

# CS 224S / LINGUIST 236 Speech Recognition and Synthesis

Dan Jurafsky

## Lecture 13: Dialogue and Conversational Agents

IP Notice:

2/15/05

CS 224S Winter 2005

1

## Outline

- **Conversational Agents**
- **Components**
  - ASR
  - NLU
  - Generation
  - Dialogue Manager
- **Dialogue Manager Design**
  - Finite State
  - Frame-based
  - Initiative: User, System, Mixed
- **VoiceXML**

2/15/05

CS 224S Winter 2005

2

## Conversational Agents

- **AKA:**
  - Spoken Language Systems
  - Dialogue Systems
  - Speech Dialogue Systems
- **Applications:**
  - Travel arrangements (Amtrak, United airlines)
  - Telephone call routing
  - Tutoring
  - Communicating with robots
  - Anything with limited screen/keyboard

2/15/05

CS 224S Winter 2005

3

## A travel dialog: Communicator

S1: Hello. You've reached the [Communicator]. Tell me your full name  
U2: *hi i'd like to fly to seattle tuesday morning*  
S3: Traveling to Seattle on Tuesday, August eleventh in the morning. Your full name?  
U4: *[John Doe]*  
S5: Hello, [John]. I'm glad to hear from you again. You can now ask me for help.  
S6: OK. Let's see I have a United flight departing Pittsburgh at ten forty five a.m. arriving into Seattle at one fifty seven p.m. Is that OK?  
U7: *Yes*  
S8: Will you return to Pittsburgh from Seattle?  
U9: *no i will go to Los Angeles*  
S10: Traveling to Los Angeles, on what date  
U11: *Sorry can we go back to talk about the previous leg again*  
S12: For the first leg, you are on a United flight 1446 departing Pittsburgh at ten forty five A.M. tomorrow, arriving in Seattle at one fifty seven P.M. Do you want to take that?  
U13: *actually i want to leave on wednesday*  
S14: a flight on wednesday.  
S15: I have a flight departing Pittsburgh at ten forty five a.m., arrives Seattle at one fifty seven p.m. Is that OK?  
U16: *Yes*

2/15

**Figure 19.1** The travel domain: a fragment from a successful conversation between a user (U) and the Communicator system (S) of Xu and Rudnicky (2000).

4

## Call routing: ATT HMIHY

S: How may I help you?  
U: *can you tell me how much it is to tokyo?*  
S: You want to know the cost of a call?  
U: *yes that's right*  
S: Please hold on for rate information.  
S: How may I help you?  
U: *what's the area code for chicago?*  
S: You want an area code.  
U: *(silence)*  
S: The area code for Chicago is 312.

**Figure 19.2** The call routing/telephone information domain: two actual conversations between a user (U) and the AT&T How May I Help You (HMIHY) system (S) (Gorin et al., 1997).

2/15/05

CS 224S Winter 2005

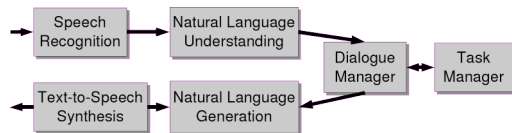
5

## A tutorial dialogue: ITSPOKE

S: I didn't understand. What force acts on all objects within close proximity to earth?  
U: *uh the force of gravity*  
S: Fine. Besides the Earth's gravitational force, are there any other forces acting on the person, keys, and elevator after the release?  
U: *no*  
S: Yep. Thus, keep in mind that after the release, the only force on the person, keys, and elevator is the force of gravity. If gravity is the only force acting on an object, will it be moving or staying still?  
U: *it will be moving*  
S: Fine. Not only are the person, keys, and elevator moving, they have only gravitational forces on them. When an object is falling and has only gravitational force on it, it is said to be in what?  
U: *freefall*

**Figure 19.3** The tutoring domain: part of an actual conversation between a student user (U) and the ITSPOKE system (S) of (Litman and Silliman, 2004), based on the Why2-Atlas text-based tutoring system (?).

