretation
- Speech synthesis

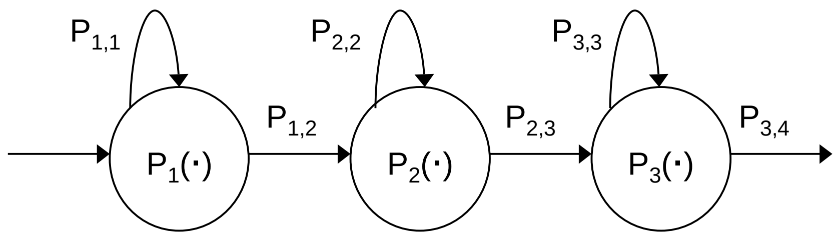
### Part II

- Discourse and dialogue
  - Discourse interpretation
  - Dialogue management
  - Response generation

- Dialogue evaluation
- Data collection

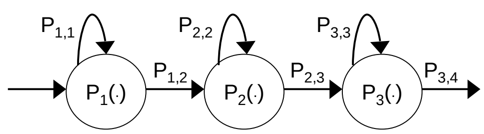
## Hidden Markov Models (HMMs)

- HMMs provide generative **acoustic models** P(**a**|**w**)
- probabilistic, non-deterministic finite-state automaton
  - state *n* generates feature vectors with density $P_n$
  - transitions from state *j* to *n* are probabilistic $P_{j,n}$






## HMMs: Single Gaussian Distribution



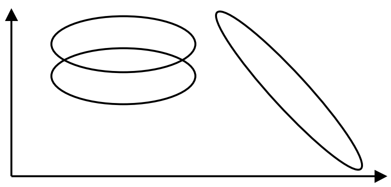

- Outgoing likelihoods: $\sum_n P_{j,n} = 1$
- Feature vector $\mathbf{a}$ generated by normal density (Gaussian) with mean $\eta$ and covariance matrix $\Sigma$

$$\mathrm{P}_n(\mathbf{a}) = \mathrm{N}(\mathbf{a} \mid \eta_n, \Sigma_n)$$
$$= (2\pi)^{-d/2} \, |\Sigma_n|^{-1/2} \, \exp(-\frac{1}{2}(\mathbf{a}-\eta_n)^T \Sigma_n^{-1}(\mathbf{a}-\eta_n))$$

## HMMs: Gaussian Mixtures

- To account for **variable pronunciations**
- Each state generates acoustic vectors according to a **linear combination** of $m$ Gaussian models, weighted by $\lambda_m$:

$$\mathrm{P}_n(\mathbf{a}) = \sum_m \lambda_{n,m} \, \mathrm{N}(\mathbf{a} \mid \eta_{n,m}, \Sigma_{n,m})$$

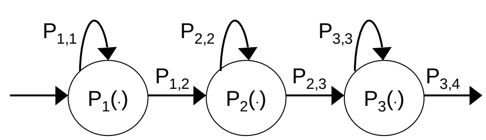


Three-component mixture model in two dimensions



## Acoustic Modeling with HMMs

- Train HMMs to represent **subword** units
- Units typically segmental; may vary in granularity
  - phonological (~40 for English)
  - phonetic (~60 for English)
  - **context-dependent triphones** (~14,000 for English): models temporal and spectral transitions between phones
  - **silence** and **noise** are usually additional symbols
- **Standard architecture** is three successive states per phone:



## Pronunciation Modeling

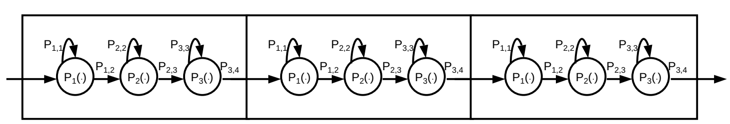
- Needed for speech recognition and synthesis
- Maps orthographic representation of words to sequence(s) of phones
- Dictionary doesn’t cover language due to:
  - open classes
  - names
  - inflectional and derivational morphology
- Pronunciation variation can be modeled with multiple pronunciation and/or acoustic mixtures
- If multiple pronunciations are given, estimate likelihoods
- Use rules (e.g. assimilation, devoicing, flapping), or statistical transducers



