Speech Recognition, Understanding and Dialog Modeling
Kuansan Wang
Microsoft Research
Acknowledgement

• Adapted from ICASSP 2002 tutorial by Alex Acero and Mazim Rahim
Components in a Spoken Dialog System

- Speech recognition
  - Speech to text
- Spoken language understanding
  - Text to meaning
- Discourse and Dialog Management
  - Meaning to actions
  - Meaning for reporting outcomes
- Natural Language Generation
  - Meaning to text
- Speech synthesis
  - Text to speech
Speech Recognition
• 1950s
  – Isolated word recognition
  – Single speaker, spectral resonance analysis
• 1960s
  – Dynamic time warping
  – Dynamic programming
  – Pattern recognition
• 1970s
  – Statistical approach: Hidden Markov Model
  – LPC for recognition (Itakura)
  – Simple dictation (IBM Tangora)
• 1980s
  – Continuous speech recognition
  – Speaker independent recognition
  – DARPA Investment (e.g. CMU Sphinx, BBN BYPLOS)
• 1990s
  – Commercialization
  – Microsoft (SAPI, Office, Windows), IBM (ViaVoice), AT&T (VRCP, ATT Direct, etc), Philips (FreeSpeech, telephony solutions), L&H (Dragon Dictate, Automobile solution), Nuance, SpeechWorks, etc.
Pattern Recognition
Formulation

• Given a speech waveform \( x = (x_1, x_2, \ldots x_n) \), find the best word sequence \( w = (w_1, w_2, \ldots w_m) \)
• Minimum error decision:

\[
\hat{w} = \arg \max_{w \in L^*} P(w \mid x) = \arg \max_{w \in L^*} P(x \mid w)P(w)
\]

• \( P(x \mid w) \): acoustic model
• \( P(w) \): language model
• \( L \): lexicon
Speech Recognition Process

Input Speech → Feature Extraction → Acoustic Model \( P(x|w) \) → Pattern Classification (argmax) → Confidence Scoring

Language Model \( P(w) \) → Word Lexicon \( L \) → "Hello World"
Feature Extraction

**Goal:** Extract *robust* features (information) from the speech that are *relevant* for ASR.

**Method:** Spectral analysis through either a bank-of-filters or through Linear Predictive Coding followed by non-linearity and normalization.

**Result:** Signal compression where for each window of speech samples where 30 or so features are extracted (64,000 b/s -> 5,200 b/s).

**Challenges:** *Robustness* to environment (office, airport, car), devices (speakerphones, cellphones), speakers (accents, dialect, style, speaking defects), noise and echo.
What Features to Use?

Short-time Analysis:

**Acoustic features:**
- Cepstrum (LPC, Filterbank, wavelets)
- Formant frequencies, pitch, prosody
- Zero-crossing rate, Energy

**Acoustic-Phonetic features:**
- Manner of articulation (e.g., stop, nasal, voiced)
- Place of articulation (labial, dental, velar)

**Articulatory features:**
- Jaw, lips, velum, tongue position

**Auditory features:**
- Mel-frequency warping, RASTA

**Temporal analysis:** Approximation of the velocity and acceleration typically through the central difference.
Feature Extraction Process

Quantization

$\alpha$

Preemphasis

$M, N$

Segmentation (blocking)

$W(n)$

Energy

Zero-crossing

Windowing

$s(t)$

$s(n)$

$s'(n)$

$s''(n)$

$s''(n)$

$s'(n)$

$s(n)$

$s(t)$

Filtering

Noise removal, Normalization

$a'(m(l))$

$a_m(l)$

$\Delta c'(m(l))$

Spectral Analysis

Pitch Formants

Bias removal or normalization

Cepstral Analysis

$cm(l)$

$\Delta c'(m(l))$

Equalization

Temporal Derivative

$\Delta c'(m(l))$

$\Delta^2 c'(m(l))$
**Acoustic Modeling**

**Goal:** Map acoustic features into distinct subword units, such as phones, syllables, words, etc.

**Hidden Markov Model (HMM):** Statistical method for characterizing the spectral properties of speech by a parametric random process. A collection of HMMs is associated with a subword unit. HMMs are also assigned for modeling extraneous events.