## Dialogue System Architecture



2/15/05

CS 224S Winter 2005

7

## ASR engine

- Standard ASR engine that we've seen
  - Speech to words
- But specific characteristics for dialogue
  - Language models could depend on where we are in the dialogue
  - Could make use of the fact that we are talking to the same human over time.
  - Barge-in (human will talk over the computer)
  - Confidence values
    - (As we will see), we want to know if we misunderstood the human.

2/15/05

CS 224S Winter 2005

8

## Language Model

- Language models for dialogue are often based on hand-written Context-Free or finite-state grammars rather than N-grams
- Why? Because of need for understanding; we need to constrain user to say things that we know what to do with.

2/15/05

CS 224S Winter 2005

9

## Language Models for Dialogue (2)

- We can have LM specific to a dialogue state
- If system just asked "What city are you departing from?"
- LM can be
  - City names only
  - FSA: (I want to (leave|depart)) (from) [CITYNAME]
  - N-grams trained on answers to "Cityname" questions from labeled data
- A LM that is constrained in this way is technically called a "restricted grammar" or "restricted LM"

2/15/05

CS 224S Winter 2005

10

## Talking to the same human over the whole conversation.

- Same speaker
- So can adapt to speaker
  - Acoustic Adaptation
    - Vocal Tract Length Normalization (VTLN)
    - Maximum Likelihood Linear Regression (MLLR)
  - Language Model adaptation
  - Pronunciation adaptation

2/15/05

CS 224S Winter 2005

11

## Barge-in

- Speakers barge-in
- Need to deal properly with this via speech-detection, etc.

2/15/05

CS 224S Winter 2005

12

## Natural Language Understanding

- Or "NLU"
- Or "Computational semantics"
- There are many ways to represent the meaning of sentences
- For speech dialogue systems, most common is "Frame and slot semantics".

2/15/05

CS 224S Winter 2005

13

## An example of a frame

- Show me morning flights from Boston to SF on Tuesday.
- SHOW:
- FLIGHTS:
- ORIGIN:
- CITY: Boston
- DATE: Tuesday
- TIME: morning
- DEST:
- CITY: San Francisco

2/15/05

CS 224S Winter 2005

14

## How to generate this semantics?

- Many methods,
- Simplest: "semantic grammars"
- CFG in which the LHS of rules is a semantic category:
  - LIST -> show me | I want | can I see|...
  - DEPARTTIME -> (after|around|before) HOUR  
| morning | afternoon | evening
  - HOUR -> one|two|three...|twelve (am|pm)
  - FLIGHTS -> (a) flight|flights
  - ORIGIN -> from CITY
  - DESTINATION -> to CITY
  - CITY -> Boston | San Francisco | Denver | Washington

2/15/05

CS 224S Winter 2005

15

## Semantics for a sentence

LIST      FLIGHTS      ORIGIN  
Show me   flights      from Boston

DESTINATION      DEPARTDATE  
to San Francisco   on   Tuesday

DEPARTTIME  
morning

2/15/05

CS 224S Winter 2005

16

## Frame-filling

- We use a parser to take these rules and apply them to the sentence.
- Resulting in a semantics for the sentence
- SHOW PICTURE OF A SEMANTIC PARSE
- We can then write some simple code
- That takes the semantically labeled sentence
- And fills in the frame.

2/15/05

CS 224S Winter 2005

17

## Other NLU Approaches

- Syntactic rules with semantic attachments
  - This latter is what is done in VoiceXML
  - This is also what's done in various Stanford and PARC grammars used in "Grammar Engineering" course
- Cascade of Finite-State-Transducers
  - In practice, many rules have no recursion
  - So don't need CFG
  - Can use finite automata instead

2/15/05

CS 224S Winter 2005

18

## Problems with any of these semantic grammars

- Relies on hand-written grammar
  - Expensive
  - May miss possible ways of saying something if the grammar-writer just doesn't think about them
- Not probabilistic
  - In practice, every sentence is ambiguous
  - Probabilities are best way to resolve ambiguities
  - We know a lot about how to learn and build good statistical models!