## Lexical HMMs

- Create compound HMM for each lexical entry by concatenating the phones making up the pronunciation
  - example of HMM for ‘lab’ (following ‘speech’ for crossword triphone)

| | | | |
|---|---|---|---|
| triphone: | ch-**l**+a | l-**a**+b | a-**b**+# |
| phone: | l | a | b |



- Multiple pronunciations can be weighted by likelihood into compound HMM for a word
- (Tri)phone models are independent parts of word models

## HMM Training: Baum-Welch Re-estimation

- Determines the probabilities for the acoustic HMM models
- **Bootstraps** from initial model
  - hand aligned data, previous models or flat start
- Allows **embedded training** of whole utterances:
  - transcribe utterance to words $w_1,\ldots,w_k$ and generate a compound HMM by concatenating compound HMMs for words: $\mathbf{m}_1,\ldots,\mathbf{m}_k$
  - calculate acoustic vectors: $\mathbf{a}_1,\ldots,\mathbf{a}_n$
- Iteratively **converges** to a new estimate
- Re-estimates all paths because states are hidden
- Provides a **maximum likelihood** estimate
  - model that assigns training data the highest likelihood



## Tutorial Overview: Outline

### Part I

- Signal processing
- Speech recognition
  - acoustic modeling
  - language modeling
  - decoding
- Semantic interpretation
- Speech synthesis

### Part II

- Discourse and dialogue
  - Discourse interpretation
  - Dialogue management
  - Response generation
- Dialogue evaluation
- Data collection

## Probabilistic Language Modeling: History

- Assigns probability P(**w**) to word sequence **w** = $w_1, w_2, \ldots, w_k$
- Bayes' Law provides a **history-based** model:

  $P(w_1, w_2, \ldots, w_k)$

  $= P(w_1)\, P(w_2|w_1)\, P(w_3|w_1,w_2) \cdots P(w_k|w_1,\ldots,w_{k-1})$
- **Cluster** histories to reduce number of parameters



## *N*-gram Language Modeling

- *n*-gram assumption clusters based on last *n*-1 words
  - $P(w_j|w_1,\ldots,w_{j-1}) \sim P(w_j|w_{j-n-1},\ldots,w_{j-2}, w_{j-1})$
  - unigrams ~ $P(w_j)$
  - bigrams ~ $P(w_j|w_{j-1})$
  - trigrams ~ $P(w_j|w_{j-2}, w_{j-1})$
- Trigrams often interpolated with bigram and unigram:

$$\hat{P}(w_3 \mid w_1, w_2) = \lambda_3 \frac{F(w_3 \mid w_1, w_2)}{\sum_k F(w_k \mid w_1, w_2)} + \lambda_2 \frac{F(w_3 \mid w_2)}{\sum_k F(w_k \mid w_2)} + \lambda_1 \frac{F(w_3)}{\sum_k F(w_k)}$$

  - the $\lambda_i$ typically estimated by maximum likelihood estimation on held out data ($F(.|.)$ are relative frequencies)
  - many other interpolations exist (another standard is a non-linear **backoff**)

## Extended Probabilistic Language Modeling

- Histories can include some indication of semantic topic
  - latent-semantic indexing (vector-based information retrieval model)
  - topic-spotting and blending of topic-specific models
  - dialogue-state specific language models
- Language models can adapt over time
  - recent history updates model through re-estimation or blending
  - often done by boosting estimates for seen words (triggers)
  - new words and/or pronunciations can be added
- Can estimate category tags (syntactic and/or semantic)
  - Joint word/category model: $P(word_1{:}tag_1,\ldots,word_k{:}tag_k)$
  - example: P(word:tag|History) ~ P(word|tag) P(tag|History)



## Finite State Language Modeling

- Write a finite-state task grammar (with non-recursive CFG)
- Simple Java Speech API example (from user's guide):

```
public <Command> = [<Polite>] <Action> <Object> (and <Object>)*;
        <Action> = open | close | delete;
        <Object> = the window | the file;
        <Polite> = please;
```

- Typically assume that all transitions are equi-probable
- Technology used in most current applications
- Can put semantic actions in the grammar

## Information Theory: Perplexity

- Perplexity is standard model of recognition complexity given a language model
- Perplexity measures the conditional likelihood of a corpus, given a language model $\mathrm{P}(.)$:

$$\mathrm{PP}(w_1,\ldots,w_N) = \mathrm{P}(w_1,\ldots,w_N)^{-1/N}$$

- Roughly the number of equi-probable choices per word
- Typically computed by taking logs and applying history-based Bayesian decomposition:

$$\log_2 \mathrm{PP} = -1/N \sum_{n=1}^{N} \log_2 \mathrm{P}(w_n \mid w_1,\ldots,w_{n-1})$$