**Advantages:** Powerful statistical method for dealing with a wide range of data and conditions, and is highly reliable for recognizing speech.
Discrete-Time Markov Process

- Example: stock market

States are observable.
Hidden Markov Models: States are not observable (hidden)

Initial state prob. = \[
\begin{bmatrix}
0.5 \\
0.2 \\
0.3 \\
\end{bmatrix}
\]

Output pdf = \[
\begin{bmatrix}
P(up) \\
P(down) \\
P(unchanged)
\end{bmatrix}
\]
Example

- I observe (up, up, down, up, unchanged, up), is it a bull market? bear market? or trading range?

  \[
  P(\text{all bull}) = 0.7*0.7*0.1*0.7*0.2*0.7*[0.5* (0.6)^5] = 1.867*10^{-4}
  \]
  \[
  P(\text{all bear}) = 0.1*0.1*0.6*0.1*0.3*0.1*[0.2* (0.3)^5] = 8.748*10^{-9}
  \]
  \[
  P(\text{all steady}) = 0.3*0.3*0.3*0.3*0.4*0.3*[0.3*(0.5)^5] = 9.1125*10^{-6}
  \]

- It’s 20 times more likely that we are in a bull market than a steady market!

- How about

  \[
  P(\text{bull,bull,bear,bull,steady,bull})= \\
  =(0.7*0.7*0.6*0.7*0.4*0.7)*(0.5*0.6*0.2*0.5*0.2*0.4)=1.382976*10^{-4}
  \]
Basic Problems in HMMs

Given acoustic observation $X$ and model $\Phi$ where $\Phi$ consists of:

- Transition probabilities
- Output probabilities
- Initial probabilities

**Evaluation:** Compute $P(X \mid \Phi)$

**Decoding:** Choose optimal state sequence

**Re-estimation:** Adjust $\Phi$ to maximize $P(X \mid \Phi)$
Evaluation: intuition

• Sum it over all possible hidden state sequences $q=(q_1, q_2, \ldots, q_k)$. Very expensive.

$$P(X \mid \Phi) = \sum_q P(X, q \mid \Phi)$$

$$= \sum_q P(X \mid q, \Phi) P(q \mid \Phi)$$

Output Probabilities

| Initial/Transition Probabilities |
**Evaluation: The Baum-Welch algorithm**

\[
\alpha_t(i) = P(X_t^i, s_t = i \mid \Phi) = \left[ \sum_{j=1}^{N} \alpha_{t-1}(j) a_{ji} \right] b_t(X_t) \quad 2 \leq t \leq T; \quad 1 \leq i \leq N \quad \text{Forward}
\]

\[
\beta_t(j) = P(X_{t+1}^T \mid s_t = j, \Phi) = \left[ \sum_{i=1}^{N} a_{ji} b_t(X_{t+1}) \beta_{t+1}(i) \right] \quad t = T - 1 \ldots 1; \quad 1 \leq j \leq N \quad \text{Backward}
\]

\[
\gamma_t(i, j) = P(s_{t-1} = i, s_t = j \mid X_1^T, \Phi) = \frac{P(s_{t-1} = i, s_t = j, X_1^T \mid \Phi)}{P(X_1^T \mid \Phi)} = \frac{\alpha_{t-1}(i) a_{ij} b_j(X_t) \beta_t(j)}{\sum_{k=1}^{N} \alpha_T(k)}
\]
**Decoding: intuition**

- Search all possible state sequences

\[
\hat{q} = \max_q P(X \mid q, \Phi) P(q \mid \Phi)
\]
Decoding: Viterbi Algorithm

Step 1: Initialization
\[ D_1(i) = \pi_i b_i(x_1), \quad B_1(i) = 0, \quad j = 1, \ldots, N \]

Step 2: Iterations
\[
\text{for } t = 2, \ldots, T \{
\text{for } j = 1, \ldots, N \{
V_t(j) = \min[V_{t-1}(i) a_{ij}] b_j(x_t)
B_t(j) = \arg\min[V_{t-1}(i) a_{ij}] \}
\}
\]

Step 3: Backtracking
The optimal score is \( V_T = \max V_t(i) \)
Final state is \( s_T = \arg\max V_t(i) \)
Optimal path is \( (s_1, s_2, \ldots, s_T) \) where
\( s_t = B_{t+1}(s_{t+1}) \quad t = T-1, \ldots, 1 \)

Optimal alignment between \( X \) and \( S \)
Reestimation: Training Algorithm

• Find model parameters $\Phi = (A, B, \pi)$ that maximize the probability $p(X|\Phi)$ of observing some data $X$
• There is no a closed-form direct solution, but can use an iterative process called EM algorithm
• We have an old parameter value $\Phi$ and we obtain a new parameter $\hat{\Phi}$ that maximizes the following auxiliary function $Q$

$$Q(\Phi, \hat{\Phi}) = E[\log P(X, q | \hat{\Phi})] = \sum_{q} P(X, q | \Phi) \log P(X, q | \hat{\Phi})$$
• It can be proved that the new parameters will yield higher likelihood scores than the old ones
EM Training Procedure