2/15/05

CS 224S Winter 2005

19

## HMMs for semantics

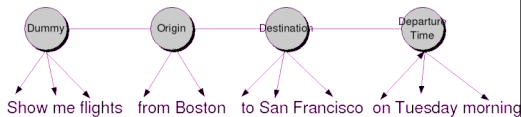
- Idea: use an HMM for semantics, just as we did for part-of-speech tagging and for speech recognition
- Hidden units:
  - Semantic slot names
    - Origin
    - Destination
    - Departure time
- Observations:
  - Word sequences

2/15/05

CS 224S Winter 2005

20

## HMM model of semantics - Pieraccini et al (1991)



2/15/05

CS 224S Winter 2005

21

## Semantic HMM

- Goal of HMM model:
  - to compute labeling of semantic roles  $C = c_1, c_2, \dots, c_n$  ( $C$  for 'cases' or 'concepts')
  - that is most probable given words  $W$

$$\begin{aligned} \operatorname{argmax}_C P(C|W) &= \operatorname{argmax}_C \frac{P(W|C)P(C)}{P(W)} \\ &= \operatorname{argmax}_C P(W|C)P(C) \\ &= \operatorname{argmax}_C \prod_{i=2}^N P(w_i | w_{i-1} \dots w_1, C) P(w_1 | C) \prod_{i=2}^M P(c_i | c_{i-1} \dots c_1) \end{aligned}$$

2/15/05

CS 224S Winter 2005

22

## Semantic HMM

- From previous slide:
 
$$= \operatorname{argmax}_C \prod_{i=2}^N P(w_i | w_{i-1} \dots w_1, C) P(w_1 | C) \prod_{i=2}^M P(c_i | c_{i-1} \dots c_1)$$
- Assume simplification:
 
$$P(w_i | w_{i-1} \dots w_1, C) = P(w_i | w_{i-1}, \dots, w_{i-N+1}, c_i)$$

$$P(c_i | c_{i-1} \dots c_1, C) = P(c_i | c_{i-1}, \dots, c_{i-M+1})$$
- Final form:
 
$$= \operatorname{argmax}_C \prod_{i=2}^N P(w_i | w_{i-1} \dots w_{i-N+1}, c_i) \prod_{i=2}^M P(c_i | c_{i-1} \dots c_{i-M+1})$$

2/15/05

CS 224S Winter 2005

23

## Generation and TTS

- Generation component
  - Chooses concepts to express to user
  - Plans out how to express these concepts in words
  - Assigns any necessary prosody to the words
- TTS component
  - Takes words and prosodic annotations
  - Synthesizes a waveform

2/15/05

CS 224S Winter 2005

24

## Generation Component

- **Content Planner**
  - Decides what content to express to user
    - (ask a question, present an answer, etc)
  - Often merged with dialogue manager
- **Language Generation**
  - Chooses syntactic structures and words to express meaning.
  - Simplest method
    - All words in sentence are prespecified!
    - "Template-based generation"
  - Can have variables:
    - What time do you want to leave CITY-ORIG?
    - Will you return to CITY-ORIG from CITY-DEST?

2/15/05

CS 224S Winter 2005

25

## More sophisticated language generation component

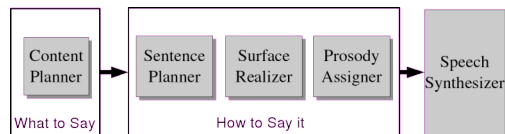
- **Natural Language Generation**
- This is a field, like Parsing, or Natural Language Understanding, or Speech Synthesis, with its own (small) conference
- Approach:
  - Dialogue manager builds representation of meaning of utterance to be expressed
  - Passes this to a "generator"
  - Generators have three components
    - Sentence planner
    - Surface realizer
    - Prosody assigner

2/15/05

CS 224S Winter 2005

26

## Architecture of a generator for a dialogue system (after Walker and Rambow 2002)



2/15/05

CS 224S Winter 2005

27

## HCI constraints on generation for dialogue: "Coherence"

- Discourse markers and pronouns ("Coherence"):

(1) Please say the date.