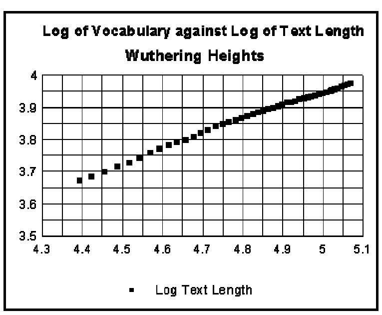
- But lower perplexity doesn't guarantee better recognition



Lucent Technologies
Bell Labs Innovations

## Zipf's Law

- Lexical frequency is inversely proportional to rank
  - Frequency($n$) = Frequency of $n$-th most frequent word
  - **Zipf's Law**: Frequency(Rank) = Frequency(1)/Rank
  - Thus: log Frequency(Rank) ∝ - log Rank




From G.R. Turner's web site on Zipf's law:
http://www.btinternet.com/~g.r.turner/ZipfDoc.htm



Lucent Technologies
Bell Labs Innovations

## Vocabulary Acquisition

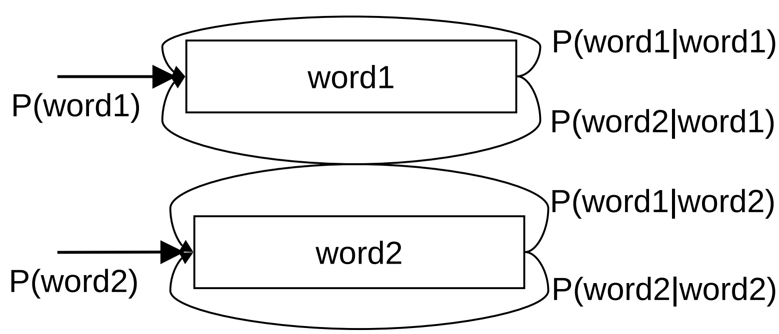
- IBM personal E-mail corpus of PDB (by R.L. Mercer)
- static coverage is given by most frequent *n* words
- dynamic coverage is most recent *n* words

| Vocabulary | Static Coverage | Dynamic Coverage | Text Size |
|---|---|---|---|
| 5,000 | 92.5 | 95.5 | 56,000 |
| 10,000 | 95.9 | 98.2 | 240,000 |
| 15,000 | 97.0 | 99.0 | 640,000 |
| 20,000 | 97.6 | 99.5 | 1,300,000 |



## Language HMMs

- Can take HMMs for each word and combine into a single HMM for the whole language (allows **cross-word** models)
- Result is usually too large to expand statically in memory
- A two word example is given by:



## Tutorial Overview: Outline

### Part I

- Signal processing
- Speech recognition
  - acoustic modeling
  - language modeling
  - decoding
- Semantic interpretation
- Speech synthesis

### Part II

- Discourse and dialogue
  - Discourse interpretation
  - Dialogue management
  - Response generation
- Dialogue evaluation
- Data collection



## HMM Decoding

- **Decoding Problem** is finding best word sequence:
  - **ArgMax** $_{w1,\ldots,wm}$ $P(w_1,\ldots,w_m \mid \mathbf{a}_1,\ldots,\mathbf{a}_n)$
- Words $w_1 \ldots w_m$ are fully determined by sequences of states
- Many state sequences produce the same words
- The **Viterbi assumption:**
  - the word sequence derived from the most likely path will be the most likely word sequence (as would be computed over all paths)

$$\phi_i(s) = \mathrm{Max}\ \ \mathrm{P}(s_1,\ldots,s_i \mid \mathbf{a}_1,\ldots,\mathbf{a}_i) = \mathrm{P}_s(\mathbf{a}_i)\,\mathrm{Max}_r\ \phi_{i-1}(r)\,\mathrm{P}_{r,s}$$