• Guess an initial parameter set
• **Expectation**: Find the formula for the conditional expectancy (usually close form)
• **Maximization**: maximize the auxiliary function with respect to the parameters (usually take derivative and set to 0).
• Repeat the E-M steps until convergence
Design Issues

- Continuous vs. Discrete HMM
- Whole-word vs. subword (phone units)
- Number of states, model parameters (e.g., Gaussians)
- Ergodic vs. Bakis
- Context-dependent vs. context-independent
Training with continuous speech

• No segmentation is needed
• Composed HMM
Context Variability in Speech

• At word/sentence level: Mr. Wright should write to Ms. Wright right away about his Ford or four door Honda.

• At phone level: /ee/ for word peat and wheel

• Triphones in speech capture coarticulation, phonetic context
**Context-dependent models**

- Triphone IY(P, CH) in speech captures coarticulation, phonetic context

- Stress: Italy vs Italian
Clustering similar triphones

• /iy/ with two different left contexts /r/ and /w/ are similar
• Cluster those triphones together
Clustering with decision trees

**Is left phone a sonorant or nasal?**
- yes: Is right phone a back-R?
- no: Is left phone /s,z,sh,zh/?

- Is right phone voiced?
  - yes: Is left phone a back-L or (is left phone neither a nasal nor a Y-glide and right phone a LAX-vowel)?
  - no: Is left phone /s,z,sh,zh/?

- **Welcome**
Goal:
Map legal phone sequences into words according to *phonotactic* rules. For example,

David /d/ /ey/ /v/ /ih/ /d/

Multiple Pronunciation:
Several words may have multiple pronunciations. For example

Data /d/ /ae/ /t/ /ax/
Data /d/ /ey/ /t/ /ax/

Challenges:
How do you generate a word lexicon automatically; how do you add new variant dialects and word pronunciations.
The Lexicon

- An entry per word, at least 100K words needed for dictation (\textit{find}: /f/ /ay/ /n/ /d/)
- Either done by hand or with letter-to-sound rules (LTS). Rules can be automatically trained with decision trees (CART): less than 8% errors, but proper nouns are hard!
- Multiple pronunciations (tomato)
**Language Model**

**Goal:**
Model “acceptable” spoken phrases, constrained by task syntax.

**Rule-based:**
Deterministic grammars that are Knowledge driven. For example,

```
flying from $city to $city on $date
```

**Statistical:**
Compute estimate of word probabilities (N-gram, class-based, CFG).
For example

```
flying from Newark to Boston tomorrow
```

![Diagram showing the process of language model with nodes for acoustic model, feature extraction, pattern classification, confidence scoring, language model, and word lexicon.]
Rewrite Rules:

1. $S \rightarrow NP\ VP$
2. $VP \rightarrow V\ NP$
3. $VP \rightarrow AUX\ VP$
4. $NP \rightarrow ART\ NP1$
5. $NP \rightarrow ADJ\ NP1$
6. $NP1 \rightarrow ADJ\ NP1$
7. $NP1 \rightarrow N$
8. $NP \rightarrow NAME$
9. $NP \rightarrow PRON$
10. $NAME \rightarrow Mary$
11. $V \rightarrow loves$
12. $ADJ \rightarrow that$
13. $N \rightarrow person$
## Chomsky Grammar Hierarchy

<table>
<thead>
<tr>
<th>Types</th>
<th>Constraints</th>
<th>Automata</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase structure grammar</td>
<td>$\alpha \rightarrow \beta$. This is the most general grammar.</td>
<td>Turing machine</td>
</tr>
<tr>
<td>Context-sensitive grammar</td>
<td>Subset of $\alpha \rightarrow \beta$ $</td>
<td>\alpha</td>
</tr>
<tr>
<td>Context-free grammar (CFG)</td>
<td>$A \rightarrow w$ and $A \rightarrow BC$, where $w$ is a terminal and $B$, $C$ are non-terminals.</td>
<td>Push down automata</td>
</tr>
<tr>
<td>Regular grammar</td>
<td>$A \rightarrow w$ and $A \rightarrow wB$</td>
<td>Finite-state automata</td>
</tr>
</tbody>
</table>
Ngrams

\[ P(W) = P(w_1, w_2, \ldots, w_n) \]
\[ = P(w_1) P(w_2 | w_1) P(w_3 | w_1, w_2) \cdots P(w_n | w_1, w_2, \ldots, w_{n-1}) \]
\[ = \prod_{i=1}^{n} P(w_i | w_1, w_2, \ldots, w_{i-1}) \]