Please say the start time.

Please say the duration...

Please say the subject...

**Bad!**

(2) First, tell me the date.

Next, I'll need the time it starts.

**Good!**

Thanks. <pause> Now, how long is it supposed to last?

Last of all, I just need a brief description

2/15/05

CS 224S Winter 2005

28

## HCI constraints on generation for dialogue: coherence (II): tapered prompts

- Prompts which get incrementally shorter:
- System: Now, what's the first company to add to your watch list?
- Caller: Cisco
- System: What's the next company name? (Or, you can say, "Finished")
- Caller: IBM
- System: Tell me the next company name, or say, "Finished."
- Caller: Intel
- System: Next one?
- Caller: America Online.
- System: Next?
- Caller: ...

2/15/05

CS 224S Winter 2005

29

## Dialogue Manager

- Controls the architecture and structure of dialogue
  - Takes input from ASR/NLU components
  - Maintains some sort of state
  - Interfaces with Task Manager
  - Passes output to NLG/TTS modules

2/15/05

CS 224S Winter 2005

30

## Four architectures for dialogue management

- Finite State
- Frame-based
- Planning Agents
- Markov Decision Processes

2/15/05

CS 224S Winter 2005

31

## Finite-State Dialogue Mgmt

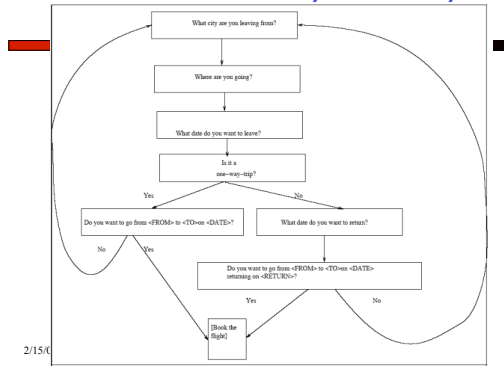
- Consider a trivial airline travel system
  - Ask the user for a departure city
  - For a destination city
  - For a time
  - Whether the trip is round-trip or not

2/15/05

CS 224S Winter 2005

32

## Finite State Dialogue Manager



2/15/05

## Finite-state dialogue managers

- System completely controls the conversation with the user.
- It asks the user a series of questions
- Ignoring (or misinterpreting) anything the user says that is not a direct answer to the system's questions

2/15/05

CS 224S Winter 2005

34

## Dialogue Initiative

- Systems that control conversation like this are **system initiative** or **single initiative**.
- "Initiative": who has control of conversation
- In normal human-human dialogue, initiative shifts back and forth between participants.

2/15/05

CS 224S Winter 2005

35

## System Initiative

- Systems which completely control the conversation at all times are called **system initiative**.
- Advantages:
  - Simple to build
  - User always knows what they can say next
  - System always knows what user can say next
    - Known words: Better performance from ASR
    - Known topic: Better performance from NLU
  - Ok for VERY simple tasks (entering a credit card, or login name and password)
- Disadvantage:
  - Too limited

2/15/05

CS 224S Winter 2005

36

## User Initiative

- User directs the system
- Generally, user asks a single question, system answers
- System can't ask questions back, engage in clarification dialogue, confirmation dialogue
- Used for simple database queries
- User asks question, system gives answer
- Web search is user initiative dialogue.

2/15/05

CS 224S Winter 2005

37

## Problems with System Initiative

- Real dialogue involves give and take!
- In travel planning, users might want to say something that is not the direct answer to the question.
- For example answering more than one question in a sentence:
  - Hi, I'd like to fly from Seattle Tuesday morning
  - I want a flight from Milwaukee to Orlando one way leaving after 5 p.m. on Wednesday.