Acoustics for state $s$ for input **a**

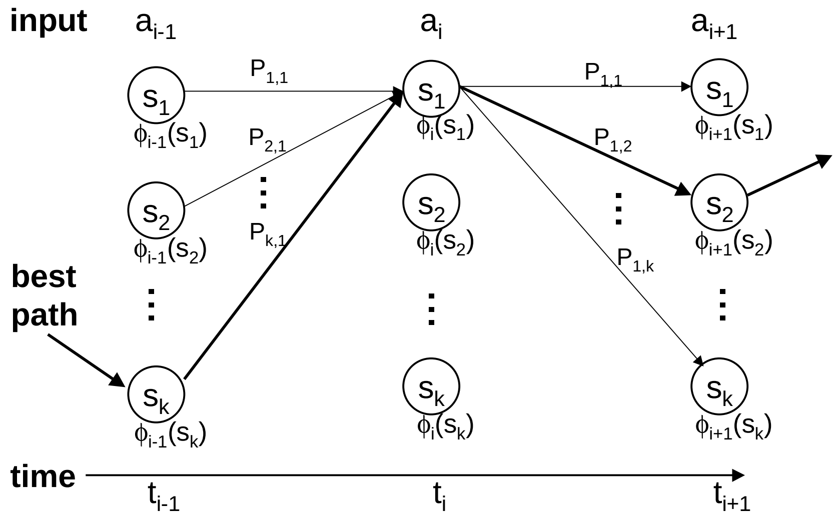
Max over previous states $r$

likelihood previous state is $r$

Transition probability from $r$ to $s$

## Visualizing Viterbi Decoding: The Trellis

$$\phi_i(s) = \mathrm{Max}\ \mathrm{P}(s_1,\ldots,s_i \mid \mathbf{a}_1,\ldots,\mathbf{a}_i) = \mathrm{P}_s(\mathbf{a}_i)\,\mathrm{Max}_r\ \phi_{i-1}(r)\,\mathrm{P}_{r,s}$$






## Viterbi Search: Dynamic Programming Token Passing

- Algorithm:
  - Initialize all states with a token with a null history and the likelihood that it's a start state
  - For each frame $a_k$
    - For each token t in state s with probability P(t), history H
      - For each state r
        - » Add new token to s with probability P(t) $P_{s,r}$ $P_r(a_k)$, and history s.H
- Time synchronous from left to right
- Allows incremental results to be evaluated

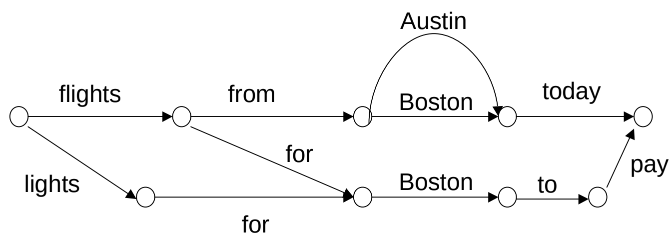
## Pruning the Search Space

- Entire search space for Viterbi search is much too large
- Solution is to **prune** tokens for paths whose score is too low
- Typical method is to use:
  - **histogram:** only keep at most n total hypotheses
  - **beam:** only keep hypotheses whose score is a fraction of best score
- Need to balance small *n* and tight beam to limit search and minimal search error (good hypotheses falling off beam)
- HMM densities are usually scaled differently than the discrete likelihoods from the language model
  - typical solution: boost language model's dynamic range, using $P(\mathbf{w})^n P(\mathbf{a}|\mathbf{w})$, usually with with $n \sim 15$
- Often include penalty for each word to favor hypotheses with fewer words



## N-best Hypotheses and Word Graphs

- Keep multiple tokens and return n-best paths/scores:
  - p1 flights from Boston today
  - p2 flights from Austin today
  - p3 flights for Boston to pay
  - p4 lights for Boston to pay
- Can produce a packed word graph (a.k.a. lattice)
  - likelihoods of paths in lattice should equal likelihood for n-best



## Search-based Decoding

- **A* search:**
  - Compute all initial hypotheses and place in priority queue
  - For best hypothesis in queue
    - extend by one observation, compute next state score(s) and place into the queue
- Scoring now compares derivations of **different lengths**
  - would like to, but can't compute cost to complete until all data is seen
  - instead, estimate with simple normalization for length
  - usually prune with beam and/or histogram constraints
- Easy to include unbounded amounts of **history** because no collapsing of histories as in dynamic programming n-gram
- Also known as **stack decoder** (priority queue is "stack")