- Trigrams Estimation

\[ P(w_i | w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})} \]
Understanding Bigrams


\[
P(\text{John} \mid <s>) = \frac{C(<s>, \text{John})}{C(<s>)} = \frac{2}{3}
\]
\[
P(\text{read} \mid \text{John}) = \frac{C(\text{John}, \text{read})}{C(\text{John})} = \frac{2}{2}
\]
\[
P(a \mid \text{read}) = \frac{C(\text{read}, a)}{C(\text{read})} = \frac{2}{3}
\]
\[
P(\text{book} \mid a) = \frac{C(a, \text{book})}{C(a)} = \frac{1}{2}
\]
\[
P(<s> \mid \text{book}) = \frac{C(\text{book}, <s>)}{C(\text{book})} = \frac{2}{3}
\]

\[
P(\text{John read a book}) = P(\text{John} \mid <s>)P(\text{read} \mid \text{John})P(a \mid \text{read})P(\text{book} \mid a)P(<s> \mid \text{book}) \approx 0.148
\]

- But we have a problem here

\[
P(\text{Mulan read a book}) = P(\text{Mulan} \mid <s>)P(\text{read} \mid \text{Mulan})P(a \mid \text{read})P(\text{book} \mid a)P(<s> \mid \text{book}) = 0
\]
Ngram Smoothing

• Data sparseness: in millions of words more than 50% of trigrams occur only once.
• Can’t assign \( p(w_i|w_{i-1}, w_{i-2}) = 0 \)
• Solution: assign non-zero probability for each ngram by lowering the probability mass of seen ngrams.

\[
P_{\text{smooth}}(w_i \mid w_{i-n+1} \ldots w_{i-1}) = \lambda P_{\text{ML}}(w_i \mid w_{i-n+1} \ldots w_{i-1}) + (1 - \lambda) P_{\text{smooth}}(w_i \mid w_{i-n+2} \ldots w_{i-1})
\]
**Perplexity**

- Cross-entropy of a language model $P$ on word sequence $W$ is
  \[ H(W) = -\frac{1}{N_w} \log_2 P(W) \]

- And its perplexity
  \[ PP(W) = 2^{H(W)} \]

- measures the complexity of a language model (geometric mean of branching factor).
Perplexity

- For digit recognition task (TIDIGITS) has 10 words, $PP=10$ and 0.2% error rate
- Airline Travel Information System (ATIS) has 2000 words and $PP=20$
- Wall Street Journal Task has 5000 words and $PP=130$ with bigram and 5% error rate
- In general, lower perplexity => lower error rate, but it does not take acoustic confusability into account: E-set (B, C, D, E, G, P, T) has $PP=7$ and has 5% error rate.
**Ngram Smoothing**

- *Deleted Interpolation* algorithm estimates $\lambda$ that maximizes probability on held-out data set
- We can also map all out-of-vocabulary words to the *unknown word*

$$P(w_i | w_{i-1}) = \frac{1 + C(w_{i-1}, w_i)}{\sum_{w_i} (1 + C(w_{i-1}, w_i))} = \frac{1 + C(w_{i-1}, w_i)}{V + \sum_{w_i} C(w_{i-1}, w_i)}$$

- Other *backoff* smoothing algorithms possible: Katz, Kneser-Ney, Good-Turing, class ngrams
Adaptive Language Models

- Cache Language Models

\[ P_{\text{cache}}(w_i \mid w_{i-n+1}...w_{i-1}) = \lambda_c P_s(w_i \mid w_{i-n+1}...w_{i-1}) + (1 - \lambda_c)P_{\text{cache}}(w_i \mid w_{i-2}w_{i-1}) \]

- Topic Language Models

- Maximum Entropy Language Models
Bigram Perplexity

• Trained on 500 million words and tested on Encarta Encyclopedia
• OOV rate measured on Encarta Encyclopedia.
  Trained on 500 million words.
WSJ Results

• Perplexity and word error rate on the 60000-word Wall Street Journal continuous speech recognition task.