2/15/05

CS 224S Winter 2005

38

## Single initiative + universals

- We can give users a little more flexibility by adding universal commands
- Universals: commands you can say anywhere
- As if we augmented every state of FSA with these
  - Help
  - Start over
  - Correct
- This describes many implemented systems
- But still doesn't allow user to say what they want to say

2/15/05

CS 224S Winter 2005

39

## Mixed Initiative

- Conversational initiative can shift between system and user
- Simplest kind of mixed initiative: use the structure of the frame itself to guide dialogue

- Slot	Question
- ORIGIN	What city are you leaving from?
- DEST	Where are you going?
- DEPT DATE	What day would you like to leave?
- DEPT TIME	What time would you like to leave?
- AIRLINE	What is your preferred airline?

2/15/05

CS 224S Winter 2005

40

## Frames are mixed-initiative

- User can answer multiple questions at once.
- System asks questions of user, filling any slots that user specifies
- When frame is filled, do database query
- If user answers 3 questions at once, system has to fill slots and not ask these questions again!
- Anyhow, we avoid the strict constraints on order of the finite-state architecture.

2/15/05

CS 224S Winter 2005

41

## Multiple frames

- flights, hotels, rental cars
- Flight legs: Each flight can have multiple legs, which might need to be discussed separately
- Presenting the flights (If there are multiple flights meeting users constraints)
  - It has slots like 1ST\_FLIGHT or 2ND\_FLIGHT so user can ask "how much is the second one"
- General route information:
  - Which airlines fly from Boston to San Francisco
- Airfare practices:
  - Do I have to stay over Saturday to get a decent airfare?

2/15/05

CS 224S Winter 2005

42

## Multiple Frames

- Need to be able to switch from frame to frame
- Based on what user says.
- Disambiguate which slot of which frame an input is supposed to fill, then switch dialogue control to that frame.
- Main implementation: production rules
  - Different types of inputs cause different productions to fire
  - Each of which can flexibly fill in different frames
  - Can also switch control to different frame

2/15/05

CS 224S Winter 2005

43

## Defining Mixed Initiative

- Mixed Initiative could mean
  - User can arbitrarily take or give up initiative in various ways
    - This is really only possible in very complex plan-based dialogue systems
    - No commercial implementations
    - Important research area
  - Something simpler and quite specific which we will define in the next few slides

2/15/05

CS 224S Winter 2005

44

## True Mixed Initiative

C<sub>1</sub>: ... I need to travel in May.  
A<sub>1</sub>: And, what day in May did you want to travel?  
C<sub>2</sub>: OK uh I need to be there for a meeting that's from the 12th to the 15th.  
A<sub>2</sub>: And you're flying into what city?  
C<sub>3</sub>: Seattle.  
A<sub>3</sub>: And what time would you like to leave Pittsburgh?  
C<sub>4</sub>: Uh hmm I don't think there's many options for non-stop.  
A<sub>4</sub>: Right. There's three non-stops today.  
C<sub>5</sub>: What are they?  
A<sub>5</sub>: The first one departs PGH at 10:00am arrives Seattle at 12:05 their time.  
The second flight departs PGH at 5:55pm, arrives Seattle at 8pm. And the last flight departs PGH at 8:15pm arrives Seattle at 10:28pm.  
C<sub>6</sub>: OK I'll take the 5ish flight on the night before on the 11th.  
A<sub>6</sub>: On the 11th? OK. Departing at 5:55pm arrives Seattle at 8pm, U.S. Air flight 115.  
C<sub>7</sub>: OK.

## How mixed initiative is usually defined

- First we need to define two other factors
- Open prompts vs. directive prompts
- Restrictive versus non-restrictive grammar

2/15/05

CS 224S Winter 2005

46

## Open vs. Directive Prompts

- Open prompt
  - System gives user very few constraints
  - User can respond how they please:
  - "How may I help you?" "How may I direct your call?"
- Directive prompt
  - Explicit instructs user how to respond
  - "Say yes if you accept the call; otherwise, say no"

2/15/05

CS 224S Winter 2005

47

## Restrictive vs. Non-restrictive grammars

- Restrictive grammar
  - Language model which strongly constrains the ASR system, based on dialogue state
- Non-restrictive grammar
  - Open language model which is not restricted to a particular dialogue state