## Multiple Pass Decoding

- Perform multiple passes, applying successively more fine-grained language models
- Can much more easily go beyond finite state or n-gram
- Can use for Viterbi or stack decoding
- Can use word graph as an efficient interface
- Can compute likelihood to complete hypotheses after each pass and use in next round to tighten beam search
- First pass can even be a free phone decoder without a word-based language model

## Measuring Recognition Accuracy

- **Word Error Rate** = $\dfrac{Insertions + Deletions + Substitutions}{Words}$
- Example scoring:
  - actual utterance: four six seven nine three three seven
  - recognizer: four oh six seven five three seven

    insert subst delete
  - WER: (1 + 1 + 1)/7 = 43%
- Would like to study **concept accuracy**
  - typically count only errors on content words [application dependent]
  - ignore case marking (singular, plural, etc.)
- For **word/concept spotting** applications:
  - **recall**: percentage of target words (concept) found
  - **precision:** percentage of hypothesized words (concepts) in target



## Empirical Recognition Accuracies

- Cambridge **HTK**, 1997; multipass HMM w. lattice rescoring
- **Top Performer** in ARPA's HUB-4: Broadcast News Task
- 65,000 word vocabulary; Out of Vocabulary: 0.5%
- Perplexities:
  - word bigram: 240 (6.9 million bigrams)
  - backoff trigram of 1000 categories: 238 (803K bi, 7.1G tri)
  - word trigram: 159 (8.4 million trigrams)
  - word 4-gram: 147 (8.6 million 4-grams)
  - word 4-gram + category trigram: 137
- Word Error Rates:
  - clean, read speech: 9.4%
  - clean, spontaneous speech: 15.2%
  - low fidelity speech: 19.5%

## Empirical Recognition Accuracies (cont'd)

- Lucent 1998, single pass HMM
- Typical of **real-time** telephony performance (low fidelity)
- 3,000 word vocabulary; Out of Vocabulary: 1.5%
- Blended models from customer/operator & customer/system
- Perplexities customer/op customer/system

| Perplexities | customer/op | customer/system |
|---|---|---|
| – bigram: | 105.8 (27,200) | 32.1 (12,808) |
| – trigram: | 99.5 (68,500) | 24.4 (25,700) |

- Word Error Rate: 23%
- Content Term (single, pair, triple of words) Precision/Recall
  - one-word terms: 93.7 / 88.4
  - two-word terms: 96.9 / 85.4
  - three-word terms: 98.5 / 84.3



## Confidence Scoring and Rejection

- Alternative to standard acoustic density scoring
  - compute HMM acoustic score for word(s) in usual way
  - baseline score for an **anti-model**
  - compute hypothesis ratio (Word Score / Baseline Score)
  - test hypothesis ratio vs. threshold
- Can be applied to:
  - free word spotting (given pronunciations)
  - (word-by-word) acoustic confidence scoring for later processing
  - verbal information verification
    - existing info: name, address, social security number
    - password

## Tutorial Overview: Outline

### Part I

- Signal processing
- Speech recognition
  - acoustic modeling
  - language modeling
  - decoding
- Semantic interpretation
- Speech synthesis

### Part II

- Discourse and dialogue
  - Discourse interpretation
  - Dialogue management
  - Response generation
- Dialogue evaluation
- Data collection



## Semantic Interpretation: Word Strings

- Content is just words
  - *System:* What is your address?
  - *User:* fourteen eleven main street
- Can also do concept extraction / keyword(s) spotting
  - *User:* My address is **fourteen eleven main street**
- Applications
  - template filling
  - directory services
  - information retrieval

## Semantic Interpretation: Pattern-Based

- Simple (typically regular) patterns specify content
- ATIS (Air Traffic Information System) Task:
  - *System:* What are your travel plans?
  - *User:* [On Monday], I'm going [from Boston] [to San Francisco].
  - Content: [DATE=Monday, ORIGIN=Boston, DESTINATION=SFO]
- Can combine content-extraction and language modeling
  - but can be too restrictive as a language model
- Java Speech API: (curly brackets show semantic 'actions')

```
public <command> = <action> [<object>] [<polite>];
    <action> = open {OP} | close {CL} | move {MV};
    <object> = [<this_that_etc>] window | door;
    <this_that_etc> = a | the | this | that | the current;
    <polite> = please | kindly;
```