• Unigrams, bigrams and trigrams were trained from 260 million words

• Smoothing mechanisms Kneser-Ney

<table>
<thead>
<tr>
<th>Models</th>
<th>Perplexity</th>
<th>Word Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unigram</td>
<td>1199.59</td>
<td>14.86%</td>
</tr>
<tr>
<td>Bigram</td>
<td>176.11</td>
<td>11.34%</td>
</tr>
<tr>
<td>Trigram</td>
<td>91.47</td>
<td>9.60%</td>
</tr>
</tbody>
</table>
**Pattern Classification**

**Goal:** Combine information (probabilities) from the acoustic model, language model and word lexicon to generate an “optimal” word sequence (highest probability).

**Method:** Decoder searches through all possible recognition choices using a Viterbi decoding algorithm.

**Challenge:** Efficient search through a large network space is computationally expensive for large vocabulary ASR.
The Problem of Large Vocabulary ASR

• The basic problem in ASR is to find the sequence of words that explain the input signal. This implies the following mapping:

Features → HMM states → HMM units → Phones → Words → Sentences

For the WSJ 20K vocabulary, this results in a network of $10^{22}$ bytes!

• State-of-the-art methods include fast match, multi-pass decoding, A* stack, and finite state transducers all provide tremendous speed-up by searching through the network and finding the best path that maximizes the likelihood function.
Weighted Finite State Transducers (WFST)

- Unified Mathematical framework to ASR
- Efficiency in time and space
- Good substitution for Probabilistic CFG (PCFG)

WFST can compile the network to $10^8$ states
14 orders of magnitude more efficient
Confidence Scoring

Goal:
Identify possible recognition errors and out-of-vocabulary events. Potentially improves the performance of ASR, SLU and DM.

Method:
A confidence score based on a hypothesis likelihood ratio test is associated with each recognized word. For example:

<table>
<thead>
<tr>
<th>Label:</th>
<th>credit please</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognized:</td>
<td>credit fees</td>
</tr>
<tr>
<td>Confidence:</td>
<td>(0.9) (0.3)</td>
</tr>
</tbody>
</table>

\[ P(W | X) = \frac{P(W)P(X | W)}{\sum_{w} P(W)P(X | W)} \]

Challenges:
Rejection of extraneous acoustic events (noise, background speech, door slams) without rejection of valid user input speech.
How to evaluate performance?

• Dictation applications: Insertions, substitutions and deletions

\[
\text{Word Error Rate} = 100\% \times \frac{\text{Subs} + \text{Dels} + \text{Ins}}{\text{No. of word in the correct sentence}}
\]

• Command-and-control: false rejection and false acceptance => ROC curves.
## ASR Performance: The State-of-the-art

<table>
<thead>
<tr>
<th>CORPUS (DARPA)</th>
<th>TYPE OF SPEECH</th>
<th>VOCABULARY SIZE</th>
<th>WORD ERROR RATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connected Digit Strings</td>
<td>Read speech</td>
<td>10</td>
<td>0.3%</td>
</tr>
<tr>
<td>Airline Travel Information</td>
<td>Spontaneous</td>
<td>2500</td>
<td>2.5%</td>
</tr>
<tr>
<td>North American Business News</td>
<td>Read speech</td>
<td>64,000</td>
<td>6.6%</td>
</tr>
<tr>
<td>Broadcast News</td>
<td>News shows</td>
<td>210,000</td>
<td>13-17%</td>
</tr>
<tr>
<td>Switchboard</td>
<td>Conversational Telephone</td>
<td>45,000</td>
<td>25-29%</td>
</tr>
</tbody>
</table>
Growth in Effective Recognition Vocabulary Size
Improvement in Word Accuracy

Switchboard/Call Home Vocabulary: 40,000 words
Perplexity: 85

10-20% relative reduction per year.
Human Speech Recognition vs ASR

Machines Outperform Humans

HUMAN ERROR (%)

0.001 0.01 0.1 1 10

100 10 1 0.1

MACHINE ERROR (%)

Digits RM-LM NAB-mic WSJ
RM-null NAB-omni SWBD WSJ-22dB
Challenges in ASR

System Performance
- Accuracy
- Efficiency (speed, memory)
- Robustness

Operational Performance
- End-point detection
- User barge-in
- Utterance rejection
- Confidence scoring

*Machines are 10-100 times less accurate than humans*
Spoken Language Understanding
Semantic Representation

• Semantic object: a linguistic unit that carries meaning
• Semantic object is often composed of a collection of semantic objects with certain relationships
  – Is-a
  – Has-a
Example:

Where is the office of Kuansan’s manager?