2/15/05

CS 224S Winter 2005

48

## Definition of Mixed Initiative

Grammar	Open Prompt	Directive Prompt
Restrictive	<i>Doesn't make sense</i>	System Initiative
Non-restrictive	User Initiative	Mixed Initiative

2/15/05

CS 224S Winter 2005

49

## VoiceXML

- Voice eXtensible Markup Language
- An XML-based dialogue design language
- Makes use of ASR and TTS
- Deals well with simple, frame-based mixed initiative dialogue.
- Most common in commercial world (too limited for research systems)
- But useful to get a handle on the concepts.

2/15/05

CS 224S Winter 2005

50

## Voice XML

- Each dialogue is a `<form>`. (**Form** is the VoiceXML word for **frame**)
- Each `<form>` generally consists of a sequence of `<field>`s, with other commands

2/15/05

CS 224S Winter 2005

51

## Sample vxml doc

```
<form>
  <field name="transporttype">
    <prompt>
      Please choose airline, hotel, or rental car. </prompt>
    <grammar type="application/x-nuance-gsl">
      [airline hotel "rental car"]
    </grammar>
  </field>
  <block>
    <prompt>
      You have chosen <value expr="transporttype">. </prompt>
    </block>
  </form>
```

2/15/05

CS 224S Winter 2005

52

## VoiceXML interpreter

- Walks through a VXML form in document order
- Iteratively selecting each item
- If multiple fields, visit each one in order.
- Special commands for events

2/15/05

CS 224S Winter 2005

53

## Another vxml doc (1)

```
<noinput>
  I'm sorry, I didn't hear you. <reprompt/>
</noinput>
- "noinput" means silence exceeds a timeout threshold

<nomatch>
  I'm sorry, I didn't understand that. <reprompt/>
</nomatch>

- "nomatch" means confidence value for utterance is too low
- notice "reprompt" command
```

2/15/05

CS 224S Winter 2005

54

## Another vxml doc (2)

```
<form>
<block> Welcome to the air travel consultant. </block>
<field name="origin">
  <prompt> Which city do you want to leave from? </prompt>
  <grammar type="application/x=nuance-gsl">
    [(san francisco) denver (new york) barcelona]
  </grammar>
  <filled>
    <prompt> OK, from <value expr="origin"> </prompt>
  </filled>
</field>
```

- "filled" tag is executed by interpreter as soon as field filled by user

2/15/05

CS 224S Winter 2005

55

## Another vxml doc (3)

```
<field name="destination">
  <prompt> And which city do you want to go to? </prompt>
  <grammar type="application/x=nuance-gsl">
    [(san francisco) denver (new york) barcelona]
  </grammar>
  <filled>
    <prompt> OK, to <value expr="destination"> </prompt>
  </filled>
</field>
<field name="departdate" type="date">
  <prompt> And what date do you want to leave? </prompt>
  <filled>
    <prompt> OK, on <value expr="departdate"> </prompt>
  </filled>
</field>
```

2/15/05

CS 224S Winter 2005

56

## Built in grammar (LM) types

```
<field name="departdate" type="date">
```

- VoiceXML 2.0 has seven built-in grammar types: **boolean, currency, date, digits, number, phone, time**
- For these, you don't need to specify an LM; there is a pre-built LM.
- Reminder: for ASR in dialogue systems, semantics is much more important than for ASR in dictation tasks.

2/15/05

CS 224S Winter 2005

57

## Another vxml doc (4)

```
<block>
  <prompt> OK, I have you are departing from
    <value expr="origin"> to <value expr="destination">
    on <value expr="departdate">
  </prompt>
  send the info to book a flight...
</block>
</form>
```

2/15/05

CS 224S Winter 2005

58

## A mixed initiative VXML doc

- Mixed initiative: user might answer a different question than the system asked
- So VoiceXML interpreter can't just evaluate each field of form in order
- User might answer field2 when system asked field1
- So need grammar which can handle all sorts of input:
  - Field1
  - Field2
  - Field 1 and field 2
  - etc