- Can be generated and updated on the fly (eg. Web Apps)



## Semantic Interpretation: Parsing

- In general case, have to uncover who did what to whom:
  - System: What would you like me to do next?
  - User: Put the block in the box on Platform 1. [ambiguous]
  - System: How can I help you?
  - User: Where is A Bug's Life playing in Summit?
- Requires some kind of parsing to produce relations:
  - Who did what to whom:
    ?(where(present(in(Summit,play(BugsLife)))))
  - This kind of representation often used for machine translation
- Often transferred to flatter frame-based representation:
  - Utterance type: where-question
  - Movie: A Bug's Life
  - Town: Summit

## Robustness and Partiality

- Controlled Speech
  - limited task vocabulary; limited task grammar
- Spontaneous Speech
  - Can have high out-of-vocabulary (OOV) rate
  - Includes restarts, word fragments, omissions, phrase fragments, disagreements, and other disfluencies
  - Contains much grammatical variation
  - Causes high word error-rate in recognizer
- Parsing is often partial, allowing:
  - omission
  - parsing fragments



## Tutorial Overview: Outline

### Part I

- Signal processing
- Speech recognition
  - acoustic modeling
  - language modeling
  - decoding
- Semantic interpretation
- Speech synthesis

### Part II

- Discourse and dialogue
  - Discourse interpretation
  - Dialogue management
  - Response generation
- Dialogue evaluation
- Data collection

## Recorded Prompts

- The simplest (and most common) solution is to record prompts spoken by a (trained) human
- Produces human quality voice
- Limited by number of prompts that can be recorded
- Can be extended by limited cut-and-paste or template filling



## Speech Synthesis

- Rule-based Synthesis
  - Uses linguistic rules (+/- training) to generate features
  - Example: DECTalk
- Concatenative Synthesis
  - Record basic inventory of sounds
  - Retrieve appropriate sequence of units at run time
  - Concatenate and adjust durations and pitch
  - Waveform synthesis

## Diphone and Polyphone Synthesis

- Phone sequences capture **co-articulation**
- Cut speech in positions that minimize context contamination
- Need single phones, diphones and sometimes triphones
- Reduce number collected by
  - phonotactic constraints
  - collapsing in cases of no co-articulation
- Data Collection Methods
  - Collect data from a single (professional) speaker
  - Select text with maximal coverage (typically with greedy algorithm), or
  - Record minimal pairs in desired contexts (real words or nonsense)



## Duration Modeling

Must generate segments with the appropriate duration

- Segmental Identity
  - /ai/ in like twice as long as /I/ in lick
- Surrounding Segments
  - vowels longer following voiced fricatives than voiceless stops
- Syllable Stress
  - onsets and nuclei of stressed syllables longer than in unstressed
- Word “importance”
  - word accent with major pitch movement lengthens
- Location of Syllable in Word
  - word ending longer than word starting longer than word internal
- Location of the Syllable in the Phrase
  - phrase final syllables longer than same syllable in other positions

## Intonation: Tone Sequence Models

- Functional Information can be encoded via tones:
  - given/new information (information status)
  - contrastive stress
  - phrasal boundaries (clause structure)
  - dialogue act (statement/question/command)

- Tone Sequence Models
  - F0 contours generated from phonologically distinctive tones/pitch accents which are locally independent
  - generate a sequence of tonal targets and fit with signal processing

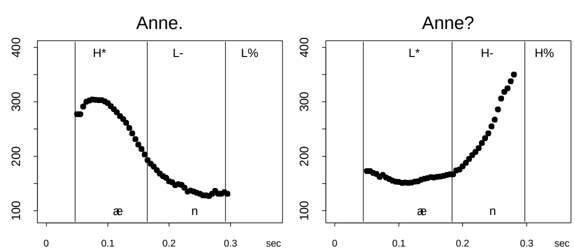

Lucent Technologies
Bell Labs Innovations

## Intonation for Function

- ToBI (Tone and Break Index) System, is one example:
  - **Pitch Accent** * (H*, L*, H*+L, H+L*, L*+H, L+H*)
  - **Phrase Accent** - (H-, L-)
  - **Boundary Tone** % (H%, L%)
  - **Intonational Phrase**