<Directory Item> <Person> by_name <Person> by_report

<Directory Query>
SLU as Pattern Recognition

• Observation: the semantic object graph (tree) represents the meaning of a sentence

• Can we treat SLU as recognizing the pattern of semantic object tree?

• Furthermore, with dialog context, can we treat SLU as a pattern recognition problem conditioned on historic semantics?
Objective: \( A_{opt} = \arg \max_A P(A|X, S_{n-1}) \)

\[ = \arg \max_A \sum_S P(A|S_n) \sum_W P(S_n|W, S_{n-1}) P(W|X, S_{n-1}) \]

(Viterbi) \[ = \arg \max_{A,S} P(A|S_n) \sum_W P(S_n|W, S_{n-1}) P(W|X, S_{n-1}) \]

- \( A \): proper actions
- \( S_n \): discourse semantics for the \( n^{th} \) turn
- From dialog point of view, S is a hidden process
- Context sensitive recognition
**Domain Experts**

**Semantic Engine**

**Rendering Engine**

- **User Inputs (X)**
  - infer user intentions
  - manage discourse and dialog context
  - apply domain knowledge
  - plan dialog actions

- **Multimedia Outputs (A)**

**Architecture**

- **Recognition Engine**
- **Semantic Engine**
- **Domain Experts**

**Middleware**
Application Grammars

What the user might do (say) $P(W|x, S_{n-1})$

Language model

Recognition Engine

What does that mean $P(S_n|W, S_{n-1})$?

Semantic model

Domain Experts

Behavior model

Rendering Engine

What to show (say) to the user $P(A|S_n)$
Language Model for Understanding using PCFG

\[ P(S; \text{Schd-Appointment}) = P(\text{Schd-Appointment}) \times P_{CFG}(\text{Schedule a Meeting with Peter} | \text{Schd-Appointment}) \]

For PCFG:

\[ P_{CFG}(\text{Schedule a Dr Who Meeting with Peter} | \text{Schd-Appointment}) = 0 \]
Smoothing with Ngram

\[
P(S) = P(SchAction \mid <s>, <s>) \times P_{CFG}(Schedule \mid SchAction) \\
\times P(a \mid SchAction, <s>) \times P(DrWho \mid SchAction, a) \\
\times P(Apnmt-Type \mid a, DrWho) \times P_{CFG}(meeting \mid Apnmt-Type) \\
\times P(Participant \mid DrWho, Apnmt-Type) \times P_{CFG}(with Peter \mid Participant) \\
\times P(<s> \mid Apnmt-Type Participant)
\]
Discourse Semantic Management

• At n-th dialog turn:
  – Attach parse tree of W to current discourse tree $S_{n-1}$
  – Resolve semantic objects into domain entities
  – Goal achieved when root node is converted successfully

• Entity Memory
  – Associative chunking memory: typed priority queue
  – Model for human’s working memory

• Reference Resolution
  – Based on entity memory
  – Resolve anaphora, ellipsis, and deixis in the same manner
“Send **it** to **those in the meeting on Wednesday**”
“On Wednesday, you had two meetings. The design meeting…”

Dynamic LM for ordinal anaphora:
• the first one = design
• the second one = review

Message id=2293

Send Mail

Recipients (People)

Meeting attendees

Existing meeting

Time

Date

Subject

Location

Wednesday
“I mean the first one”
"I mean the first one"
“Message sent. Anything else?”

Send Mail

Message
id=2293

xdh, kuansanw, hon,…
Who manages the dialog?

System Initiative User

Please say collect, calling card, third number.

How can I help you?
Example: Mixed-Initiative Dialog

- **System Initiative**
  
  System: Please say *just* your departure city.
  User: Chicago
  System: Please say *just* your arrival city.
  User: Newark

- **Mixed Initiative**
  
  System: Please say your departure city
  User: I need to travel from Chicago to Newark tomorrow.

- **User Initiative**
  
  System: How may I help you?
  User: I need to travel from Chicago to Newark tomorrow.

Long dialogs but easier to design

Shorter dialogs (better user experience) but more difficult to design
**Dialog Evaluation**

**User Satisfaction** is the ultimate objective of a dialog system

- Task completion rate
- Efficiency of the dialog interaction
- Usability of the system
- Perceived intelligibility of the system
- Quality of the audio output
- Perplexity and quality of the response generation

**Machine learning** techniques can be applied to predict user satisfaction (e.g., Paradise framework)
Reading Materials