2/15/05

CS 224S Winter 2005

59

## VXML Nuance-style grammars

- Rewrite rules
  - Wantsentence -> I want to (fly|go)
- Nuance VXML format is:
  - () for concatenation, [] for disjunction
  - Each rule has a name:
  - Wantsentence (I want to [fly go])
  - Airports [(san francisco) denver]

2/15/05

CS 224S Winter 2005

60

## Mixed-init VXML example (3)

```
<noinput> I'm sorry, I didn't hear you.
<reprompt/> </noinput>

<nomatch> I'm sorry, I didn't understand that.
<reprompt/> </nomatch>

<form>
  <grammar type="application/x=nuance-gsl">
    <![CDATA[
```

2/15/05

CS 224S Winter 2005

61

## Grammar

```
Flight ( ?[
  (i [wanna (want to)] [fly go])
  (i'd like to [fly go])
  (((i wanna)(i'd like a)] flight)
]
[
  ([from leaving departing] City:x) {<origin $x>}
  ( [(?going to)(arriving in)] City:x) {<dest $x>}
  ([from leaving departing] City:x
  [(?going to)(arriving in)] City:y) {<origin $x> <dest $y>}
]
?please
)
```

2/15/05

CS 224S Winter 2005

62

## Grammar

```
City [ [(san francisco) (s f o)] {return( "san francisco, california")}
      [(denver) (d e n)] {return( "denver, colorado")}
      [(seattle) (s t x)] {return( "seattle, washington")}
]
]]> </grammar>
```

2/15/05

CS 224S Winter 2005

63

## Grammar

```
<initial name="init">
  <prompt> Welcome to the air travel consultant. What are your travel plans?
</prompt>
</initial>
<field name="origin">
  <prompt> Which city do you want to leave from? </prompt>
  <filled>
    <prompt> OK, from <value expr="origin"> </prompt>
  </filled>
</field>
```

2/15/05

CS 224S Winter 2005

64

## Grammar

```
<field name="dest">
  <prompt> And which city do you want to go to? </prompt>
  <filled>
    <prompt> OK, to <value expr="dest"> </prompt>
  </filled>
</field>
<block>
  <prompt> OK, I have you are departing from <value expr="origin">
    to <value expr="dest">. </prompt>
  send the info to book a flight...
</block>
</form>
```

2/15/05

CS 224S Winter 2005

65

## Summary: VoiceXML

- Voice eXtensible Markup Language
- An XML-based dialogue design language
- Makes use of ASR and TTS
- Deals well with simple, frame-based mixed initiative dialogue.
- Most common in commercial world (too limited for research systems)
- But useful to get a handle on the concepts.

2/15/05

CS 224S Winter 2005

66

## User-centered dialogue system design

---

1. Early focus on users and task:
  - interviews, study of human-human task, etc.
2. Build prototypes:
  - Wizard of Oz systems
3. Iterative Design:
  - iterative design cycle with embedded user testing

2/15/05

CS 224S Winter 2005

67

## Dialogue system Evaluation

---

- Whenever we design a new algorithm or build a new application, need to evaluate it
- How to evaluate a dialogue system?
- What constitutes success or failure for a dialogue system?

2/15/05

CS 224S Winter 2005

68

## Task Completion Success

---

- % of subtasks completed
- Correctness of each questions/answer/error msg
- Correctness of total solution

2/15/05

CS 224S Winter 2005

69

## Task Completion Cost

---

- Completion time in turns/seconds
- Number of queries
- Turn correction ration: number of system or user turns used solely to correct errors, divided by total number of turns
- Inappropriateness (verbose, ambiguous) of system's questions, answers, error messages

2/15/05

CS 224S Winter 2005

70

## User Satisfaction

---

- Were answers provided quickly enough?
- Did the system understand your requests the first time?
- Do you think a person unfamiliar with computers could use the system easily?

2/15/05

CS 224S Winter 2005

71

## Summary

---

2/15/05

CS 224S Winter 2005

72