    <Pitch Accent>+ <Phrase Accent> <Boundary Tone>

statement vs. question example:




source: *Multilingual Text-to-Speech Synthesis*, R. Sproat, ed., Kluwer, 1998


Lucent Technologies
Bell Labs Innovations

## Text Markup for Synthesis

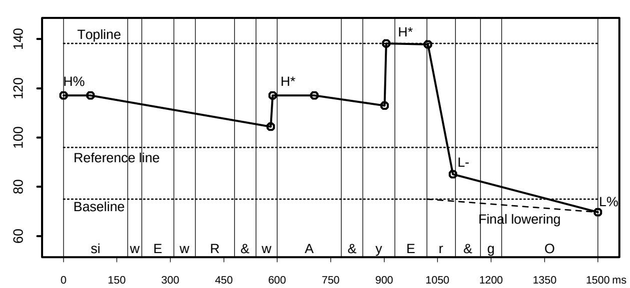
- Bell Labs TTS Markup
  - r(0.9) L*+H(0.8) ***Humpty*** L*+H(0.8) ***Dumpty*** r(0.85) L*(0.5) ***sat on a*** H*(1.2) ***wall***.
  - **Tones**: Tone(Prominence)
  - **Speaking Rate**: r(Rate) and pauses
  - **Top** Line (highest pitch); **Reference** Line (reference pitch); **Base** Line (lowest pitch)

- SABLE is an emerging standard extending SGML
  http://www.cstr.ed.ac.uk/projects/sable.html
  - marks: emphasis(#), break(#), pitch(base/mid/range,#), rate(#), volume(#), semanticMode(date/time/email/URL/...), speaker(age,sex)
  - Implemented in **Festival** Synthesizer (free for research, etc.):
    http://www.cstr.ed.ac.uk/projects/festival.html



## Intonation in ***Bell Labs TTS***

- Generate a sequence of F0 targets for synthesis
- Example:
  - We were away a year ago.
  - phones: w E w R & w A & y E r & g O
  - Default Declarative intonation: (H%) H* L- L% [question: L* H- H%]




source: *Multilingual Text-to-Speech Synthesis*, R. Sproat, ed., Kluwer, 1998

## Signal Processing for Speech Synthesis

- Diphones recorded in one context must be generated in other contexts
- Features are extracted from recorded units
- Signal processing manipulates features to smooth boundaries where units are concatenated
- Signal processing modifies signal via ‘interpolation’
  - intonation
  - duration



## The Source-Filter Model of Synthesis

- Model of features to be extracted and fitted
- Excitation or Voicing Source(s) to model sound source
  - standard wave of glottal pulses for voiced sounds
  - randomly varying noise for unvoiced sounds
  - modification of airflow due to lips, etc.
  - high frequency (F0 rate), quasi-periodic, choppy
  - modeled with vector of glottal waveform patterns in voiced regions
- Acoustic Filter(s)
  - shapes the frequency character of vocal tract and radiation character at the lips
  - relatively slow (samples around 5ms suffice) and stationary
  - modeled with LPC (linear predictive coding)

## Barge-in

- Technique to allow speaker to interrupt the system's speech
- Combined processing of input signal and output signal
- Signal detector runs looking for speech start and endpoints
  - tests a generic speech model against noise model
  - typically cancels echoes created by outgoing speech
- If speech is detected:
  - Any synthesized or recorded speech is cancelled
  - Recognition begins and continues until end point is detected



## Speech Application Programming Interfaces

- Abstract from recognition/synthesis engines
- Recognizer and synthesizer loading
- Acoustic and grammar model loading (dynamic updates)
- Recognition
  - online
  - n-best or lattice
- Synthesis
  - markup
  - barge in
- Acoustic control
  - telephony interface
  - microphone/speaker interface

## Speech API Examples

- SAPI: Microsoft Speech API (rec&synth)
  - communicates through COM objects
  - instances: most systems implement all or some of this (Dragon, IBM, Lucent, L&H, etc.)
- JSAPI: Java Speech API (rec & synth)
  - communicates through Java events (like GUI)
  - concurrency through threads
  - instances: IBM ViaVoice (rec), L&H (synth)
- (J)HAPI: (Java) HTK API (recognition)
  - communicates through C or Java port of C interface
  - eg: Entropics Cambridge Research Lab's HMM Tool Kit (HTK)
- Galaxy (rec & synth)
  - communicates through a production system scripting language
  - MIT System, ported by MITRE for DARPA Communicator